ISSN 1213-6972 *Volume 28, Number 1-2, 2020*

Journal of WSCG

An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.

EDITOR – IN – CHIEF

Václav Skala

Vaclav Skala – Union Agency

Journal of WSCG

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia
Faculty of Applied Sciences
Univerzitni 8
CZ 306 14 Plzen
Czech Republic
http://www.VaclavSkala.eu

Managing Editor: Vaclav Skala

Printed and Published by:
Vaclav Skala - Union Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic

Published in cooperation with the University of West Bohemia
Univerzitní 8, 306 14 Pilsen, Czech Republic

Hardcopy: ***ISSN 1213 – 6972***
CD ROM: ***ISSN 1213 – 6980***
On-line: ***ISSN 1213 – 6964***

ISSN 1213-6972 *Volume 28, Number 1-2, 2020*

Journal of WSCG

An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.

EDITOR – IN – CHIEF

Václav Skala

Vaclav Skala – Union Agency

Journal of WSCG

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia
Faculty of Applied Sciences
Univerzitni 8
CZ 306 14 Plzen
Czech Republic
http://www.VaclavSkala.eu

Managing Editor: Vaclav Skala

Printed and Published by:
Vaclav Skala - Union Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic

Published in cooperation with the University of West Bohemia
Univerzitní 8, 306 14 Pilsen, Czech Republic

Hardcopy: ***ISSN 1213 – 6972***
CD ROM: ***ISSN 1213 – 6980***
On-line: ***ISSN 1213 – 6964***

Journal of WSCG

Editor-in-Chief

Journal of WSCG URLs: http://www.wscg.eu or http://wscg.zcu.cz/jwscg

Editorial Board

Journal of WSCG

Board of Reviewers 2020

Rodrigues,J. (Portugal)
Rojas-Sola,J. (Spain)
Raja,S. (India)
Santos,L. (Portugal)
Sarfraz,M. (Kuwait)
Segura,R. (Spain)
Shamima,Y. (USA)
Shum,H. (U.K.)
Sintorn,E. (Sweden)
Sundstedt,V. (Sweden)
Takahasi,S. (Japan)
Teschner,M. (Germany)
Tokuta,A. (USA)
Toler-Franklin,C. (USA)
Trapp,M. (Germany)
Tytkowski,K. (Poland)
Vanderhaeghe,D. (France)
Wiegreffe,D. (Germany)
Wu,S. (Brazil)
Wuensche,B. (New Zealand)
Yoshizawa,S. (Japan)
Zwettler,G. (Austria)

Journal of WSCG

Vol.28, No.1-2, 2020

Contents

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.28, No.1-2, 2020

Accelerating Ray Tracing with Origin Offsets

T. Zoet
Utrecht University
Domplein 29
3512 JE Utrecht
ts.zoet@gmail.com

J. Bikker
Utrecht University
Domplein 29
3512 JE Utrecht
bikker.j@gmail.com

Abstract

A significant portion of rendering time in ray tracing consists of the traversal of an acceleration structure. While methods designed to reduce traversal cost often target the top of the tree, we instead propose a method that targets the lowest levels, by offsetting ray origins just before starting traversal. Our method moves ray origins out of nodes deep in the tree, avoiding the scattered memory access associated with these rarely visited nodes. We propose a precomputed set of hemispheres, placed on the surfaces, which is guaranteed not to contain any geometry, thus allowing rays starting inside a hemisphere to be moved to the hemispheres boundary. Our method is compatible with most acceleration structures and does not require access to the actual traversal implementation.

Keywords

Ray tracing, acceleration structures

1 INTRODUCTION

Ray tracing has wide applications. It is used in collision detection algorithms, simulations of physical phenomena and graphics. Recent advances in ray tracing hardware made real-time ray tracing accessible to the masses. In the past year, the first video games that employ ray tracing for rendering were released.

To perform efficient intersection tests between rays and a collection of objects, acceleration structures are used. The two most common acceleration structures in modern high-performance renderers, the kD-tree and bounding volume hierarchy (BVH), aim to bring the average case time complexity of tracing a single ray down to near logarithmic.

In this paper, we propose a method that aims to improve ray tracing performance by offsetting extension rays just before starting traversal. This is based on the observation that many rays visit the leaf node they originate from without finding an intersection. These unnecessary visits are relatively expensive, due to a scattered memory access pattern. Our aim is to prevent these visits by moving the ray origins along the ray direction. This shortening will often move the rays out of the leaf nodes.

In our method a set of hemispheres is placed on the surface of the scene geometry. The radius of each hemisphere is such that it does not to contain any geometry. For each ray origin we fetch the nearest hemispheres, which are then used to replace the ray origin by the intersection of the ray and the hemispheres. If the origin is moved far enough, the ray is moved out of the leaf node it originated from and potentially one or more nodes higher in the acceleration structure. Our aim is to outweigh the overhead of offset calculation by the reduced traversal cost.

2 RELATED WORK

Accelerating the intersection of rays and scene geometry is achieved using various data structures and algorithms. These methods can be put into roughly three categories:

1. Acceleration structures. The most commonly used acceleration structures, the kD-tree and BVH, aim to bring the cost of tracing a single ray from linear to near logarithmic time. They do this by recursively partitioning space and objects, respectively.

2. Amortization of traversal cost. By grouping rays together in packets, the cost of fetching the acceleration structure data can be amortized over many rays.

3. Efficient traversal of the acceleration structure. These methods often make use of high-level knowledge about rays, such as the difference between nearest-hit and any-hit (occlusion) rays, or connectivity between rays, to guide traversal towards parts of the tree that are more likely to intersect the rays.

The above categorization will not cover all methods. Some can be put into more than one category, whereas others will defy all categorization. The following sections provide a brief summary of the main works in each category, providing a context for the contributions in this paper.

2.1 Acceleration structures

Fujimoto et al. [FTI86] use a uniform grid to store scene geometry. As a ray traverses the grid, it intersects the objects in each cell it passes. Glassner [Gla84] uses an adaptive scheme, placing objects in a recursively partitioned octree. Cleary and Wyvill [CW88] provide an analysis of the uniform grid, showing that it, at that time, outperformed hierarchical methods.

Earlier, the kD-tree (or binary tree) was introduced by Bentley [Ben75], although not in the context of ray tracing. The kD-tree has found wide application. Goldsmith and Salmon [GS87] introduce a simple heuristic to guide the construction of spatial subdivision trees. MacDonald and Booth [MB90] propose the refined surface area heuristic (SAH) and show that it performs much better than using a fixed split order.

Clark [Cla76] introduced the BVH, which partitions objects rather than space. Construction of the BVH is similar to the kD-tree, and allows the use of e.g. the surface area heuristic.

Havran [Hav00] provides an in-depth analysis of a wide range of acceleration structures, including the kD-tree and BVH, and concludes that the kD-tree gives the best ray tracing performance. Wald [Wal07] greatly improves the speed of BVH construction and Stich [SFD09] solves the problem of scenes with non-uniformly sized triangles. Together, this closes the performance gap between the kD-tree and BVH. Today the BVH is the most widely used acceleration structure in renderers.

2.2 Amortization

Wald et al. [WSBW01] group several rays together in a single packet. The size of this packet is generally chosen to be the width of the vector registers on the CPU. For SSE capable CPUs, packet traversal yields a 2-3x performance improvement when compared to single ray traversal.

Reshetov et al. [RSH05] search for the first node in an acceleration structure for which both subtrees contain a leaf node that intersects a group of rays. Once the entry point has been found, individual rays start traversal at this point. The process is further optimized by Fowler et al. [FCM09], who find deeper entry points, with fewer traversal steps.

Overbeck et al. [ORM08] use large packets of up to 1024 rays and propose a traversal algorithm specifically aimed at reflection and refraction rays. These rays suffer from quickly degrading coherence as the number of bounces for each path increases.

2.3 Improved traversal

Boulos and Haines [BH10] show that occlusion rays often dominate the total rendering time. It is therefore worthwhile to devise specialised traversal strategies for occlusion rays and nearest-hit rays. MacDonald and Booth [MB90] were the first to propose an alternative traversal algorithm. They add neighbor links between the leaves of a kD-tree. Once a ray has been traced, any extension ray originating at the intersection point can then be traced by following these links. This traversal method prevents visiting many internal nodes, at the cost of additional memory usage. Havran et al. [HBZ98] show that ropes result in a 10-20% reduction of total rendering time, compared to standard kD-tree traversal. The gains are later improved to up to 35% by Havran and Bittner [HB07]. Hendrich et al. [HPMB19] use convex frustum shafts to cull parts of a BVH. Djeu et al. [Djeu09] reduce the traversal time of occlusion rays, by marking leaf nodes contained by geometry as volumetric occluders and terminating traversal in these nodes. Ize and Hansen [IH11] introduce the Ray Termination Surface Area Heuristic (RTSAH), which significantly reduces the number of visited nodes for occlusion rays.

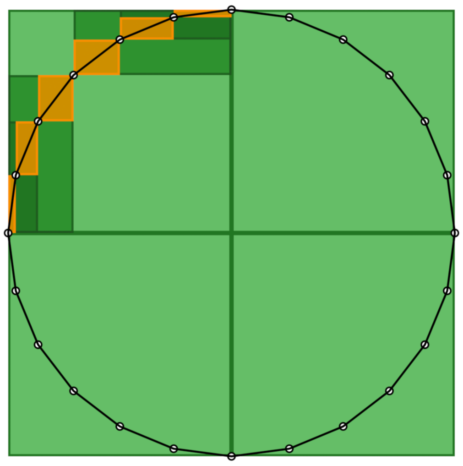


Figure 1: Part of a BVH built on a circle. In shades of green internal nodes. In orange leaf nodes. Note that the distance required to move a ray starting on the surface out of a leaf node is small.

The work presented in this paper falls under the third category. We build a set of hemispheres on top of the acceleration structure that are guaranteed not to contain any geometry. We use the hemispheres to calculate offsets for extension rays, skipping deep parts of the hierarchy.

3 OFFSETTING RAYS

Testing a ray against a leaf node is more expensive than testing against an internal node. This has two main reasons. First, leaf nodes are visited less frequently than the shallower internal nodes, and are therefore less likely to be in the cache. The same holds for the primitive data in the leaf node. Second, intersecting a leaf node involves intersecting the primitives it stores. Most methods that aim to accelerate traversal focus on skipping internal nodes or amortizing traversal costs over multiple rays and quickly reaching the leaf nodes. We argue that it could be beneficial to focus on skipping leaf nodes, avoiding scattered memory access and primitive intersection tests.

Consider rendering an opaque, tessellated sphere (Figure 1). After the primary rays have been traced, extension rays starting on the sphere's surface will not hit any geometry, as the sphere is a convex object and all rays will travel away from its surface. Still, traversal will first traverse down into the leaf node where the previous ray ended, since the new ray's origin is contained in the leaf's bounds. Offsetting the ray origin along a short distance could be enough to move it out of the leaf node. A larger offset would then cull additional nodes above and adjacent to the leaf node. While a sphere is the best-case scenario due to its convexity, the idea of offsetting ray origins extends to more complex scenes.

To calculate offsets we use an auxiliary data structure, in the form of a set of hemispheres. In a preprocessing step we place one or more hemispheres on each triangle. At each hemisphere location, we determine the nearest intersection on the positive side of the plane. The distance to this intersection is then stored as the radius of the hemisphere. The volume of this hemisphere is thus guaranteed to be empty. During rendering, for each ray, we retrieve the hemispheres attached to the triangle it originates from. The origin offset is then calculated using a ray-sphere intersection test. If the ray intersects multiple hemispheres, the greatest intersection distance is used. After offsetting the ray origin, traversal is started. Note that our method is oblivious to the chosen acceleration structure and the actual traversal, and thus orthogonal to other optimizations.

We limit our discussion to polygonal scenes and reflected rays, although the concepts are potentially applicable to other scenes and transmitted rays.

In the next two subsections we will describe various potential hemisphere placement strategies. Section 3.1 covers how the hemisphere locations can be chosen such that they best achieve our goal: moving the rays out of as many leaf nodes as possible. While the most optimal solution is of course placing an infinite number of hemispheres, to optimize execution time a balance between traversal savings and overhead must be found. Section 3.2 describes low-level optimizations to reduce overhead, such as a carefully chosen memory layout, caching behaviour and vectorization.

3.1 Hemisphere sets

The set of hemispheres must be chosen in such a way that the average ray origin offset is maximal. At the same time, we wish to limit the number of hemispheres,

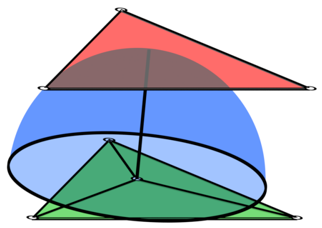


Figure 2: A hemisphere (blue) placed in the center of a triangle (green). The radius is limited by the triangle (red) above it.

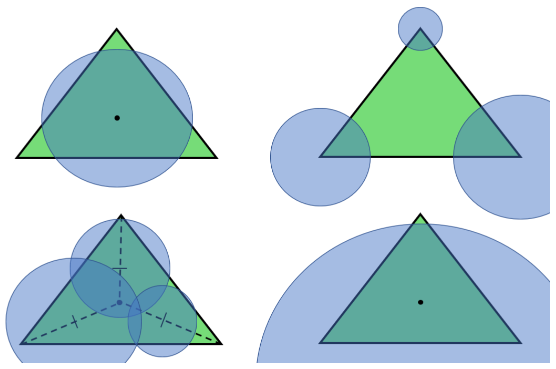


Figure 3: Visualization of the placement strategies. Top row: center sphere set and vertex sphere set. Bottom row: median sphere set and polygon sphere set.

to reduce query overhead. We investigate four placement strategies that balance these factors. The strategies are visualized in Figure 3.

Center Sphere Set. The simplest method is to store a single hemisphere located at the center of each triangle (Figure 2). At this location the hemisphere will in general have the most potential for moving a ray out of the triangle's bounding box.

Vertex Sphere Set. Alternatively, we can place a hemisphere on each triangle vertex. This method is more resistant to occluding geometry at the center: if only one area of the triangle is occluded there is still potential for a large enough hemisphere being placed on one or more of the vertices to offset nearby rays far enough to leave the triangle bounds. There is an important downside: if a vertex is part of a concave corner of the surface geometry the radius of the hemisphere will be 0.

Median Sphere Set. To solve the problem of concave geometry we can move the hemispheres closer to the triangle center. One option is to place the hemispheres halfway between each vertex and the triangle center, the medians, i.e. the line segments from the vertices to the midpoint of the opposite edges. Together, these hemispheres can cover a large area of the triangle.

Polygon Sphere Set. In some situations the center sphere set could perform better with only a slight modification. Consider a scene with many groups of adjacent triangles lying in the same plane, essentially forming (non-convex) polygons. A triangle in such a group could make use of the hemisphere of one of its neighbours. This neighbouring hemisphere might be much larger if the triangle itself lies closer to an occluding part of the scene than the neighbour. If the neighbouring hemisphere covers a large part of the triangle, it can offset many rays for much larger distances than the triangle's own hemisphere. Selecting the optimal hemisphere from all neighbours is a hard optimization problem. A potentially good approximation would be to select the largest of all hemispheres and assigning this to each triangle of the polygon.

3.2 Optimization

By placing a fixed number of hemispheres per triangle, retrieval has constant time complexity. It also allows keeping the querying code simple, minimizing the strain on the instruction cache. Furthermore, it opens up options for aligning the data to cache lines and efficient vectorized intersection tests.

Center Sphere Set. The center sphere set has the lowest storage requirements. A hemisphere record requires 16 bytes: 4 for the radius and 12 for the position, which comfortably fits in a single cache line. Alternatively, the hemisphere position can be deduced from the triangle vertex positions. Although this limits storage to 4 bytes, this is only beneficial if the vertex data is in the cache.

Vertex Sphere Set. This method uses 3 times more memory than the center sphere set. This may still fit in a single cache line, although padding to 64 bytes may be necessary to avoid extra cache line accesses. Alternatively, the hemisphere data can be stored with the vertex data, adding only the hemisphere radius to each vertex position. Obviously, this change will impact other parts of the renderer. It could degrade performance for the acceleration structure or might simply be impossible to implement (e.g. when using indexed triangles, where vertices can be shared by multiple triangles).

Median Sphere Set. The median sphere set can be used with or without the center hemisphere. Also storing the center hemisphere will not add much overhead. If the medians are recalculated from the vertices, the first step is calculating the center, and storing the 4^{th} radius allows aligning the data to 16 bytes. If the hemisphere centers are stored, adding the extra hemisphere will increase the per triangle data to 64 bytes, exactly a cache line.

Polygon Sphere Set. The polygon sphere set can be stored and queried in two distinct ways. First, using the method for the center sphere set. In that case the same optimizations can be applied. Second, using an indexed lookup. Since hemispheres are used by multiple triangles, they need not be stored multiple times. This can reduce memory overhead, if we assume that we store the hemisphere centers as well as the radii. Total memory usage would be one index per triangle to fetch the hemisphere, and one hemisphere per polygon. The number of polygons will depend on the nature of the scene and how many triangles are in the same plane as their neighbours.

Further optimization can be done using SIMD instructions. For example, using SSE allows querying the center sphere set with 4 rays at once, amortizing memory cost over the 4 rays.

4 RESULTS

The scenes used in our experiments are shown in Table 1. For each of the scenes several viewpoints are selected. Results are averaged over these viewpoints. Three ray classes are tested to account for different behaviours: ambient occlusion (AO) rays, diffuse rays and shadow rays. AO rays are rendered using several radii set to a fixed percentage of the scene size. Each primary ray that finds an intersection is used to generate exactly one AO ray. For the diffuse renders each path is extended in a random direction until it finds a light, leaves the scene or reaches a maximum recursion depth. For the shadow rays a single point light is added to each scene.

Two types of statistics are collected: the number of visited nodes per ray (split by internal nodes and leafs) and runtimes. Note that for the diffuse renders the number of nodes is calculated per ray rather than per path. Generating and tracing primary rays, writing to the image buffer or any other steps are excluded from the measurements.

All experiments were run single-threaded on a 2.60GHz Intel i7-6700HQ with 8GB DDR4 RAM. For BVH construction and intersection tests the Intel Embree ray tracing library (version 3.5.2) was used [WWB*14]. All images were rendered at 4096x4096 pixels.

4.1 Traversal

Table 2 shows the number of visited nodes and leafs for AO rays. The rays were given a length, or radius, of a percentage of the scene size. The **baseline** column shows the absolute number of visited internal nodes and leaves. These numbers increase as the radius increases. When using any of the hemisphere sets to offset the rays, the average number of visited nodes and leaves decreases, as shown by each column corresponding to the respective hemisphere set.

The median hemisphere set consistently outperforms all other sets. This is only by a small margin however, suggesting that a larger number of hemispheres per triangle has little added benefit. When comparing the results of the vertex sphere set with the center sphere set, we see that it performs better in some scenes, but worse in others. This seems related to the nature of the geometry:

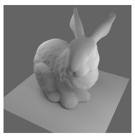
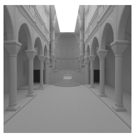
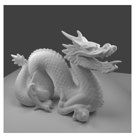
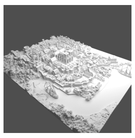
								
buddha	bunny	sibenik cathedral	dragon	hairball	living room	rungholt	sponza crytek	sponza dragon
1.1M	144k	76K	871K	2.9M	581K	5.8M	262K	1.1M
16.6	2.2	1.1	13.3	43.9	8.9	88.7	4.0	17.3

Table 1: Overview of the scenes used in the experiments. For each scene we report the triangle count and the memory cost (in MB) of the 16-byte center sphere set.

the architectural scenes that contain more concave geometry, such as the Cathedral, make the vertex sphere set less useful. The polygon sphere set shows no improvement over the center sphere set, suggesting that selecting the largest hemisphere is a poor optimization strategy.

Scenes such as Buddha, Bunny and Dragon show a reduction of visited leaf nodes by up to 85% for short rays. For longer rays reductions are not as large, although still at least 30%. The strong reduction for short rays is due to many rays receiving an offset that is larger than their length, causing traversal to terminate in the root node. Figure 4 shows this for the Dragon scene: the fraction of rays for which the calculated offset (using the center sphere set) is larger than the ray length is high for short radii and then drops quickly. For scenes that do not contain both large open spaces and convex geometry, this fraction drops much faster.

The Rungholt scene contains almost exclusively axis-aligned geometry. Because of this, the majority of extension rays do not start inside the bounding box of the leaf node the primary ray ended in, and there is little improvement with offsets. Also shown in Table 2 are results for the Rungholt scene rotated by 15°. The baseline shows that the number of visited internal nodes increases by about 20%, while the number of visited leaves doubles. When using offsets, this increase is partially undone.

The Sponza Dragon scene is the Sponza Crytek scene with the dragon model added in its center. While the Sponza Crytek scene shows negligible reduction for long rays, with the dragon relative improvements become larger, showing that the sphere sets adapt to varying geometry. The Sponza Dragon scene could be seen as a common use case for e.g. games, where detailed character models move through buildings.

Table 3 and Table 4 show the results for diffuse and shadow rays. The overall behaviour for all scenes and sphere sets is similar to that of the ambient occlusion rays, with some slight differences. The scenes in which the AO rays saw the largest reductions do not fare as well. Conversely, some of the worst performers for long AO rays (such as the Living Room and Sponza Crytek scenes) do show more significant reductions.

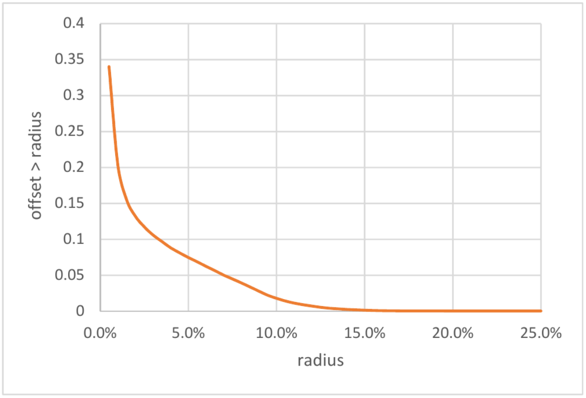


Figure 4: Fraction of AO rays for which the offset is larger than the radius (in the Dragon scene with the center sphere set).

4.2 Runtimes

The traversal results have shown that the median sphere set consistently outperforms all other methods. However, this is only by a small margin. The center sphere set shows improvements within a range of several percent. Because the center sphere set uses only a quarter of the memory that is required by the median sphere set, the results in this section only cover the center sphere set. The implementations of the alternatives have not been thoroughly optimized, and would probably not perform better, as calculating the offsets is mostly memory bound. The center hemispheres were stored as a simple array, retrievable using the triangle index. Each hemisphere occupies 16 bytes: 3 floats for the center, 1 float for the radius.

Figure 5 shows the ambient occlusion ray trace times. For short rays (1% of the scene size) the Dragon scene is traced using only 73% (827ms vs 1132ms) of the original time. As the ray length increases this difference becomes smaller, with 85% (1339ms vs 1576ms) at 50% of the scene size. Rungholt is traced slower by up to 3% at all ray lengths. For all but the shortest rays, the same happens for the Cathedral and Sponza Crytek scenes. In all cases, performance improvements are in accordance with the traversal improvements reported in the previous section: more saved traversal steps means shorter trace times.

scene	radius	baseline	center	vertex	median	polygon
buddha	1%	7.71 / 0.80	46% / 19%	43% / 17%	44% / 15%	45% / 19%
buddha	5%	8.45 / 0.94	61% / 34%	59% / 32%	59% / 31%	60% / 34%
buddha	10%	8.73 / 1.01	63% / 39%	61% / 38%	62% / 37%	63% / 40%
buddha	25%	8.90 / 1.06	64% / 43%	63% / 42%	63% / 41%	64% / 44%
bunny	1%	6.50 / 0.26	47% / 40%	38% / 32%	41% / 33%	45% / 39%
bunny	5%	7.10 / 0.33	63% / 54%	58% / 47%	59% / 48%	62% / 53%
bunny	10%	7.40 / 0.39	66% / 61%	62% / 55%	63% / 56%	66% / 60%
bunny	25%	7.55 / 0.52	71% / 72%	68% / 68%	69% / 68%	70% / 71%
cathedral	1%	5.78 / 0.12	68% / 84%	79% / 93%	57% / 73%	80% / 86%
cathedral	5%	6.16 / 0.18	85% / 90%	89% / 95%	81% / 83%	91% / 91%
cathedral	10%	6.60 / 0.26	89% / 94%	93% / 97%	86% / 89%	92% / 94%
cathedral	25%	7.51 / 0.62	93% / 98%	95% / 99%	90% / 96%	95% / 98%
dragon	1%	6.32 / 0.43	55% / 17%	49% / 15%	50% / 15%	54% / 17%
dragon	5%	7.08 / 0.57	66% / 40%	62% / 39%	62% / 39%	66% / 40%
dragon	10%	7.43 / 0.65	69% / 50%	66% / 49%	66% / 49%	69% / 50%
dragon	25%	7.41 / 0.66	71% / 53%	67% / 52%	68% / 52%	70% / 53%
hairball	1%	9.18 / 1.45	76% / 58%	71% / 64%	66% / 43%	73% / 58%
hairball	5%	12.12 / 2.74	93% / 81%	93% / 83%	91% / 74%	92% / 81%
hairball	10%	14.16 / 3.76	94% / 87%	94% / 89%	93% / 83%	94% / 87%
hairball	25%	16.08 / 4.66	96% / 90%	95% / 92%	94% / 87%	95% / 90%
livingroom	1%	5.17 / 0.10	60% / 70%	64% / 71%	49% / 64%	77% / 77%
livingroom	5%	5.62 / 0.18	84% / 85%	84% / 85%	77% / 82%	88% / 88%
livingroom	10%	5.92 / 0.24	90% / 89%	89% / 89%	86% / 87%	93% / 92%
livingroom	25%	6.39 / 0.35	92% / 93%	92% / 93%	90% / 92%	95% / 95%
rungholt	1%	6.55 / 0.22	83% / 99%	80% / 98%	81% / 98%	82% / 99%
rungholt	5%	7.59 / 0.44	93% / 100%	91% / 99%	92% / 99%	93% / 100%
rungholt	10%	8.07 / 0.54	94% / 100%	92% / 99%	93% / 99%	94% / 100%
rungholt	25%	8.18 / 0.59	95% / 100%	93% / 99%	94% / 99%	95% / 100%
rungholtrotated	1%	7.98 / 0.45	77% / 70%	73% / 65%	74% / 64%	78% / 82%
rungholtrotated	5%	9.15 / 0.74	88% / 84%	86% / 82%	86% / 81%	90% / 91%
rungholtrotated	10%	9.49 / 0.89	90% / 88%	88% / 86%	89% / 86%	92% / 93%
rungholtrotated	25%	9.66 / 0.95	91% / 90%	89% / 88%	90% / 88%	92% / 94%
sponzacrytek	1%	10.33 / 0.18	86% / 80%	90% / 79%	79% / 74%	90% / 82%
sponzacrytek	5%	11.05 / 0.28	95% / 89%	95% / 88%	92% / 86%	96% / 90%
sponzacrytek	10%	11.62 / 0.39	96% / 93%	96% / 92%	94% / 91%	96% / 93%
sponzacrytek	25%	9.16 / 0.60	98% / 98%	98% / 97%	98% / 97%	99% / 98%
sponzadragon	1%	10.96 / 0.26	86% / 71%	87% / 70%	82% / 67%	87% / 73%
sponzadragon	5%	12.50 / 0.43	93% / 85%	93% / 84%	92% / 83%	94% / 86%
sponzadragon	10%	13.66 / 0.60	95% / 91%	95% / 90%	94% / 89%	95% / 91%
sponzadragon	25%	11.60 / 0.74	97% / 95%	97% / 95%	96% / 95%	97% / 96%

Table 2: Ambient occlusion traversal. **radius**: ray length relative to scene size. **baseline**: #visited internal nodes / #visited leaf nodes per ray without offset. **sphere sets**: visited nodes / leaves relative to baseline.

The diffuse trace times are displayed in Figure 6. The results follow the same pattern as for the AO rays: a loss of 3% for Rungholt and gains up to 20% for the Dragon. The results for shadow rays, as shown in Figure 7, again follow the same pattern.

When combining the observations about visited nodes and trace times, we can draw an important conclusion about the computational overhead of offset calculation: this overhead is small. Consider the Rungholt scene, where the reduction of visited leaf nodes is 1% or less for all ray types, and trace times increase by only 3%.

5 CONCLUSION

We presented a simple auxiliary data structure to accelerate ray tracing. We precompute a set of hemispheres located on the surface of the scene's geometry that are guaranteed not to contain any objects. When given an extension ray, we retrieve the hemispheres that start on the same geometric primitive and offset the ray origin. With a large enough offset, the ray will be moved out of the leaf node it started in, thus saving expensive intersection tests at the bottom of the tree during traversal. Additional internal nodes may also be culled if the ray is moved out of their bounding box.

Improvements vary between scenes. In the most optimal case, the number of visited leaf nodes can be reduced by several factors. Additionally, there is a large reduction of the number of visited internal nodes. However, this is only for short occlusion rays in scenes containing large open spaces and convex geometry. On the

scene	baseline	center	vertex	median	polygon
buddha	9.93 / 2.38	68% / 32%	67% / 31%	67% / 29%	68% / 32%
bunny	8.12 / 2.24	74% / 45%	70% / 42%	71% / 40%	73% / 45%
cathedral	10.09 / 2.29	94% / 91%	96% / 97%	91% / 86%	96% / 93%
dragon	9.49 / 2.42	74% / 48%	71% / 42%	71% / 40%	74% / 49%
hairball	19.62 / 8.39	95% / 83%	95% / 87%	94% / 78%	95% / 84%
livingroom	9.21 / 2.18	92% / 87%	92% / 88%	90% / 82%	95% / 92%
rungholt	9.35 / 1.02	95% / 99%	93% / 97%	94% / 97%	95% / 99%
rungholtrotated	11.99 / 2.16	89% / 70%	87% / 66%	88% / 65%	92% / 85%
sponzacrytek	16.20 / 4.20	96% / 93%	96% / 94%	94% / 91%	97% / 94%
sponzadragon	18.77 / 4.87	95% / 91%	95% / 91%	94% / 89%	95% / 92%

Table 3: Diffuse traversal. **baseline**: #visited internal nodes / #visited leaf nodes per ray without offset. **sphere sets**: visited nodes / leaves relative to baseline.

scene	baseline	center	vertex	median	polygon
buddha	10.18 / 0.40	72% / 47%	70% / 44%	71% / 44%	72% / 47%
bunny	10.42 / 0.15	81% / 64%	78% / 57%	79% / 58%	80% / 64%
cathedral	7.27 / 0.05	93% / 89%	94% / 94%	90% / 80%	96% / 90%
dragon	8.15 / 0.18	71% / 34%	69% / 32%	69% / 32%	70% / 34%
hairball	16.66 / 1.43	97% / 90%	97% / 91%	96% / 85%	96% / 89%
livingroom	6.38 / 0.05	91% / 77%	90% / 76%	88% / 70%	93% / 83%
rungholt	6.96 / 0.08	93% / 99%	91% / 97%	92% / 98%	92% / 99%
rungholtrotated	8.78 / 0.20	88% / 79%	86% / 72%	87% / 73%	87% / 82%
sponzacrytek	15.61 / 0.09	97% / 86%	96% / 83%	95% / 81%	97% / 87%
sponzadragon	17.71 / 0.14	96% / 78%	95% / 76%	95% / 74%	96% / 79%

Table 4: Shadow traversal. **baseline**: #visited internal nodes / #visited leaf nodes per ray without offset. **sphere sets**: visited nodes / leaves relative to baseline.

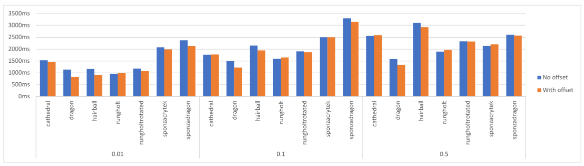


Figure 5: Ambient occlusion ray trace times.

other end of the spectrum there are scenes for which there is next to no reduction of visited nodes. Changes in actual trace time range from speedups of up to 25% to a loss of 3%. This last number indicates that the overhead of offset calculation is low.

The memory overhead added by the hemisphere set is linear in the scene size, requiring a fixed number of bytes per triangle. The optimized hemisphere set used in this paper uses 16 bytes per triangle. The hemisphere set is also completely independent of the acceleration structure, making it compatible with any form of traversal of the BVH or even alternative acceleration structures.

6 FUTURE WORK

Only the center sphere set implementation was thoroughly optimized. While the alternatives offer little extra reduction of visited nodes at the cost of several times more memory overhead, it might be worthwhile to investigate their performance when properly optimized.

The polygon sphere set could perform better than the center sphere set with a better heuristic for selecting the optimal hemisphere. Taking into account additional properties, such as the area of overlap between a hemisphere and the triangles, might result in a higher quality hemisphere set.

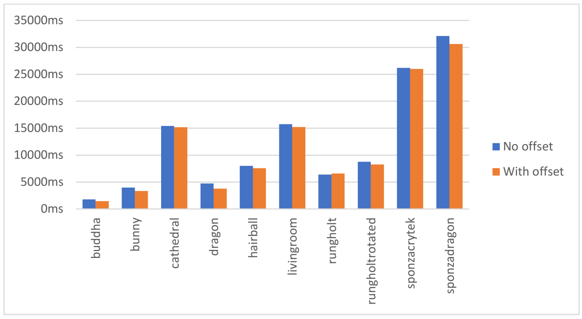


Figure 6: Diffuse ray trace times.

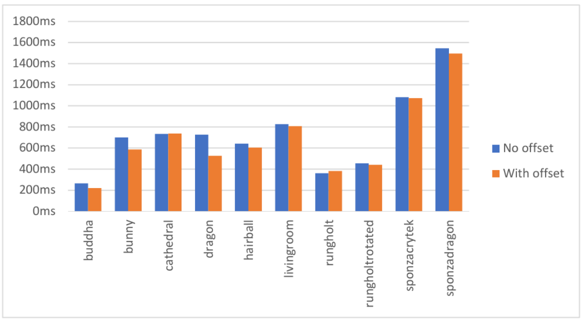


Figure 7: Shadow ray trace times.

Concave geometry has been shown to result in little reduction of visited nodes. This may be improved by using other methods to offset rays, such as other geometric primitives that better represent empty spaces in these areas.

A deciding factor for the viability of hemisphere sets is how they perform in a GPU renderer. Offset calculation will have a different overhead and the traversal algorithms used on the GPU tend to be different.

Efficient construction of the hemisphere sets was not a subject of this work. This will become especially relevant for dynamic scenes, where quick reconstruction of (parts of) the hemisphere set will be needed.

7 ACKNOWLEDGEMENTS

Test scenes are taken from Morgan McGuire's Computer Graphics Archive [McG17]. We thank the anonymous reviewers for their valuable comments.

8 REFERENCES

[Ben75] BENTLEY J. L.: Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 9 (1975), 509–517.

[BH10] BOULOS S., HAINES E.: Sorted bvhs. *Ray Tracing News 23*, 2 (2010), 6.

[Cla76] CLARK J. H.: Hierarchical geometric models for visible surface algorithms. *Communications of the ACM 19*, 10 (1976), 547–554.

[CW88] CLEARY J. G., WYVILL G.: Analysis of an algorithm for fast ray tracing using uniform space subdivision. *The Visual Computer 4*, 2 (1988), 65–83.

[FCM09] FOWLER C., COLLINS S., MANZKE M.: Accelerated entry point search algorithm for real-time ray-tracing. In *Proceedings of the 25th Spring Conference on Computer Graphics* (2009), ACM, pp. 59–66.

[FTI86] FUJIMOTO A., TANAKA T., IWATA K.: Arts: Accelerated ray-tracing system. *IEEE Computer Graphics and Applications 6*, 4 (1986), 16–26.

[Gla84] GLASSNER A. S.: Space subdivision for fast ray tracing. *IEEE Computer Graphics and applications 4*, 10 (1984), 15–24.

[GS87] GOLDSMITH J., SALMON J.: Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications 7*, 5 (1987), 14–20.

[Hav00] HAVRAN V.: *Heuristic ray shooting algorithms*. PhD thesis, Ph. d. thesis, Department of Computer Science and Engineering, Faculty of â, 2000.

[HB07] HAVRAN V., BITTNER J.: Ray tracing with sparse boxes. In *Proceedings of the 23rd Spring Conference on Computer Graphics* (2007), ACM, pp. 49–54.

[HBZ98] HAVRAN V., BITTNER J., ZÁRA J.: Ray tracing with rope trees. In *14th Spring Conference on Computer Graphics* (1998), pp. 130–140.

[HPMB19] HENDRICH J., POSPÍŠIL A., MEISTER D., BITTNER J.: Ray classification for accelerated bvh traversal. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 49–56.

[IH11] IZE T., HANSEN C.: Rtsah traversal order for occlusion rays. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 297–305.

[MB90] MACDONALD J. D., BOOTH K. S.: Heuristics for ray tracing using space subdivision. *The Visual Computer 6*, 3 (1990), 153–166.

[McG17] MCGUIRE M.: Computer graphics archive, July 2017. `https://casual-effects.com/data`. URL: `https://casual-effects.com/data`.

[ORM08] OVERBECK R., RAMAMOORTHI R., MARK W. R.: Large ray packets for real-time whitted ray tracing. In *2008 IEEE Symposium on Interactive Ray Tracing* (2008), IEEE, pp. 41–48.

[RSH05] RESHETOV A., SOUPIKOV A., HURLEY J.: Multi-level ray tracing algorithm. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 1176–1185.

[SFD09] STICH M., FRIEDRICH H., DIETRICH A.: Spatial splits in bounding volume hierarchies. In *Proceedings of the Conference on High Performance Graphics 2009* (2009), ACM, pp. 7–13.

[Wal07] WALD I.: On fast construction of sah-based bounding volume hierarchies. In *2007 IEEE Symposium on Interactive Ray Tracing* (2007), IEEE, pp. 33–40.

[WSBW01] WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive rendering with coherent ray tracing. In *Computer graphics forum* (2001), vol. 20, Wiley Online Library, pp. 153–165.

[WWB*14] WALD I., WOOP S., BENTHIN C., JOHNSON G. S., ERNST M.: Embree: a kernel framework for efficient cpu ray tracing. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 143.

Clustered Grid Cell Data Structure for Isosurface Rendering

Fredrik Nysjö
Centre for Image Analysis, Dept. of Information Technology,
Uppsala University, Sweden
{fredrik.nysjo}@it.uu.se

ABSTRACT

Active grid cells in scalar volume data are typically identified by many isosurface rendering methods when extracting another representation of the data for rendering. However, the use of grid cells themselves as rendering primitives is not extensively explored in the literature. In this paper, we propose a cluster-based data structure for storing the data of active grid cells for fast cell rasterisation via billboard splatting. Compared to previous cell rasterisation approaches, eight corner scalar values are stored with each active grid cell, so that the full volume data is not required during rendering. The grid cells can be quickly extracted and use about 37 percent memory compared to a typical efficient mesh-based representation, while supporting large grid sizes. We present further improvements such as a visibility buffer for cluster culling and EWA-based interpolation of attributes such as normals. We also show that our data structure can be used for hybrid ray tracing or path tracing to compute global illumination.

Keywords
Point-based rendering, Visibility, Ray tracing

1 INTRODUCTION

Isosurface rendering requires finding the intersections of the isosurface for a particular isovalue in the data. The scalar volume data, for example, a computed tomography (CT) image, is typically sampled with tri-linear interpolation when finding intersections and computing smooth normals for shading. If dynamic isovalue is not required, indirect volume rendering techniques generally provide fast rendering and good data reduction. Mesh-based methods such as Marching Cubes (MC) [15] and dual contouring [11] find the active grid cells (the grid cells containing isosurface intersections) of the data and extract a polygonal representation of the isosurface. Point-based methods such as elliptical weighted-average (EWA)-based surface splatting [22, 1] typically extract a point for each active grid cell and render these points as small disks. Sparse storage of larger voxel bricks and hierarchical data structures such as octrees can also be used to provide better data reduction and faster traversal to direct volume rendering techniques such as isosurface raycasting.

Compared to the previously mentioned techniques, the use of grid cells themselves as rendering primitives is not extensively explored in the literature. Parker et al. [20] describe a method for analytically computing ray-surface intersections for active grid cells during isosurface raycasting. Zhang et al. [21] propose storing active edges instead of active grid cells, which are rendered by disk splatting or expanded and rendered as small point clouds. Cell rasterisation (CR) was proposed by Liu et al. [13] as a fast method for rasterising the active grid cells of a volume. Active grid cells are rasterised as point primitives via so-called billboard splatting, and the original volume data is sampled in the cells in the fragment shader to find intersections and compute smooth normals. Compared to indirect volume rendering techniques such as mesh- and point-based methods, cell rasterisation provide the same isosurface as isosurface raycasting. Liu et al. also use a data structure for fast sorting and traversal of the cells in front-to-back order for proper transparency rendering. However, their method provide no data reduction, since only grid positions are stored with the cells, and therefore the original volume data needs to be available in GPU memory for sampling during rendering.

In this paper, we propose a memory efficient data structure for storing the active grid cells of a volume. The data structure is easy to implement, does not require the original volume data to be available during rendering, and supports ray tracing in addition to rasterisation. Our main contributions in the paper are:

- A clustered grid cell data structure (CGC) for storing grid positions and corner scalar values of active grid cells. The data structure supports large grid sizes

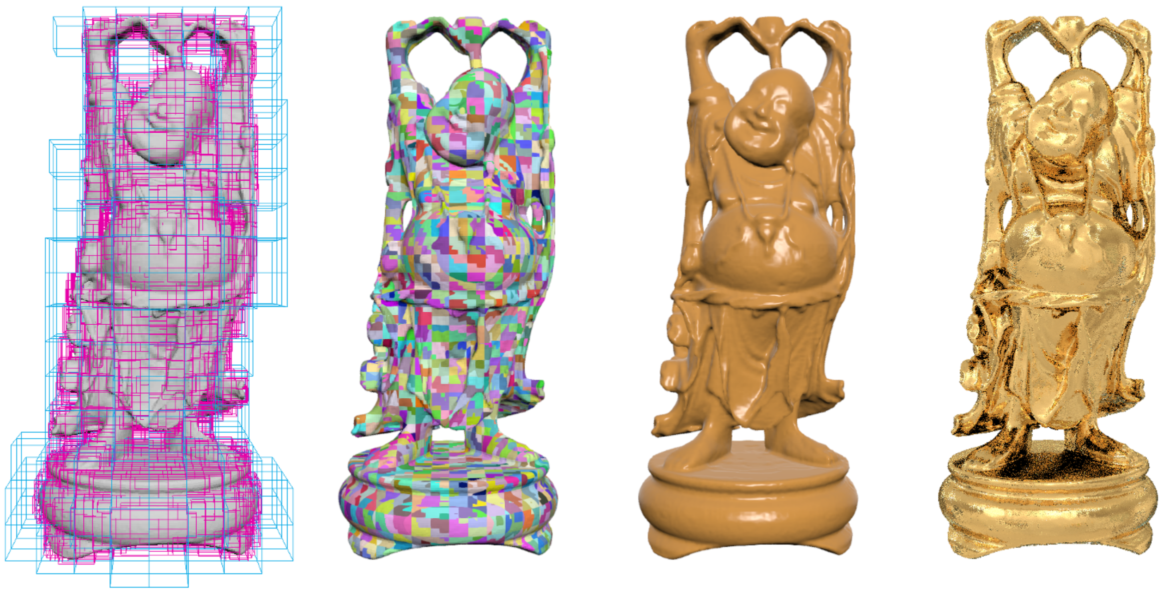


Figure 1: Our proposed data structure for isosurface rendering: Left: 16^3 block volume with each block (blue wireframe) storing a pointer to a list of clusters (pink wireframes) of active grid cells extracted from a voxelised model; Middle-left: isosurface rasterised with cell rasterisation using scalar values stored with active grid cells and visibility culling of individual clusters (indicated by colors); Middle-right: final shaded isosurface after EWA-based normal interpolation and deferred normalized Blinn-Phong shading; Right: hybrid ray tracing using our data structure with cell rasterisation for primary rays and path tracing with three bounces and an environment map for secondary rays (after one frame, using 1 sample per pixel). Each cluster shown in the first two images consists of up to 120 active grid cells and an axis-aligned bounding box. A lower resolution (256^3 voxels) voxelisation of the Buddha model was used to clearer show the clusters and the effect of the attribute interpolation.

($128K \times 128K \times 128K$) and can be used for fast cell rasterisation via billboard splatting.

- Improvements to the cell rasterisation including visibility culling of clusters and EWA-based interpolation for smooth attribute interpolation of normals and other attributes.
- Showing that the proposed data structure can be used for hybrid ray tracing or path tracing to compute global illumination.

2 RELATED WORK

For direct isosurface raycasting, larger bricks or tiles are generally used instead of individual grid cells for data reduction and empty space skipping, as in Hadwiger et al. [7]. The voxel database (VDB) data structure by Museth [19] is commonly used in offline rendering and uses a shallow tree of internal nodes and leaf nodes to efficiently store volume data in tiles. Recently, Hoetzlein [9] also introduced GVDB for real-time rendering on the GPU. Our data structure shares some similarities with the VDB data structure, but does not, for example, provide random access to voxels, and is restricted to isosurface rendering by using smaller tiles grouped into clusters.

Sparse voxel octrees (SVOs) have been proposed for storage and ray tracing of large voxel models [12]. The SVO typically stores surface voxels in the leaf nodes of the octree, and is traversed during ray tracing. Jablonski et al. [10] propose an alternative method for rendering SVOs via billboard splatting and a screen-space voxel buffer. Large number of individual voxels can also be efficiently rasterised as cubes via billboard splatting and fast ray-box intersection tests [16]. However, for high quality isosurface rendering, a drawback of traditional SVOs representing voxels as small cubes is the limited contour information they provide, leading to a blocky appearance unless each voxel is projected to less than a few pixels. Efficient sparse voxel octrees (ESVOs) [12] improves this by storing additional contour information in the octree and voxels. Heitz et al. [8] use filtered signed distance fields and differential cone tracing to render SVOs with accurate anti-aliased contours. Marcus [17] proposes a SVO data structure storing scalar values (signed distances in their case) at the corners of each voxel, which is similar to the corner values we store for cell rasterisation, but used during ray tracing of the SVO.

Visibility culling can be important in rasterisation to improve performance and reduce overdraw. Livnat et al. [14] perform point-based view-dependent isosurface extraction from an octree, using a visibility framebuffer to prune non-visible regions. They further compute surface normals in a post-processing step for far

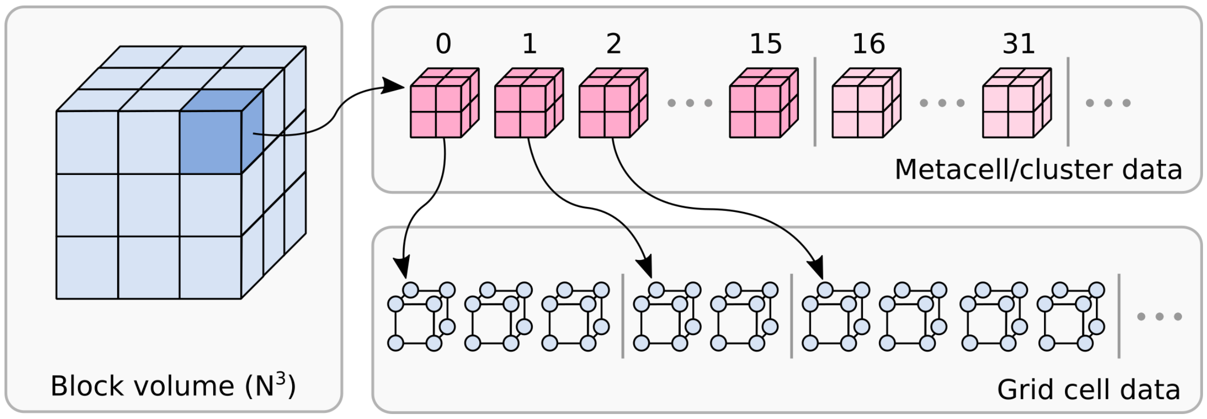


Figure 2: Overview of the clustered grid cell (CGC) data structure for storing active grid cells. Each block in the block volume (blue) stores a pointer to metacells (red) extracted in the block. The metacells are grouped into clusters of 16 metacells, with each metacell storing a pointer to the corner values of up to eight grid cells. The last metacell in each cluster stores a cluster bound instead of a pointer. Blocks, metacells, and grid cell corners are stored in separate buffers on the GPU. The memory layout for blocks and metacells is further described in Table 1.

points (minified surfaces), while using MC triangle normals for near points (magnified surfaces). For general occlusion culling of clustered geometry, a hierarchical Z-buffer [5] is often used in combination with pre-computed cluster bounds for conservative culling of clusters. For back-face culling, a normal cone can similarly be pre-computed and stored with each cluster. In deferred splatting [6] and the method for cluster culling we use in this paper based on a visibility buffer, the visibility culling is not conservative but does not require generating a hierarchical Z-buffer or storing cluster bounds or normal cones.

Rasterisation of small primitives (small triangles and points) can further lead to bad GPU utilisation when each primitive only occupies a few pixels. Evans [4] presents a point-based method replacing the standard rasterisation pipeline with stochastic splatting in a compute shader. Splatting is performed using 64-bit atomic instructions, thereby avoiding executing GPU threads for non-covered fragments in 2×2 pixel quads or fragments with zero alpha value. A drawback of this approach is that 64-bit atomic instructions are only available and exposed to the programmer on game consoles (PlayStation 4) and on certain NVIDIA GPUs (Maxwell and later generations).

3 METHODS

In Section 3.1, we describe the proposed CGC data structure for storing clusters of active grid cells, and the isosurface extraction. In Sections 3.2–3.4, we describe cell rasterisation using billboard splatting, and present further improvements such as a visibility buffer for back-face and occlusion culling and EWA-based attribute interpolation of attributes such as normals. In the final Sections 3.5 and 3.6, we discuss memory usage and hybrid ray tracing.

Block data	**Size**	**Metacell data**	**Size**
Metacell count	32 bit	Grid position (XYZ)	48 bit
Metacell pointer	32 bit	Cell mask	16 bit
Min value	32 bit	Cell pointer	32 bit
Max value	32 bit		
Total:	128 bit	**Total:**	96 bit

Table 1: Memory layout for blocks and metacells. Blocks and metacells are stored in separate GPU buffer textures, using 16 bytes per block and 12 bytes per metacell. Corner scalar values for active grid cells are stored in a separate buffer texture.

3.1 Data Structure

An overview of our proposed data structure is shown in Figures 1–2. We store extracted grid cells in a linear buffer on the GPU. Eight corner values per cell are stored as two RGBA values, with the component type matching the scalar precision of the volume data. To avoid storing a grid position with each cell, we group up to eight neighboring cells into a $2 \times 2 \times 2$ metacell storing a pointer, a cell mask, and a cell pointer. The metacells are stored in a second linear buffer, with each segment of 16 consecutive metacells representing a cluster.

For the isosurface extraction, we divide the volume data into $16 \times 16 \times 16$ number of blocks, and extract the active grid cells of each block in Morton order. To prevent metacells of the same cluster from spilling over into different volume blocks, we pad the last cluster of each block with empty metacells such that its size become 16. We also use the last metacell of each cluster to store a cluster bound instead of cell metadata. We perform the isosurface extraction on the CPU, and upload extracted metacells and grid cells to separate GPU buffer textures.

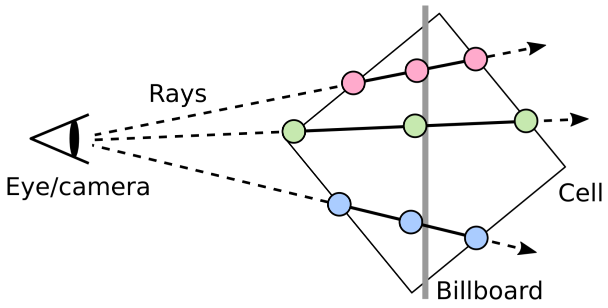


Figure 3: Cell rasterisation using billboard splatting: for each fragment rasterised for the billboard (gray line intersecting the cell), a ray-box intersection test is performed with the cell, and interpolated samples (indicated in colors) computed along the ray to determine the isosurface intersection. In our implementation, three samples per ray is used, which was found sufficient.

The memory layout for blocks and metacells is shown in Table 1. In our current implementation, grid positions are stored with 16 bit per component, allowing a maximum grid size of $128K \times 128K \times 128K$ voxels to be represented.

3.2 Cell Rasterisation

The basic idea of cell rasterisation using billboard splatting is illustrated in Figure 3. In the original cell rasterisation paper [13], a vertex shader is used to render active cells as a point sprites (`GL_POINTS` primitives). For each rasterised fragment of a point sprite billboard, a ray-box intersection test is performed with the cell, and the original volume data is sampled along the ray with tri-linear interpolation to find isosurface intersections. When an intersection is found, a normal gradient is also computed from the volume data for shading.

For efficient rendering of the clusters of grid cells in our data structure, we replace the vertex shader of previous approaches with a tesselation shader taking a single patch (`GL_PATCHES` primitive) per cluster as input and expands it into points for the cells in the cluster. Clusters that were culled in the last frame (Section 3.3) are rendered at $1/8$ rate (with at most on point per metacell per frame) until they become visible. Stored corner values are interpolated with manual tri-linear interpolation in the fragment shader. For the ray-box intersection test, we use the efficient slab test mentioned in Majercik et al. [16], which also can be used to rasterise cells as cubic voxels. To allow smooth interpolation of normals and other attributes for shading, we rasterise cells to a geometry buffer (G-buffer) and compute deferred normalized Blinn-Phong shading in a separate pass after attribute interpolation (further described in Section 3.4). A cell normal is computed in the tesselation shader from the corner values, and further used for back-face culling.

3.3 Visibility Buffer

To further improve the rasterisation performance, we introduce a visibility buffer to avoid performing full cell rasterisation of clusters that are fully occluded or back-facing. The visibility buffer stores a bit per cluster indicating the visibility after each frame. An RG32UI texture is used during the cell rasterisation pass to capture cluster IDs visible cells. After the cell rasterisation, we clear the current visibility buffer bits and update the visibility buffer from the ID texture in a separate compute shader pass. Similar to the disk-based deferred splatting approach by Guennebaud et al. [6], we render cells of non-visible clusters as single pixel splats for updating the visibility buffer. During fast camera movements, non-visible clusters becoming visible might result in flickering artifacts. We therefore introduce a second visibility buffer to record of new clusters becoming visible, and perform full cell rasterisation for those clusters in a second pass. During fast camera movements, there might still be some flickering artifacts, which can be seen in the accompanying video.

3.4 EWA-based Attribute Interpolation

Normals and other attributes should be smoothly interpolated before computing shading. However, our data structure does not allow random access to neighboring grid cells. For smooth interpolation of normals and other attributes, we introduce a second EWA-based attribute splatting pass in which cells are rendered as round splats with a weighted footprint, and attributes accumulated to the G-buffer. The results of the EWA-based interpolation is shown in Figure 4.

Laine et al. [12] propose performing interpolation as a post-processing step (after ESVO ray tracing) by computing a weighted average of neighboring pixels in a Poisson disk centered around each target pixel. The interpolation can use a large number of samples (up to 96) per pixel, which may be expensive on GPUs with low memory bandwidth. Specular surfaces are also not handled correctly because the interpolation is performed after shading. Jablonski et al. [10] propose a smaller 3×3 Gaussian filter for interpolating G-buffer normals before shading. We investigated replacing the post-processing in [12] with a stochastic interpolation scheme using temporal anti-aliasing (TAA) and fewer samples per frame. However, our EWA-based interpolation provided smoother normals and was possible to use without TAA. Another option would be storing additional corner values (for example, 32 instead of 8) with each grid cell, at the cost of significantly increased memory usage.

3.5 Memory Usage

Average memory usage per grid cell in our data structures, when including storage for metacells and the

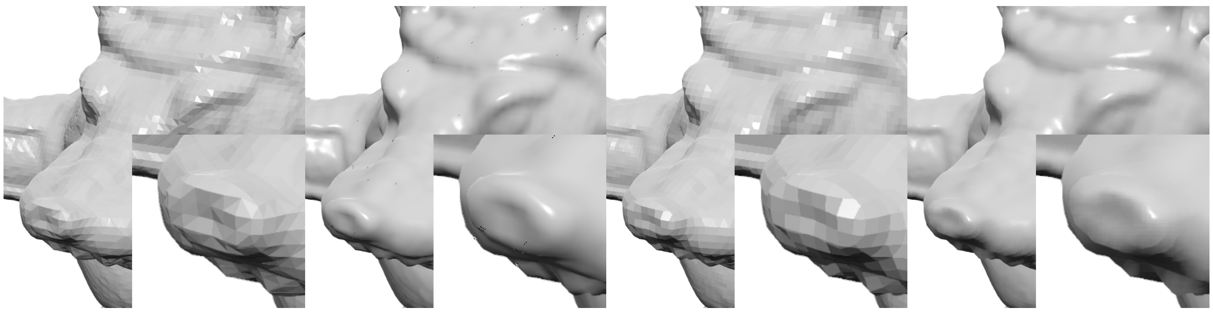


Figure 4: Isosurface comparison for voxelised Armadillo model: (a) MC mesh (without pre-computed normals); (b) isosurface from cell rasterisation with sampling in the original volume data; (c)–(d) isosurface from cell rasterisation of our CGC data structure, without and with EWA-based normal interpolation. For our method, the normal gradient of each cell was computed in the vertex shader from the stored corner values. A lower resolution (256^3 voxels) voxelisation of the Armadillo model was used to better show the effect of the normal interpolation.

block volume, is 11 bytes for 8-bit scalar data and 19 bytes for 16-bit scalar data. If 4-bit scalars provide sufficient precision for the application, for example, when scalars represent coverage values or truncated signed distances such as in some sculpting applications, the average memory usage may be further reduced to 7 bytes per cell. In addition to storage for the main data structure, we allocate 0.13 MB visibility buffers with capacity for 1M unique clusters, which was found sufficient even for the hairball dataset.

3.6 Hybrid Ray Tracing and Path Tracing

The data structure presented in this paper can also be used directly for ray tracing or path tracing of the extracted isosurface. Using the pre-computed cluster bound stored in the last metacell of each cluster, intersections with rays can be efficiently computed hierarchicaly by traversing and testing blocks, clusters, metacells, and individual voxels with the same efficient slab test used in the cell rasterisation. Shadow and ambient occlusion rays only need to fetch the metacell data, whereas additional bounces for path tracing also need to fetch grid cell data in order to compute a normal at each intersection point. Primary rays can be either traced or rasterised (hybrid ray tracing). Results from a path tracing implementation using hybrid ray tracing are shown in Figures 1 and 6.

4 EXPERIMENTS AND RESULTS

We implemented our method in OpenGL 4.5 and C++. All experiments were performed on an AMD Ryzen 7 2700 with 32GB RAM and an NVIDIA GeForce GTX 1070, and on a separate laptop with an Intel Core i7-6700HQ with 16GB RAM and an NVIDIA GeForce GTX 965M to evaluate performance on a lower-end GPU. Frame times were measured via OpenGL timer queries (`GL_ARB_timer_query`) for the scenes in

Volume	Resolution ($W \times H \times D$)	Type	Size (GB)
Dragon	$512 \times 512 \times 512$	8 bit	0.13
Plastic Skull	$512 \times 512 \times 512$	8 bit	0.13
Stag beetle CT	$832 \times 832 \times 494$	16 bit	0.68
Chameleon CT	$1024 \times 1024 \times 1080$	8 bit	1.13
Raptor	$2048 \times 1024 \times 512$	8 bit	1.07
Buddha	$1024 \times 2048 \times 1024$	8 bit	2.15
Hairball	$1024 \times 1024 \times 1024$	8 bit	1.07

Table 2: Volume datasets used for evaluation.

Figure 5 rendered at 1920×1080 resolution. To generate the non-CT volume datasets in Table 2 for the test scenes, a CPU-based implementation of the slicemap-based binary voxelisation method of Eisemann et al. [3] was used. Each input mesh was voxelised to three times a target resolution, followed by downsampling to generate a coverage representation.

A comparison of memory usage and extraction times for our data structure (CGC) and a typical efficient mesh-based representation extracted with MC is presented in Table 3. The size of the original volume data is also presented in Table 2. For the MC implementation, we used the code from [2] after modifying it to generate indexed meshes without duplicate vertices, and store vertex positions in 16-bit signed normalized format with a scaling factor. While the MC memory usage could be further reduced using compression or converting indexed meshes into triangle strips, such optimization would require further mesh processing and increased extraction time. A performance comparison of MC, original cell rasterisation (CR) [13] using sampling in the original volume data, and our cell rasterisation using the CGC data structure is presented in Table 4. A corresponding isosurface comparison is also shown in Figure 4.

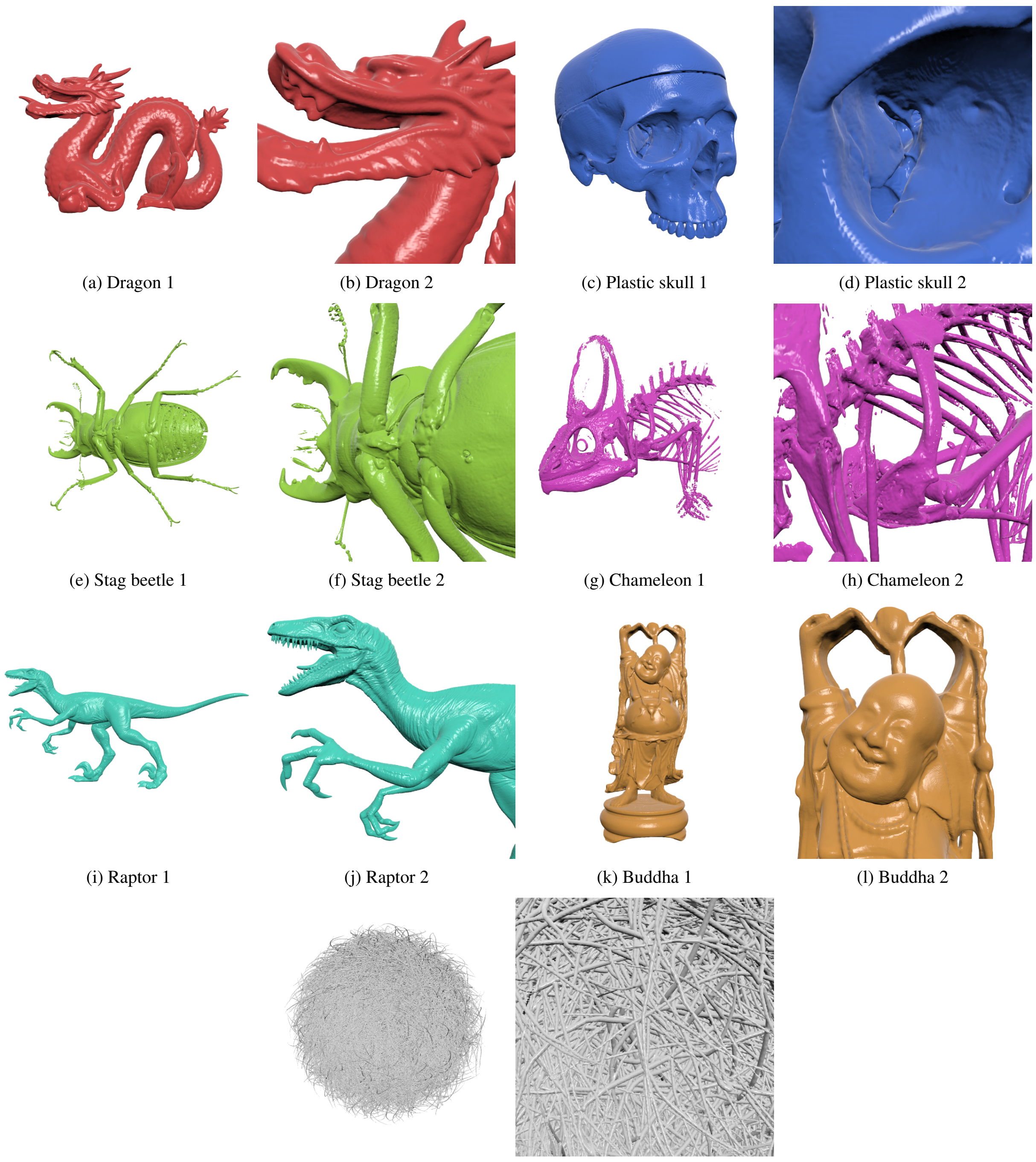

(a) Dragon 1 (b) Dragon 2 (c) Plastic skull 1 (d) Plastic skull 2
(e) Stag beetle 1 (f) Stag beetle 2 (g) Chameleon 1 (h) Chameleon 2
(i) Raptor 1 (j) Raptor 2 (k) Buddha 1 (l) Buddha 2
(m) Hairball 1 (n) Hairball 2

Figure 5: Test scenes for the datasets in Table 2.

5 DISCUSSION

In this paper, the clustered grid cell (CGC) data structure for isosurface rendering was proposed. We demonstrated that this data structure can be used for fast cell rasterisation via billboard splatting, as well as for hybrid ray tracing of isosurfaces extracted from large volumes. The data structure does not require the original volume data to be available during rendering, and uses about 37% memory compared to a typical mesh-based representation.

The cell rasterisation performance of our method was comparable to mesh rendering, except for the larger test scenes with high depth complexity, in which our method was faster and benefited from the visibility culling. Good use cases for our method would include digital sculpting and visualization of data that need stor-

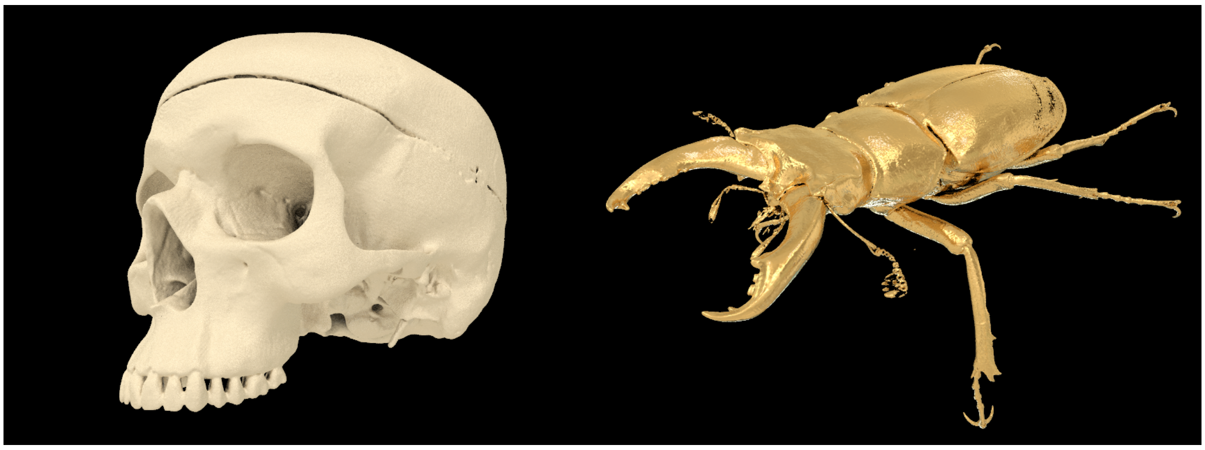


Figure 6: Hybrid ray tracing: Plastic skull dataset (512^3 voxels) and Stag beetle CT dataset ($834 \times 834 \times 494$ voxels) rendered with our data structure using cell rasterisation for primary rays and path tracing with three bounces and an HDR environment map for secondary rays (using 1 sample per pixel). Both scenes are rendered at interactive rate (less than 16 milliseconds per frame) at 1920×1080 resolution, with images showing result after 100 frames.

	CGC (ours)				MC [15]			
Volume	**Active cells** (count)	**Metacells** (count)	**Memory** (MB)	**Extraction** (time, s)	**Vertices** (count)	**Triangles** (count)	**Memory** (MB)	**Extraction** (time, s)
Dragon	668,804	182,400	7.5	0.8	668,860	1,337,714	20.1	1.5
Plastic skull	2,107,816	569,888	23.7	0.9	2,108,153	4,216,444	63.2	1.7
Stag beetle CT	3,140,758	769,536	59.5	2.6*	3,045,308	6,099,032	91.5	4.1*
Chameleon CT	3,492,569	911,088	38.9	11.4	3,430,406	6,861,504	102.9	12.4
Raptor	2,864,152	765,136	32.1	10.7	2,864,515	5,729,028	85.9	11.1
Buddha	8,464,696	2,264,208	94.9	17.6	8,464,912	16,929,714	253.9	24.3
Hairball	36,034,268	8,952,736	395.7	8.3	36,585,310	73,442,740	1,100.8	16.8

Table 3: Memory usage and isosurface extraction times for the datasets in Table 2, for our CGC data structure using the cell format in Table 1, and with corresponding MC indexed triangle meshes extracted and shown for comparison. For MC meshes, memory usage is without vertex normals and with indices stored using 4 bytes per index and vertex positions stored in 16-bit signed normalized format (with scaling factor computed from mesh bounds) using 6 bytes per vertex. The same normalized isovalue 0.5 was used for all datasets.

ing additional attributes per voxel or cell. A limitation of our method is the need to use deferred rendering for the EWA-splatting of normals and other attributes. However, as we demonstrate, the data structure can also be used for cell rasterisation with sampling in the original volume data.

It could be interesting to explore approaches using compute-based rasterisation for far grid cells and traditional rasterisation for near grid cells. We also aim to investigate if clusters could be more efficiently rendered with mesh shaders that were recently introduced with the NVIDIA Turing GPU architecture. Other future work could be improving ray tracing performance by re-arranging the clusters of min-max blocks into bounding volume hierarchies.

The source code for our implementation is available at `https://bitbucket.org/FredrikNysjo/grid_cells`.

ACKNOWLEDGEMENTS

The author would like to thank Filip Malmberg and Ingela Nyström for valuable input on the manuscript and the method. Further thanks to Johan Nysjö and Ingrid Carlbom for comments and discussion, and to the anonymous reviewers for their feedback. The STL mesh for the plastic skull dataset was obtained from a CT scan provided by Uppsala University hospital. The Dragon, Buddha, and Hairball models were downloaded from Morgan McGuire's Computer Graphics Archive [18], whereas the Raptor model was downloaded from the AIM@Shape Shape Repository (`http://visionair.ge.imati.cnr.it/ontologies/shapes`). Remaining datasets were obtained from `https://www.cg.tuwien.ac.at/research/publications/2005/dataset-stagbeetle` and the Digital Morphology library (`http://digimorph.org`).

	MC [15]		CR [13]		CGC (ours)	
Scene	**GTX 1070** (time, ms)	**GTX 965M** (time, ms)	**GTX 1070** (time, ms)	**GTX 965M** (time, ms)	**GTX 1070** (time, ms)	**GTX 965M** (time, ms)
Dragon 1	0.3	0.8	0.4	1.2	0.6 (0.4)	1.9 (1.1)
Dragon 2	0.3	0.9	0.6	1.7	0.7 (0.5)	2.2 (1.3)
Plastic skull 1	0.9	2.5	0.7	1.9	1.0 (0.6)	2.8 (1.7)
Plastic skull 2	1.0	3.0	1.2	3.3	1.5 (0.9)	4.0 (2.5)
Stag beetle 1	1.4	3.9	0.9	2.5	1.2 (0.9)	3.5 (2.5)
Stag beetle 2	1.8	6.5	1.5	4.4	1.6 (1.1)	4.9 (3.3)
Chameleon 1	1.5	4.3	1.4	3.5	2.0 (1.4)	5.2 (3.6)
Chameleon 2	1.5	3.6	1.3	3.6	1.6 (1.0)	4.9 (3.0)
Raptor 1	1.2	3.6	1.4	4.1	2.1 (1.5)	6.3 (4.4)
Raptor 2	1.2	3.0	1.0	2.8	1.5 (0.9)	4.5 (2.7)
Buddha 1	3.9	11.3	4.5	12.1	7.0 (4.9)	17.9 (12.4)
Buddha 2	4.0	9.2	1.7	4.5	2.5 (1.7)	7.0 (4.6)
Hairball 1	25.9	61.5	6.3	18.4	8.5 (6.2)	24.3 (18.4)
Hairball 2	26.5	66.4	6.2	18.0	6.7 (5.1)	19.3 (15.4)

Table 4: Performance comparison when rendering the scenes in Figure 5 at 1920×1080 pixels render target resolution on two different GPUs (GTX 1070 and GTX 965M), using Marching Cubes (MC), cell rasterisation with sampling in the original volume data (CR [13]), and cell rasterisation of our data structure (CGC). For the CR implementation, the visibility buffer and cluster sorting of our method is used instead of the original data structure. The times show GPU timings in milliseconds of the rasterisation time (depth and G-buffer pass). Rasterisation time for our method without EWA-splatting is also presented in the parentheses.

6 REFERENCES

[1] BOTSCH, M., HORNUNG, A., ZWICKER, M., AND KOBBELT, L. High-Quality Surface Splatting on Today's GPUs. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics* (2005), SPBG'05, Eurographics Association, pp. 17–24.

[2] BOURKE, P. Polygonising a scalar field. `http://paulbourke.net/geometry/polygonise`. Accessed on January 1, 2020.

[3] EISEMANN, E., AND DÉCORET, X. Single-Pass GPU Solid Voxelization for Real-Time Applications. In *Proceedings of Graphics Interface 2008* (2008), pp. 73–80.

[4] EVANS, A. Learning from Failure: a Survey of Promising, Unconventional and Mostly Abandoned Renderers for 'Dreams PS4', a Geometrically Dense, Painterly UGC Game. Presented at the 'Advances in Real-Time Rendering Course' at ACM SIGGRAPH '15 (2015).

[5] GREENE, N., KASS, M., AND MILLER, G. Hierarchical z-buffer visibility. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), ACM, pp. 231–238.

[6] GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. Deferred Splatting. *Computer Graphics Forum 23*, 3 (2004), 653–660.

[7] HADWIGER, M., SIGG, C., SCHARSACH, H., BÜHLER, K., AND GROSS, M. Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces. In *Computer Graphics Forum* (2005), vol. 24, pp. 303–312.

[8] HEITZ, E., AND NEYRET, F. Representing Appearance and Pre-Filtering Subpixel Data in Sparse Voxel Octrees. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics* (2012), pp. 125–134.

[9] HOETZLEIN, R. K. GVDB: Raytracing Sparse Voxel Database Structures on the GPU. In *Proceedings of High Performance Graphics* (2016), pp. 109–117.

[10] JABLONSKI, S., AND MARTYN, T. Real-Time Voxel Rendering Algorithm based on Screen Space Billboard Voxel Buffer with Sparse Lookup Textures. In *Proceedings of the 24th Int. Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2016), WSCG'2016, pp. 27–36.

[11] JU, T., LOSASSO, F., SCHAEFER, S., AND WARREN, J. Dual Contouring of Hermite Data. In *ACM transactions on graphics (TOG)* (2002), vol. 21, pp. 339–346.

[12] LAINE, S., AND KARRAS, T. Efficient Sparse Voxel Octrees - Analysis, Extensions, and Implementation. Tech. rep., NVIDIA Research, 2010.

[13] LIU, B., CLAPWORTHY, G. J., AND DONG, F. Fast Isosurface Rendering on a GPU by Cell

Rasterization. *Computer Graphics Forum 28*, 8 (2009), 2151–2164.

[14] LIVNAT, Y., AND TRICOCHE, X. Interactive Point-Based Isosurface Extraction. In *Proceedings of Visualization '04* (2004), pp. 457–464.

[15] LORENSEN, W. E., AND CLINE, H. E. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics 21*, 4 (1987), 163–169.

[16] MAJERCIK, A., CRASSIN, C., SHIRLEY, P., AND MCGUIRE, M. A Ray-Box Intersection Algorithm and Efficient Dynamic Voxel Rendering. *Journal of Computer Graphics Techniques (JCGT) 7*, 3 (2018), 66–81.

[17] MARCUS, R. Level-of-Detail Independent Voxel-Based Surface Approximations. Tech. rep., Utrecht University, 2017.

[18] MCGUIRE, M. Computer graphics archive, July 2017. Accessed on January 1, 2020.

[19] MUSETH, K. VDB: High-Resolution Sparse Volumes with Dynamic Topology. *ACM Transactions on Graphics (TOG) 32*, 3 (2013), 27.

[20] PARKER, S., SHIRLEY, P., LIVNAT, Y., HANSEN, C., AND SLOAN, P.-P. Interactive Ray Tracing for Isosurface Rendering. In *Proceedings of the Conference on Visualization'98* (1998), pp. 233–238.

[21] ZHANG, H., AND KAUFMAN, A. Interactive Point-based Isosurface Exploration and High-quality Rendering. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1267–1274.

[22] ZWICKER, M., PFISTER, H., VAN BAAR, J., AND GROSS, M. Surface Splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001), ACM SIGGRAPH '01, ACM, pp. 371–378.

Smooth Map Deformation Using Integral Images

Molchanov Vladimir
Westfälische Wilhelms-Universität Münster
Schlossplatz 2
48149 Münster, Germany
molchano@uni-muenster.de

Lars Linsen
Westfälische Wilhelms-Universität Münster
Schlossplatz 2
48149 Münster, Germany
linsen@uni-muenster.de

ABSTRACT

Deformation of geographical maps is a powerful tool used in geospatial data exploration systems. Virtual lenses help the user to retrieve local detail information without losing the global overview, while contiguous cartograms exploit map deformation for representing statistical data as areas of regions. In most applications, the map deformations should be smooth and computationally efficient. We propose a novel map deformation approach based on integral images computed for 2D density distributions. Depending on the prescribed density function, the resulting deformed map represents an approximation to a respective contiguous cartogram or may serve as a user-steered virtual lens. Our technique is suitable for use in highly interactive geospatial data exploration systems due to its algorithmic and computational efficiency. We test the proposed method on several artificial and real-world datasets and discuss directions for future work.

Keywords

Contiguous cartogram, geospatial visualization, smooth map deformation, RadViz, generalized barycentric coordinates projection, integral image

1 INTRODUCTION

Visual data representation becomes increasingly challenging with growing complexity of modern datasets. The data complexity can be enhanced by including more detailed information as well as due to inhomogeneity of data components. However, effective visualizations should be simple, compact, and intuitive to remain practical. Therefore, a thorough static representation of modern data on standard computer displays is often impossible.

Advanced visualization systems solve the problem of displaying complex data in real-time by allowing the user to interact with the graphical data representations. Then, dynamical changes of views steered by the user disclose certain parts or aspects of the underlying data on demand. Hiding irrelevant and highlighting important information as well as a proper choice of the presented level of detail may significantly reduce visual clutter and information overload. Thus, interactive systems make it possible for the user to navigate through, explore, transform, and analyze even very large and complex datasets.

Geo-spatial data visualization is an example of an application field with high demands on handling data at different spatial scales. General interfaces used in the interactive systems for working with multiple levels of details include *zooming*, *overview+detail*, and *focus+context* [CKB09]. Zooming may easily lead to either suppressing local information or to losing spatial awareness and the global overview [BDW$^+$08]. Overview+detail approaches integrate different degrees of data granularity in one view at the cost of introducing spatial discontinuities between the scales. Finally, the focus+context metaphor is usually realized in the form of a lens, which deforms spatial domain by expanding a region of interest while compressing areas outside the region of interest. In this case, finer details of the region of interest become locally visible, while the global overview remains preserved by avoiding discontinuities and displaying the entire context.

Cartograms are a graphical representation of statistical data over geographical regions. E.g., scalar values can be visually encoded as areas of the regions, to which these values are assigned. Contiguous cartograms are computed by continuous deformation of the original map while conserving adjacencies (or neighborhoods) of the map regions. Thus, such cartograms are topologically accurate and therefore more intuitive for the user. Cartograms are usually not aimed at representing additional data at different spatial scales as in the focus+context approach. However, formally, contiguous cartograms can be considered as a simultaneous application of lenses to all map regions according to their scalar values that determine the regions' magnification factors.

The problem of contiguous cartogram construction can be formulated as follows: For a given non-negative scalar density function representing geospatial statistical data, find a smooth transformation of the map that equalizes the density distribution, i.e., integral density in any region becomes proportional to the region area. In our work, we propose a novel globally smooth map deformation method. The deformation parameters are computed based on integral images computed for a given density distribution. We allow the user to interactively change the discrete density function, thus, steering the local deformation of the map. The proposed method is algorithmically and computationally efficient. Thus, the method can be used in interactive geospatial data exploration systems.

Our contributions include:

- We demonstrate how integral images can be used for pseudo-cartogram construction and discover algorithmic equivalence of existing methods for pseudo-cartogram computation and the RadViz (Generalized Barycentric Coordinates) visualization algorithm.
- We improve the existing map deformation algorithm by considering tilted integral images and sliding anchor points, which results in novel smooth non-linear transformations.
- We modify the proposed map-deformation algorithm to function as a magnification lens of user-defined geometry in interactive geospatial data exploration applications.
- We perform numerical tests examining properties and efficiency of the proposed methods.
- We discuss possible application scenarios and future work directions.

2 RELATED WORK

Virtual lenses in visualization may act not only as pure magnification tools, but also to query detail information and to filter or modify data representations in a region of interest. A taxonomy of virtual interactive lenses according to the types of data and user tasks was recently presented in an extensive survey by Tominski et al. [TGK$^+$17]. *Fisheye views* proposed by Furnas [Fur86] are based on local degree of interest and provide a smooth integration of two levels of details. While classical lenses have regular shape (circle or rectangular), *JellyLenses* proposed by Pindat et al. [PPCP12] dynamically adapt their geometry to the content in the focus. Haunert and Sering [HS11] modified fish-eye projectiond applied to road networks by reducing the distortion in dense network regions. Wu et al. [WLLM13] developed a technique for content-aware resizing task. The method is based on an iterative optimization procedure to compute a deformation map.

While virtual lenses are commonly an example of local distortion-based visualizations, the construction of contiguous cartograms usually requires a global map deformation. The main idea of *rubber-sheet algorithms* is a deformation of a mesh superimposed to the map. Most grid-based methods are iterative and require a number of parameters to be correctly set up by the user. The *Carto3F* method by Sun [Sun13] is a rubber-sheet algorithm that guarantees topological integrity and computationally outperforms its predecessors when implemented on the GPU. The diffusion-based density-equalizing map by Gastner and Newman [GN04] is topologically correct as well but potentially impairs shape preservation of the map regions [Sun13].

Pseudo-cartograms proposed by Tobler [Tob86] provide approximations to contiguous cartograms by distorting a rectangular mesh along two coordinate axes x and y independently. Pseudo-cartograms may serve as an initial state for applying other iterative cartogram-construction algorithms. Keim et al. [KPS$^+$03] proposed a real-time *HistoScale* method for computing pseudo-cartograms. The algorithm first computes 1D histograms of geospatial data in two Euclidian dimensions and then transforms the map by a proper rescaling of map regions in each histogram bin to fulfill the cartogram condition. *RadialScale* and *AngularScale* density-equalizing approaches developed by Bak et al. [BSS$^+$09] distort a map in radial and angular directions and, thus, can be seen as an analogy of the pseudo-cartograms by Tobler in polar coordinates.

The cartogram quality is characterized by its statistical accuracy, the topological accuracy of the geometry as well as the polygonal complexity of the resulting layout. Numerical measures for cartogram quality were summarized by Alam et al. [AKV15]. A recent overview of the state of the art in cartograms was presented by Nusrat and Kobourov [NK16]. Aesthetic appearance is an important but hardly numerically measurable aspect of cartograms. Therefore, despite of existence of many fully automatic algorithms, there is a need in graphical interactive systems providing manual user control over cartogram creation. An example of such a system was recently presented by Kronenfeld [Kro18].

User interactions with graphical views are very practical in multidimensional data exploration systems. Recently, Sacha et al. [SZS$^+$17] revealed seven common interaction scenarios of the user interactions with visual encodings of dimensionality reduction algorithm outcomes. *RadViz* proposed by Hoffman [HGM$^+$97] is a well-known projection method which maps multidimensional samples in a 2D domain according to their values and the anchor points placed on a circle. The multidimensional coefficients act as attracting forces of the anchor nodes representing dimensions for the projected samples. Cheng et al. [CXM17] identified three types of errors in classical RadViz layout and minimized the errors by properly placing the anchor points. The authors also demonstrated equivalence of RadViz to the *Generalized Barycentric Coordinates* (GBC) interpolation method by Meyer et al. [MBLD02]. Nam and Mueller [NM13] used GBC for projecting multidimensional data describing geographical locations. The *RadViz++* method proposed by Pagliosa and Telea [PT19] addresses the scalability issue of the RadViz metaphor.

Summed-area tables or *Integral Images* (InIm) were originally introduced by Crow [Cro84] and applied to object detection in image analysis by Viola and Jones [VJ02]. Computation of InIm at arbitrary angles was studied by Chin et al. [CGT08]. Ehsan et al. [ECuRM15] addressed the problem of efficient parallel computation of InIm. Nowadays, many libraries (e.g., OpenCV [Bra00] used in our work) provide out-of-the-box functions for InIm computation.

We demonstrate how InIm can be used for pseudo-cartogram construction. By doing so, we discover algorithmic equivalence of existing methods for pseudo-cartogram computation and the RadViz/GBC algorithms. We improve these pseudo-cartogram computation algorithms by using tilted InIm. Finally, we show how to apply our method for virtual lenses and contiguous cartogram applications.

3 BACKGROUND AND NOTATIONS

Let an object $\mathscr{O}$, e.g., a continent or a country, be represented on a geographical map as a union of non-overlapping regions $\mathscr{R}_i$, $i = 1,\ldots,n$, e.g, states or provinces. Let $a_i > 0$ be the area fractions of $\mathscr{R}_i$ in $\mathscr{O}$. There may also be a special region $\mathscr{B}$ representing the background, i.e., the part of the map that does not belong to $\mathscr{O}$. Let $\{v_i\}$ be statistical data assigned to all regions $\mathscr{R}_i$. For cartograms, typical examples of v_i are population or voting results in $\mathscr{R}_i$. Then, density distribution $\rho(x,y)$ is a scalar function, which satisfies the relation

$$\int_{\mathscr{R}_i} \rho(x,y)\,\mathrm{d}x\,\mathrm{d}y = v_i, \qquad i = 1,\ldots,n. \tag{1}$$

Note that the density functions used in our work are defined up to a global positive constant factor, which does not influence the results. A particular solution of Equation (1) is a piecewise constant $\rho(x,y)$ that in each region $\mathscr{R}_i$ is equal to a value v_i/a_i for a constant a_i. The density values in background $\mathscr{B}$ can be arbitrary as long as no data are assigned to the background region. We discuss this aspect in Section 5.3.

In our work, we propose an efficient and simple algorithm for a smooth deformation of a geographical map given as a (one- or three-channel) 2D texture based on a scalar-valued discrete 2D density function. For simplicity of exposition, we assume that both map and density textures have the same resolution. All computations are performed in texture coordinates ranging between 0 and 1 in horizontal and vertical directions. The resolution of the textures is arbitrary and can be chosen by the user. Note that we use the same notations for both continuous and discrete functions, e.g., density $\rho(x,y)$, to simplify the exposition, as to not lead to confusion.

Our algorithm extensively uses InIm computed for the density texture and is defined as follows. The value of InIm at any location (x,y) is the sum of values in all the pixels that are both to the left and above location (x,y) [ECuRM15], which we denote by $\alpha(x,y)$. In fact, we need four integral values per pixel $\alpha(x,y)$, $\beta(x,y)$, $\gamma(x,y)$, and $\delta(x,y)$ representing the sums of pixel values in all four rectangular domains as shown in Figure 1a. Whereas $\alpha(x,y)$ is the conventional InIm and can be directly computed by a standard function call in many image processing libraries, the other three values can be easily expressed via $\alpha(x,y)$ by

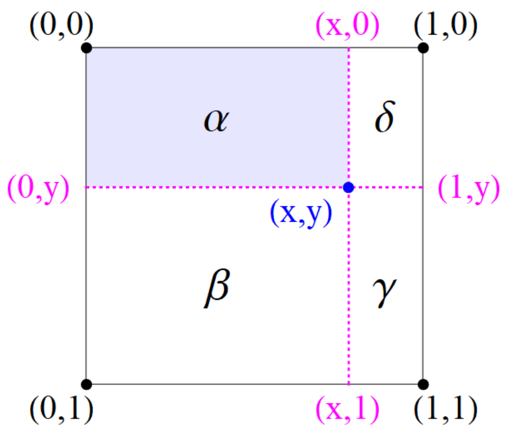
$$\beta(x,y) = \alpha(x,1) - \alpha(x,y), \tag{2}$$
$$\gamma(x,y) = \alpha(1,1) - \alpha(x,1) - \alpha(1,y) + \alpha(x,y), \tag{3}$$
$$\delta(x,y) = \alpha(1,y) - \alpha(x,y). \tag{4}$$

Without loss of generality, we can assume that

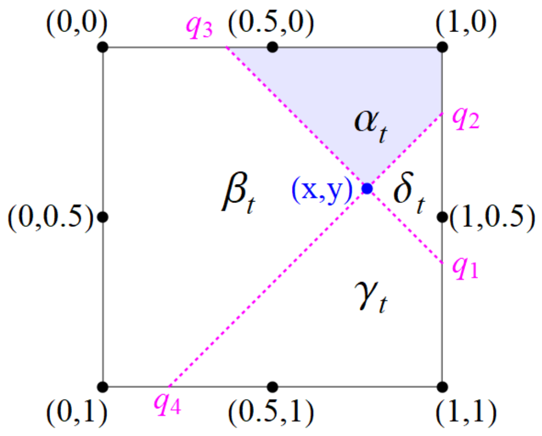
$$\alpha(x,y) + \beta(x,y) + \gamma(x,y) + \delta(x,y) \equiv \alpha(1,1) = 1, \tag{5}$$

i.e., all integral coefficients sum up to unity, which can be reached by a proper scaling of density $\rho(x,y)$. In addition to the listed InIm values, we propose to use four integral values $\alpha_t(x,y)$, $\beta_t(x,y)$, $\gamma_t(x,y)$, and $\delta_t(x,y)$ of a tilted InIm, see Figure 1b. Again, $\alpha_t(x,y)$ can be computed using existing functionality (we use function `integral` of OpenCV [Bra00] in our implementation). For computing the other three values, one may call the same function for a tilted InIm of the density texture rotated by 90°, 180°, and 270°, correspondingly. Obviously, tilted integral coefficients also sum up to unity, i.e., an analog of Equation (5) holds for $\alpha_t(x,y)$, $\beta_t(x,y)$, $\gamma_t(x,y)$, and $\delta_t(x,y)$:

$$\alpha_t(x,y) + \beta_t(x,y) + \gamma_t(x,y) + \delta_t(x,y) \equiv 1. \tag{6}$$



(a) integral coefficients at location (x,y)



(b) tilted integral coefficients at location (x,y)

Figure 1: InIm computations: (a) The four coefficients computed at location (x,y) stand for integrals of a density function over respective rectangular regions. (b) The four additional coefficients can be computed for the same location as integrals over tilted regions.

4 EQUIVALENCE OF PSEUDO-CARTOGRAM CONSTRUCTION AND GBC PROJECTION

Contiguous cartogram construction algorithms aim at equalizing a given density distribution by smoothly deforming the geographical map. The deformation has to be smooth in order to preserve neighborhood relation of the map region and avoid region overlapping, thus, supporting readability of the result. Many cartogram methods iteratively deform a spatial mesh and then apply a texture mapping technique for computing cartograms.

The pseudo-cartogram algorithm by Tobler [Tob86] computes an exact cartogram for separable density functions, i.e., when $\rho(x,y) = f(x)\cdot g(y)$ for some functions f and g. In this particular case, Tobler proposed a transformation $(x,y) \mapsto (x',y')$ given by

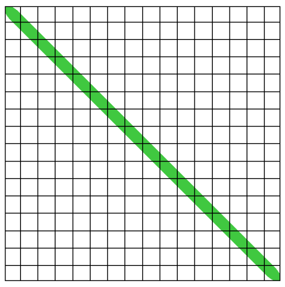
$$(x',y') = \left(\int_0^x f(t)\,\mathrm{d}t,\ \int_0^y g(t)\,\mathrm{d}t \right), \qquad (7)$$

which constructs a perfect cartogram. For a non-separable density function, an approximate solution can be obtained by [Tob86]

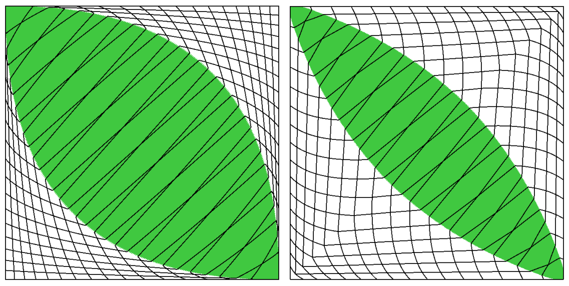
$$\begin{aligned}(x',y') &= \left(\int_0^1 \rho(x,y)\,\mathrm{d}y,\ \int_0^1 \rho(x,y)\,\mathrm{d}x \right)\\ &= (\alpha+\beta,\ \alpha+\delta), \qquad (8)\end{aligned}$$

when using the notations from Figure 1a.

Limitations of the method by Tobler come from the fact that spatial dimensions are transformed separately. Then, in particular, lines of a rectangular mesh remain straight lines after transformation. The accuracy of the pseudo-cartogram highly depends on how close the given density function can be approximated by a separable function. E.g., if $\rho(x,y)$ is constant along a diagonal of the texture and vanishes elsewhere, the pseudo-cartogram algorithm results in a trivial mapping, i.e., does not deform the map at all, see Figure 2a.



(a) Tobler's pseudo-cartogram



(b) 4 sliding anchors (c) 8 sliding anchors

Figure 2: Synthetic dataset illustrating limitations of Tobler's approach. The density is set to 1 in the green diagonal stripe and to 0 elsewhere. While Tobler's pseudo-cartogram (a) does not deform the artificial map, our map deformations using 4 (b) or 8 sliding anchors (c) perform significantly better in enlarging the populated region and contracting the empty background area.

Before proposing our cartogram method, we first demonstrate that the mapping in Equation (8) is equivalent to a RadViz or GBC projection of the texture pixels. GBC maps a multidimensional sample $(a_1,\ldots,a_k)$ to a 2D domain according to formula [CXM17]

$$\mathbf{z} = \sum_{j=1}^{k} \frac{a_j}{\sum a_i} \cdot \mathbf{v}_j, \tag{9}$$

where $\{\mathbf{v}_j\}$ are anchor points in the projection domain each representing a data dimension. The RadViz anchor points are usually placed in a circular layout. GBC generalizes the projection formula for the arbitrary configuration of the anchor-points distribution. Now, let a pixel in the undistorted map be characterized by InIm coefficients $(\alpha, \beta, \gamma, \delta)$ and let the anchor points be $\mathbf{v}_1 = (1,1)$, $\mathbf{v}_2 = (1,0)$, $\mathbf{v}_3 = (0,1)$, and $\mathbf{v}_4 = (0,0)$. Then, according to Equation (9) and when taking into account that the ImIn coefficients sum up to unity, see Equation (5), we obtain

$$\begin{aligned}\mathbf{z} &= \alpha\cdot(1,1)+\beta\cdot(1,0)+\gamma\cdot(0,1)+\delta\cdot(0,0)\\ &= (\alpha+\beta,\, \alpha+\delta),\end{aligned} \tag{10}$$

which is equal to the mapping in Equation (8) for $\mathbf{z} \equiv (x',y')$. Usually, the sample coefficients in GBC are interpreted as attracting force magnitudes acting between the sample and the respective anchor points. In our case, there are four forces, whose magnitudes are the InIm coefficients and which pull the pixel towards the respective texture corners. The resulting force induces the pixel position in the deformed map.

5 PROPOSED METHOD

5.1 Deformation Formulae

In order to alleviate limitations of the pseudo-cartogram method proposed by Tobler, we extend the deformation map in Equation (10) by taking into account the tilted InIm coefficients $(\alpha_t, \beta_t, \gamma_t, \delta_t)$. We test two options for assigning anchor nodes to the coefficients. First, in analogy to the procedure above, the nodes can be fixed and set to the midpoints of the boundary edges of the texture, resulting in a mapping

$$\begin{aligned}(x',y') &= \alpha_t(x,y)\cdot(0.5,1)+\beta_t(x,y)\cdot(1,0.5)+\\ &\quad \gamma_t(x,y)\cdot(0.5,0)+\delta_t(x,y)\cdot(0,0.5)\\ &= \left(\beta_t+\frac{\alpha_t+\gamma_t}{2},\, \alpha_t+\frac{\beta_t+\delta_t}{2}\right).\end{aligned} \tag{11}$$

However, the resulting map is clearly suboptimal, see Figure 3c, since the mapping in Equation (11) does not cover the whole texture space, i.e., regions close to the texture corners remain empty for any possible integral coefficient. Thus, the transformation in Equation (11) is impractical and we list it only for the sake of complete exposition.

Instead, we propose to use anchor points which depend on the coordinates of the mapped pixel, namely

$$\begin{aligned}(x',y') &= \alpha_t(x,y)\cdot(x,1)+\beta_t(x,y)\cdot(1,y)+\\ &\quad \gamma_t(x,y)\cdot(x,0)+\delta_t(x,y)\cdot(0,y)\\ &= (\beta_t+x\cdot(\alpha_t+\gamma_t),\, \alpha_t+y\cdot(\beta_t+\delta_t)).\end{aligned} \tag{12}$$

The new adaptive anchor points are the intersection points of the dashed lines in Figure 1a with the sides of the domain boundary. We refer to this transformation as a *four sliding anchor map*. The mapping in Equation (12) is much more flexible when compared to Tobler's pseudo-cartogram. Note that the straight lines of the mesh become curved after the transformation, see Figure 2b.

Tobler's pseudo-cartogram is by construction accurate for separable density distributions but it may perform poorly otherwise. Our transformation in Equation (12) instead is not exact for separable $\rho(x,y)$ but is more flexible. We further extend our four sliding anchor map by combining it with a flexible version of Tobler's psudo-catrogram. We propose to modify Tobler's method by using sliding anchors instead of the four fixed texture corners used in Equation (9) and average the resulting mapping with our transformation in Equation (12), such that each pixel is mapped using *eight sliding anchors*. The proposed deformation formula has then the form:

$$\begin{aligned}(x',y') = \frac{1}{2}\Big(&\alpha\cdot q_1(x,y)+\beta\cdot q_2(x,y)+\\ &\gamma\cdot q_3(x,y)+\delta\cdot q_4(x,y)+\\ &(\beta_t+x\cdot(\alpha_t+\gamma_t),\, \alpha_t+y\cdot(\beta_t+\delta_t))\Big),\end{aligned} \tag{13}$$

where the new adaptive anchor points $q_i(x,y)$, $i = 1,\ldots,4$, are the intersection points of the dashed lines in Figure 1b with the sides of the domain boundary. The anchors can be computed according to these formulae:

$$\begin{aligned}\text{if } y<x: &\quad q_1(x,y)=(1,\,1+y-x),\\ &\quad q_3(x,y)=(x-y,\,0);\\ \text{if } y\ge x: &\quad q_1(x,y)=(1-y+x,\,1),\\ &\quad q_3(x,y)=(0,\,y-x);\\ \text{if } x+y<1: &\quad q_2(x,y)=(x+y,\,0),\\ &\quad q_4(x,y)=(0,\,x+y);\\ \text{if } x+y\ge 1: &\quad q_2(x,y)=(1,\,x+y-1),\\ &\quad q_4(x,y)=(x+y-1,\,1).\end{aligned} \tag{14}$$

The proposed deformation can be applied to every pixel of the undistorted map. However, in our tests, we map only nodes of a regular rectangular mesh according to Equation (13) and then use texture mapping technique for interpolating the map within the grid cells. For better understanding of the deformation outcome, the sparse grid lines after transformation are shown in the results in Figure 2c.

5.2 Algorithm

The complete proposed deformation algorithm includes following five steps:

1. Prepare density texture $\rho(x,y)$, i.e., based on geostatistical data.
2. Compute textures containing integral coefficients.
3. Define a mesh over the geographical map, which will be deformed.
4. Apply deformation formula (13) for computing the new positions of the mesh's vertices.
5. Interpolate the map within each deformed mesh cell. This can be performed using a texture mapping technique.

Note that due to Equation (2), only one texture $\alpha(x,y)$ is sufficient for computing $\beta(x,y)$, $\gamma(x,y)$, and $\delta(x,y)$. Moreover, due to Equation (6), only three tilted InIms are needed. Therefore, in total, only four textures with InIms are required for computing the proposed deformations (12) and (13).

5.3 User Interactions with Density Texture

The density value for background region $\mathscr{B}$ is typically not defined. However, its choice may significantly affect the resulting deformation map. If the assigned density value is high, object $\mathscr{O}$ (the non-background regions) will generally be contracted by the transformation. If the background density is close to the average density of object $\mathscr{O}$, the size of object $\mathscr{O}$ will tend to remain close to its original size. Also, its shape may remain close, but the the internal borders of the regions within $\mathscr{O}$ will change. Finally, vanishing density in $\mathscr{B}$ will result in mappings minimizing the background size while maximizing the total area of object $\mathscr{O}$. Thus, such a mapping can be interpreted as a global zooming transformation, which aims at efficient usage of available empty space in the map.

The map deformation algorithms proposed in the previous section can be executed at interactive rates, thus, they can be used in interactive applications. One possible application scenario of the methods is a user-steered magnification of an area of interest similar to a virtual lens. Since the resulting mapping is global and smooth, there is no need for a costly computation of the transition region between the magnified region and the rest of the map.

6 RESULTS

For our experiments we used geospatial data in the form of shape files provided at `www.gadm.org`. All maps are given in a longitude-latitude coordinate system.

In our first numerical experiment, we compare the different proposed deformation formulae for exploring their character, best application cases, and limitations. Figure 3a shows the original map of the states of Italy and the resulting deformed maps. In all cases, the density distribution is piecewise constant being equal to 1.0 for all states and equal to 0.0 for background. Thus, ideally the cartogram would deform the map such that the background area vanishes. With r being the ratio of the background to country area, Tobler's pseudo-cartogram reduces the value of r from originally 3.18 to 1.44. The map for the tilted InIm approach with fixed anchors in Figure 3c was already discussed above. It does not make use of the whole texture space, since the four fixed anchors are placed in the centers of the texture sides. The deformation with four sliding anchors computed according to Equation (12) reduces the background area the most efficiently resulting in $r = 1.06$. The combination of this transformation with the flexible Tobler map (eight sliding anchors) gives $r = 1.22$, see Figures 3d and 3e. The fact that the four sliding anchors have a lower value r than the eight sliding anchors can be understood, if one considers that the shape of Italy is dominantly diagonal in the longitude-latitude coordinates. The non-linear shape of the deformed mesh lines is clearly visible in the proposed maps.

In our second test, we examine how the density value assigned to the background region affects the resulting deformation. Figure 4a shows undistorted map of the states of the Czech Republic. Constant density values are assigned to each region according to population data in 2011. The average density in the foreground regions is equal to unity. We test the proposed algorithms with four and eight sliding anchor nodes with background density $\rho_0 = 1.0$, 0.5, and 0.0. As ρ_0 decreases, the foreground regions occupy larger parts of the map, i.e., the background area diminishes. Note that in contrast to diagonally spread map of Italy in our first test, the Czech Republic map is horizontally elongated. Consequently, the method with eight sliding anchor nodes performs better in contracting the empty region, see Figures 4g and 4h. In Tobler's pseudo-cartogram in Figure 4b the most dense region of Prague (pink area in the north-west) leads to expansion the Praque region, but also of other regions that share the same longitude or latitude coordinates with Praque irrespective of their real population density. The expansion of the Prague region in our proposed methods instead has a more localized character as can be seen from the curved mesh lines, e.g., in Figure 4g. Therefore, shapes of the re-

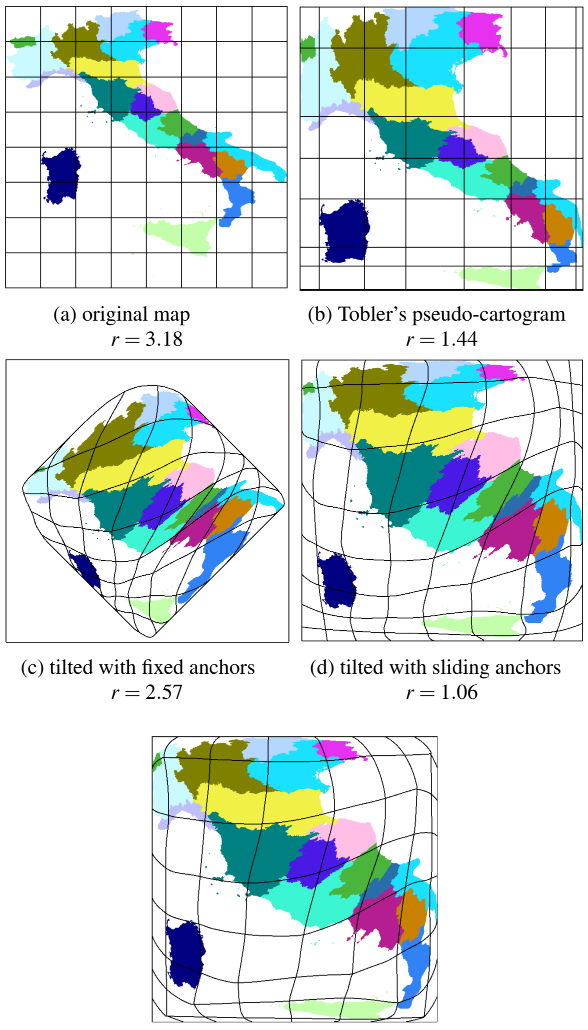

(a) original map
$r = 3.18$
(b) Tobler's pseudo-cartogram
$r = 1.44$
(c) tilted with fixed anchors
$r = 2.57$
(d) tilted with sliding anchors
$r = 1.06$
(e) combined deformation
$r = 1.2$

Figure 3: Comparison of different transformations of a map of Italy. (a) Original map with an overlaid regular mesh. Constant density values of 0.0 and 1.0 are assigned to the land and background regions, correspondingly. (b) Tobler's pseudo-cartogram. (c) Deformations with fixed anchors according to Equation (11). (d) Deformations with 4 sliding anchors according to Equation (12). (e) Deformations with 8 sliding anchors according to Equation (13). Ratios of the background to country area r are presented for all maps.

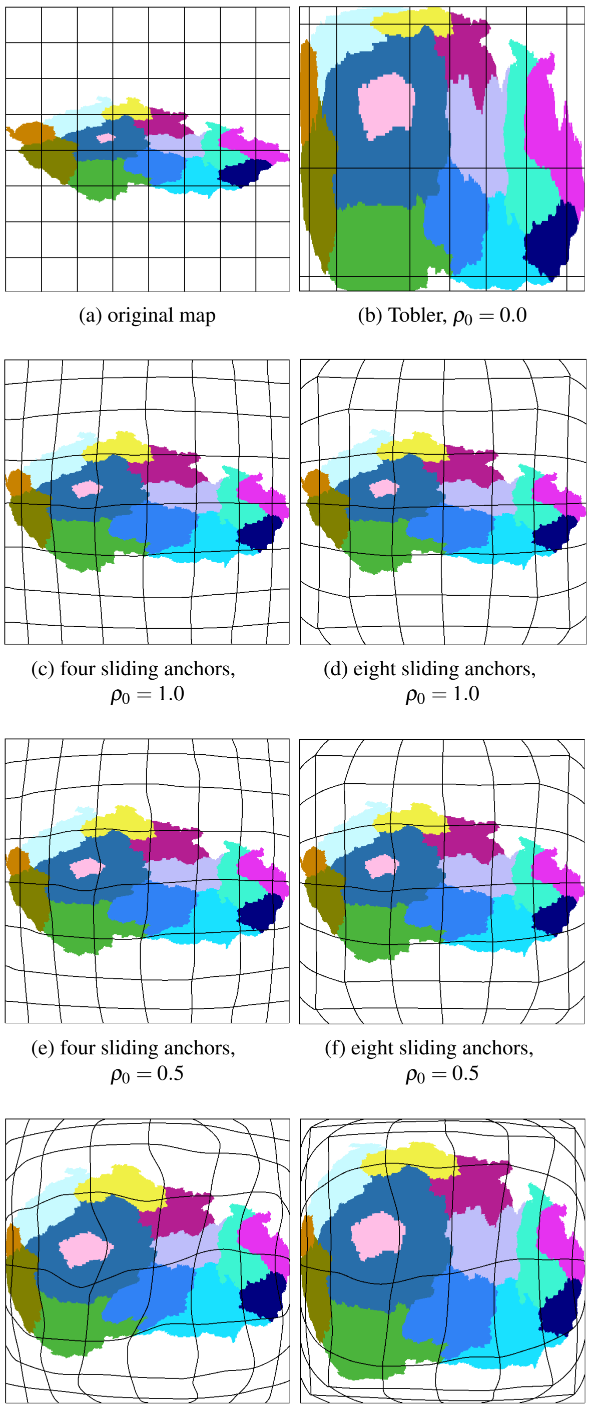

(a) original map
(b) Tobler, $\rho_0 = 0.0$
(c) four sliding anchors,
$\rho_0 = 1.0$
(d) eight sliding anchors,
$\rho_0 = 1.0$
(e) four sliding anchors,
$\rho_0 = 0.5$
(f) eight sliding anchors,
$\rho_0 = 0.5$
(g) four sliding anchors,
$\rho_0 = 0.0$
(h) eight sliding anchors,
$\rho_0 = 0.0$

Figure 4: Dependence of the foreground region's shape preservation in the deformed map on the background density value ρ_0.

gions in our maps are better preserved when compared to the Tobler map.

Next, we measured computational times required for the main steps of the proposed algorithms. We found that it takes, on average, 53ms for computing all eight InIm textures of size 1024^2 and 6.5ms for mapping 128^2 nodes of a regular mesh. The execution times were measured by averaging $1,000$ runs of the respective steps. Integral images are computed using the OpenCV 3.3.1 library. The user interface is developed in Qt 5.11. All relevant computations are performed on a PC with an i7-7700K 4.20GHz CPU. Texture mapping is executed using OpenGL 4.6.

Finally, we demonstrate how the proposed methods can be used in interactive exploration systems as a virtual lens. The user selections of regions of interest on a to-

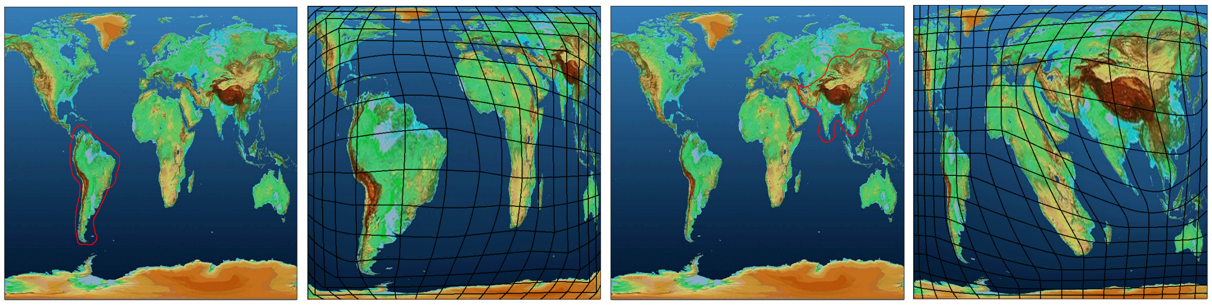

(a) user selection
(b) eight sliding anchors, $\rho_1 = 13.0$
(c) user selection
(d) four sliding anchors, $\rho_1 = 13.0$

Figure 5: Focus+context zooming of a user-selected region: (a,c) User selections on the original map. (b,d) Transformed maps with density value $\rho_1 = 13.0$ assigned to the selected regions, while the area outside the selected region keeps density value 1.0.

pographical world map are shown Figure 5a and 5c, respectively. The selections are performed interactively using a lasso tool, i.e., the region does not need to be predefined. Then, the user interactively increases the constant density value assigned to the selected area from an initial value of $\rho_1 = 1.0$ to a new value of 13.0, while the background density (everything outside the selected region) remains equal to 1.0. The results of the transformation for the two selections are shown in Figures 5b and 5d. We observe that the selected regions appear as being zoomed in, while the rest of the original map is smoothly deformed.

In the accompanying video, we provide further test cases, demonstrate a prototype of our geospatial data exploration system, and explain possible user interactions with the tool.

7 CONCLUSION AND FUTURE WORK

In our work we showed how Tobler's pseudo-cartograms can be computed using InIm, i.e., an InfoVis task is solved by a projection method (RadViz or its generalization GBC) using an image processing tool. We illustrated the limitation of the method by constructing a simple synthetic dataset, for which Tobler's method is ineffective. We furthermore showed the equivalence of the pseudo-cartogram construction and the GBC projection method. Using tilted InIm, we proposed new deformation algorithms with four and eight sliding anchor nodes, correspondingly, which result in non-linear non-separable globally smooth mappings. Therefore, adjacency of the subregions of the original map is perfectly preserved in the transformed one, which is one of a desired properties of contiguous cartograms. We showed how these techniques can be used as virtual lenses for mapping regions of arbitrary shapes. Algorithmic simplicity and computational efficiency of the proposed algorithms allow for using them in highly interactive applications.

Many existing map deformation algorithms are iterative. Since the proposed mappings are explicit, disadvantages and limitations of iterative methods (i.e., need in setup of non-intuitive parameters, preventing overlapping of subregions, and computational burden) are avoided.

Future directions of our research include:

- Using InIm tilted by arbitrary angles (not only multiples of 45°), we may arbitrarily increase the number of anchor nodes. The properties of the limiting deformation map as the number of anchor increases are of theoretical and practical interest.
- Since the positions of the anchor points are arbitrary, it may be possible to construct deformations defined for non-rectangular domains.
- All steps of the proposed algorithm allow for parallel computing. Thus, despite the fact that our computation times are already low and allow for interactive applications, computation times could further be significantly reduced by implementing our methods on the GPU.
- Application of the proposed algorithm to regularization of arbitrary distributed point clouds is technically possible. Certain adaptations for efficient computation of the density function are necessary.

8 ACKNOWLEDGMENTS

This work was supported by DFG grant MO 3050/2-1.

9 REFERENCES

[AKV15] Md. Jawaherul Alam, Stephen G. Kobourov, and Sankar Veeramoni. Quantitative measures for cartogram generation techniques. *Computer Graphics Forum*, 34(3):351–360, 2015.

[BDW$^+$08] Thomas Butkiewicz, Wenwen Dou, Zachary Wartell, William Ribarsky, and Remco Chang. Multi-focused geospatial analysis using probes. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1165–1172, 2008.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[BSS$^+$09] Peter Bak, Matthias Schaefer, Andreas Stoffel, Daniel A. Keim, and Itzhak Omer. Density equalizing distortion of large geographic point sets. *Cartography and Geographic Information Science*, 36(3):237–250, 2009.

[CGT08] Tat-Jun Chin, Hanlin Goh, and Ngan-Meng Tan. Exact integral images at generic angles for 2D barcode detection. In *2008 19th International Conference on Pattern Recognition*, pages 1–4, December 2008.

[CKB09] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.*, 41(1), January 2009.

[Cro84] Franklin C. Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 207–212, New York, NY, USA, 1984. Association for Computing Machinery.

[CXM17] Shenghui Cheng, Wei Xu, and Klaus Mueller. RadViz deluxe: An attribute-aware display for multivariate data. *Processes*, 5(4), 2017.

[ECuRM15] Shoaib Ehsan, Adrian F. Clark, Naveed ur Rehman, and Klaus D. McDonald-Maier. Integral images: Efficient algorithms for their computation and storage in resource-constrained embedded vision systems. *Sensors (Basel)*, 15(7):16804–16830, July 2015.

[Fur86] G. W. Furnas. Generalized fisheye views. *SIGCHI Bull.*, 17(4):16–23, April 1986.

[GN04] Michael T. Gastner and M. E. J. Newman. Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences*, 101(20):7499–7504, 2004.

[HGM$^+$97] Patrick Hoffman, Georges Grinstein, Kenneth Marx, Ivo Grosse, and Eugene Stanley. DNA visual and analytic data mining. In *Proceedings of the 8th Conference on Visualization*, VIS '97, pages 437–442, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[HS11] J. Haunert and L. Sering. Drawing road networks with focus regions. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2555–2562, December 2011.

[KPS$^+$03] Daniel A. Keim, Christian Panse, Matthias Schäfer, Mike Sips, and Stephen C. North. HistoScale: An efficient approach for computing pseudo-cartograms. In *14th IEEE Visualization 2003 (VIS 2003)*, Piscataway, N.J, 2003.

[Kro18] Barry J. Kronenfeld. Manual construction of continuous cartograms through mesh transformation. *Cartography and Geographic Information Science*, 45(1):76–94, 2018.

[MBLD02] Mark Meyer, Alan Barr, Haeyoung Lee, and Mathieu Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools*, 7(1):13–22, 2002.

[NK16] Sabrina Nusrat and Stephen Kobourov. The state of the art in cartograms. *Computer Graphics Forum*, 35(3):619–642, June 2016.

[NM13] J. E. Nam and K. Mueller. TripAdvisor^{N-D}: A tourism-inspired high-dimensional space exploration framework with overview and detail. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):291–305, February 2013.

[PPCP12] Cyprien Pindat, Emmanuel Pietriga, Olivier Chapuis, and Claude Puech. JellyLens: Content-aware adaptive lenses. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, pages 261–270, New York, NY, USA, 2012.

[PT19] Lucas Pagliosa and Alexandru Telea. RadViz++: Improvements on radial-based visualizations. *Informatics*, 6:16, Apr 2019.

[Sun13] Shipeng Sun. A fast, free-form rubber-sheet algorithm for contiguous area cartograms. *International Journal of Geographical Information Science*, 27(3):567–593, 2013.

[SZS$^+$17] D. Sacha, L. Zhang, M. Sedlmair, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):241–250, January 2017.

[TGK$^+$17] C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann. Interactive lenses for visualization: An extended survey. *Computer Graphics Forum*, 36(6):173–200, September 2017.

[Tob86] Waldo R. Tobler. Pseudo-cartograms. *The American Cartographer*, 13(1):43–50, 1986.

[VJ02] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2002.

[WLLM13] Yingcai Wu, Xiaotong Liu, Shixia Liu, and Kwan-Liu Ma. ViSizer: A visualization resizing framework. *IEEE Transactions on Visualization and Computer Graphics*, 19(02):278–290, February 2013.

AICO, Artificial Intelligent COach

Vicent Sanz Marco
Osaka University
1-32 Toyonaka city,
560-0043, Osaka,
vicent.sanzmarco@aist.go.jp

Norimichi Ukita
Toyota
Technological
Institute
2-12-1 Nagoya,
468-8511
Japan
ukita@toyota-ti.ac.jp

Natsuko Kobayashi
Osaka University
1-32 Toyonaka city,
560-0043, Osaka,
kobayashi-
nat@cmc.
osaka-u.ac.jp

Morito Matsuoka
Osaka University
1-32 Toyonaka city,
560-0043, Osaka,
matsuoka@cmc.
osaka-u.ac.jp

ABSTRACT

Choosing effective strategies before playing against an opponent team is a laborious task and one of the main challenges that American football coaches have to cope with. For this reason, we have developed an artificial intelligent American football coach (AICO), a novel system which helps coaches to decide the best defensive strategies to be used against an opponent. Similar to coaches who prepare a winning game plan based on their vast experience and previously obtained opponents' statistics, AICO uses power of machine learning and video analysis. Tracking every player of the last recorded matches of the opponent team, AICO learns the strategies used by them and then calculates how successfully their own defensive strategies will perform against them. We have used 7350 videos in our experiments obtaining that AICO can recognize the opponent's strategies with about 93% accuracy and provides the successful rate of each strategy to be used against them with 94% accuracy.

Keywords
Computer Vision, image and Video Processing, Pattern Recognition, Sport, AI

1 INTRODUCTION

Nowadays, the advancement of neural network applications has been a useful tool to develop automatic analysis systems to help with sport analysis, being an active research topic in the last years [Jia16, Fre19]. In this research, we have created an artificial American football coach to help the coaches to determine the best strategies to be used against opponent teams. In order to assist coaches with this laborious task, we have created AICO, a novel video analysis recognition system which works together with machine learning.

Recognition and tracking players in the field is made by video analysis, which determines the strategies used by both teams, offensive and defensive ones. To the best of our knowledge, other research related to American football is only focused on professional teams, not being easily accessible by other kind of teams, such as small teams or amateurs. There are studies to detect players in the field [Rie13, Dir18], track player's movement [Yam13], recognize offensive strategies [Atm13, Sid09].However, all these investigations required professional resources, such as, broadcast videos obtained from TV [Ste17] or the utilization of specialized and expensive devices to track players [Bur17].

For this reason, we have developed AICO, an inexpensive and versatile system that can be used everywhere. AICO is an artificial recognition system which can be used by any type of American football team, since it does not require any previous device installation on players, stadium or field.

American football coaches spend a considerable time studying American football videos from opponent teams, trying to discern strengths and weakness and use them for their own benefit. AICO is a novel automatic system developed to help coaches in that matter. AICO gathers the opponents movements and strategies in order to discover the best strategies to be used against that team, lightening the amount of work made by the coaches involved in the analysis of the opponent videos.

On the other hand, since during our research we could not find any available American football video dataset containing match plays, we have created an American football video dataset containing about 7350 plays of different teams, making possible the use of this dataset in further video recognition comparisons in future projects.

2 AMERICAN FOOTBALL BACKGROUND

In American football, there are two teams with 11 players who compete in four quarters of 15 minutes. In every play, one of the teams is defined as the offensive team, the one in possession of the ball, while the other team is defined as the defensive team.

According to American football regulations [Goo18], the offensive team needs to move the ball forward at least 10 yards. This is why the field has clearly marked yardage lines on it. The offensive team has 4 attempts, called *downs* in American football, to either score or gain 10 or more yards. If the ball is moved that far, the count resets, and the team earns another set of four downs to try go a further 10 yards. If the offensive team does not reach 10 or more yards in the 4 downs, the defensive team gains the possession of the ball and changes its role to offensive team. In this paper, a *play* is defined as a down of the offensive team.

To know if a strategy was successful or not, we are using the American football analytic made by Football Outsider [Out19] which focuses on advanced statistical analysis of the NFL and is run by professional sport journalists. An offensive strategy is defined as *successful* by Football Outsiders if it gains at least 40% of the yards-to-go on the first down, 60% of yards-to-go on the second down and 100% of yards-to-go on third or fourth down. Otherwise, the strategy is defined as *unsuccessful*.

American football coaches have a *playbook* containing all the offensive and defensive strategies their team can play during a match. The offensive strategies are separated in *passing strategies* and *running strategies*. In *passing strategies*, the quarterback attempts to pass the ball to another player. On other hand, in *running strategies*, the quarterback runs with the ball or gives the ball to a closest player that makes the run. A team has numerous passing and running strategies in their playbook. The defensive strategies are divided depending on the amount of players used in the front line. For our experiments, the playbook used has about 100 offensive strategies (60 passing and 40 running) and 100 defensive strategies.

To determine the offensive strategies to be used against an opponent team, coaches choose the best strategies players performed in the previous games or training sessions. However, for selecting defensive strategies is a different story.

In sports, team coaches analyze the opponents teams in order to find the best strategy to play against them. American football is not an exception. Thanks to our collaboration with *Kosei Gakuen High School* American football (KSS Lotus) team, we verified that coaches visualize the last matches of the next rival, checking every offensive play used by the opponent team. Figure 1

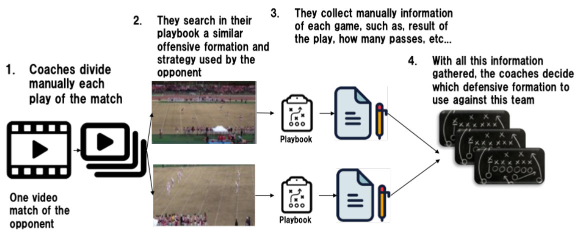

1. Coaches divide manually each play of the match
One video match of the opponent
2. They search in their playbook a similar offensive formation and strategy used by the opponent
Playbook
Playbook
3. They collect manually information of each game, such as, result of the play, how many passes, etc...
4. With all this information gathered, the coaches decide which defensive formation to use against this team

Figure 1: Flowchart showing how the coaches decide their defensive strategies against an opponent

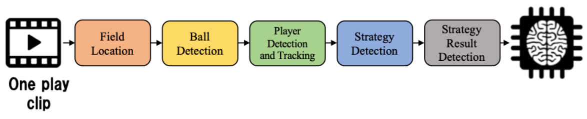


Figure 2: Preprocessing data to train AICO from a play clip

shows a flowchart describing how coaches gather manually the information about the opponent team. In every play, coaches compare their offensive strategies in the playbook to the opponent strategy used, trying to match it with the most similar one in their playbook. They also collect other relevant information, such as, how many passes failed or who was the most relevant player of the other team. After having collected all the information from the opponent team, coaches sit together and discuss the best defensive formations and strategies to be used against that team. This work is manually done by coaches, however, in this paper, we propose an alternative system, AICO. A novel video analysis system together with neural networks that can automatically gather the relevant information of the opponent team and determines the best defensive strategies to be used against that team.

3 ALGORITHM

One of the two main goals pursued by AICO is the creation of an automatic video analysis system. The flowchart showed in Figure 1 represents how coaches examine a specific opponent team using the opponent match recordings. This procedure made by the coaches is repeated for every opponent team. Similar to the way coaches get information about their opponents, the video analysis system implemented is responsible to obtain the data from the recordings of the last opponent team matches. A strategy recognition algorithm has been developed to analyze the last video matches of the opponent team and detect strategies used in every play. The algorithm detects the offensive strategies of the team chosen as opponent, and also calculates how successful were the defensive strategies of the other team against them. From now on, this strategy recognition process is going to be called *preprocessing step*, whose overview chart can be found in Figure 2.

The second main goal of AICO uses the results obtained in the *preprocessing step* to learn the strategies of a specific adversary and provide the most reliable defensive strategies to be used against this opponent team, calculating the percentage of success of every strategy provided by the coaches. As consequence, AICO is trained independently for every opponent, obtaining different AICOs for every team analyzed.

3.1 Field Location

AICO currently works only using one single-camera and there are no restrictions on the camera used to record the match as long as it is in a fixed position with a tripod and has at least a quality of 720p.

After receiving the full match of an opponent team, AICO uses an improved Chen's algorithm [Che14], with 92% of detection accuracy, in order to recognize different sequences of plays. We have modified Chen's algorithm to detect the plays in any kind field, due to the algorithm only work if the field has grass, but this is not always the case. In the labeling step, AICO classifies every play depending on the actions the teams are performing. AICO only keeps offensive and defensive plays (downs), which are labeled as *play clips*, discarding other kind of plays such as, field goal plays and extra point plays.

For each *play clip*, AICO performs Direct Linear Transform (DLT) [Har03] to detect what part of the field the camera is recording. DLT algorithm is used to resolve the homography matrix H between the first frame of the *play clip* and a digital American football field model. From now on, this digital American football field model will be referred to as the *football model*. Since the system works in homogeneous coordinates, a point (x,y) from the real field and a point (x',y') from the *football model* can be expressed as:

$$c\begin{bmatrix}x'\\y'\\1\end{bmatrix}=H\begin{bmatrix}x\\y\\1\end{bmatrix} \quad (1)$$

where c is any non-zero constant, and

$$H=\begin{bmatrix}h_1 & h_2 & h_3\\h_4 & h_5 & h_6\\h_7 & h_8 & h_9\end{bmatrix} \quad (2)$$

To resolve Equation 1, the first row is divided by the third row:

$$-h_1x-h_2y-h_3+(h_7x+h_8y+h_9)x'=0 \quad (3)$$

and the second row is divided by the third row:

$$-h_4x-h_5y-h_6+(h_7x+h_8y+h_9)y'=0 \quad (4)$$

As consequence, Equation (3) and (4) can be expressed in a matrix form:

$$A_ih=0 \quad (5)$$

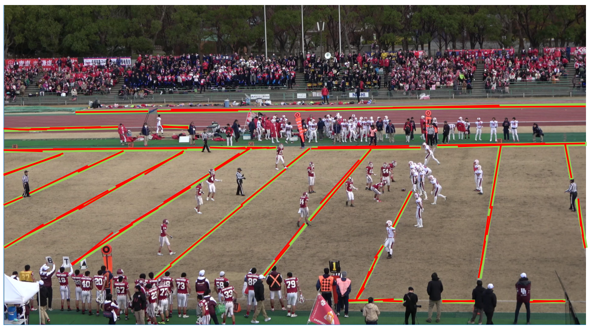


Figure 3: Hough transform lines are represented in red while RANSAC lines are in green

where $A_i=\begin{bmatrix}-x & -y & -1 & 0 & 0 & 0 & x'x & x'y & x'\\0 & 0 & 0 & -x & -y & -1 & y'x & y'y & y'\end{bmatrix}$ and $h=\begin{bmatrix}h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9\end{bmatrix}^T$.

Since each correspondent point in the first frame (x,y) and its relative (x',y') in the *football model* provides 2 equations, 4 correspondent points are enough to calculate H [Ela08]. AICO requests the user to provide 2 matching reference points for the first frame and other 2 for the *football model*.

AICO uses the homography matrix H to localize the field boundaries. Once it is obtained, AICO will only focus on the players located inside the field, not considering anything or anyone out of it.

When all the *play clips* sent are obtained from the same point of view, the homography calibration only needs to be made once and can be used by all of them. Only when a *play clip* is sent to AICO from a different perspective, it will be necessary to introduce the 4 matching reference points to calculate H again.

To resolve this issue, AICO has an automatic algorithm to detect if the *play clip* has a different perspective from the previous one. This algorithm uses Hough transform [Bal87] to detect the lines in the first frame. After these lines are obtained, the algorithm uses RANSAC [Chu03] to join lines that are detecting the same line. AICO stores these RANSAC lines together with the 4 matching reference points.

Figure 3 shows an example of the lines obtained from Hough transform (red lines) and the RANSAC lines (green lines) obtained. AICO will store these RANSAC lines or used them to compare with the RANSAC lines of the previous *play clip*.

When a new *play clip* is analyzed, AICO compares all RANSAC lines of the first frame of this *play clip* to the all RANSAC lines obtained from the previous *play clip*. When the location of the RANSAC lines of both play clips coincide, with a difference of less than 10%, AICO assumes that the location of the camera is the same in both *play clips*. As consequence, the 4 reference points of the previous *play clip* are used for this new *play clip*, and the user does not need to introduce them again.

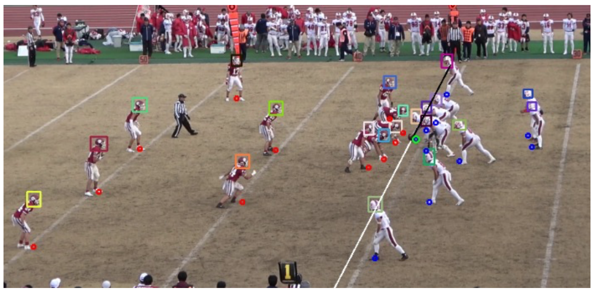


Figure 4: Example of ball line detection and player detection

3.2 Ball Detection

The American football regulation establishes that, at the beginning of every play, the ball has to be on the ground, and only the *Center* player from the offensive team is allowed to touch the ball with one hand. This *Center* player is the one who passes the ball to the quarterback at the beginning of each play. Team strategy performances start right after the ball starts moving, so this is the crucial moment for tracking players and determining the strategies of both teams.

As consequence, to determine when the play starts, it is necessary to track the ball and detect when the ball is moved for first time. When the ball is moved, AICO starts tracking players in order to determine the offensive and defensive strategies used in that *play clip*.

AICO uses the Convolutional Neural Network (CNN) Inception_v2 [Jof15] in the first frame of each *play clip* to locate the ball. This CNN has been trained to detect only American football balls.

Once the ball is detected, it is tracked via Sparse Collaborative appearance Model (SCM) [Zho14]. This tracker algorithm has been demonstrated to be one of the best state-of-the-art tracker models [Wy015]. AICO uses Equation (1), where (x, y) are the ball tracker coordinates, and H is the homography matrix calculated in the field location step, to locate the ball in the *football model*.

When the *Center* player passes the ball to the quarterback, the ball is not only moved in the real world, but it is also moved in the *football model*. We have defined that if the ball is moved 1 yard or more in the *football model*, then the tracker following the ball is deleted and the frame is frozen. This frame is used to detect all players in the field in the player detection step.

Additionally, AICO draws an artificial line that crosses the initial ball position and goes parallel to the closest yard line in the field. This line, called *ball line*, is used by the player detection step to determine if a player is part of the offensive or defensive team. Figure 4 shows an example of the ball line detection. AICO only uses one line, however it has been represented by two lines (black and white) in the figure to make it easy to see the ball position.

3.3 Player Detection and Tracking

After the ball has been moved to the quarterback, AICO freezes the frame and uses the CNN Inception_v4 [Sze17] to detect every player in the field. This neural network has been trained to detect only American football players. So referees are not included in the detection.

Inception_v4 frames all the players detected in the field. Once players are detected, AICO treats each player's frame separately. We have seen in our experiments that tracking the whole body of the players results in lost tracking of the player assigned. However, if the helmet is tracked instead of the body, the tracking accuracy obtained is very high (around 95%). For this reason, another CNN (Inception_v2) is performed on each player's frame. This inception_v2 has been trained to detect the players' helmet in each player's frame.

The helmet detected of the player is assigned to the Siamrpn++ [Li19] tracker. As consequence, each player has each on personal Siamrpn++ tracker.

However, the position calculated in the *football model* using the helmet position does not allow AICO to localize the correct position of each player in the *football model*. To obtain the correct position of the player in the *football model*, it is necessary to use their feet. Thus, for every helmet tracker, AICO also established a *location point*. This *location point* is the lower point of the line that goes from the middle of the helmet tracker to the bottom of the player's frame. The *location point* follows the helmet tracker, so, wherever the helmet tracker moves, the location point is always defined with the tracker. As consequence, the correct position of the player in the *football model* is determined using the *location point* together with the homography matrix H calculated in the field location step (Equation (1)).

Figure 5 shows an example of the player detection procedure. In this example, Inception_v4 detects the player and creates the green frame. Afterwards, Inception_v2 recognizes the helmet and defines the blue frame. It is possible to create a vertical line (yellow line) between the middle width point of the helmet frame and the player's frame bottom. The intersection between this yellow line and the player's frame bottom is represented as a red dot. This red dot is the *location point* of that player.

In addition, AICO can automatically determine if the player is in the offensive or defensive team. Once every player in the field is located at the beginning of the play, AICO searches the *Center* player. This player is the closest player to the ball and it is always part of the offensive team. In Figure 4, the *location point* of the *Center* player is marked by a green circle.

After having located the *Center* player, AICO uses the *ball line* to determine if the offensive team is on the

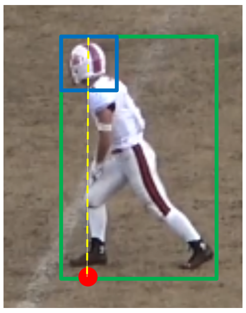


Figure 5: Player detection method

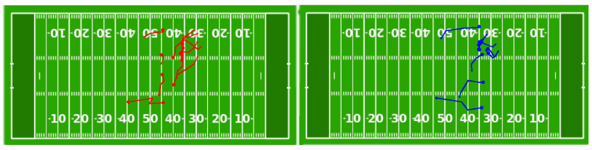


Figure 6: Tracking obtained after player detection step is finished. Defensive tracking on the left and offensive tracking on the right

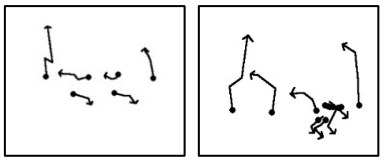


Figure 7: Converted images of the offensive (right) and defensive (left) tracking used to determine the strategies of both teams

right or left side of the ball. Consequently, using the *ball line* and the location player, AICO can define if players are part of the defensive or offensive team.

Once all players are detected and tracked, AICO creates two images: one has the tracking of the offensive players and the other contains the tracking of the defensive players. Figure 6 shows an example of these two images. As it is shown in Figure 7, these two images are sent to an application that uses OpenCV [Kae16] to clean, rotate and convert them in images that can be used to determine the strategy used by both teams in the strategy detection step.

Defensive strategies in coach playbooks do not include players in the front line, since their main responsibility is to stop the other team offensive front line and execute always the same movement. Therefore, the OpenCV application removes the defensive front line players in the defensive converted image.

3.4 Strategy Detection

AICO compares the offensive and defensive strategies obtained in player detection step to the coach playbook, looking for the most similar strategies contained in the playbook. To compare the images AICO uses OpenCV.

For this comparison, the strategies in the playbook are converted in images and then compared to the converted images obtained in the player detection step. AICO uses Optical Character Recognition(OCR) to detect the text contained in the playbook images.

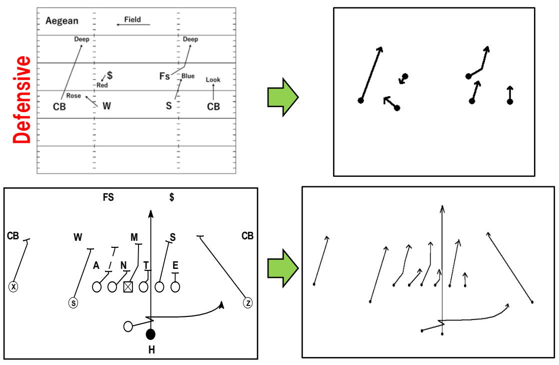


Figure 8: Example of strategy image conversion

Figure 8 shows the strategy conversion using the OpenCV application (right) from the playbook (left). All strategies in the playbook use the same tags to define the players position, such as, *CB* for cornerback player or *H* for running back player. These tags are used by the OpenCV application to find the initial position of each player, as well as they are used to localize the closest line that determines where the movement line of the player starts. Then, player tags are replaced by dots and the remaining text is removed from the final image converted. Additionally, the player movement lines are converted in similar lines that the ones AICO uses in the converted tracking images of the players generated in the player detection step. In the Figure 8, the defensive strategy called *Aegan* has only 6 players, since the 5 players from the defensive front line are not included because they are not relevant to the strategy. The defensive front line players are not relevant due to their only objective is to go forward and tackle the quarterback independently of the defensive strategy used.

The converted strategies are stored in a database that can be used by AICO in the future, this conversion process is executed only once per strategy. In this database is store the original strategy, the name of the strategy and the converted strategy. The coaches can use several playbooks and increase the strategies that AICO contains in its database. As consequence, AICO will have more strategies to compare with, achieving a more precise detection of the strategy used by the teams. For our experiments, AICO uses a strategy database of around 100 offensive and 100 defensive strategies.

AICO uses OpenCV to search the most similar strategies from the playbook that matches the converted tracking result images obtained in the player detection step. *SIFT* algorithm [Low04] is used to compare the tracking image of the offensive and defensive teams against all the offensive and defensive strategies contained in the playbook, respectively.

The offensive and defensive strategies with the highest successful comparison percentage from the list are identified as the strategies performed by the offensive and defensive teams. This percentage is determined by the confidence match of the converted tracking image and the strategy image from the playbook. However, if the highest percentage is lower than 65%, then AICO assumes that the playbook does not contain the strategy tracked. In this case, AICO sends an email to the coach to inform that a new strategy has been detected. At this point, the coaches can create a new strategy and send this new strategy to AICO. Then AICO will compare and match this new strategy in the play.

The output of this strategy detection step are two strategy images from the playbook, offensive and defensive, which match the converted tracking images obtained in the player detection step.

3.5 Strategy Result Detection

At this point, AICO can detect and track players, and determine the strategy used by both teams in a play. However, AICO does not know yet if the strategies used by each team were successful or not. For this purpose, AICO needs to count the downs of the offensive team and calculate how many yards the offensive team has gain/lose in each down.

Regarding the count of downs, AICO needs to identify if the team continue being the same in the next play, if that is the case, the down counter of that team is increased. If it is not the same team, the possession of the ball has changed, so the down counter needs to be set to 1 and the ball position is stored as *initial ball* position. This *initial ball* position is used at the end of this step to determine which strategy is successful in each down.

To detect the team, AICO uses the helmet color of the *Center* player detected in the player detection step. AICO stores the color of the helmet and compares it to the helmet color of the *Center* player in the previous *clip play*. In case the color is the same, then the offensive team has not changed and the down counter is increased by 1. Otherwise, the offensive and defensive team has switched roles.

To determine the yards gain by a team, AICO checks the ball position of the *play clip* and compares how many yards has moved from the previous *play clip*. AICO uses the orientation of the offensive team to determine if the yards gained were positive or negative from the previous *play clip*.

AICO uses the yards calculation obtained by the offensive team, the down counters and the *initial ball* position to determine if the strategy used in the *play clip* was successful or not. Therefore, according to Football Outsiders, AICO defines an offensive strategy as

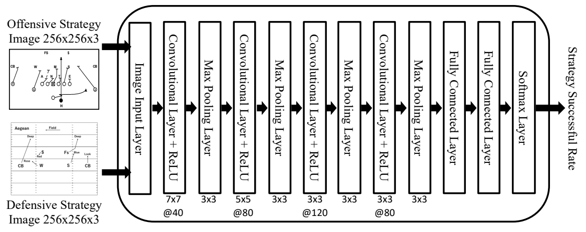


Figure 9: The CNN structure of AICO

successful if it gains at least 4 yards (40%) for the first down, 6 yards (60%) for the second down and 10 yards (100%) for third or fourth down compared to the *initial ball* position. Otherwise, the offensive strategy is defined as *unsuccessful*. Regarding the defensive strategy used in that *play clip*, it is consider *unsuccessful* when the offensive strategy is labeled as *successful* and vice versa.

After all the *play clips* of a whole match have been analyzed, AICO asks the coaches to give the names of both teams that participated in the match. AICO identifies both teams using the color of the player's helmets. Lists of all the offensive strategies used by each team are stored. There is one list per team. These offensive strategy lists are used later by the coaches to select the offensive strategy that they want AICO to compare against their defensive strategy.

3.6 AICO

AICO adopts the CNN structure shown in Figure 9 which has 12 deep neural network layers: one image input layer, four pairs of convolutional and max pooling layers, two fully connected layers and one softmax layer.

The filter size and the number of filters of each convolutional, ReLU and Max Pooling layers are set by parameter fine-tuning, shown below them in the figure. For example, the first convolutional layer has a 40 filter of $7x7$ size. Both fully connected (FC) layers multiply the input by a weight matrix and then adds a bias vector. The first FC layer uses a weight matrix of $50x1280$ numbers and a bias vector of 50 numbers. The second FC layer uses a weight matrix of $18x50$ and a bias vector of 18 numbers. The last layer is a softmax layer that uses the softmax function, also known as the multiclass generalization of logistic regression [Gho18]. Last but not least, AICO CNN structure output is the success rate of the strategy input against the other team.

The dataset utilized for training this CNN contains information gathered in the *preprocessing step* from several *play clips* of the target team, containing only offensive *play clips* of that team. Every dataset entrance contains the offensive strategy image used by the target team in a *play clip*, the defensive strategy image used against the target team in the same *play clip*, and a label confirming if the offensive strategy was *successful* or not.

Since AICO needs to be trained individually for each opponent team, there will be a specific AICO per every opponent team. For instance, if the coaches want information from 3 teams, there will be three different AICOs trained using different datasets related to each team.

After AICO has finished its training and now it can be used by the coaches. AICO provides a list of offensive strategies used by a team and the neural network trained for that specific team. This neural network has two inputs: an offensive strategy and a defensive strategy image with a size of $256x256$ matrix. As consequence, both images will be resized before they are input in the CNN. The offensive strategy image is an image selected previously by the coaches from the offensive strategy list of the target team. The defensive strategy image is an image of a defensive strategy provided by the coaches.

In summary, AICO examines the defensive strategies received from the American football coaches against the offensive strategy of the target team, and returns the effectiveness of using this defensive strategy against that offensive strategy of the opponent team.

4 EXPERIMENTS

AICO performance has been tested using real-world American football videos. Since we could not find available public datasets, we requested to KSS Lotus team to provide us American football video matches from diverse teams.

KSS Lotus team has supplied videos from 5 different teams, which are considered as opponents, playing against other teams. There is a total of 10 matches per opponent team. These 10 matches are used as a ground truth due to we know in advance the strategies used in each play by each team. A total of 5 additional matches where KSS Lotus team played against that 5 teams, one match per team, together with the respective strategies, has been provided as well.

The 50 matches of the opponent teams together with the strategies used by both teams in each *play clip* were used to train AICO. The matches of KSS Lotus team are used to verify AICO's strategy prediction accuracy. All of these 55 matches has been recorded using a Sony FDR-AX60 video camera with a quality of 720p.

KSS Lotus coaches segmented the 55 match videos into *play clips*, and we grouped them in a dataset. In total, the dataset contains about 7350 *play clips*, where 6700 are *play clips* from the 5 opponent teams and 650 are the *play clips* where KSS Lotus team is playing against one of the 5 opponent teams. Based on

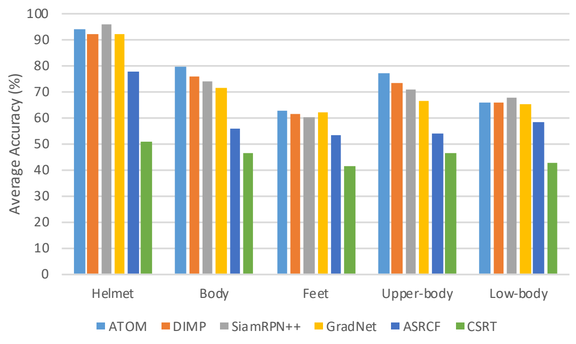


Figure 10: Tracking accuracy comparison using different parts of the player's body

the video matches used in the experiments, an American football team performs around 65 offensive and 70 defensive plays per match. This number can vary if one of the teams is stronger than the other team in the match. As a result, the dataset created contains about 650 offensive and 700 defensive plays of each opponent team, no counting the match against KSS Lotus team. The 6700 *play clips* are used to examine AICO performance on all the experiments together with the playbook strategies, however, the 650 *play clips* where KSS Lotus team played as well as the playbook are used to verify the defensive strategy selection effectiveness. This dataset containing all the *play clips* will be made publicly available in future projects.

4.1 Tracking Accuracy

Due to the movement of the players, a tracker can jump from one player to another, since one player is overlapped by another player in the video. As a consequence, one player may have two trackers, having lost the other player his tracker. It makes difficult to determine the strategy used by the teams since the tracker is not following the correct player. For this reason, it is necessary to achieve a high tracking accuracy.

In this experiment, we have used 100 *play clips* where we manually defined the position of each player as a ground truth. After having these ground truth videos, we have tested the following trackers from the *seventh visual object tracking vot2019 challenge* [Kri19]: ATOM, DIMP, SiamRPN++, GradNet and ASRCF. Each of this trackers has been trained to track American football players using the datasets created for the players detection. We have tested as well the CSRT tracker [Luk17] provided by OpenCV. Additionally, these trackers has been tested using different parts of the players body: helmet, body, feet, upper-body and low-body.

Figure 10 shows the tracking accuracy of the trackers tested using different parts of the player's body. This tracking accuracy is obtained by comparing the ground truth movement of a player together with his tracked movement. As we can see in the results, the player's helmet is the part of the body that obtained the highest tracking accuracy, and the best tracker is SiamRPN++ achieving around 96.075% accuracy. For this reason, AICO tracks the helmet of each player using SiamRPN++.

4.2 Neural Networks

In this research, the neural networks utilized are required to achieve a high accuracy performance. We compared the performance of the following CNN classifiers: Inception [Iof15], Resnet [Kai16] and Mobilenet [How17]. Regarding Inception models, they have been tested using the four versions currently available (*Inception_v1*, *Inception_v2*, *Inception_v3*, *Inception_v4*), and *Inception-ResNet-v2* which is a hybrid inception module using Resnet and has a similar computational cost than *Inception_v4*. Additionally, the following deep residual networks(ResNets) are also tested: *Resnet_v1_50*, *Resnet_v1_101*, *Resnet_v1_152*, *Resnet_v2_50*, *Resnet_v2_101*, *Resnet_v2_152* and *Resnet_v2_200*. Last but not least, diverse configurations of Mobilenet with 224 as input resolution have been added to the experiments: *Mobilenet_v1*, *Mobilenet_v1_075*, *Mobilenet_v1_050* and *Mobilenet_v1_025*. All these models are built upon TensorFlow GPU.

To the best of our knowledge, there are not any datasets available to detect American football balls, player's helmet or the players in the field. For this reason, we have created 3 different datasets using images and videos provided by KSS Lotus team from season 2013 to 2018. These datasets are composed of 2000, 15000 and 20000 images of balls, helmet and players, respectively, and they will be publicly available for everyone in the future. For the experiments, these datasets have been used to train each CNN classifier.

Figure 11 shows the average accuracy of detecting the ball, the helmet and the player using different neural networks. Inception_v4 achieves the highest accuracy (92.41%) on detecting the players in the field compared to the other CNN classifiers. Regarding the ball and helmet detection, the best option for both is Inception_v2 since it always detects them and has the fastest inference time, compared to other CNN that achieves the same accuracy. Since the ball is always detected, AICO can always define the *goal line* that is utilized to determine if the offensive strategy was *successful* or not.

As a result, AICO uses two Inception_v2 to detect the helmet and the ball, and one Inception_v4 to detect the players in the field.

4.3 Strategy Selection Accuracy

In the experiments, AICO has used the KSS Lotus playbook to detect the most similar strategies performed by each team in a *play clip*. This detection mechanism is similar as the coaches made in real life (Figure 1). After comparing the strategies selected by AICO to the strategies selected by KSS Lotus coaches in each *play clip*, we have verified that the 8% missing recognition of the players in the field made by Inception_v4 does not affect the detection of the correct strategy. As a result, about 95.65% and 90.58% of the times AICO recognizes the same defensive and offensive strategies as the coaches, respectively. Regarding the offensive strategies, 88.58% and 92.58% are achieved for running and passing offensive strategies, respectively.

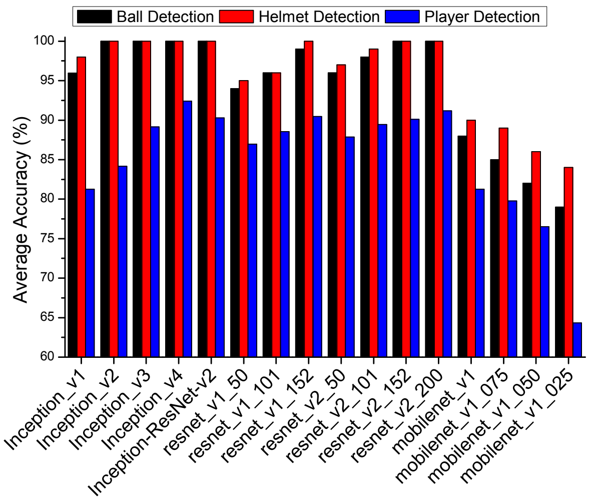


Figure 11: Average accuracy of detecting the ball, the helmet and the players using different CNN classifiers

4.4 Strategy Prediction Effectiveness

For this evaluation, the process is divided in three steps. First, AICO is trained using different amounts of an opponent match videos. In the second step, we examine the KSS Lotus team match against this opponent. In this analysis, we obtain the defensive strategies used by KSS Lotus team together with the successful rate of each defensive strategy used per offensive strategy used by the opponent team. For example, if KSS Lotus team performs the defensive strategy *A* 4 times against the offensive strategy *B* of the opponent team, being 3 times successfully defended, then strategy *A* has achieved 75% of successful rate.

In the third step, we input to AICO the strategy image of *A* together with the strategy image of *B*, and then AICO returns the successful rate of using *A* against *B*. For the experiment results, it is compared the successful rate of *A* against *B* obtained by AICO to the one obtained in the second step.

Figure 12 shows a successful rate example of using a some defensive strategy, such as *Zombie*, against to an offensive strategy like *Crunch_Read*. In blue it is represented the successful rate obtained by KSS in the

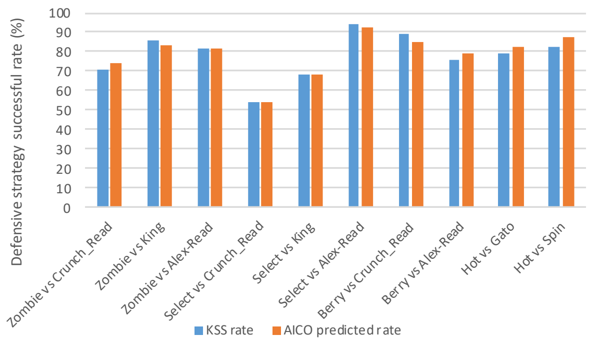


Figure 12: Example of some defensive strategy successful rate accuracy against offensive strategies

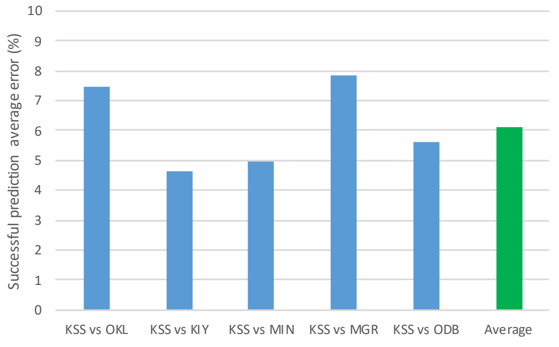


Figure 13: Successful prediction average error made by AICO using the 5 matches where KSS is playing

match, and in orange the successful rate predicted by AICO.

As we can see, AICO's successful rate matching up with that of KSS indicates the high performance level of AICO to predict how well a defensive strategy will perform in the real world.

Figure 13 shows the average error of predicting the successful rate using 5 matches of KSS playing against different opponents. AICO achieves 6.11% of error prediction, confirming that AICO can help the coaches to predict how successful will be a defensive strategy against to a particular offensive strategy

5 CONCLUSIONS

This paper presents a novel video recognition system working together with a neural network structure, AICO, which has been developed to analyze the opponent team strategies to successfully obtain the best defensive strategies to use against that adversary. Video analysis task made by American football coaches may become easier and more precise thanks to AICO.

In addition, during this research, we have created a dataset that contains about 7350 *play clips* based on different American football teams which will become publicly available in future projects.

This video analysis recognition system can work with any type of recording, even if only a single camera has been used, and it can easily be used by any American football team at any field, since it does not require previous installation in the stadium, field or players.

AICO achieves a 93.12% of accuracy in strategy detection of both teams in a play compared to the coaches judgment. Furthermore, AICO obtains a successful rate prediction with around 94% of accuracy determining the successful rate of a defensive strategy against an offensive strategy of an opponent team. As a result, AICO becomes a reliable artificial coach advisor to help coaches in their opponent strategy analysis.

ACKNOWLEDGEMENT

Thank you very much to Fujitsu and Kosei Gakuen High School American football (KSS Lotus) for their support and help to develop AICO.

6 REFERENCES

[Atm13] I. Atmosukarto, B. Ghanem, S. Ahuja, K. Muthuswamy, and N. Ahuja. Automatic recognition of offensive team formation in american football plays. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 991–998, 2013.

[Bal87] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. In *Readings in computer vision*, pages 714–725. Elsevier, 1987.

[Bur17] M. Buchheit and B. M. Simpson. Player-tracking technology: half-full or half-empty glass? *International journal of sports physiology and performance*, 12(s2):S2–35, 2017.

[Che14] S. Chen, Z. Feng, Q. Lu, B. Mahasseni, T. Fiez, A. Fern, and S. Todorovic. Play type recognition in real-world football video. In *IEEE Winter Conference on Applications of Computer Vision*, pages 652–659. IEEE, 2014.

[Chu03] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003.

[Dir18] C. Direkoglu, M. Sah, and N. E. OConnor. Player detection in field sports. *Machine Vision and Applications*, 29(2):187–206, 2018.

[Ela08] E. Dubrofsky and R. J. Woodham. Combining line and point correspondences for homography estimation. In *Advances in Visual Computing*, pages 202–213, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Fre19] M. Frey, E. Murina, J. Rohrbach, M. Walser, P. Haas, and M. Dettling. Machine learning for position detection in football. In *2019*

6th Swiss Conference on Data Science (SDS), pages 111–112, June 2019.

[Gho18] J. Ghosh, Y. Li, R. Mitra, et al. On the use of cauchy prior distributions for bayesian logistic regression. *Bayesian Analysis*, 13(2):359–383, 2018.

[Goo18] R. Goodell. Official playing rules of the national football league. *National Football League: New York, NY, USA*, 2018.

[Har03] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[How17] A. G. Howard, Z. Menglong, C. Bo, K. Dmitry, and W. Wang. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[Iof15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML '15*, 2015.

[Jia16] H. Jiang, Y. Lu, and J. Xue. Automatic soccer video event detection based on a deep neural network combined cnn and rnn. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 490–494. IEEE, 2016.

[Jof15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[Kae16] A. Kaehler and G. Bradski. *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. " O'Reilly Media, Inc.", 2016.

[Kai16] K. He, H. Kaiming, Z. Xiangyu, R. Shaoqing, and J. Sun. Identity mappings in deep residual networks. In *ECCV '16*, 2016.

[Kri19] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. Cehovin Zajc, O. Drbohlav, A. Lukezic, A. Berg, et al. The seventh visual object tracking vot2019 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[Lee16] N. Lee and K. M. Kitani. Predicting wide receiver trajectories in american football. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.

[Li19] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[Luk17] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and M. Kristan. Discriminative correlation filter with channel and spatial reliability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6309–6318, 2017.

[Out19] Football outsiders, 2019. URL `https://www.footballoutsiders.com/`.

[Rie13] W. Riedel, D. Guillory, and T. Mwangi. Ee368 project: Football video registration and player detection. 2013.

[Sid09] B. Siddiquie, Y. Yacoob, and L. Davis. Recognizing plays in american football videos. *University of Maryland, Tech. Rep*, 111, 2009.

[Ste17] M. Stein, H. Janetzko, A. Lamprecht, T. Breitkreutz, P. Zimmermann, B. Goldlücke, T. Schreck, G. Andrienko, M. Grossniklaus, and D. A. Keim. Bring it to the pitch: Combining video and movement data to enhance team sport analysis. *IEEE transactions on visualization and computer graphics*, 24(1):13–22, 2017.

[Sze17] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[Wy015] Y. Wu, J. Lim, and M. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, Sep. 2015.

[Yam13] T. Yamamoto, H. Kataoka, M. Hayashi, Y. Aoki, K. Oshima, and M. Tanabiki. Multiple players tracking and identification using group detection and player number recognition in sports video. In *IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society*, pages 2442–2446. IEEE, 2013.

[Zho14] W. Zhong, H. Lu, and M. Yang. Robust object tracking via sparse collaborative appearance model. *IEEE Transactions on Image Processing*, 23(5):2356–2368, May 2014.

Strategies for Training Deep Learning Models in Medical Domains with Small Reference Datasets

Gerald A. Zwettler
RG Advanced Information Systems and Technology (AIST), Department of Software Engineering, University of Applied Sciences Upper Austria
Softwarepark 11
4232 Hagenberg, Austria
gerald.zwettler@fh-hagenberg.at

David R. Holmes III
Biomedical Analytics and Computational Engineering Lab, Dept. of Physiology and Biomedical Engineering, Mayo Clinic College of Medicine
200 1st St SW,
MN 55905, Rochester, USA
holmes.david3@mayo.edu

Werner Backfrieder
Medical Informatics, Department of Software Engineering, University of Applied Sciences Upper Austria
Softwarepark 11
4232 Hagenberg, Austria
werner.backfrieder@fh-hagenberg.at

ABSTRACT

With the steady progress of Deep Learning (DL), powerful tools are now present for sophisticated segmentation tasks. Nevertheless, the generally very high demand for training data and precise reference segmentations often cannot be met in medical domains when processing small and individual studies or acquisition protocols. As common strategies, reinforcement learning or transfer learning are applicable but coherent with immense effort due to domain-specific adjustment. In this work the applicability of a U-net cascade for training on a very low amount of abdominal MRI datasets of the parenchyma is evaluated and strategies to compensate for the lack of training data are discussed. Although the model accuracy when training on 13 MRI volumes with achievable JI=89.41 is rather low, results are still good enough for manual post-processing utilizing a Graph cut (GC) approach with medium demand for user interaction. This way, the DL models are retrained, when additional test data sets become available to subsequently improve the classification accuracy. With only 2 additional GC post-processed datasets, the accuracy after model re-training is increased to JI= 89.87. Besides, the applicability of Generative Adversial Networks (GAN) in the medical domain is evaluated discussing to synthesize axial CT slices together with perfect ground truth reference segmentations. It is shown for abdominal CT slices of the parenchyma, that in case of lack of training data, synthesized slices, that can be derived at arbitrary number, help to significantly improve the DL training process when only an insufficient amount of data is available. While training on 2,200 real images only leads to accuracy JI=88.75, the enrichment with 2,200 additional images synthesized from a GAN trained on 5,000 datasets only leads to an increase up to JI=92.02. Even if the DL model is exclusively trained on 4,400 computer-generated images, the classification accuracy on real-world data is notable with JI=90.81.

Keywords
Medical Image Segmentation, Deep Learning, Generative Adversial Networks, Graph cut.

1. INTRODUCTION

The research and development on preferably automated, generic and precise segmentation strategies for processing medical image datasets has been of high importance since the very first computed tomographic image acquisition devices in the early 1970s. Since then many semi-automatic segmentation concepts have been proposed, but most of them conserve the medical diagnostician and its experience-based evaluation as central criterion for the final decision. While there are some few off-the-shelf applications available for specific diagnostic domains [Chr18], in general computer-aided diagnostics is still achieved in a user-centric process utilizing tools and frameworks for semi-automated image processing [Str15].

1.1 Field of Medical Application

Whenever quantitative evaluation is required for computer-based planning of a surgery, evaluating the success of the therapy or progress of a degenerative disease in a follow up study [Agg11], a precise segmentation of the target structures is inevitable. With emerging progress in 3D visualization with respect to the mixed reality continuum and the 3D print of anatomical structures as 3D models for surgery planning, training and education [Squ18] the fields of application for medical image processing are steadily growing.

A broad range of semi-automated image processing tools is available for user-centric processing in particular diagnostic domains. Nevertheless, evaluation of datasets with these tools requires a high level of experience in both, the medical and the technical domain to prevent from misapplication, errors and subjective bias of the results. To close this gap, the application of standardized image processing chains for processing medical tomographic datasets is recommended in radiographer training [Zwe13].

1.2 State of the Art

Since the beginnings of computer-based medical image analysis, besides semi-automatic and user-oriented methods such as Region Growing, Graph cut or Live-Wire [Son13], shape-modeling approaches have been a strategy for automation of the segmentation process in specific medical domains. Rudimental a priori knowledge regarding shape is incorporated for deformable models [McI96] or Level Set segmentation [Set99] while statistical shape models [Coo92] are trained on a large dataset of reference shapes representing the expected anatomical variability. While statistical shape models lead to a domain-specific compact representation of shape due to PCA, the generic use is still hard to achieve as extraction of corresponding landmarks is very domain-specific and difficult for non-prominent structure shapes. Incorporating the expected intensity profile besides shape makes Active Appearance Models [Coo98] relevant in several diagnostic domains but their most significant field of application is still human face recognition.

Advances in available GPU hardware and machine learning frameworks such as Tensorflow/Keras lead to an immense boost in Deep Learning (DL) applications, facilitating practical use of formerly only theoretically specified concepts. Conventional Feed Forward networks, already applied in medical domains such as multi-modal image fusion [Zha11] are steadily enriched by an increasing number of hidden layers thus increasing the overall number of trainable parameters. All common Deep Learning Concepts found their way and application in specific medical domain. For instance self-organizing maps neural networks [Koh97] that are due to their grid/graph nature good for complex clustering tasks, were successfully applied for classification of renal diseases too [Van98]. Advanced semantic interpretation of input signals is of high relevance in natural langue, optical character or audio processing and was significantly boosted by recurrent neural networks introduced as long/short-term-memories (LSTM) [Hoc97] allowing incorporation of historical contextual aspects for an increased classification accuracy. A key trigger for the technological progress of Deep Learning is the development of convolutional neural networks (CNN). Instead of expert or machine driven feature selection in the machine learning domain, with convolutional networks the search for domain-specific and adequate features is now handled within the training process. The possible multi-resolution convolution pyramids and depth of 1200 layer for some CNN structures thereby significantly outperform classic convolution approaches such as Haar Cascades [Vio01] of sequentially applied weak classifiers. Another key development in Deep Learning is generative adversial networks (GAN) [Goo14] opening up totally new domains of application. GANs thereby are composed of a classification (discriminator) and a convolution network (generator) both alternatingly trained. While the generator tries mimicking the given reference samples with synthesized data, the discriminators loss indicates if the fake and real data are differentiable. As fields of application, the synthesis of handwritings, paintings, and medical data [Yi19] or general enrichment of the training data to prevent from over-fitting [Fri18] are of relevance.

1.3 Related Work

With few datasets available, i.e. around 10-20 volumes only, common Deep Learning models generally cannot be trained to highest accuracy. Applying drop-out and data augmentation strategies, the risk for over-fitting can be significantly reduced [Gao19]. Besides data augmentation, the generation of synthesized data is another strategy for enriching the reference database as successfully shown for liver segmentation [Yan17]. Nevertheless, due to the black-box nature of Deep Learning models, dealing with results below required accuracy is difficult.

A marker-based U-net model was proposed in [Sak19] that is simultaneously trained on both, the automatic classification of the target anatomical structure and to thereby obey markers optionally placed by the user for a post-processing classification run. While training a correction mechanism together with the net unmasks much of accept as is nature of Deep learning models, the problem of insufficient data for proper training still remains.

Another approach to incorporate the human expert in the computer-based diagnostics is to interpret the DL computer outcome as a “third eye” [Fou19], i.e. using deep learning recognition algorithms to solely improve visual diagnostics in medicine.

1.4 Training Deep Learning Models on Small Training Datasets

In this work the applicability of Graph cuts, a segmentation concept known since two decades, is evaluated in a totally different context, namely as post-processing for segmentations of an insufficiently trained Deep Learning model. That way, the black box nature of DL models is overcome and adequate correction of the results by experts becomes possible in a semi-automated way. The test data classifications significantly improved and validated that way by human experts in image processing are then used to enrich the data basis for model retraining. Thereby, a generic strategy for allowing expert-centered post-processing of DL models is introduced allowing to correct the seldom cases where obvious segmentation errors are introduced.

Furthermore, a strategy for synthesizing artificial medical image datasets together with highly correlating reference segmentations is evaluated. The medical fake images are mixed with real data at different mixing ratios and used for training a U-net cascade. As GANs generally require a huge amount of input data for training, a chicken-egg problem would arise. Thus, in this work it is shown that even GANs trained on a low amount of reference images can synthesize images that help to train DL models.

2. MATERIAL

As diagnostic domain in this research work, the parenchyma from CT and MRI imaging modality is chosen. The 131 CT datasets from the Medical Segmentation Decathlon [Sim19] with provided ground truth expert reference segmentations are used for training a GAN as well as for evaluations of real and synthesized data on the training process of a U-net cascade [Zwe20] with the class *liver* and *tumor* being merged. Graph cut based post-processing of classification results from a weakly trained U-net cascade are evaluated on 20 parenchyma MRI datasets in axial CAIPIRINHA non-contrast, breath-holding fat-suppressed AX CAIPI VIBE FS protocol [Mor15]. From the MRI datasets, 10 are without and 10 with Hepatocellular Liver Lesions (HCC). In case of a lesion, this class is merged with the parenchyma region leading to a binary classification problem. The MRI datasets are segmented using spline tracing (live-wire) and auto-tracing (region growing) available from Analyze software [Rob98].

2.1 Data Preparation and Pre-processing

For both, CT and MRI datasets, the same pre-processing strategies are applied. To balance the varying slice thicknesses, the z-spacing is adjusted to the x/y inter-slice spacing using cubic interpolation for the intensity dataset and binary shape interpolation [Raj03] for the reference segmentations. To limit the extent of the input slices, an area of 352×288 pixels, referring to an average axial width-to-height aspect-ratio, is extracted based on the segmentation ROI. To allow for data augmentation during the training process, a safety margin of 10px along the borders is applied, leading to original slice extent of 372×308 pixels.

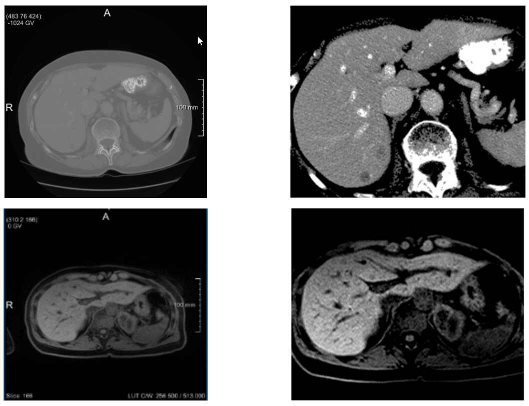


Figure 1. Slice Pre-processing for CT (top row) and MRI (bottom row).

To normalize the intensity profile w.r.t the target anatomical structure, μ_{liver} and σ_{liver} are evaluated per dataset based on the segmentation mask. An intensity transform similar to windowing is applied to shift the mean object intensity to 127.0 per dataset applying a scale factor $s = \frac{115}{3 \cdot \sigma_{liver}}$, see Eqn. 1.

$$\mathrm{T}(a_i) = \begin{cases} MAX(127 - |a_i - \mu_{liver}| \cdot s, 0) & a_i \leq \mu_{liver} \\ MIN(127 + |a_i - \mu_{liver}| \cdot s, 255) & a_i > \mu_{liver} \end{cases} \quad (1)$$

Preprocessing on slice #100 for the first CT and MRI volume respectively with size restriction to 372×308 pixel and intensity transformation is shown in Fig. 1.

3. METHODOLOGY

The Deep Learning Network to be used in this paper for validation on low or synthesized data is a U-net [Ron15] applied as cascade with combining axial, sagittal and coronal views [Zwe20], see section 3.1. Although the Deep Learning model processes the input in a 2D slice-wise manner, results of Jaccard Index JI=.9529 and Dice Coefficient DC=.9759 are achievable with slice-mini-batch optimization and training on solid 22,000 images from 100 volumes.

3.1 U-net Cascade for Liver Segmentation

The input axial images are transformed to sagittal and coronal view with the 2D slices then each classified by a separately trained U-net, see Fig. 2.

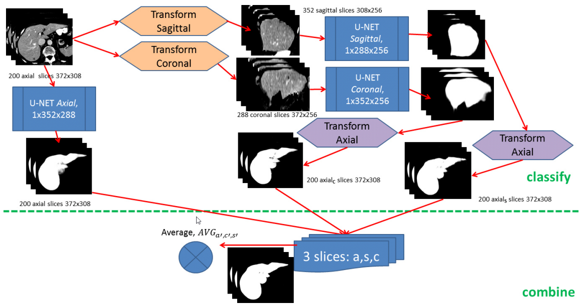


Figure 2. Slice-wise processing in axial, sagittal and coronal view with final results as average.

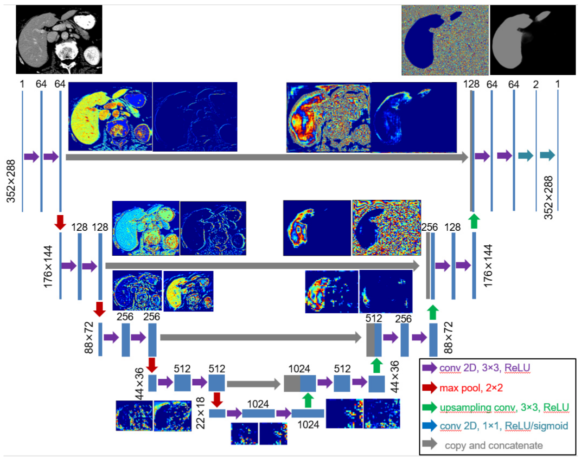


Figure 3. U-net adapted from [Ron15] with exemplary kernel output per hierarchy level.

The U-nets thereby expect an unsigned char single channel 352×288 image as input and incorporate 176×144, 88×72, 44×36 and 22×18 resolution hierarchies for the classic encoder/decoder pattern, see Fig. 3. For matter of illustration, in Fig. 1 kernel output feature images are visualized at different levels of hierarchy.

For training, an Adam Optimizer [Kin14] with learning rate $lr = 5 \cdot 10^{-6}$ using *cross-entropy* as loss is utilized to train the model at a $batchSize = 32$ and $patience = 12$ up to *200* epochs.

3.2 Data Augmentation

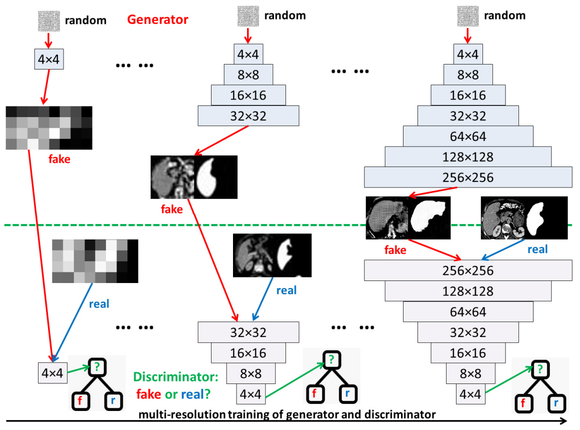
A general key strategy for training is application of data augmentation, thereby slightly varying the input data. With data augmentation over-fitting can be reduced not only if training on small datasets but in general. For this research work, the following geometric and intensity transformations are applied on the input data at random scale:

- *transX* and *transY*: translation in x-direction and y-direction of the current slice
- *rot*: rotation around the image center
- *intMul*: linear scale of the image intensities leading to brighter or darker pixel values
- *intAdd*: additive manipulation of the intensities within the window, leading to a uniform shift for full scalar range

For training of the U-net cascade as delineated in section 3.1 and [Zwe20], the augmentation scale is randomly varied within range $[16, 16, 10, .1, 30]$ for *transX, transY, rot, intMul* and *intAdd* respectively.

3.3 GAN for Medical Image Synthesis

A GAN architecture with Generator as convolutional neural network fed by random input and a Discriminator as binary classificatory (fake or real) is chosen.



random
Generator
4×4
8×8
16×16
32×32
64×64
128×128
256×256
fake
real
Discriminator: fake or real?
multi-resolution training of generator and discriminator

Figure 4. Adapted GAN architecture [Ker17].

The GAN is trained in a progressive way using 7 levels of resolution from 4×4 to 256×256, see Fig. 4 for illustrated levels 1 (4×4), 4 (32×32) and 7 (256×256). For training, an *AdamOptimizer* with a learning rate of 0.00015 and an iteration count *iter* increasing with the hierarchy level l as $iter = 20{,}000 + l * 32{,}000 + l^2 * 3{,}200$ is chosen, thus leading to iterations between [55.200;400.800]. As input for the Discriminator 5,000 CT slices non-overlapping with the test-set are randomly arranged in batches of size 2 with data augmentation applied. To resample the input size between 256×256 used for the GAN and 352×288 used for the U-net cascade, bilinear interpolation is applied. An important aspect is the synthesis of an accurate ground-truth reference mask besides the fake medical images. Thus, the single-channel input tensor of size 2×256×256×1 with values in [0.0;1.0] is extended to 2×256×256×2 with the associated reference mask as additional channel.

3.4 Graph cut Post-Processing

To allow for user-guided post-processing of the DL results, a fitness function for N_4 Graph cut processing combining both, original image properties and DL segmentation results is required, namely:

- *ORIG*: horizontal (H) and vertical (V) edges of the original intensity profile after applying intensity shift, c.f. Equ. 1.
- *EXP*: ORIG damped / amplified by a difference image from the expected intensity level after smoothing (median $r = 1$ and Gauss $r = 5, \sigma = 2.5$).
- *S1* and *S4*: H/V edges from the binary segmentation results from axial, sagittal, coronal and combined with 1 and 4 hits per voxel respectively.

Besides S1 and S4, the 2- and 3-hit cases are omitted due to lack in entropy. The cumulated fitness function is composed by applying Equ. 2 with function *s()* scaling to $[0; w_i]$ and the weights optimized via Evolution Strategy.

$$\max\begin{pmatrix} s(ORIG_H, w_0), s(EXP_H, w_1), \\ s(S1_H, w_2), s(S4_H, w_3) \end{pmatrix} \tag{2}$$

For Evolution Strategy optimization of the weights, recombination (*μ/ρ+, λ*) with *epochs=100*, *batchSize=8*, *populationSize=8*, children *λ=32, mutationChance=0.4, mutationRate=0.25* dropping by 4% each epoch is applied.

To reduce the required user-interaction to a minimum, the FG and BG seeds are derived from the combined DL segmentation via skeleton calculation.

4. IMPLEMENTATION

Model training and testing is implemented in Python version 3.7.3 using Tensorflow 2.0 beta together with Keras. The Python image processing is built upon OpenCV and numpy for fast matrix operations. To provide the model with training data, a `DataGenerator` class is derived from `Sequence` base class providing data augmentation functionality.

5. RESULTS

For evaluation, the *Sørensen-Dice coefficient* (DSC) and the Jaccard index (JI) are calculated from reference mask R and result foreground region S of image I with $R \subseteq I, S \subseteq I$ and pixel $(x, y) \in R \cup S$. Additionally, the normalized surface distance (NSD) [Lap19] evaluates the FP and FN error based on the 3D distance-map based proximity to the next correct border position, see Equ. 3-5.

$$DSC(R,S) = \frac{2 \cdot |R \cap S|}{|R| + |S|} \tag{3}$$

$$NSD(R,I) = 1 - \frac{\sum_{x,y} [\![R_{x,y} \neq I_{x,y}]\!] \cdot D(R)_{x,y}}{\sum_{x,y} D(R)_{x,y}}, \quad D_{x,y}(R) = dist_{Euc}(surf(R)) \tag{4}$$

$$JI(R,S) = \frac{|R \cap S|}{|R| + |S| - |R \cap S|} \tag{5}$$

The process steps discussed in this paper, namely data preparation, pre-processing, GAN and U-net cascade model training and validation/test are performed on a *Colfax SX9600 GPU Rack* with 2×I*ntel Xeon Gold 6148 2.4GHZ* processors and 768GB of DDR4 memory with 2667MHZ clock frequency split into 24 partitions of 32GB each. The system runs *CentosOS 7.6* operating system and provides for fast tensor calculation 8 GPU cores, namely 4× *NVIDIA Volta Titan V 12G* and 4× *NVIDIA Tesla V100 32G*. For training and evaluation, parallelization was omitted and only one single Tesla core used in a sequential manner.

5.1 GAN Synthesis of Medical Images

Utilizing the GAN structure trained on 5,000 CT slices, 10,000 synthesized images are generated that meet the range criterion for cumulated discriminator and generator loss as $|D_{loss}| + |G_{loss}| \leq 160.0$, see Fig. 5 for images 10 (upper left), 20 (upper right), 30 (lower left) and 40 (lower right).

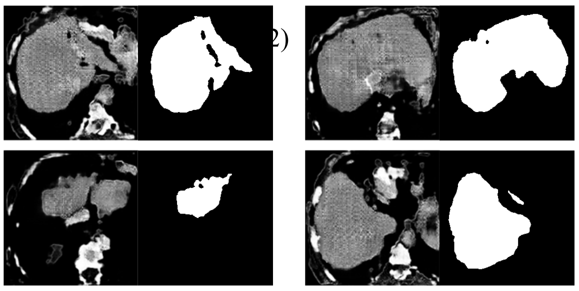


Figure 5. Synthesized Images with Reference.

The synthesized images show a slightly different intensity profile with μ_{real}=68.268 [26.52;106.70] and σ_{real}=16.420 for the 640 real and μ_{fake}=69.341 [26.827;104.022] and σ_{fake}=15.872 for the 640 fake images according to HYBRID_T in Tab. 1. The generated binary regions are of average size $\mu_{sizeRefREAL}$=3,836,278 [6,156;7,809,799] and $\mu_{sizeRefFAKE}$=3,678,637 [0;8.235,480] respectively. It seems that the loss-check for the data synthesis leads to an increase in caudal and cranial slices with smaller parenchyma cross-sections, areas that generally are weaker in axial U-nets and necessitate for additional training [Zwe20].

5.1.1 GAN Synthesis

For testing the applicability of GAN-synthesized data on training the U-net cascade, real and fake CT slices of the parenchyma are utilized for training and test as enlisted in Tab. 1. Training and validation of the U-net utilizes different real CT slices (#9000-15000) as used for the GAN training (first 5000) while then for test, the CT slices #22500-27000 are used for extraction of 640 slices. With respect to the generated fake data, the synthesized slices for U-net training and test are separated datasets too.

It is further evaluated, how different the images synthesized for FAKE_T according to Tab. 1 and the ones used for training are, i.e. that the random number generator utilized by Tensorflow does not lead to redundancy, see Fig. 6 for the first three generated images and their most similar images from the fake training database evaluating the difference images (mid) and the reference mask match (right).

purpose	dataset	#real	#fake
TRAIN	REAL_SMALL	2,200	0
	REAL	4,400	0
	FAKE	0	4,400
	HYBRID	2,200	2,200
	HYBRID_LARGE	2,200	6,600
TEST	REAL_T	640	0
	FAKE_T	0	640
	HYBRID_T	640	640

Table 1. Small train and test ensembles for validation on the axial U-net.

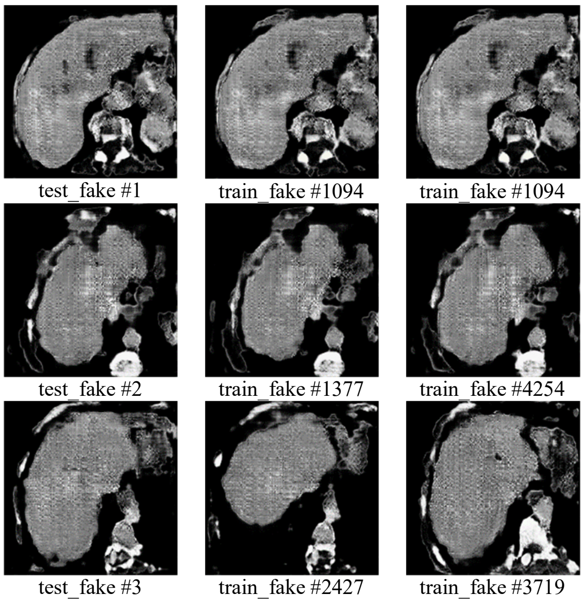


Figure 6. GAN synthesized image similarity.

As the same GAN model is utilized, depending on the random seeds the similarity at least implies, the slices might result from a same virtual volume.

Nevertheless, the accuracy of a potential model on the synthesized data is not as relevant as the difference in the real data from the synthesized samples, see Fig. 7. Here the GAN being trained on a distinctive dataset prevents from potential bias. For the real-data, separation into slices for train/validation/test that do not originate from the same volume is important. It ensures that not only the slices but the entire tomographic information is distinctive, very relevant to prevent from bias due to high z-resolution.

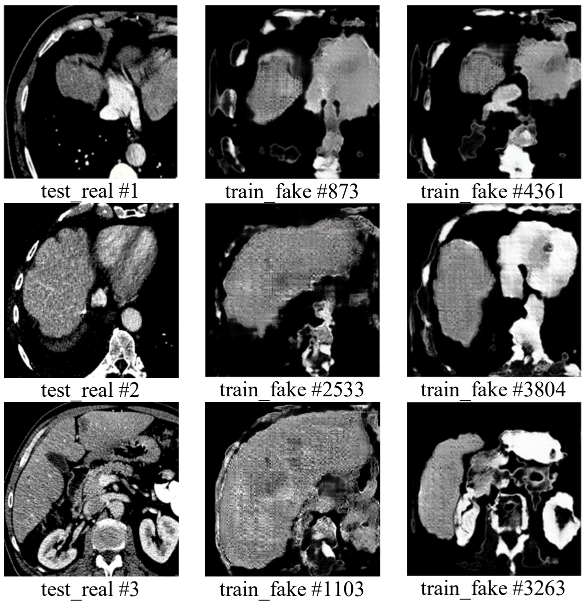


Figure 7. Similarity of real images with GAN-synthesized images in train and test.

model	dataset		
	REAL_T	FAKE_T	HYBRID_T
REAL	92.3572	89.9044	91.1111
REAL_SMALL	88.7514	87.7616	88.3140
FAKE	90.8081	95.7848	93.0345
HYBRID	92.0237	94.9955	93.4568
HYBRID_LARGE	92.9145	97.6974	95.1660

Table 2. JI on axial U-net with Synthesized Data from 5k GAN.

5.1.2 Training with synthesized data

With the proposed datasets, 5 axial U-net models are trained with the mean JI evaluated as percentage-accuracy on the 3 test datasets, see Tab. 2. With 4,400 compared to 2,200 datasets used for training, REAL significantly outperforms REAL_SMALL.

Comparing the performance on real data with the models HYBRID and HYBRID_LARGE it becomes obvious that synthesized data significantly boosts the training process. The model FAKE only trained on 4,400 synthesized data achieves solid JI=90.808 on the REAL_T test data. In Fig. 8 results of REAL_SMALL model (second row) and HYBRID_LARGE model (third row) on REAL_T test data for slices #1-3 (first row) with color-encoded FN (red) and FP (blue) are presented.

The training process for 97 epochs on the HYBRID_LARGE model is shown in Fig. 9. Due to heterogeneity of the data and applied data augmentation, the validation accuracy follows the trend of the test accuracy indicating no over-fitting. The loss is reduced from 0.7 to 0.18 within the first epoch and then drops approximately indirect proportional to the loss with a sink at epoch 46.

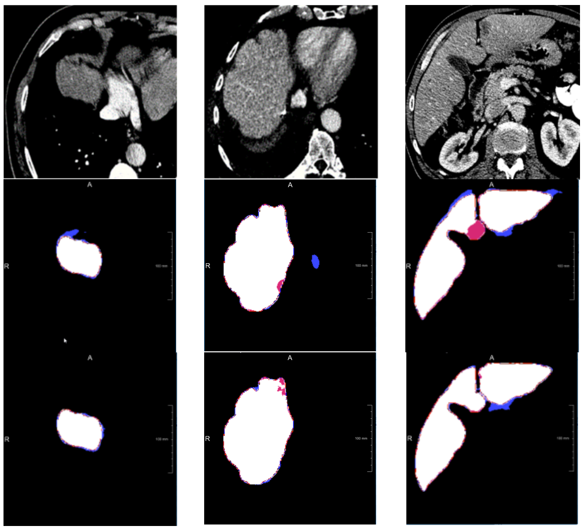


Figure 8. FN, FP and correct classification with HYBRID model on REAL_T test data.

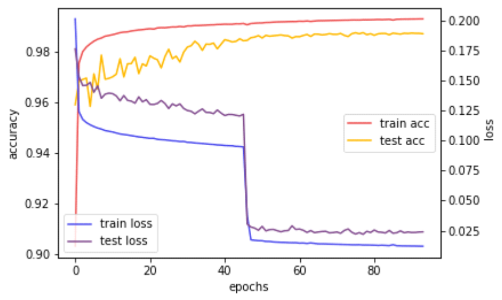


Figure 9. Loss and accuracy of train/validation per epoch when training HYBRID_LARGE.

5.2 Graph cut Post-processing

In this section, the applicability of Graph cut as post-processing tool for DL classifications as well the influence of enriched datasets on the training process is evaluated.

5.2.1 Training U-net cascade on MRI data

Training on 13 test datasets (#2-14, 7 without and 6 with HPC; 3,132 slices) gives solid results for the axial, sagittal and coronal models with significant boots when combining with 2/3 majority voting ($AVG_{a,c,s}$) for tests on 420 slices of datasets #1 and #14 (1 with and 1 without HPC), see Tab. 3.

model	DSC	JI	NSD
ax	92.1154	85.3834	92.3271
ax_s	91.9219	85.0513	88.6966
ax_c	92.5392	86.1144	93.2273
$AVG_{a,c,s}$	94.4096	89.4111	97.0472

Table 3. U-net cascade trained with 13 datasets in axial, sagittal, coronal and combined $AVG_{a,c,s}$.

5.2.2 Graph cut for post-processing

Utilizing evolution strategy, after 100 epochs evaluating on the CT parenchyma data with automatic application of the Graph cut post-processing, the fitness-function weights get optimized to w_0=0.287, w_1=0.217, w_2=0.419, w_3=0.641, c.f. Fig. 10.

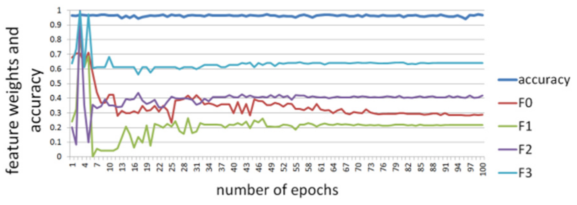


Figure 10. Evolution Strategy-based optimization of the Graph cut fitness function.

With only the weight ratio relevant, w_3 for the cumulated DL result is of highest importance, while the original image aspects (w_0, w_1) still allow for adaption to the image profile, see Fig. 11 for original images, combined DL segmentation and the horizontal edges derived from the fitness function for slices 30, 100 and 190 of MRI dataset #1.

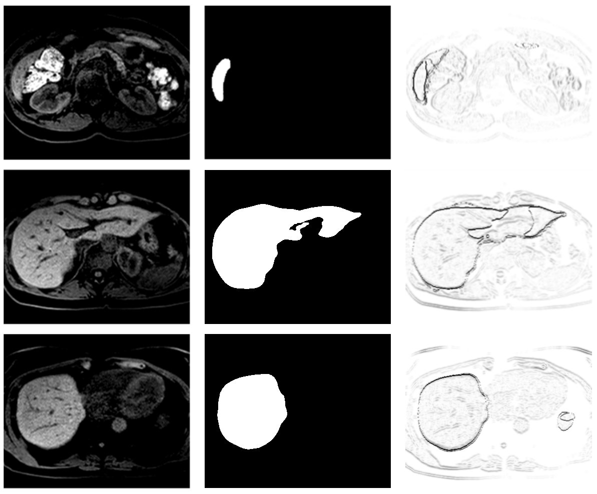


Figure 11. Horizontal GC fitness image.

With the FG and BG seeds automatically derived from the combined segmentations, the required amount of user interaction is kept to a minimum; see Fig. 12 for skeleton, GC result and mismatch for slices 30, 100 and 190 of MRI dataset #1.

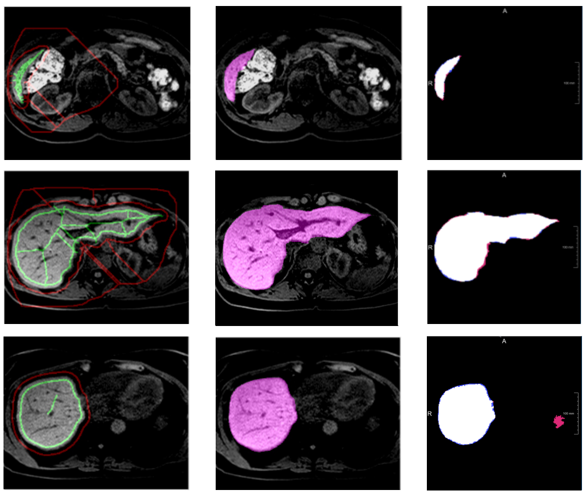
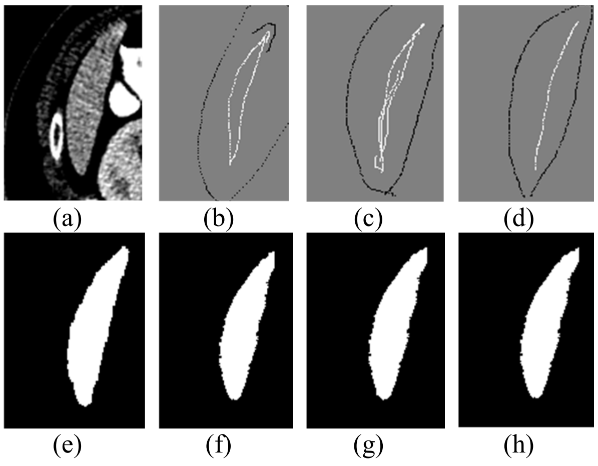
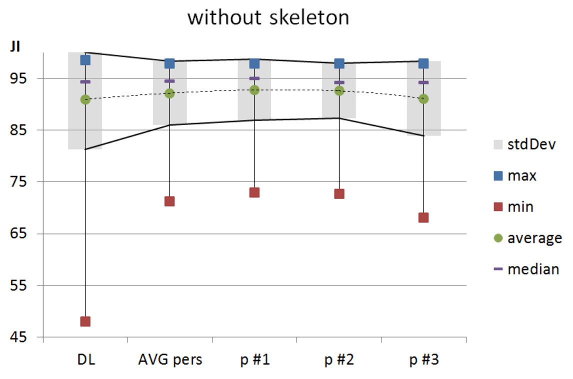


Figure 12. GC segmentation based on skeletons.

To allow for tests with enriched datasets, the remaining 5 MRI datasets (2 without and 3 with HPC; 1,238 slices in total) are segmented by a medical expert utilizing Live-wire (LW) to provide rough reference segmentations. From these 5 datasets, two (484 slices) are classified by the U-net cascade ($AVG_{a,c,s}$) as described in sections 3.1 and 5.2.1. The two datasets thereby only achieve JI=82.1644, DSC=90.2091 and NSD=91.5809. After the GC post-processing, the accuracy is increased to JI=88.8186, DSC=94.0782 and NSD=98.1452 with the semi-automated processing lasting for 152min (18.8sec per slice on average). To evaluate robustness of the Graph cut post-processing step, three experts evaluate the same subset of n=30 randomly selected slices. Although the FG skeletons placed by the experts in a manual way vary, the segmentation outcome is very stable, see Fig. 13.





(i)

Figure 13. Slice #28 robustly segmented by three medical experts utilizing GC with different FG and BG skeletons.

Slice #28 without skeleton support (a) and expected ground truth (e) shows suboptimal DL result JI=.877 and can be improved by all test persons (f-h) in range [.911;.920] even with very different GC skeleton interpretations (b-d). Utilizing GC for manual DL post-processing, potential invalid DL results get improved by test persons p#1-#3 and variability of the JI accuracy is generally decreased, see Fig. 13 (i).

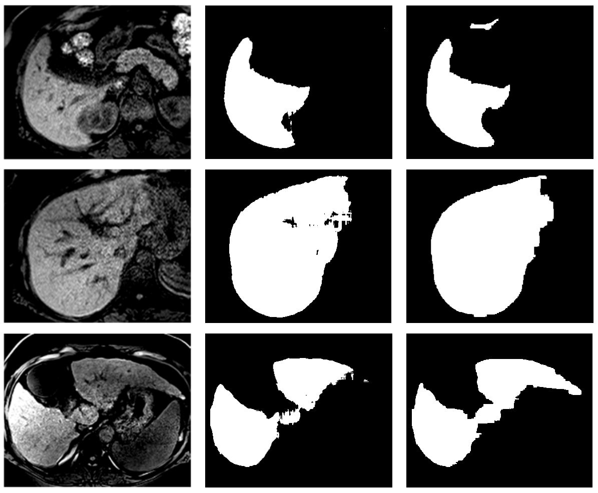


Figure 14. MRI add-on data with original, result from $AVG_{a,c,s}$ and result from Graph cut.

As a rough Live-wire segmentation with immanent inaccuracies is used as base for comparison, the accuracies before and after GC correction allow for relative comparison only, see Fig. 14 comparing original slice, initial result from $AVG_{a,c,s}$ and after GC correction for add-on slices #100, 150 and 1100 respectively.

5.2.3 Re-training the U-net on enriched data

The training of the 4 models for the U-net cascade, as described in section 5.2.1, is re-run with enriched data. The initial 13 datasets thereby get enriched by 484 slices (1 dataset with and 1 without HPC) previously corrected by user-guided Graph cut post-processing, see Tab. 4. Furthermore, all 5 add-on datasets (1,238 slices) are used to directly train the U-net cascade based on the rough Live-wire segmentation provided, see Tab. 5.

model	DSC	JI	NSD
ax	93.2937	87.4303	96.0364
ax_s	92.1216	85.3940	85.8697
ax_c	92.9323	87.1230	93.4942
$AVG_{a,c,s}$	94.6671	89.8742	97.4724

Table 4. Results on Models trained with 13 datasets enriched by 2GC datasets

model	DSC	JI	NSD
ax	92.6168	86.2488	94.3139
ax_s	92.1746	85.4850	87.9913
ax_c	92.1341	85.4155	91.9131
$AVG_{a,c,s}$	94.5163	89.6028	97.0967

Table 5. Results on Models trained with 13 datasets enriched by 5 LW datasets.

A comparison of the achievable test accuracy for the models trained on the 13, the 13+2GC datasets and the 13+5 LW datasets is provided in Fig. 15.

A visual representation of the tomographic segmentation achievable by the $AVG_{a,c,s}$ model trained on 13 + 2GC is given in Fig. 16 with FP (blue) and FN (red) visualized for test dataset #1 with errors in cranial and caudal direction and high accuracy in the mid slices.

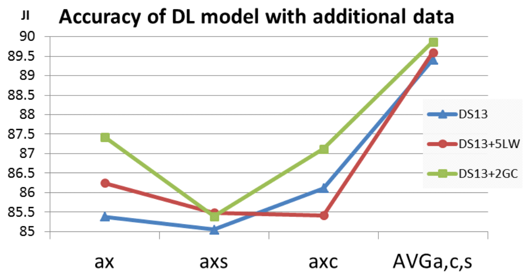


Figure 15. Achievable JI accuracy for DS13, 2GC add-on and 5LW add-on, c.f. Fig.12-14.

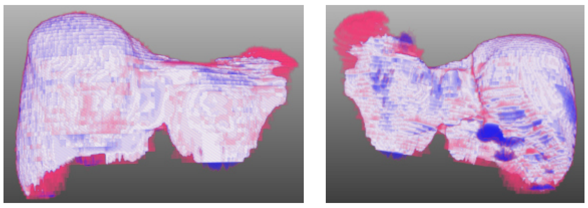


Figure 16. 3D segmentation of test dataset #1.

6. DISCUSSION AND CONCLUSION

Although the texture of the GANs at increased mask size does not look natural for a human inspector, yet the synthesized data is highly applicable for training of DL Models with an abstraction from pure visual input to implicit features anyway. The results in Tab. 2 indicate that enriching the data with synthesized data boosts the training process similar to data augmentation. In case of lack of real-world data, the gap can be closed. As models REAL (4,400 real) and HYBRID_LARGE (2,200 real, 6,600 fake) perform at comparable accuracy, the impact of synthetic data seems of course not equal to enriching with real data but leading to an improvement with increasing number as compared to models REAL_SMALL and HYBRID. As synthesized data can be generated in arbitrary numbers at very high variability it is a good option to compensate for lack of real-world testing data. It has to be stated, that training the GAN itself at a high accuracy necessitates for a sufficient amount of real data too.

For the GAN used in this paper, 5,000 input slices from 24 CT volumes were used. Thus, for application domains with a very low number of samples, a kind of chicken-egg-problem arises as the GAN needs to be trained first in a sufficient way. Here the application of data augmentation as applied for the U-net training helps to significantly reduce the demand for training data. Furthermore, a reasonable batch size, e.g. 16 or 32, which was not possible due to resource limits of the DL HW, will help to reduce the data demand for GAN training. The presented encoding of intensity profile and binary reference segmentation as a 2-channel tensor allows for generation of medical data together with very precise ground-truth. To reduce the memory demand by a factor of two, both the intensity and the reference channel could be encoded in a single channel.

The applicability of GC as post-processing tool is given due to fitness function weights optimized with Evolution strategy to balance between original image intensity and segmentation edges. Using two additional input data first weakly classified by the U-net cascade at only JI=82.1644 and then post-processed by GC allows to iteratively enrich the training dataset. As the reference segmentations generally originate from different tools and experts, this heterogeneity affects the training process too. It is shown, that a small set of segmentations prepared with region growing, Graph cut and Live-wire trains well with the U-net cascade. As shown in Fig. 16, adding data of adequate quality can help to improve the training and test quality.

Thus, if in specific diagnostic domains or for particular studies the initially available amount of test data is insufficient to train DL models at highest accuracy, an incremental approach for data revision and model re-training was presented. In future, training of GANs with less input slices and the combination with GC will be evaluated.

7. ACKNOWLEDGMENTS

Thanks to the BIR research team at Mayo Clinic, Rochester, MN for valuable discussion, support and the provided GPU infrastructure.

8. REFERENCES

[Agg11] Aggarwal, A., Vig, R., Bhadoria, S., and Dethe, C.G., Role of Segmentation in Medical Imaging: A Comparative Study. Int. Journal of Comp. Applic. 29(1), 2011.

[Chr18] Christensen, A., and Wake, N. Wohler Report: Medical image processing software. http://www.wohlersassociates.com, 2018.

[Coo92] Cootes, T.F., Taylor, C.J., Cooper, D.H., and Graham, J., 1992. Training Models of Shape from Sets of Examples. Proc. of the British Machine Vision Conference, 1992.

[Coo98] Cootes, T.F., Edwards, G.J., and Taylor, C.J., 1998. Active Appearance Models. Proc. of the 5th Europ. Conf. on Computer Vision, 1998.

[Fou19] Fourcade, A., and Khonsari, R.H. Deep learning in medical image analysis: A third eye for doctors. J Stomatol Oral Maxillofac Surg 120 (2019) 279–288, 2019.

[Fri18] Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. Neurocomputing, 2018.

[Gao19] Gao, H., Pei, J., and Huang, H. Demystifying Dropout. Proc. of the 36th Inter. Conf. on Machine Learning, PMLR, 2019.

[Goo14] Goodfellow , I.J., Pouget Abadie , J., Mirza, M., Xu, B., Warde Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. Proc. of the 27th Int. Conf. on Neural Information Proc. Sys., vol. 2, 2014.

[Hoc97] Hochreiter, S., and Schmidhuber, J. Long Short-Term Memory. Neural Comp. 9(8), 1997.

[Kin14] Kingma, D.P., and Ba, J.L. Adam: A method for stochastic optimization. Int. Conf. on Learning Representations (ICLR), 2014.

[Koh95] Kohonen, T. Self-Organizing Maps. 3rd ed. Springer, 1997.

[Lap19] Laplante, P.A. (ed.). Encyclopedia of Image Processing. CRC Press/Taylor & Francis Publishing, 2019.

[McI96] McInerney, T., and Terzopoulos, D. Deformable Models in Medical Image Analysis : A Survey. Medical Image Analysis 1(2), 1996.

[Mor15] Morani, A.C., Vicens, R.A., Wei, W., Gupta, S., Vikram, R., Balachandran, A., Reed, B.J., Ma, J., Qayyum, A., and Szklaruk, J. T1-weighted (T1W) gradient recall echo volumetric interpolated breath-hold examination. J Comput Assist Tomogr. 39(2), 2015.

[Raj03] Rajagopalan, S., Karwoski, R. A., Robb, R. A., Ellis, R. E., and Peters, T. M. Shape-Based Interpolation of Porous and Tortuous Binary Objects. MICCAI 2003, 2003.

[Rob98] Robb, R.A., Hanson, D.P., Karwoski, R.A., Larson, A.G., Workman, E.L. and Stacy, M.C., 1989. Analyze: a comprehensive, operator-interactive software package for multidimensional medical image display and analysis. Comput Med Imaging Graph 13(6), 1998.

[Ron15] Ronneberg, O., Fischer, P., and Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI, 2015.

[Sak19] Sakinis, T., Milletari, F., Roth, H., Korfiatis, P., Kostandy, P., Philbrick, K., Akkus, Z., Xu, Z., Xu, D., and Erickson, B.J. Interactive segmentation of medical images through fully convolutional neural networks. MICCAI, 2019.

[Set99] Sethian, J.A. Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press, 1999.

[Sim19] Simpson, A., Antonelli, M., Bakas, S., Bilello, M., Farahani, K., Ginneken, B., Kopp-Schneider, A., Landman, B., Litjens, G., Menze, B., Ronneberger, O., Summers, R., Bilic, P., Christ, P., Do, R., Gollub, M., Golia-Pernicka, J., Heckers, S., Jarnagin, W. and Cardoso, M.J., 2019. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. CoRR, 2019.

[Son13] Sonka, M., Hlavac, V., and Boyle, R. Image Processing, Analysis, and Machine Vision. CENAGE Learning, 4th ed., 2014.

[Squ18] Squelch, A. 3D printing and medical imaging. Journal Med Radiat Sci. 65(3), 2018.

[Str15] Strakos, P., Jaros, M., Karasek, T., Kozubek, T., Vavra, P., and Jonszta, T. Review of the Software Used for 3D Volumetric Reconstruction of the Liver. Int. Journal of Computer and Information Engineering 9(2), 2015.

[Van98] Van Biesen, W., Sieben, G., Lameire, N., and Vanholder, R. Application of Kohonen neural networks for the non-morphological distinction between glomerular and tubular renal disease. Nephrol Dial Transplant 13(1), 1998.

[Vio01] Viola, P., and Jones, M. Rapid Object Detection using a Boosted Cascade of Simple Features. Conf. on Computer Vision and Pattern Recognition ‘01, 2001.

[Yan17] Yang, D., Xu, D., Zhou, S.K., Georgescu, B., Chen, M., Grbic, S., Metaxas, D., and Comaniciu, D. Automatic Liver Segmentation Using an Adversarial Image-to-Image Network. MICCAI, 2017.

[Yi19] Yi, X., Walia, E., and Babyn, P. Generative Adversarial Network in Medical Imaging: A Review. Medical Image Analysis vol. 58, 2019.

[Zha11] Zhang, J., and Wang, X.W. The application of feed forward neural network for the X ray image fusion. J. Phys., 2011.

[Zwe13] Zwettler, G., and Backfrieder, W. Generic Model-Based Application of Modular Image Processing Chains for Medical 3D Data Analysis in Clinical Research and Radiographer Training. Proc. of IWISH, 2013.

[Zwe20] Zwettler, G.A., Backfrieder, W., and Holmes III, D.R. Pre- and Post-processing Strategies for Generic Slice-wise Segmentation of Tomographic 3D datasets Utilizing U-Net Deep Learning Models Trained for Specific Diagnostic Domains. Proc. of VISAPP, 2020.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.28, No.1-2, 2020

Simulated Annealing to Unfold 3D Meshes and Assign Glue Tabs

Thorsten Korpitsch
TU Wien
Vienna, Austria
e0152943@student.tuwien.ac.at

Shigeo Takahashi
University of Aizu
Fukushima, Japan
takahashi@acm.org

Eduard Gröller
TU Wien
VRVis Reasearch Center
Vienna, Austria
groeller@cg.tuwien.ac.at

Hsiang-Yun Wu
TU Wien
Vienna, Austria
wu@cg.tuwien.ac.at

ABSTRACT

3D mesh unfolding transforms a 3D mesh model into one or multiple 2D planar patches. The technique is widely used to fabricate papercrafts, where 3D objects can be reconstructed from printed paper or paper-like materials. The applicability, visual quality, and stability of such papercraft productions is still challenging since it requires a reasonable formulation of these factors. In this paper, we unfold a 3D mesh into a single connected 2D patch. We also introduce glue tabs as additional indicators in order to provide users with extra space to apply glue for better reconstruction quality. To improve space efficiency, we do not apply glue tabs on every edge, while still guaranteeing the stability of the constructed paper model. A minimum spanning tree (MST) describes possible unfoldings, whereas simulated annealing optimisation is used to find an optimal unfolding. Our approach allows us to unfold 3D triangular meshes into single 2D patches without shape distortions, and employing only a small number of glue tabs. A visual indicator scheme is also incorporated as a post-process to guide users during the model reconstruction process. Finally, we qualitatively evaluate the applicability of the presented approach in comparison to the conventional technique and the achieved results.

Keywords

3D mesh unfolding, Simulated annealing, Glue tabs, Graph theory

1 INTRODUCTION

Papercraft is a popular art form, where people create 2D or 3D objects from cardboard or paper as shown in Figure 1. To achieve this, a 3D mesh representing the object is unfolded into a single or multiple 2D patches, which can then be printed and used to reconstruct the model in 3D. Possible models range from simple ones, such as paper aeroplanes, to complex ones, such as buildings. Recently, papercraft approaches are also used in combination with self-folding materials to form structures in an automatic fashion [1]. As demonstrated by Takahashi et al. [19], unfolding a 3D triangular mesh into a single patch, in contrast to multiple patches, eases the reconstruction process for users. However, finding an unfolding, in which no pair of faces overlaps with one another and without distorting the original mesh is a difficult task. More specifically, it has been proven to be an NP-complete problem [8]. Reconstruction of a model can be hard even with indicators that show which edges should be glued together [19].

Glue tabs are essential as they allow users to build clean 3D models and guide them during the reconstruction process. In this paper, we propose a technique to unfold a 3D mesh with a minimum number of glue tabs. This eases the reconstruction of models by guiding edges that should be glued together, as well as providing users sufficient surface to apply glue. The addition of glue tabs increases the complexity of finding an overlap-free unfolding, due to the combinatorial complexity of selecting edges where to attach glue tabs. We calculate minimum spanning trees of the dual graph of the 3D mesh, which describe possible unfoldings. Simulated annealing optimisation is used to find an overlap-free unfolding. Glue tabs are pre-calculated and treated analogue to mesh faces when unfolding the 3D model.

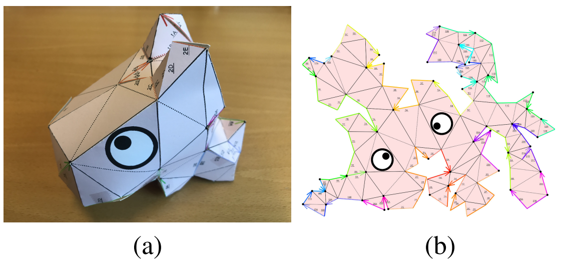


Figure 1: An example of a papercraft model, including (a) a 3D cow model and its corresponding (b) single unfolded patch.

We incorporate trapezoidal glue tabs due to their popularity. Advantageously a trapezoid consists of two triangles, so that we can consider glue tabs as additional faces of the 3D mesh and apply similar overlap-detection when searching for a feasible solution. Furthermore, this shape gives users more space than a simple triangle. It is also preferable to a rectangle as it takes less space. As the glue tabs size increases, the difficulty of finding an overlap-free unfolding also increases. Our contributions include the following:

- Introduction of a minimum number of glue tabs to mesh unfolding problems.
- A new meta-heuristic approach to finding unfoldings of 3D meshes.
- Proof that the number of necessary glue tabs can be optimally pre-computed for each unfolding patch.

The remainder of this paper is structured as follows: Section 2 discusses conventional mesh unfolding and optimisation techniques. Section 3 describes the concepts used in this paper. In Section 4, we describe necessary steps of the method. Section 5 brings insight into the implementation. Section 6 shows several experimental results generated using our approach and evaluates quantitatively and qualitatively. Section 7 summarises the findings and provides an outlook on future work.

2 RELATED WORK

This section focuses on previous work done on the topic of optimising the unfolding of 3D models and also explains the differences to the proposed approach.

2.1 Optimised Unfolding of 3D Meshes

Mesh unfolding has different applications, such as creating papercraft models [16, 19] and the creation of models from self-folding materials [6]. Some techniques employ mesh deformation [2, 12], in order to relax the problem. Mitani et al. [12] and Chang et al. [2] propose methods that allow mesh simplification as a pre-processing step. Also, the theoretical perspectives [14] of the problem have also been explored. Some authors study different types of target meshes, for example, orthogonal polyhedra [20]. The most relevant work to ours is by Takahashi et al. [19], who proposed a genetic-based algorithm to find a single connected patch for printing purposes. They unfold 3D models without distorting or editing the original 3D model. The key concept of unfoldability is borrowed from topological surgery, in order to guarantee an unfolded patch can be stitched together along the corresponding cut edges.

Contrary to the work done by Takahashi et al. [19] our paper explores a meta-heuristic simulated-annealing approach to find unfoldings. Simulated annealing is easy to implement compared to genetic algorithms and it is more likely to find an optimal solution compared to a greedy algorithm [16].

Another key conventional approach is investigated by Straub et al. [16]. They explored unfolding and attaching glue tabs on all cut edges. They also explored the removal of overlaps by introducing new subdivisions to the mesh. A greedy algorithm optimises the unfolding and resolves overlaps. Glue tabs that have been added to an unfolding are optimised, i.e. changed in size to avoid overlaps, after an initial unfolding has been found. For printability, the unfolding is then separated into multiple cut-out sheets if it does not fit on a single one.

In this paper, the proposed algorithm examines all possible glue tabs in advance and selects a minimum number thereof at each unfolding iteration. To the best of our knowledge, the simultaneous handling of unfolding and glue tabs has not been explored in the state-of-the-art. Our technique is an improvement of the approach studied by Takahashi et al. [19], where reconstructing a 3D model is facilitated if the mesh is unfolded into just a single patch.

2.2 Optimisation Techniques

Since the 3D mesh unfolding problem is an NP-complete problem [8], optimisation techniques are often used to find a solution. Trying all combinations would not be practical in most of the cases. Many optimisation techniques are well explored, including greedy algorithms [5] or heuristic optimisation techniques [11]. Simulated annealing is a well-known optimisation technique [9] that is widely applied in computer science [4] and other scientific fields [13, 18].

```
Data: Configuration $P$, Max Temperature $t_{max}$
Result: Optimised Configuration $P$
1 Set $t = t_{max}$;
2 while $t > 0$ do
      /* Random step to generate $P'$ from $P$ */
3     Create $P'$ from $P$;
4     if $energy(P') \leq energy(P)$ then
5         Set $P = P'$;
6     else if $rand(0,1) \geq exp(-(energy(P) - energy(P'))/k_B/t)$ then
7         Set $P = P'$;
8     Decrease $t$;
9 end
```

Algorithm 1: The pseudo-code of simulated annealing [9].

Algorithm 1 depicts the concept of a simulated annealing approach, where a configuration P is optimised by minimising its energy. In each iteration, a new configuration P' is created, and the energy is compared to the previous configuration through the function $energy(*)$. In each iteration, there is a probability to take a worse configuration, to avoid getting stuck in local minima, as shown at Line 6 in Algorithm 1. k_B is a constant, which should be adjusted and determined through experiments.

Optimising 3D Mesh unfolding with simulated annealing has the distinct advantage over greedy algorithms, that it is less likely to get stuck in a local minimum.

Compared to genetic algorithms, simulated annealing is easier to implement and is good at approximating a global optimum. Therefore, we chose simulated annealing as the optimisation approach in our unfolding.

3 DEFINITION OF KEY CONCEPTS

In this section, the terminology used in this paper is defined and key elements are described.

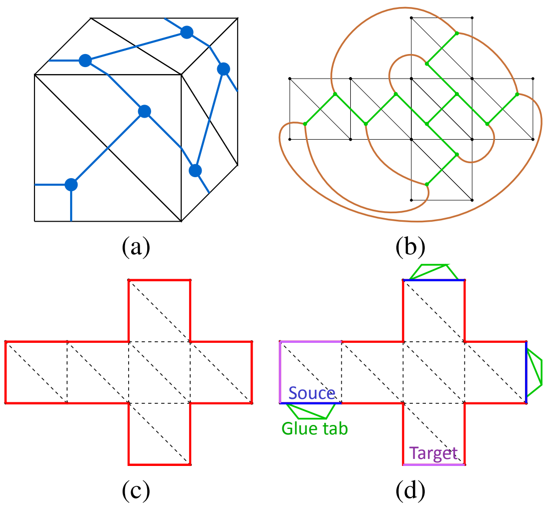


Figure 2: (a) A triangular mesh M (black) and its dual graph D (blue). (b) The MST T (green) and remaining edges (brown) in the dual graph. (c) Bend-Edges (dotted) and Cut-Edges (red). (d) Trapezoidal glue tabs (green) with source and target edges.

Triangular Mesh M. The input for the algorithm is a triangular mesh M (see black edges in Figure 2 (a)). A mesh M is defined as a triple (V_M, E_M, F_M), where V_M, E_M, and F_M represent the sets of vertices, edges, and faces, respectively.

Dual Graph D. Graph $D = (V_D, E_D)$ is the dual graph of the mesh M. For each face $f \in F_M$, a dual vertex in V_D is first assigned, followed by connecting dual vertices using a dual edge E_D, if the two faces that enclose the dual vertices are adjacent in M. In other words, the dual graph D has an edge if and only if the two corresponding faces of M are separated by an edge in the mesh [7].

The dual graph is then used to find an unfolding, since a dual edge connects neighbouring faces. A dual edge can either represent an edge that is cut or an edge that is used for folding the papercraft. Note that each triangular mesh has only one corresponding dual graph, which can be computed efficiently. This makes it a very compelling concept for 3D mesh unfolding.

Minimum Spanning Tree (MST) T. From the dual graph D, a MST T can be computed. Given a weight $c(e)$ to each edge $e \in E_D$, a minimum spanning tree $T = (V_T, E_T)$, where $V_T = V_D$ and $E_T \subseteq E_D$, is computed by minimising the cost function $\sum_{e \in E_T} c(e)$ [3]. The MST in combination with the dual graph is a viable concept to calculate possible unfoldings, as explained by Straub et al. [16]. To compute a MST, we need a weight for each edge in the dual graph. Initially, we assign a random weight between $(0,1)$ for each edge in our experiment. In practice, the weights are adjusted to generate a better MST as the optimisation proceeds.

Unfolding. An unfolding is defined as the planar representation of a 3D mesh, and is computed by mapping faces onto a 2D surface. One can unfold the faces of a mesh one after another by referring to the MST T described previously. This process is called mesh unfolding. We follow the strategy of Takahashi et al. [19], and define that the goal of an unfolded mesh will not contain overlapping areas. A correct unfolding should have only occlusion-free areas (see Figure 2 (b)).

Bend-Edge. The dual edge of $e \in E_T$ is defined as a Bend-Edge (see dotted edges in Figure 2(c)), where users can fold their papercraft by bending Bend-Edges.

Cut-Edge. The dual edge of each $e \in E_D \backslash E_T$ (brown edges in Figure 2(b)) is called a Cut-Edge (red edges in Figure 2(c)). Glue tabs can be applied to Cut-Edges.

Glue Tab. A glue tab is defined as an additional region that does not exist in the original mesh. The idea of incorporating these regions has been widely used to apply glue for attaching Cut-Edges of a papercraft. It can have different forms, while in this paper, we introduce trapezoid glue tabs (Figure 2(d)) as our default setting, due to its popularity in the community.

4 UNFOLDING MESHES WITH ADDITIONAL GLUE TABS

Figure 3 shows the step by step (S1)-(S5) process of our approach. We will describe the high-level idea here, followed by the detail explanation in the corresponding subsections.

Once a triangular mesh M is loaded (Figure 3(a)), in (S1) the corresponding dual graph is computed (the blue graph in Figure 3(b)). In (S2), a MST T is computed (the green tree in Figure 3(c) and the Cut-Edges in red). Based on this T, a minimum number of glue tabs is selected (Figure 3(d)). The next step is to unfold the 3D mesh and assign glue tabs referring to the MST T (Figure 3(e)). If any overlaps occur, we create a different MST by changing an edge weight, and return and continue with (S2). Otherwise, we optimise the unfolding as described in Section 4.5, which results in an unfolded patch with better space utilisation (Figure 3(e)). Figure 3(f) shows additional visual indicators added during the post-processing. Steps (S1)-(S5) will be explained in the following subsections.

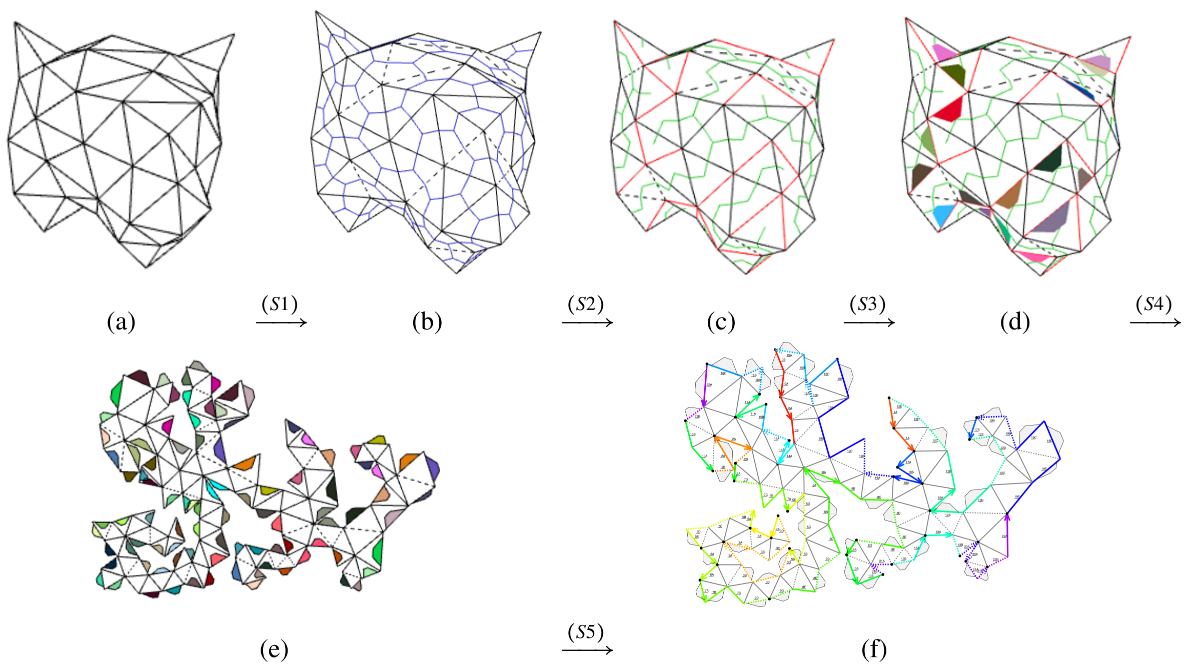


Figure 3: Steps (S1)-(S5) of the proposed unfolding process. (a) The input 3D mesh. (b) The dual graph (blue) of (a). (c) MST (green) and Cut-Edges (red) of (b). (d) Minimal number of glue tabs. (e) The optimised unfolding, and (f) the patch after the post-processing.

4.1 Dual Graph and Preparation (S1)

The dual graph D is computed initially (Figure 3(b)). We iterate through all faces of the mesh and save a dual edge for each pair of faces that share an edge. To compute a MST later, each new dual edge is initially assigned a random weight between $(0,1)$. Note that the weight can be also computed based on the angle bounded by the two adjacent faces or other measures. We choose a random approach here due to its better performance in comparison to other measures. More sophisticated measures, such as mesh structure analysis, will be conducted in a future study.

Glue Tab Pre-Computation

In this step, we also initialise glue tabs. After computing the dual graph, we compute all possible glue tabs, in order to link each dual edge to its corresponding glue tab. For each edge $e \in E_M$ a glue tab is computed. The height of the glue tab is 20% of the face size it will be glued to. The long base length is 80% and the short base length is 40% of the edge length it is attached to, as shown in Figure 2(d). These parameters are determined empirically to avoid an infeasible solution space.

4.2 MST Computation (S2)

A MST T is computed from the dual graph D using the well-known Kruskal's algorithm [10]. Recall that the dual edges of E_T are all Bend-Edges, while the dual edges $e \in E_D \backslash E_T$ are all Cut-Edges (red edges in Figure 3(c)).

4.3 Glue Tabs Selection (S3)

In our approach, we only select certain Cut-Edges to assign glue tabs. This is because assigning glue tabs on every Cut-Edge will limit the number of feasible solutions in the search space and increase the effort needed when reconstructing the 3D model. Therefore, we assign only a minimum number of glue tabs that will result in a stable constructed 3D model. For each vertex, we allow at most one edge associated with this vertex without a glue tab, so that the created papercraft will not contain an open hole after being built. This is achieved by visiting all pre-computed glue tabs and selecting a minimum set of them based on the lemma below.

This implies that each vertex $v \in V_M$ of a mesh can only be incident to one edge $e \in E_M$ that is a Cut-Edge, and will not be fixed with a glue tab. This stability is reasonable because adding more glue tabs will not yield a more stable reconstructed model, while increasing the reconstruction effort as shown in our user study.

Minimum Number of Glue Tabs

The aforementioned concept can be summarised as:

Definition 4.1. Given a mesh $M = (V_M, E_M)$ and the corresponding dual graph $D = (V_D, E_D)$, the graph $C = (V_C, E_C)$, where $V_C = V_D$ and $E_C = E_D \backslash E_T$, contains only Cut-Edges that are not assigned a glue tab. If $\forall v \in V_C \mid degree(v) \leq 1$ holds true, the reconstruction of the 3D model is stable. The function $degree(v)$ returns the vertex degree of a vertex v.

Figure 4 gives an example to demonstrate this property. The input mesh M (black) and the corresponding dual graph D (blue) are shown in Figure 4(a). Different MST T (green trees in Figure 4(b) and (c)) can be synthesised from D. The graph C in Definition 4.1 can be determined as follows. Assume that the graph C is the union of n connected components $C_i = (V_{C_i}, E_{C_i})$, where $i = 1,..,n$ and $1 \leq n < |E_C|$. For each connected component C_i we assign glue tabs, under the condition that $\forall v \in V_C \mid degree(v) \leq 1$ holds.

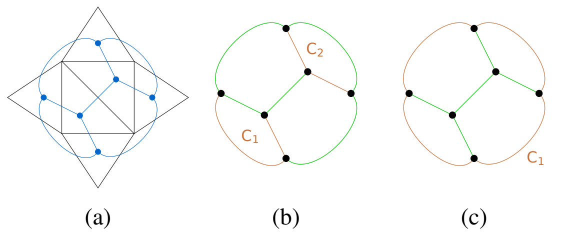


Figure 4: (a) An unfolded mesh M (black) and its dual graph D (blue). (b) A MST (green) of (a) and its E_C resulting in two path components C_1 and C_2 (brown). (c) Another MST T (green) of (a) that defines the unfolding and its corresponding E_C as a cycle component C_1 (brown).

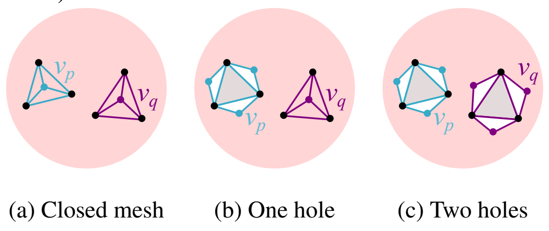


Figure 5: An example of multiple cycles.

We first investigate the properties of C_i. Suppose we have a closed manifold mesh. Then there are two types of C_i that can be taken into consideration. One type is a path (brown graph in Figure 4(b)), and the other one is a cycle (brown paths in Figure 4(c)). This is because the dual vertex of a face in a triangular mesh connects to three dual edges. At least one dual edge belongs to the MST, and thus each vertex in C_i can have a maximum degree of two. We further summarise the combinations of cycles and paths into three cases. Case 1 contains only one cycle, Case 2 contains only paths, and Case 3 contains cycles and paths. Figure 5 shows the reason why the cycles-only case does not exist. A cycle C_i happens only when we cut a mesh into multiple patches, or open a vertex. If we open vertices on a closed manifold mesh, the Euler Formula does not hold: $(V_{C_i} - n) + \sum_{p=1}^{n} degree(v_p) - (E_{C_i} + \sum_{p=1}^{n} degree(v_p)) + F_{C_i} \neq 1$. This leads to the condition that our unfolded patch is not a single connected planar graph, which violates our assumption.

Since C_i can be only a cycle or a path, the glue tab assignment can be done independently. In practice, the problem is equal to retrieving a minimum edge cover of a cycle or a path, which can be calculated in $O(|E_{C_i}|)$ time.

Based on this fact, we can devise an algorithm that finds the minimum number of glue tabs for C_i. In Case 1, C_i is a path, as shown in Figure 4(b). We pick a vertex $v \in V_D$ with $degree(v) = 1$ as the start vertex, then we iterate over the path and attach a glue tab on every second edge. More specifically, the number of glue tabs necessary of a path is exactly $f(C_i) = \lfloor \frac{|E_{C_i}|}{2} \rfloor$. In Case 2, if C_i is a cycle, as shown in Figure 4(c), we can pick an arbitrary starting edge for the path and the number of glue tabs can be computed as $f(C_i) + 1$.

Lemma 4.1. *The minimum number of glue tabs for a mesh M can be computed in $O(n)$ time. The total glue tab number is $\sum_{i=1}^{n} f(C_i)$, where n is the number of connected components defined in Definition 4.1.*

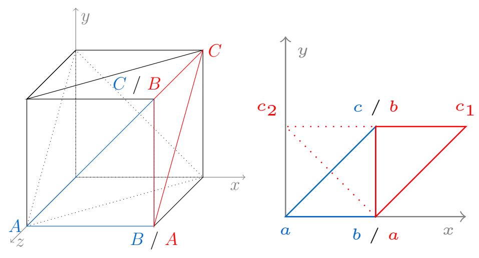
Proof. Note that $f(C_i)$ returns the minimum number of glue tabs, can be proven by induction. Let $P(n)$ be the statement that $f(C_i) = \lfloor \frac{|E_{C_i}|}{2} \rfloor$ determines the minimum number of glue tabs necessary for a path. The base case is $P(1)$. For $|E_{C_i}| = 0$, $\lfloor \frac{0}{2} \rfloor = 0$ is true. Let us consider the step from n to $n+1$ and assume that $P(n)$ is true. Suppose $n+1$ is odd, this means that n is even and $P(n+1) = P(n) + 1$, in case n is odd, $P(n+1) = P(n)$. This leads to the fact that $P(n+1) = \lfloor \frac{|E_{C_i}|}{2} \rfloor$ holds true. The case of a cycle can be proven similarly. Based on this fact, we now show that $\sum_{i=1} f(C_i)$ is minimal by contradiction. We assume that $\sum_{i=1} f(C_i)$ is not minimal, which implies that $\exists f(C_i)$ having a larger value than its minimum. This constitutes a contradiction to our previous statement. ∎

4.4 Mesh unfolding using the MST (S4)

Figure 6 shows how an unfolding of the mesh is computed. Each face is defined by three points A, B, and C in 3D and a, b and c in the 2D representation. For the first face, as shown in blue in Figure 6, we set the vertex a to $(0,0)$, b to $(0,\overline{AB})$. Then we calculate the position of c using the following Equation 1.

$$\begin{aligned} s &= \|(B-A)\times(C-A)\|/(\overline{AB})^2 \\ d &= (B-A)\cdot(C-A)/(\overline{AB})^2 \\ c_x &= a_x + d(b_x - a_x) - s(b_y - a_y) \\ c_y &= a_y + d(b_y - a_y) + s(b_x - a_x) \end{aligned} \tag{1}$$

For every following triangle we already have the positions of two vertices and only need to calculate the position for the last vertex c, which has two possible positions c_1 and c_2 as shown in Figure 6(b). We select the position that is on the opposite side of the shared edge. If a glue tab is attached to a face it is unfolded analogue to mesh triangles.



(a) (b)

Figure 6: (a) A 3D model of a box, and (b) the unfolding of the first two faces.

4.4.1 *Overlap Detection*

After an unfolding is found, we detect overlaps to determine if the unfolding is correct. An overlap is defined if two faces are intersected partly with each other, or if a face is entirely located inside another face. For glue tabs, the definition is similar. In total, three different cases of overlaps can occur, including either face-face, glue tab-glue tab, or face-glue tab overlaps.

If an overlap occurs, the Sutherland-Hodgman Clipping algorithm [17] is used to calculate the overlapped area. This algorithm finds the vertices of the intersection polygon created by the two faces. The overlap can be described as a polygon p. The area is calculated with the shoelace formula $Area = \frac{1}{2}\left|\sum_{i=1}^{k-1} x_i y_{i+1} + x_k y_1 - \sum_{i=1}^{k-1} x_{i+1} y_i - x_1 y_k\right|$ [15], where x_i and y_i are the coordinates of the i-th point in p. This area is then used as the $energy(P)$ function to describe the quality of the unfolding.

Face-face overlaps are checked before glue tab related overlaps are checked to improve the performance. This is because if faces overlap with each other, an overlap-free unfolding cannot be found even if the glue tabs are overlap-free. Moreover, the performance can be improved by limiting the number of faces that need to be checked. The overlap detection does not need to be done if faces share Bend-Edges, as no overlap is possible due to how we handle the unfolding.

4.4.2 *Spatial Optimisation*

After the simulated annealing process comes to an end, we try to find an unfolding that enables better spatial efficiency, see Figure 7. The energy-function is thereby redefined as $energy(P) = spread(\text{x-axis}) + spread(\text{y-axis})$, where $spread(\text{x-axis})$ measures the distance of the vertex with the highest value on the x-axis minus the vertex with the lowest value on the x-axis. We use simulated annealing again, whereas the algorithm now discards all unfoldings with overlaps.

4.5 Post-Processing (S5)

Multiple strategies are introduced to improve the visual quality, namely the clean look of the reconstructed model and guide users with reconstruction using visual indicators.

Colour Coding and Edge Numbering. According to the stitching algorithm proposed by Takahashi et al. [19], boundary edges that will be glued together, are assigned the same colour. We follow their strategy to generate our visual indicators.

Bend-Edge Coding. Each Bend-Edge is coded to distinguish between a mountain-fold or a valley-fold, as proposed by Takahashi et al. [19]. A mountain-fold is represented by a solid line and a valley-fold is displayed by a dotted line (see Figure 3(c)).

5 IMPLEMENTATION DETAILS

The system is implemented using an Intel Core i7 CPU (4x3.3 GHz, 4MB L3 Cache) and 8 GB RAM. The source code is written in C++17 using OpenGL ver. 4.5 to visualise the results and CGAL ver. 4.13 to manage the graph data structure. QT 5.13 is employed to provide a graphical user interface, and CMake ver. 3.14 is used to build the process. GCC 9.2 is used to compile the sources on Ubuntu 18.04.02 LTS.

We limit in the simulated annealing process the maximum number of iterations to $t = 100{,}000$ and set $k_B = 0.002$ based on our empirical experience.

Then we employ the simulated annealing process. Additionally, a face-face overlap is weighed ten times higher than a glue tab-glue tab and a glue tab-face overlap. This prioritisation should force the algorithm to solve the mesh unfolding first, as glue tab related overlaps could be resolved using post-processing.

At the end of every iteration, if P is set to P', we visualise the unfolding. The annealing process terminates if $energy(P') = 0$. If until $t = 0$ we do not find a correct unfolding we terminate unsuccessfully. The spatial optimisation is limited to $5{,}000$ iterations.

6 RESULTS AND EVALUATION

In this section, we show several experimental results and discuss the findings in our evaluation.

6.1 Experimental Results

Figures 3, 7 and 8 show the results generated using our system. In Figure 7, we compare the unfolding results that are considered without and with spatial optimisation. In Figure 7(b), the triangles are smaller than the ones in (c). Figure 8 show other interesting models, where the glue tabs are highlighted in the 3D meshes. The testing data specification is summarised in Table 1.

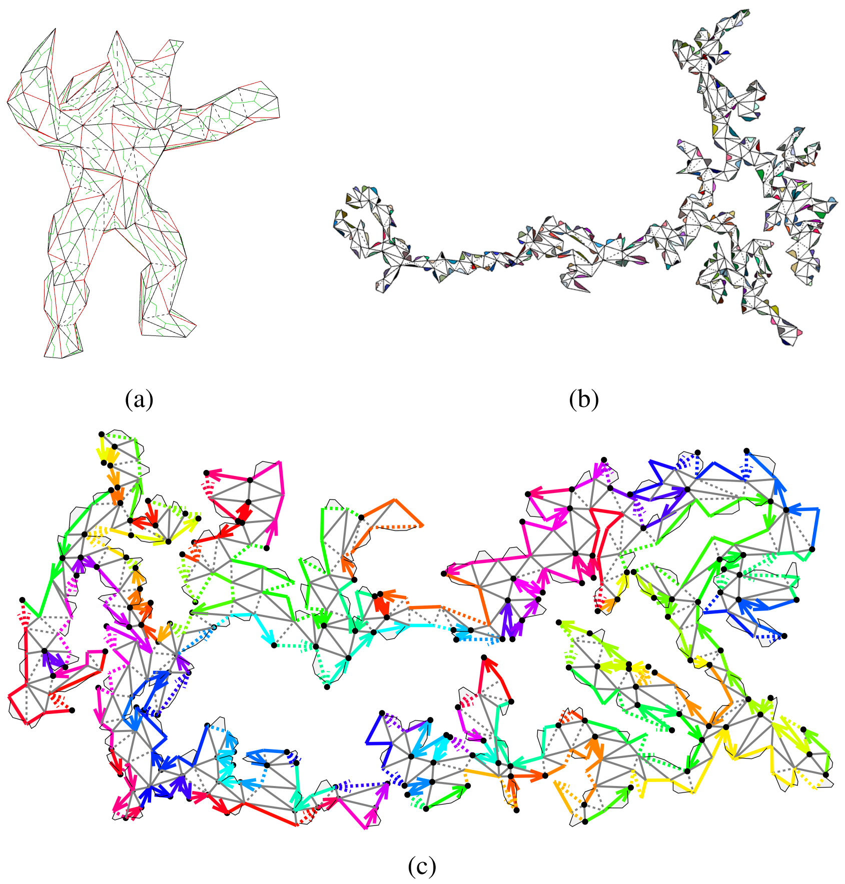


Figure 7: (a) Armadillo mesh. (b) Unfolding. (c) Spatially optimised unfolding.

6.2 Quantitative Evaluation

Figure 8(a)-(c) shows different models and their unfoldings. Sufficient results can be achieved within 100,000 iterations limit to find an unfolding. With a limit of 5,000 iterations for spatial optimisation compact results can be achieved. More iterations did not yield consistently more compact results.

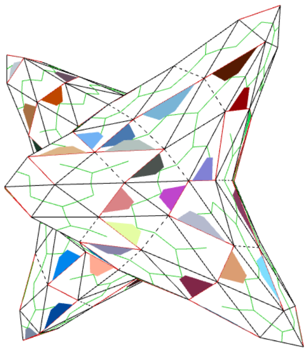
Glue tabs certainly impact the amount of time necessary to find an unfolding, but they are not the only influence. The structure of the model also impacts the time necessary to find an unfolding. While some models have a similar number of faces, the time to unfold can differ substantially. This can be attributed to the geometric properties of the models, as spherical models seem to unfold consistently faster than others. Further, the time discrepancy can be attributed to the random-walk of the simulated annealing process, where the time to unfold can greatly differ for the same model.

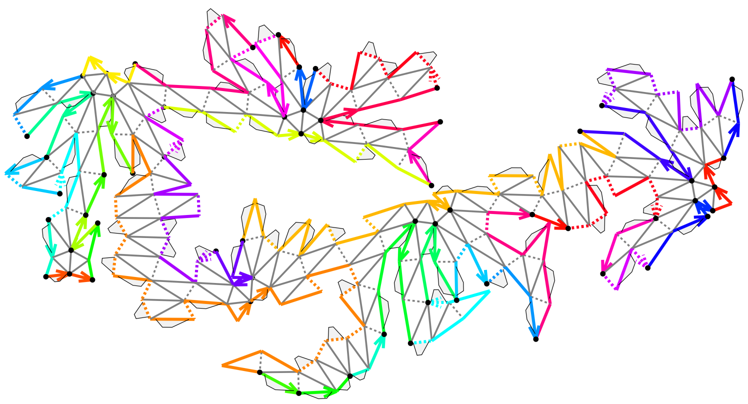
To evaluate the algorithm, we conduct an experiment by unfolding a variety of 3D models. The results are summarised in Table 1 using glue tabs as described in Section 4.1. In the experiment, a maximum of 100,000 iterations is used to limit the computational time, as an experimental value.

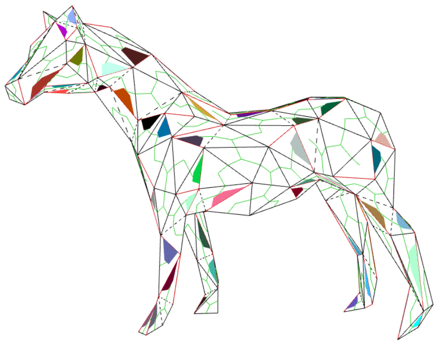
Table 1: Table showing the average unfolding performance for different models. V_M is the number of vertices, E_M the number of edges and F_M the number of faces.

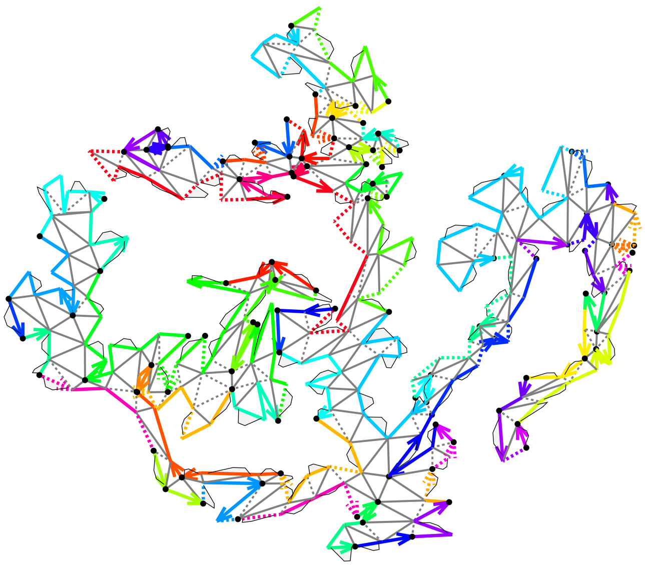
					Time (seconds)	
Model		V_M	E_M	F_M	Ours	Brute-force
Icosa		12	30	20	0	0
Star		14	36	24	8	19
Star-Sqrt3		38	108	72	31	>6000
Tiger	(Fig. 3)	58	168	112	65	-
Star-PNsplit	(Fig. 8(a))	110	324	216	625	-
Horse	(Fig. 8(b))	153	453	302	946	-
Hand	(Fig. 8(c))	170	504	336	1377	-
Bunny-348		176	522	348	976	-
Armadillo	(Fig. 7)	195	579	386	730	-
Moneybox-392		196	586	392	2200	-

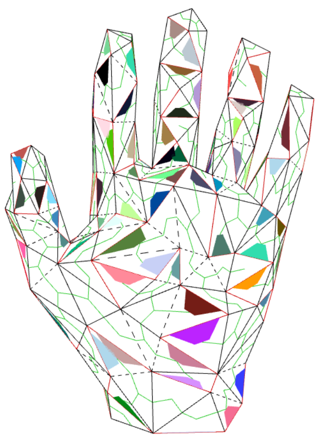
The performance of the presented approach is not only influenced by the number of faces, but also by the size of the glue tabs. The bigger the glue tabs are, the more iterations are needed to find a feasible unfolding. In the worst case, an unfolding might no longer be possible. Table 1 shows that not only the number of faces influences the computational time for finding a solution, but also shows that the randomness of simulated annealing plays an important role. A comparison with a brute-force approach is conducted to investigate the feasibility of the solution space. The brute-force approach cal-


(a.1)


(a.2)


(b.1)


(b.2)


(c.1)

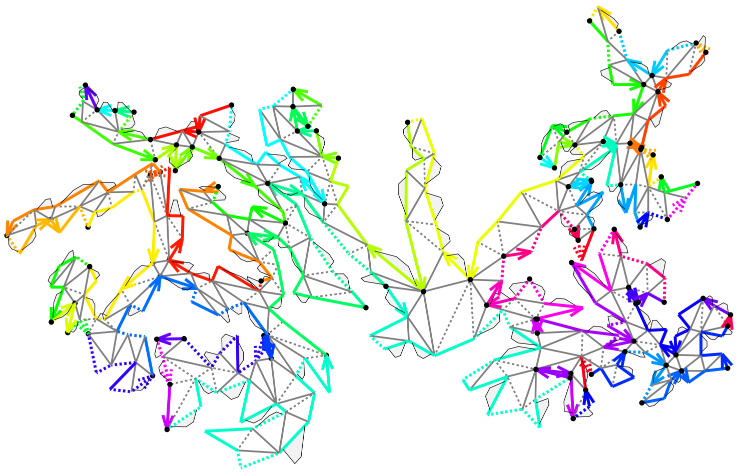

(c.2)

Figure 8: (a.1) Star mesh with 216 faces. (a.2) Unfolding of the star mesh. (b.1) Horse mesh with 302 faces. (b.2) Unfolding of the horse mesh. (c.1) Hand mesh with 336 faces. (c.2) Unfolding of the hand mesh.

culates all possible minimum spanning trees, as well as all possible glue tab positions until a solution is found. Only small models can be solved using the brute-force approach. As the number of faces increases using the brute-force approach becomes infeasible.

6.3 Qualitative Evaluation

To evaluate the impact of glue tabs on the reconstruction process as well as the visual quality of the reconstructed models, we conducted a user study with five participants (P1)-(P5) between the age 22 and 32 with different educational backgrounds. Three of them have a computer science background, and one has a finance background. The participants were asked to construct the tiger model (Figure 9) twice. Once is with glue tabs (Figure 3(f)), and the other time without. Apart from the glue tabs, both unfoldings have the same visual indicators, as explained in Section 4.5. Models with glue tabs were prepared with double-sided tape, where participants still needed to remove the protective covering. For models without glue tabs, we prepared glue strips. At the beginning of the study, participants were first instructed on how to construct the papercraft. We prepared an unfolded octagon with glue tabs and visual indicators, which the participants used to practice how to follow the visual indicators. To avoid a learning effect, P2-P4 started with the model with glue tabs (O1), and P1 and P5 began with the model without glue tabs (O2).

Table 2: Time (in minutes:seconds) for participants to reconstruct models. M/F marks the gender of the participant. Bold values are the time achieved in the first round. † indicates which model participants found easier to reconstruct, while ‡ marks the preferred visual model.

	P1 (M)	P2 (M)	P3 (F)	P4 (M)	P5 (M)
w. glue tabs	26:15	**23:10**‡	**14:08**†‡	**19:08**‡	22:15†‡
wo. glue tabs	**44:40**†‡	18:47†	19:09	15:04†	**failed**

Table 2 shows the time spent for each model. Most participants required less time for the second model reconstruction, while P3 spent more time for the second one. Her second model does not have glue tabs, which implies that the model with glue tabs is sufficiently easier for her. P5 failed to construct the model without glue tabs after trying for an hour. We further analyse and compare the time reduction of the two model orders, which suggests that glue tabs improve the amount of time necessary for reconstruction. The time needed for reconstruction for P1 decreased by 18:25 and for P4 by 4:04, who started without and continued with glue tabs respectively, while it increased for P3 by 5:01 and for P2 by 4:23, who both started with glue tabs.

We also asked participants, which model was easier to create. Opinions of participants differed. P1, P2, and P4 said that the model without glue tabs was easier, although P1 spent almost half time for the glue tab model. P2 remarked that without pre-cut glue stripes they would find the reconstruction with glue tabs easier. P3 emphasised that the glue tab model is significantly easier, as fewer edges need to be glued and edges are easier to connect. However, the last glue tab is problematic, since it cannot be fixed from inside.

We also asked participants to evaluate which constructed model has a better visual appearance. P2, P3, P4 and P5 perceived that the model with glue tabs is cleaner, while P1 prefers the model without glue tabs. P1 stated that it is easier to use stripes in the process.

Figure 9 shows the results from P3. We highlight two parts of the tiger. With glue tabs, sometimes the edges do not stick together well, as shown at the upper part of the picture, while edges fixed with glue strips are less problematic. On the contrary, in the lower part, the glue tabs stick together well.

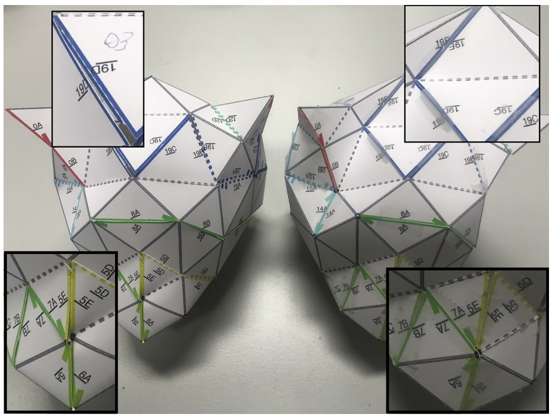


Figure 9: Tiger with glue tabs (left) and without (right) reconstructed by P3.

6.4 Limitations and Discussion

Multiple factors could limit the presented approach. The approach tends to solve the problem in global way, while disregarding local overlaps. Thus, small local overlaps are harder to solve as their changes to $energy(P)$ is rather insignificant. Another limitation is that the current approach is not globally optimal, since we initially limit the glue tab placement to a certain degree. One limitation inherited from the glue tab design is that one can hardly connect the last two faces, since users cannot access the inside of the model anymore to apply counter-pressure.

7 CONCLUSION AND FUTURE WORK

In this paper, we present a new approach to unfold 3D meshes, by adding a minimal number of glue tabs to aid users with the reconstruction. We also demonstrate the applicability of the approach.

To improve the performance and lower the impact of glue tabs on the performance, a glue tab can be adjusted if an overlapping area is rather small. This would increase the solution space for the problem.

In the future, we plan to incorporate mesh structure analysis to guarantee that semantic structures, such as the ears of a bunny, are kept together in the unfolded patch. This could improve the reconstruction process.

ACKNOWLEDGEMENTS

This paper was partly written incollaboration with the VRVis Competence Center. VRVis is funded by BMVIT, BMWFW, Styria, SFG and Vienna Business Agency in thescope of COMET - Competence Centers for Excellent Technologies (854174), which is managed by FFG.

REFERENCES

[1] Byoungkwon An, Ye Tao, Jianzhe Gu, Tingyu Cheng, Xiang 'Anthony' Chen, Xiaoxiao Zhang, Wei Zhao, Youngwook Do, Shigeo Takahashi, Hsiang-Yun Wu, et al. Thermorph: Democratizing 4d printing of self-folding materials and interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[2] Yi-Jun Chang and Hsu-Chun Yen. Improved algorithms for grid-unfolding orthogonal polyhedra. *International Journal of Computational Geometry & Applications*, 27(01n02):33–56, 2017.

[3] David Cheriton and Robert Endre Tarjan. Finding minimum spanning trees. *SIAM Journal on Computing*, 5(4):724, 1976.

[4] Anton Dekkers and Emile Aarts. Global optimization and simulated annealing. *Mathematical Programming*, 50(1-3):367–393, 1991.

[5] Ronald A. DeVore and Vladimir N. Temlyakov. Some remarks on greedy algorithms. *Advances in computational Mathematics*, 5(1):173–187, 1996.

[6] Samuel M. Felton, Michael T. Tolley, ByungHyun Shin, Cagdas D. Onal, Erik D. Demaine, Daniela Rus, and Robert J Wood. Self-folding with shape memory composites. *Soft Matter*, 9(32):7688–7694, 2013.

[7] Jonathan L. Gross and Jay Yellen. *Handbook of graph theory*. CRC Press, 2004.

[8] Thomas Haenselmann and Wolfgang Effelsberg. Optimal strategies for creating paper models from 3D objects. *Multimedia Systems*, 18(6):519–532, 2012.

[9] Scott Kirkpatrick, Daniel Gelatt, and Mario Vecchi. Optimization by simulated annealing. *Science*, 220(4598), 1983.

[10] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[11] Kwang Y. Lee and Mohamed A. El-Sharkawi. *Modern heuristic optimization techniques: theory and applications to power systems*, volume 39. 2008.

[12] Jun Mitani and Hiromasa Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 259–263, 2004.

[13] Jean Pannetier, J. Bassas-Alsina, Juan Rodriguez-Carvajal, and Vincent Caignaert. Prediction of crystal structures from crystal chemistry rules by simulated annealing. *Nature*, 346(6282):343, 1990.

[14] Geoffrey C. Shephard. Convex polytopes with convex nets. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 78, pages 389–403, 1975.

[15] Martin Šlapák, Josef Vojtěch, and Radek Velc. Automated numerical calculation of sagnac correction for photonic paths. *Optics Communications*, 389:230–233, 2017.

[16] Raphael Straub and Hartmut Prautzsch. Creating optimized cut-out sheets for paper models from meshes. *Karlsruhe Reports in Informatics 2011*, 36, 2011.

[17] Ivan Edward Sutherland and Gary Wesley Hodgman. Reentrant polygon clipping. *Communications of the ACM*, 17(1):32–42, 1974.

[18] Jon M. Sutter, Steve L. Dixon, and Peter C. Jurs. Automated descriptor selection for quantitative structure-activity relationships using generalized simulated annealing. *Journal of Chemical Information and Computer Sciences*, 35(1):77–84, 1995.

[19] Shigeo Takahashi, Hsiang-Yun Wu, Seow Hui Saw, Chun-Cheng Lin, and Hsu-Chun Yen. Optimized topological surgery for unfolding 3D meshes. *Computer Graphics Forum*, 30:2077–2086, 2011.

[20] Zhonghua Xi, Yun-Hyeong Kim, Young J. Kim, and Jyh-Ming Lien. Learning to segment and unfold polyhedral mesh from failures. *Computers & Graphics*, 58:139–149, 2016.

Hallucinating Very Low Resolution Face Images to 16x magnification with Age based Attributes

Jihwan Kim
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
okpo65@gmail.com

Sunghee Choi
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
sunghee@kaist.edu

ABSTRACT

Face hallucination is a type of super resolution that restores very low resolution (8×8 pixel) to high resolution (128×128 pixel) face images. Since unique facial features caused by age, e.g.wrinkles, are ignored during restoration, restored face images can be somewhat dissimilar to the original faces, particularly for older people. To solve this problem, we construct a pipeline network to restore face images more realistically by including age attribute, predicted from the low resolution image. Predicted age attribute is divided into young and old groups, where the aging network is the last pipeline stage and only applied when the original face image includes old age attributes. Thus, older people tend to be restored with wrinkles and features similar to their original appearance. Restored images are compared qualitatively and quantitatively with images created by existing methods. We show that the proposed method maintains and restores age related personality features, such as wrinkles, producing higher structural similarity index than other methods.

Keywords
Face hallucination, Pipeline network, Age, Personality, Deep learning

1 INTRODUCTION

Face hallucination (FH) restores a low resolution face image (LR) to high resolution (HR), and is particularly important for face recognition and restoration systems, such as surveillance or CCTV [HHST17]. Generally, FH supports 16 scale restoration for 8×8 pixel and 8 scale restoration for 16×16 pixels. Larger images (16×16 pixel) have considerably more information compared to 8×8 pixel images, which allows employing prior information, such as heatmaps, during their restoration. In contrast, 8×8 pixel images have limited information, which makes restoration difficult. Figure 1 shows that it is difficult to distinguish face shape at all for 8×8 pixel images, whereas 16×16 pixel images can be divided into eye, nose, mouth, and face regions by eye. This paper considers the more challenging 8×8 pixel image restoration.

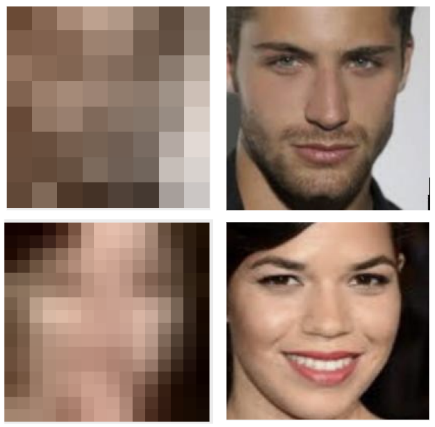


Figure 1: Typical 8×8 and 16×16 pixel images (left) and their corresponding ground truth images (right)

Several methods have been proposed previously to implement FH. For example, Dahl et al. [DNS17] used pixelCNN, Yu et al. [yu2018face] used auto-encoder, Chen et al. [CTL$^+$18] used GAN, and Huang et al. [HHST17] used the wavelet transform. However, these methods have ill-posed problems, producing many HR solutions for a single LR face image. Lower image quality increases restoration difficulty because more information regarding face feature and personality details are lost. Consequently, we cannot solve the FH one-

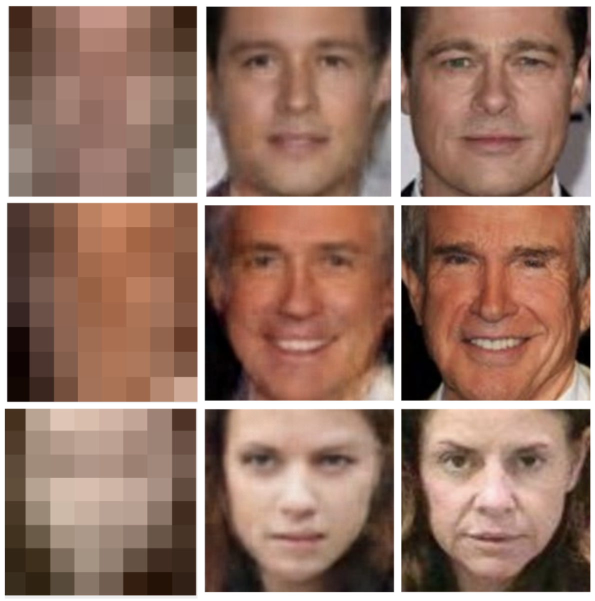


Figure 2: Restoration from 8×8 to 128×128 pixel images using a wavelet based convolutional network approach [HHST17]

to-many problem between LR and possibly many HRs, even if we use large training datasets [YFHP18]. Therefore, this paper proposes a pipeline based network that includes facial age attributes to mitigate the ill-posed FH problem. Very low resolution images (8×8 pixel) have lost considerable facial information compared to their ground truth images, and it is difficult to restore precisely because learning ignores age related facial features (e.g. wrinkles). Figure 2 shows some typical example restored images from 8×8 pixel input images, which all have critical missing details, such as wrinkles. Hence the restored images look much younger than their ground truth. Figure 3 shows that the structural similarity index (SSIM), a measure of image similarity to ground truth, decreases relatively linearly with increasing subject (and hence face image) age, being considerably lower for 80 than 20 year old subjects. To solve this problem, we create a pipeline network learning model that can restore face images including age attributes by including age information in the pipeline network to help produce an appropriate facial image for their age.

The proposed pipeline network incorporates three stages.

1. Age estimation. We apply an Wide ResNet CNN to estimate age [RTVG15] from the original low resolution image. Young and old attributes are determined by this predicted age.

2. First restoration. 8×8 pixel images are restored to 128×128 pixel images using a wavelet based convolutional network (CNN) approach following [HHST17].

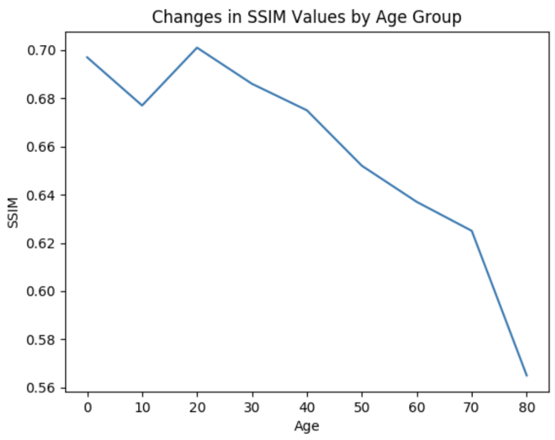


Figure 3: Structural similarity index (SSIM) for restored images with respect to subject age

3. Apply age attributes. We apply an aging general adversarial network (GAN) [ZSQ17] only for images with known age attributes to include subject's age related facial features (e.g. wrinkles).

We assume that restoring face image with facial features like wrinkles mitigates ill-posed problems by making it look similar to the original. Experimental results demonstrate that our pipeline network restores much more similarly to their original images especially for older people than other methods.

2 RELATED WORK

Many methods have been proposed for FH to solve the ill-posed problem. Yu et al. [YFG$^+$18] restored facial images using a heatmap of the face, i.e., structural information, to solve the ill-posed problem caused by pose variation or misalignment. Chen et al. [CTL$^+$18] used a super resolution encoder-decoder structure to generate facial images, creating more realistic images by employing a GAN structure with loss function being the difference between ground truth and generated images. Huang et al. [HHST17] converted the super resolution process to wavelet coefficient prediction for high resolution images. The high resolution image was then reconstructed using the predicted wavelet coefficients, considering texture and topology information from the original low resolution image. In addition to restoring low to high resolution images, recognition algorithms have been proposed to distinguish individuals by including facial features, such as eigenfaces [HYBK08].

Dahl et al. [DNS17] used pixelCNN to restore very low resolution images (8×8 pixel), employing a stochastic model network rather than linear interpolation to predict each pixel value in the high resolution images. However, each pixel RGB value must be separately trained in the model, requiring very high computational complexity. Therefore, restoring for scales above 4 is very difficult. Yu et al. [YFHP18] proposed an FH method by adding facial attributes to maintain face detail. After restoring the low resolution image by interpolation, a facial attribute vector was inserted into the residual block and then trained. However, they only supported 8 scale restoration from 16×16 pixels, and restored the image using a random attribute rather than actual face image attributes. In contrast, the proposed approach supports 16 scale reconstruction for the challenging 8×8 pixel input resolution. We obtain the age attribute from the low resolution image and then use this in the pipeline to restore a facial image more likely reflecting the subject's own personality.

Tamura et al. [TKM96] proposed a neural network that identifies gender with more than 90% accuracy at 8×8 pixels image, where the network learned image gender using face shapes, cheekbone shapes, and shading. Wang et al. [WCY+16] showed that recognition performance from low resolution images is much lower than ground truth images for face, digit and font recognition.

A key factor in identifying faces is the general patterns of aging facial marks (e.g. wrinkles). Jain et al. [JP09] attempted to increase recognition accuracy by detecting facial-marks such as wrinkles. Ling et al. [LSRJ07] showed that recognition error was higher when there was a large gap between actual and expected age.

3 APPROACH

Overview Structure

Figure 4 shows three stages of our pipeline network. Stage 1 (age estimation) predicts age attribute directly for the low resolution image (8×8 pixel), using an age classification network [RTVG15] with Wide ResNet [ZK16] structure. We only assigned age attributes to the image if estimated age is above 40. Stage 2 (restoration) uses a wavelet based CNN [HHST17] to restore the facial image. Finally, stage 3 (aging) applies the aging network [ZSQ17] for images with predicted age above 40. Predicted age from stage 1 and the restored image from stage 2 are input to the age GAN to produce the final restored image.

Dataset

We use the UTKFace dataset [ZSQ17] with age, gender, and race labels comprising 23,704 images, all aligned and cropped. This dataset has been used in many fields, including face detection, age progression/regression, and age estimation. However, more than half of this dataset subjects are in their 20-30s, which could produce age bias when estimating subject age for FH. Therefore, we constructed an age-balanced UTKFace dataset, with even distribution between young and old groups (less and more than 40 years old, respectively) by randomly removing some of the young group images. The age-balanced UTKFace dataset includes 13,656 images with 6,591 young and 7,065 older group images. Figure 5 compares the original and age-balanced UTKFace dataset distributions. The age-balance case is clearly more evenly distributed across the identified subject ages. Both datasets are employed separately for training and their results compared.

Age Estimation

The first stage in the pipeline estimates subject age from the low resolution images using Wide ResNet CNN [ZK16] and the UTKFace dataset. The age classification network comes from [RTVG15]. Wide ResNet solves the neural network depth problem by modifying the ResNet structure and increasing the number of CNN output channels rather than the number of CNN layers. We use the UTKFace images, scaled down to 8×8 pixel. First, the low resolution image is converted to 64×64 pixels by bicubic interpolation. Then image features are extracted using Wide ResNet CNN, and age values were extracted using the softmax activation function. We add data augmentation and dropout to improve estimation accuracy.

After predicting the age, we divide the dataset into young and old groups around the 40 year cutoff based on the predicted age. The aging network, in the final pipeline stage is only applied to images in the older group. Table 1 compares actual and predicted young and old attributes. Accuracy for subjects in their 40s and 50s is relatively low, since they are close to the cutoff, but overall matching rate is acceptable. The hit ratio are more important for 60+ aged subjects, where facial features such as wrinkles are very common. In particular, subjects in their 80s are quite accurately classified (80%).

type	hit	total	ratio
0s	2,967	3,059	96.99
10s	1,392	1,531	90.92
20s	6,712	7,343	91.41
30s	3,533	4,537	77.87
40s	857	2,245	38.17
50s	1,337	2,299	58.16
60s	893	1,318	**67.75**
70s	524	699	**74.96**
80s	407	504	**80.75**
all	18,749	23,704	79.1

Table 1: Age classification accuracy. (Notes: Hit = when predicted and actual age bracket agree.)

Wavelet based CNN

The second stage in the pipeline uses a wavelet based CNN [HHST17] to restore low resolution (8×8 pixel) to high resolution face (128×128 pixel) images. Rather

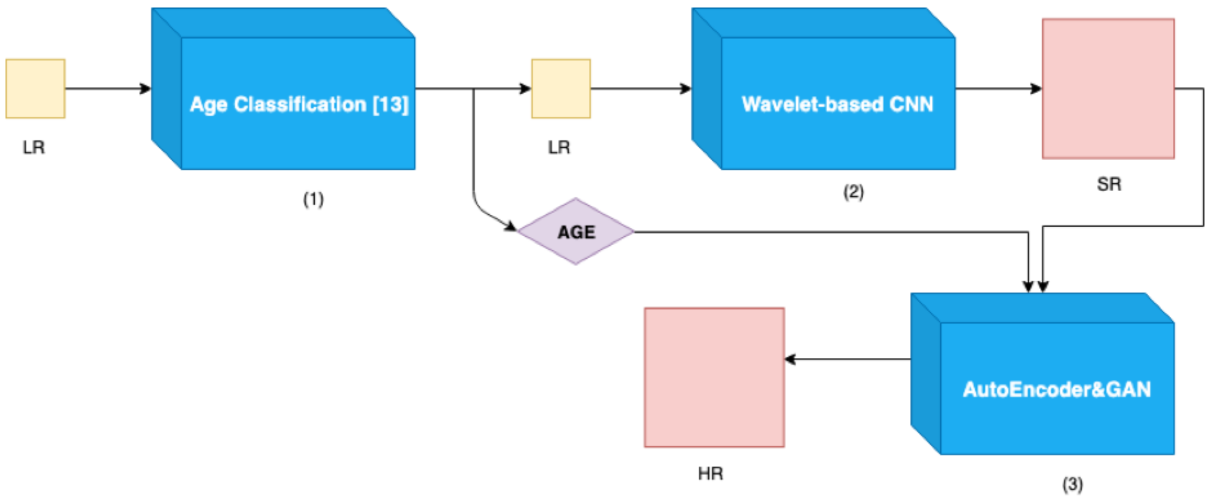


Figure 4: Proposed pipeline network

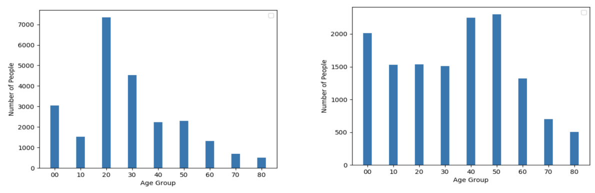


Figure 5: Population distribution by age group for (left) original UTKFace and (right) age-balanced UTKFace datasets

than simply reconstructing the high resolution image, wavelet coefficients for the high resolution image are predicted and restored while maintaining global topology and facial structural textual information.

Embedding Net

This is the process of mapping the low resolution image $(3 \times w \times h)$ to the feature map. We use 3×3 filters at 1 stride and output them to the final output $(N_e \times h \times w)$ without going through upsampling and downsampling. (N_e is the channel size of the last layer)

Wavelet Prediction Net

If n is the level of wavelet packet decomposition, then magnification r is $r = 2^n$ called the scaling factor and N_w is 4^n as the number of wavelet coefficients. The level of packet decomposition determines the scaling factor and the number of wavelet coefficients. Wavelet prediction net is composed with N_w parallel independent subnets that takes the output from the embedding net as input and predicts wavelet coefficients for high-resolution images. Each subnet learns the same image size as the embedding net used 3×3 filters with 1 stride $(3wh)$. Predicting the wavelet coefficient values for all subnets will result in a wavelet coefficient of $N_w \times 3 \times w \times h$. Let C = $(c_1, c_2, ..., c_{N_w})$, $\hat{C}$ = $(\hat{c}_1, \hat{c}_2, ..., \hat{c}_{N_w})$ be the ground-truth wavelet coefficients, predicted wavelet coefficients, respectively, and let W = $(\lambda_1, \lambda_2, ..., \lambda_{N_w})$ be the weight matrix to prioritize which wavelet coefficients are relatively important while restoration. As you can see the loss function below, wavelet loss function attempts to minimize the MSE loss of each wavelet coefficients.

$$l_{wavelet}(\hat{C}, C) = \|W^{1/2} \odot (\hat{C} - C)\|_F^2$$

$$= \sum_{i=1}^{N_w} \lambda_i \|\hat{c}_i - c_i\|_F^2$$

Reconstruction Net

The reconstruction net converts the full size of the wavelet image, $N_w \times 3 \times w \times h$, to the existing image size of $3 \times (r \times w) \times (r \times h)$. The deconvolution layer has a $r \times r$ filter and proceeds to r stride. The formula below shows the overall change in image resolution while learning with wavelet-based CNN. As mentioned before, embedding net maps low resolution images $(3 \times w \times h)$ to feature maps $(N_e \times h \times w)$. The feature maps are then divided into N_w wavelet coefficient images $(3 \times w \times h)$ in the wavelet prediction net. Since N_w is r^2, we can adjust the resolution of high resolution images $(3 \times (r \times h) \times (r \times w))$. The definition of wavelet based CNN networks [HHST17] can be formulated as follows:

$$\Psi : R^{3 \times h \times w} \rightarrow R^{N_e \times h \times w}$$
$$\varphi_i : R^{N_e \times h \times w} \rightarrow R^{3 \times h \times w}, i = 1, 2, ..., N_w$$
$$\phi : R^{N_w \times 3 \times h \times w} \rightarrow R^{3 \times (r \times h) \times (r \times w)}$$

Aging network

The aging network [ZSQ17] includes an encoder to convert face images into a latent vector and a GAN to generate face images from the vector. The encoder maps all images to latent vectors and collects them to form a latent space with age on the horizontal axis and personality on the vertical axis. We predict the latent vector for the age progression/regression image in this latent space by moving the horizontal axis, and then put this latent vector into the GAN to generate the aged image. Traditional GAN uses latent vectors as random sampling, producing blurry images. However, this network creates facial images that retain specific personalities by inserting the latent vector that encodes the actual image into the GAN.

Discriminator on z

The discriminator on z (D_z) is trained to construct a prior distribution (uniform distribution) when mapping latent vectors z from the encoder into latent space. D_z is trained to discriminate z from the encoder and random sampling z^* from prior distributions. It selects a uniform distribution as prior and builds a competition with the encoder to ensure z is evenly distributed in latent space.

Discriminator on Face Image

The discriminator on facial images (D_{image}) is designed to produce realistic images similar to the discriminator used in traditional GANs. The generator takes a latent vector as input and created facial images to trick the discriminator, and the discriminator is trained to distinguish this generated image from the actual image. D_{image} also includes an age label to create a more realistic image for older subjects.

4 EXPERIMENTS

The proposed pipeline network is supported by Ubuntu 16.04 LTS. All pipeline stages are implemented using tensorflow and pytorch. We also use Anaconda V3 because each pipeline step uses a different python version, and Anaconda makes it easy to build other environments by putting development environments in separate containers. We use the age-balanced UTKFace and UTKFace datasets for training, and put the age attribute through the pipeline to restore low resolution facial images to high resolution facial images, while maintaining their personality. If the subject's predicted age (from stage 1 of the pipeline) exceeded 40 years, the old age attribute is assigned, and only those images go through the aging network. The second pipeline stage primarily restore face images. The final pipeline stage (aging network) is only performed on facial images assigned the old age attribute in the first step.

Qualitative Result

Figure 6 compares the proposed pipeline results qualitatively with current best-practice methods that support 16 scale from 8×8 pixel images. The proposed pipeline results are more similar to ground-truth image compared with the other methods, more successfully restoring subject personality, such as age-related wrinkles. Bicubic interpolation and SRCNN [DLHT15] fail to restore face shape, and SRGAN [LTH$^+$17] creates blurry facial images due to the unstable GAN. Wavelet based CNN [HHST17] largely restores facial texture, but the over-smoothing causes subjects to look much younger than their actual age.

Quantitative Result

Tables 2 and 3 compare SSIM by age group for the age-balanced UTKFace and UTKFace datasets, respectively. The proposed pipeline generally achieves higher SSIM than the other methods for subjects older than 40 years that had the age attribute set.

Age	SRCNN	SRGAN	Wavelet	Ours
0s	0.513	0.696	0.675	0.698
10s	0.447	0.653	0.651	0.664
20s	0.435	0.667	0.666	0.68
30s	0.442	0.664	0.652	0.667
40s	0.443	0.65	0.644	**0.659**
50s	0.435	0.632	0.625	**0.652**
60s	0.444	0.634	0.622	**0.652**
70s	0.43	0.609	0.598	**0.629**
80s	0.398	**0.579**	0.551	0.577

Table 2: Structural similarity index scores for the considered image restoration methods on the age-balanced UTKFace dataset

Age	SRCNN	SRGAN	Wavelet	Ours
0s	0.486	0.706	0.697	0.721
10s	0.418	0.66	0.677	0.69
20s	0.408	0.676	0.701	0.714
30s	0.41	0.671	0.686	0.698
40s	0.421	0.665	0.675	**0.684**
50s	0.418	0.649	0.652	**0.66**
60s	0.414	0.641	0.637	**0.642**
70s	0.417	0.623	0.625	**0.65**
80s	0.373	0.582	0.565	**0.612**

Table 3: Structural similarity index scores for the considered image restoration methods on the UTKFace dataset

5 CONCLUSION

We propose a pipeline network to restore very low resolution images to be similar to ground truth by including an age attribute. The proposed pipeline includes three stages: age estimation, wavelet based CNN, and aging network. We construct an age-balanced UTKFace dataset for training to avoid biases due to UTKFace dataset images being weighted toward a certain age.

We compare restored face images qualitatively with current best practice methods, and confirm that the proposed pipeline network produce facial images that restore subject personality due to age, such as wrinkles. Quantitatively, the proposed pipeline generally achieves superior SSIM scores than the other considered methods.

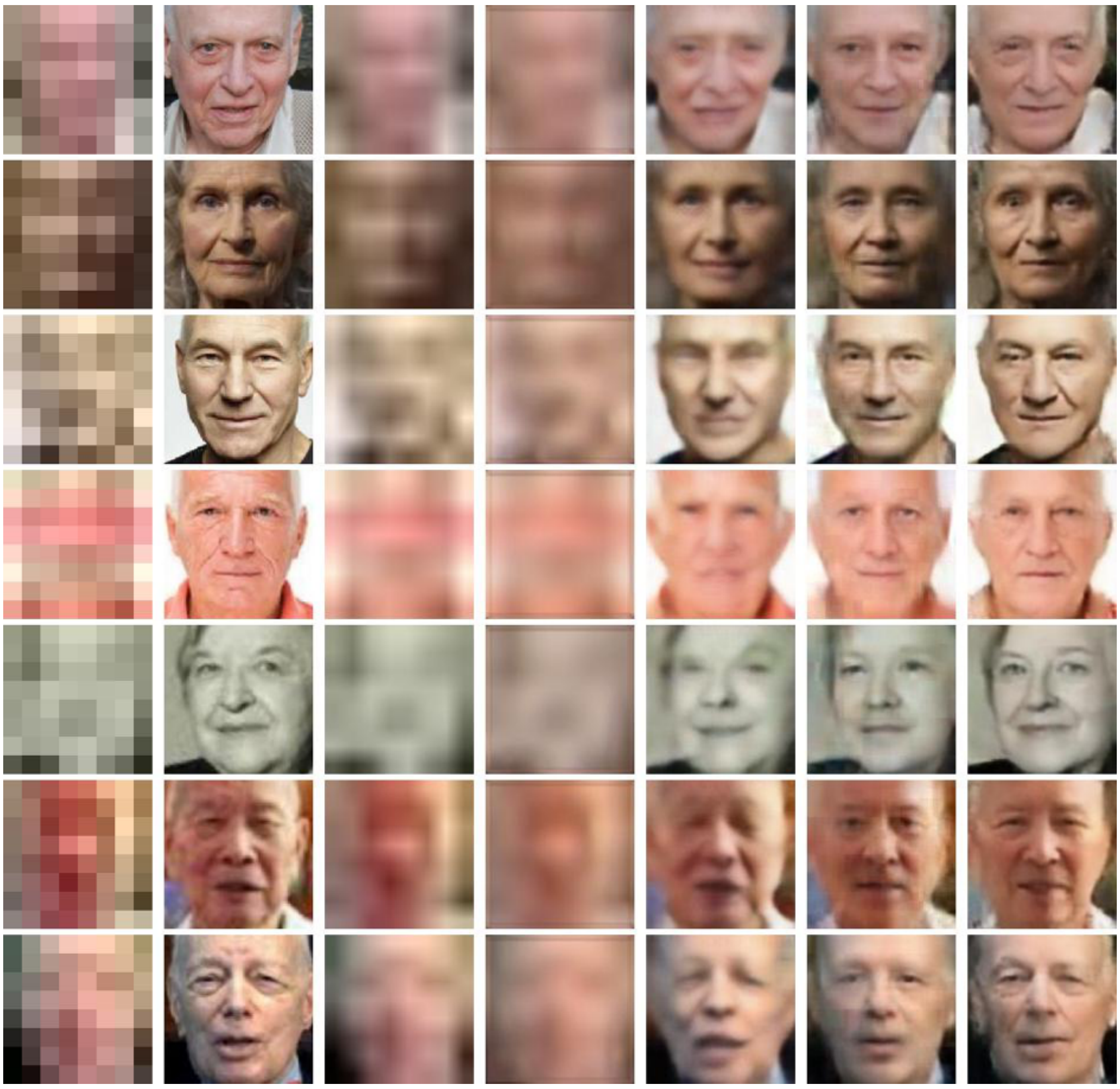


Figure 6: Image restoration outcomes for example 8 × 8 pixel input images. From left: input, ground truth, bicubic interpolation, SRCNN, SRGAN, Wavelet based CNN, proposed pipeline.

Future study will expand the image restoration pipeline to include not only age but also factors representing individual's facial features, such as gender, race, and facial shape. We will also reduce the pipeline process by restoring images with the age attribute in the upscaling process.

6 ACKNOWLEDGEMENTS

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-01158, Development of a Framework for 3D Geometric Model Processing)

7 REFERENCES

[CTL+18] Yu Chen, Ying Tai, Xiaoming Liu, Chunhua Shen, and Jian Yang. Fsrnet: End-to-end learning face super-resolution with facial priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2492–2501, 2018.

[DLHT15] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[DNS17] Ryan Dahl, Mohammad Norouzi, and Jonathon Shlens. Pixel recursive super resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5439–5448, 2017.

[HHST17] Huaibo Huang, Ran He, Zhenan Sun, and Tieniu Tan. Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1689–1697, 2017.

[HYBK08] Pablo H Hennings-Yeomans, Simon Baker, and BVK Vijaya Kumar. Simultaneous super-resolution and feature extraction for recognition of low-resolution faces. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[JP09] Anil K Jain and Unsang Park. Facial marks: Soft biometric for face recognition. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 37–40. IEEE, 2009.

[LSRJ07] Haibin Ling, Stefano Soatto, Narayanan Ramanathan, and David W Jacobs. A study of face recognition as people age. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[LTH$^+$17] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[RTVG15] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 10–15, 2015.

[TKM96] Shinichi Tamura, Hideo Kawai, and Hiroshi Mitsumoto. Male/female identification from 8$\times$ 6 very low resolution face images by neural network. *Pattern recognition*, 29(2):331–335, 1996.

[WCY$^+$16] Zhangyang Wang, Shiyu Chang, Yingzhen Yang, Ding Liu, and Thomas S Huang. Studying very low resolution recognition using deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4792–4800, 2016.

[YFG$^+$18] Xin Yu, Basura Fernando, Bernard Ghanem, Fatih Porikli, and Richard Hartley. Face super-resolution guided by facial component heatmaps. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 217–233, 2018.

[YFHP18] Xin Yu, Basura Fernando, Richard Hartley, and Fatih Porikli. Super-resolving very low-resolution face images with supplementary attributes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 908–917, 2018.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016.

[ZSQ17] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5810–5818, 2017.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.28, No.1-2, 2020

Application of a New Greyscale Descriptor for Recognition of Erythrocytes Extracted from Digital Microscopic Images

Dariusz Frejlichowski
West Pomeranian University of Technology, Szczecin
Faculty of Computer Science and Information Technology
Żołnierska 52, 71-210, Szczecin, Poland
dfrejlichowski@wi.zut.edu.pl

ABSTRACT

In the paper an algorithm for description of greyscale objects extracted from images is applied for recognition of human red blood cells visible on digital microscopic images. This is a part of an approach for automatic (or semi-automatic) diagnosis of selected diseases based on the deformation of erythrocytes. The disease is concluded by means of the recognition of types of red blood cells visible on an digital microscopic image, stained using MGG method and converted into greyscale. The applied algorithm is based on the polar transform of pixels belonging to an object and a specialized method for constituting the resultant description, where derived coordinates are put into matrix, in which the row corresponds to the distance from the centre, and the column—to the angle. This assumption was previously applied only for shape features. The proposed algorithm includes several auxiliary steps, e.g. median and low-pass filtering in order to pre-process the extracted object in greyscale. The algorithm is experimentally evaluated and analysed. It is compared with four other greyscale descriptors, namely: Scale-Invariant Feature Transform, Gabor filter, Polar-Fourier Greyscale Descriptor, and the approach based on polar transform and projections.

Keywords
Image Recognition, Erythrocytes Identification, Greyscale Descriptor, Polar Transform

1 INTRODUCTION

In the paper a new algorithm for greyscale object description is applied to the problem of recognition of red blood cells extracted from digital microscopic images. The identification of particular types of erythrocytes is applied for the automatic (or semi-automatic) diagnosis of some selected diseases. It results from the fact that deformed erythrocytes cannot properly deliver oxygen. Hence, blood circulation is not regulated. Two exemplary diseases that can be concluded this way are anaemia and malaria. In both cases the appearance of cells is changed, what gives the possibility of performing the automatic analysis based on digital microscopic images using computer vision algorithms. In case of works described in the paper, the digital microscopic images stained by means of the MGG (May-Grunwald-Giemsa) method are applied. Considering the approaches applied for the problem, the template matching utilizing the greyscale as a feature is used. The template matching is a general approach in which the classes of objects for the identification are represented by templates stored in the template base. Usually one or a few templates represent a class. However, both templates and test objects are not stored in their original appearance. Instead, they are represented (described) using particular methods, so-called descriptors. These descriptors apply some low-level features, e.g. shape, texture, colour or greyscale. These algorithms are very popular nowadays, thanks to their fast and easy derivation [Ver15]. In the paper the greyscale as a feature is applied. This comes from the conclusion that in real world, greyscale can bring important and useful information, sometimes better than in case of other features [Fre19]. A new algorithm for greyscale object representation is used for the problem of red blood cells identification. It applies the polar transform of pixels and a method for deriving the description, where obtained new coordinates are put into matrix, in which the row corresponds to the distance from the centre of an object, and the column—to the angle. This proposition was taken from an algorithm for shape representation [Rau94] and was not used for the greyscale so far. The applied method includes several additional steps, e.g. median and low-pass filtering in order to pre-process the extracted object in greyscale.

The proposed algorithm is applied in the problem of red blood cells recognition and the obtained results are described in this paper. However, in order to analyse the efficiency of the method, it is compared experimentally with four other greyscale descriptors: Scale-Invariant Feature Transform [Low04], Gabor filter [Kum18], Polar-Fourier Greyscale Descriptor [Fre11], and the approach based on polar transform and projections [Fre18].

The rest of the paper is organized as follows. The second section describes the related works, both by means of the erythrocytes analysis, and the greyscale applied as a feature. The third section provides the description of the algorithm prosed in the paper. The fourth section gives the basic information about the algorithms selected for the comparison with it. The fifth section presents the results of the experiment, and finally, the last section concludes the paper and gives some discussion about future plans.

2 RELATED WORKS

The usage of computer vision in the analysis of cells visible on the digital microscopic blood images is popular. However, various applications are taken into account, e.g. counting the cells without their identification (e.g. [Ven13]), analysis of all visible objects (e.g. [Ran07]), sometimes more complicated and hence less efficient. Even when only one type is taken into consideration, usually leukocytes are analysed (e.g. [Son02], [Sab04]). However, some progress in the examination of red blood cells was made—several algorithms were applied for this purpose so far, e.g. morphological operators [DiR02], threshold selection techniques [Ros06], histogram [Dia09], deformable templates [Bro00], and polar-logarithmic transform [Lue05]. In [Fre10] three shape descriptors based on polar transform were investigated, namely Log-Pol-Fourier, UNL-Fourier and Point Distance Histogram. Very popular lately Deep Convolutional Neural Networks were applied to the problem of automatic identification of malaria infected erythrocytes [Don07].

Usually, for the recognition of red blood cells the shape is applied as the feature representing an object. However, the greyscale is also applicable (e.g. [Fre11], [Fre18], [Fre19]). This feature is less popular than other ones. Additionally, the algorithms are usually utilized for the whole image, yet they can be applied for the extracted object as well. In this manner, Scale-Invariant Feature Transform (SIFT) is especially popular [Low04]. A modification of this method (so-called extended SIFT) was also proposed [KeY04]. The Speeded Up Robust Features (SURF) approach is a second popular algorithm designed for this goal [Bay08]. Similarly, the Scale-Invariant Shape Features (SISF) were utilized in the problem of detection based on greyscale [Jur04]. On the other hand, human detection was also an area of interest, and for this task the Histograms of Oriented Gradients (HoG) were applied [Dal05]. The histogram for a scene in greyscale was also used [Chi11].

Texture analysis can be in some application very close to the usage of greyscale descriptors. Some examples of texture descriptors that could be applied in that way are: Local Binary Pattern with Local Phase Quantization [Nan16a], connectivity indexes in local neighbourhoods [Flo16a], Gaussian Markov Random Fields [Dha14], Gabor features [Kum18], non-uniform patterns [Nan16b], genetic algorithms [Wan17], local fractal dimensions [Flo16b], Fisher tensors with 'bag-of-words' [Far14], and approach based on morphology [Apt11].

3 PROPOSED APPROACH

The algorithm described in this paper is based on the idea given by T. W. Rauber and A. S. Steiger-Garcao for shape representation [Rau94]. Proposed by them shape representation, called UNL (named after the Universidade Nova de Lisboa), was based on the assumption that derived polar coordinates for silhouette points were put into a matrix, in which the row corresponds to the distance from centroid, and the column—to the angle. The difference in the resultant representation from the traditional polar transform for a shape can be easily noticed—see Figure 1.

In this paper above-mentioned idea is extended to the greyscale object representation. An example of a greyscale object and its transformed representation by means of the proposed approach is given in Figure 2.

The proposed algorithm is similar to the Polar-Fourier Greyscale Descriptor, proposed in [Fre11]. The differ-

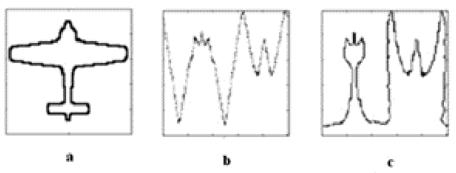


Figure 1: The difference in shape representation for object (a) between polar transform (b) and UNL-transform (c).

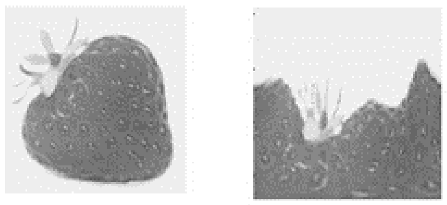


Figure 2: The result of the transformed greyscale object by means of the proposed approach proposed: original object on the left and its transformed representation on the right.

ence between the two approaches is explained in Section 4.3.

Similarly to the Polar-Fourier Greyscale Descriptor, the proposed algorithm applies some additional pre-processing steps, improving the obtained representation, namely: median filtering, low-pass filtering, the construction of the constant rectangle containing the object, filling the gaps with the background colour, and resizing the transformed image to the constant size.

The algorithm for computing the resultant greyscale object representation is given in details below.

Step 1. Median filtering of the input sub-image I, with the kernel size 3.

Step 2. Low-pass filtering, realized through the convolution with mask 3×3 pixels, and normalization parameter equal to 9.

Step 3. Calculation of the centroid by means of the moments [Hup95]. Firstly, m_{00}, m_{10}, m_{01} are derived, and then the centroid $O = (x_c, y_c)$:

$$m_{pq} = \sum_x \sum_y x^p y^q I(x,y), \tag{1}$$

$$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}}. \tag{2}$$

Step 4. Finding the maximal distances d_{maxX}, d_{maxY} for X- and Y-axis respectively from the boundaries of I to O.

Step 5. Expanding the image into both directions by $d_{maxX} - x_c$ and $d_{maxY} - y_c$ and filling in the new parts using constant greyscale level, e.g. 127.

Step 6. Derivation of new coordinates and insertion in the image P, in which the row corresponds to the distance from centroid (ρ_i), and the column—to the angle (θ_i):

$$\rho_i = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2},$$

$$\theta_i = \operatorname{atan}\left(\frac{y_i - y_c}{x_i - x_c}\right). \tag{3}$$

Step 7. Resizing P to the constant rectangular size, $n \times n$, e.g. $n = 128$.

Step 8. Derivation of 2D Fourier transform:

$$C(k,l) = \frac{1}{HW} \left| \sum_{h=1}^{H} \sum_{w=1}^{W} P(h,w) \cdot \exp^{\left(-i\frac{2\pi}{H}(k-1)(h-1)\right)} \exp^{\left(-i\frac{2\pi}{W}(l-1)(w-1)\right)} \right|, \tag{4}$$

where:
H, W—height and width of P,
k—sampling rate in vertical direction ($k \geq 1$ and $k \leq H$),
l—sampling rate in horizontal direction ($l \geq 1$ and $l \leq W$),
$C(k,l)$—the coefficient of discrete Fourier transform in k-th row and l-th column,
$P(h,w)$—value in the resultant image plane with coordinates h, w.

Step 9. Selection of the spectrum sub-part, e.g. 10×10 size and concatenation into vector V.

4 THE ALGORITHMS APPLIED FOR THE EXPERIMENTAL COMPARISON

The proposed algorithm, described in the previous section, was compared and experimentally investigated with several other algorithms for greyscale object representation. The selection was based on an assumption that a popular method for object localization in greyscale images and the one popular for texture representation should be evaluated. Both are applied in object description as well. Also, two previous algorithms for greyscale objects representation proposed by the Author were used. That gave four algorithms that were compared with the proposition given in the paper. Their description is given briefly in the following subsections.

4.1 Scale-Invariant Feature Transform

The Scale-Invariant Feature Transform (SIFT) was proposed by D. G. Lowe [Low04], [Low11]. The description obtained using it is invariant to rotation and scaling. The algorithm is very popular as a method for feature detection. However, it can be easily applied as a representation of an object in greyscale. Therefore, in works described in this paper it is utilized that way. Moreover, some approaches were proposed in order to speed-up the derivation of the representation [Hey07].

4.2 The Gabor descriptor

The Gabor filter is mainly used as a texture descriptor [Zha09]. However, its practical usefulness in representing the grey levels lead to its applicability also in a form of a greyscale descriptor. Similarly to the previously mentioned algorithm, the Gabor filter is applied in the experiments described in the following section.

4.3 The Polar-Fourier Greyscale Descriptor

The Polar-Fourier Greyscale Descriptor was proposed in [Fre11] and is in fact a source for the descriptor applied in this paper. Hence, it is very similar to the algorithm described in the previous section. The significant difference occurs in **Step 6**. In the original Polar-Fourier Greyscale Descriptor simple polar transform is applied. The proposed in this paper approach is similarly to the UNL-transform more sophisticated, what was explained in the introductory part of Section 3.

4.4 The Descriptor based on Polar Transform and Vertical and Horizontal Projections

The approach applying the polar transform and projections has similar initial stages as the Polar-Fourier Greyscale Descriptor, however the rest is different, since instead of Fourier transform the vertical and horizontal projections are applied. The whole descriptor is given as follows [Fre18]:

Step 1. Median filtering of the input image I, with the kernel size 3.

Step 2. Low-pass filtering, realized through the convolution with mask 3×3 pixels, and normalization parameter equal to 9.

Step 3. Calculation of the centroid by means of the moments [Hup95]. Firstly m_{00}, m_{10}, m_{01} are derived, and later the centroid $O = (x_c, y_c)$:

$$m_{pq} = \sum_x \sum_y x^p y^q I(x,y), \tag{5}$$

$$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}}. \tag{6}$$

Step 4. Transforming I into polar coordinates (resultant image is denoted as P), by means of the formulas:

$$\rho_i = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2},$$

$$\theta_i = \operatorname{atan}\left(\frac{y_i - y_c}{x_i - x_c}\right). \tag{7}$$

Step 5. Resizing P to the constant rectangular size, $n \times n$, e.g. $n = 128$.

Step 6. Deriving the horizontal and vertical projections of P:

$$H_i = \sum_{j=1}^{n} P_{i,j}, \quad V_j = \sum_{i=1}^{n} P_{i,j}. \tag{8}$$

Step 7. Concatenating the obtained vectors H and V into one, $C = HV$, representing an object.

5 EXPERIMENTAL RESULTS

Before the representation and identification of particular red blood cells, they have to be segmented and extracted. For this purpose the approach proposed in [Fre10] was applied. It starts with the conversion of the input image into greyscale. Later the modified histogram thresholding is applied in order to obtain the binary image. Then, particular objects can be localized. This process is performed by tracing regions of each separate objects. Only objects entirely placed within a processed image are considered. The area of extracted regions is analysed in order to reject thrombocytes and leukocytes. In order to limit the processed objects to erythrocytes, if a region is larger (for leukocytes) or smaller (for thrombocytes) then it is rejected (the difference in size for these three main objects of interest is provided in Figure 3). This assumption results from the fact that leukocytes (e.g. monocytes) have the size of 40 μm, thrombocytes—ca. 2 μm, and the size of erythrocytes varies between 6 and 12 μm. Additionally, this process rejects some occluded objects, what is a benefit, since they are difficult to recognise. Nevertheless, it is still possible that some undesirable objects remain, if their area is similar to erythrocytes. In order to avoid this problem, the analysis of object's histogram is applied, since this feature varies for different particles—thrombocytes and leukocytes have almost black parts inside. Moreover, the histogram equalisation of the image before the binarisation is performed, what reduces the number of occluded shapes.

As a result of the above-described, by means of using the established coordinates of a single cell, the rectangular subpart of the greyscale image with it is extracted and processed in next stages.

The experiments were performed using 55 May-Grunwald-Giemsa stained microscopic images magnified 1,000 times. All of them were converted into greyscale, and then every cell was localised and extracted separately by means of the above-described approach. For each object the proposed algorithm as

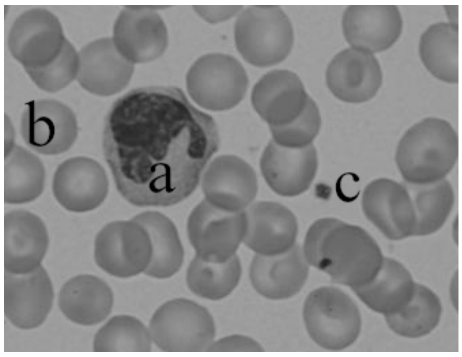


Figure 3: Three different types of human blood cells, varying in size: a) leukocyte, b) erythrocyte, c) thrombocyte [Omi14].

well as the algorithms described in the previous section were applied in order to obtain the descriptions. They were matched using the dissimilarity measure with the stored templates that were represented in the same way. Twelve classes of erythrocytes were applied. For each class five template objects were used. That gave in result 60 templates, since 12 classes of erythrocytes were considered.

The number of cells within a single MGG image varies significantly. In case of the images applied for the experiments, this number varies from a dozen to more than a hundred. In order to emphasize the number of analysed and processed objects, some statistical information will be given. It is limited to the number of extracted erythrocytes in the most limited case. After the rejection of larger (leukocytes and occluded cells) and smaller (thrombocytes) objects the overall number of processed objects of interest was equal to 2,772. The average for an image was (rounded) 50, and the median value was 44. The smallest number of properly extracted erythrocytes was equal to 10 objects, and the highest was 102.

The obtained efficiency for particular erythrocyte types and greyscale descriptors applied in the experiment is given in Table 1. Additionally, in the table average efficiency for all analysed descriptors is provided.

6 CONCLUDING REMARKS AND FUTURE PLANS

In the paper an algorithm for description of greyscale objects extracted from digital images was applied for recognition of human red blood cells visible on digital microscopic images. This is a part of an approach for automatic (or semi-automatic) diagnosis of selected diseases based on the deformation of erythrocytes. The disease is concluded by means of the recognition of types of red blood cells visible on a digital microscopic image, stained using MGG method and converted into greyscale.

The proposed algorithm was experimentally investigated in comparison with four other greyscale descriptors. It obtained the best results by means of the efficiency in recognizing the types of erythrocytes.

Nevertheless, future works are planned on further improvement in the representation of objects extracted from digital images and represented using greyscale as a feature. Another approaches are planned to be analysed.

Moreover, since the representation of cells was the main topic of the works described in this paper, more attention should be put on the classification stage. Here, the simplest method based on the template matching was applied. It was assumed that some more sophisticated and novel approaches should provide even better overall results. Hence, the second direction of research should be related to the analysis of the methods belonging to the last steps of the method. For example, application of some deep learning algorithms, very popular nowadays, would definitely lead to better results.

Finally, the Electron Microscopy seems to be a tempting object of future works. The tasks would be slightly different in that case, e.g. the rejection of erythrocytes could be performed in order to pre-process an image for other applications. However, since objects visible on EM images are spatial, some significant modifications of the proposed description algorithm should be made.

7 REFERENCES

[Apt11] Aptoula, E., Lefèvre, S.: Morphological texture description of grey-scale and color images. Advances in Imaging and Electron Physics, Vol. 169, pp. 1–74, 2011.

[Bay08] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L. SURF: Speeded Up Robust Features. Computer Vision and Image Understanding, Vol. 110, Iss. 3, pp. 346–359, 2008.

[Bro00] Bronkorsta, P.J.H., et al. On-line detection of red blood cell shape using deformable templates. Pattern Recognition Letters, Vol. 21, Iss. 5, pp. 413–424, 2000.

[Chi11] Chin, T.J., Suter, D., Wang, H. Boosting histograms of descriptor distances for scalable multiclass specific scene recognition. Image and Vision Computing, Vol. 29, Iss. 4, pp. 241–250, 2011.

[Dal05] Dalal, N., Triggs, B. Histograms of oriented gradients for human detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), Vol. 1, pp. 886–893, 2005.

[Dha14] Dharmagunawardhana, C., Mahmoodi, S., Bennett, M., Niranjan, M. Gaussian Markov random field based improved texture descriptor for image segmentation. Image and Vision Computing, Vol. 32, Iss. 11, pp. 884–895, 2014.

[Dia09] Dıaz, G., Gonzalez, F. A., Romero, E. A semi-automatic method for quantification and classification of erythrocytes infected with malaria parasites in microscopic images. Journal of Biomedical Informatics, Vol. 42, Iss. 2, pp. 296–307, 2009.

[DiR02] Di Ruberto C, Dempster A, Khan S, Jarra B. Analysis of infected blood cell images using morphological operators. Image and Vision Computing, Vol. 20, pp. 133–146, 2002.

[Don07] Dong, Y. et al., Evaluations of deep convolutional neural networks for automatic identification

	SIFT	Gabor	P-F GD	polar transform and projections	proposed descriptor
schistocyte	85%	75%	69%	61%	92%
dacrocyte	93%	86%	83%	72%	95%
acantocyte	98%	94%	98%	89%	100%
echinocyte	96%	81%	89%	88%	95%
ovalocyte	78%	70%	74%	69%	86%
normocyte	94%	95%	97%	86%	100%
stomatocyte	90%	87%	90%	83%	98%
codocyte	80%	76%	72%	65%	80%
spherocyte	100%	98%	100%	93%	100%
leptocyte	100%	98%	97%	91%	100%
annular eryth.	82%	80%	76%	64%	84%
drepanocyte	92%	78%	81%	74%	90%
average	**91%**	**85%**	**86%**	**78%**	**93%**

Table 1: Average recognition rates (efficiency) obtained for particular classes of erythrocytes by means of experimentally investigated greyscale descriptors.

of malaria infected cells. 2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), Orlando, FL, pp. 101–104, 2017.

[Far14] Faraki, M., Harandi, M.T., Wiliem, A., Lovell, B.C. Fisher tensors for classifying human epithelial cells. Pattern Recognition, Vol. 47, Iss. 7, pp. 2348–2359, 2014.

[Flo16a] Florindo, J.B., Landini, G., Bruno, O.M. Three-dimensional connectivity index for texture recognition. Pattern Recognition Letters, Vol. 84, pp. 239–244, 2016.

[Flo16b] Florindo, J.B., Bruno, O.M. Local fractal dimension and binary patterns in texture recognition. Pattern Recognition Letters, Vol. 78, pp. 22–27, 2016.

[Fre10] Frejlichowski D. Pre-processing, extraction and recognition of binary erythrocyte shapes for computer-assisted diagnosis based on MGG images. In: Bolc, L. et al. (eds.): ICCVG 2010, Part I. Lecture Notes in Computer Science, Vol. 6374, pp. 368–375, 2010.

[Fre11] Frejlichowski, D. Identification of erythrocyte types in greyscale MGG images for computer-assisted diagnosis. IbPRIA 2011, Lecture Notes in Computer Science, Vol. 6669, pp. 636–643, 2011.

[Fre18] Frejlichowski, D. A new algorithm for greyscale objects representation by means of the polar transform and vertical and horizontal projections. ACIIDS 2018: Intelligent Infor-mation and Database Systems, Lecture Notes in Artifical Intelligence, Vol. 10752, pp. 617–625, 2018.

[Fre19] Frejlichowski, D. Low-Level Greyscale Image Descriptors Applied for Intelligent and Contextual Approaches. In: Nguyen N., Gaol F., Hong TP., Trawiński B. (eds) Intelligent Information and Database Systems. ACIIDS 2019. Lecture Notes in Computer Science, Vol. 11432, pp. 441–451, 2019.

[Hey07] Heymann, S. Maller, K., Smolic, A., Froehlich, B. and Wiegand, T. SIFT implementation and optimization for general-purpose GPU. Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 317–322, 2007.

[Hup95] Hupkens, T.M., de Clippeleir, J. Noise and intensity invariant moments. Pattern Recognition Letters, Vol. 16, Iss. 4, pp. 371–376, 1995.

[Jur04] Jurie, F., Schmid, C. Scale-invariant shape features for recognition of object categories. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), Vol. 2, II-90–II-96, 2004.

[KeY04] Ke, Y., Sukthankar, R. PCA-SIFT: a more distinctive representation for local image descriptors. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), Vol. 2, II-506–II-513, 2004.

[Kum18] Kumar, M., Singh, Kh. M. Retrieval of headneck medical images using Gabor filter based on power-law transformation method and rank BHMT. Signal, Image and Video Processing, Vol. 12, Iss. 5, pp. 827–833, 2018.

[Low04] Lowe, D. G. Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, Vol. 60, Iss. 2, pp. 91–110, 2004.

[Low11] Low, Y.-Q., Lee, S.-W., Goi, B.-M., and Ng, M.-S. A New SIFT-based camera calibration method for hybrid dual-camera. In: A. Abd Manaf et al. (Eds.): ICIEIS 2011, Part II, CCIS, Vol. 252, pp. 96–103, 2011.

[Lue05] Luengo-Oroz, M. A., Angulo, J., Flandrin, G., Klossa, J. Mathematical morphology in polar-logarithmic coordinates. Application to erythrocyte shape analysis. In: Marques, J. S., Perez de la Blanca, N., Pina, P. (eds.) IbPRIA 2005. Lecture Notes in Computer Science, Vol. 3523, pp. 199–205, 2005.

[Nan16a] Nanni, L., Melucci, M. Combination of projectors, standard texture descriptors and bag of features for classifying images. Neurocomputing, Vol. 173, pp. 1602–1614, 2016.

[Nan16b] Nanni, L., Brahnam, S., Lumini, A. A simple method for improving local binary patterns by considering non-uniform patterns. Pattern Recognition, Vol. 45, Iss. 10, pp. 3844–3852, 2012.

[Omi14] Omid, S., Rabbani, H., Talebi, A., and Banaem, H.U. Selection of the best features for leukocytes classification in blood smear microscopic images. In SPIE Medical Imaging, pp. 90410P. International Society for Optics and Photonics, 2014.

[Ran07] Ranzato, M. et al. Automatic recognition of biological particles in microscopic images. Pattern Recognition Letters, Vol. 28, Iss. 1, pp. 31–39, 2007.

[Rau94] Rauber, T. Two-dimensional shape description. Technical report, University Nova de Lisboa, Portugal, 1994.

[Ros06] Ross, N.E., Pritchard, C.J., Rubin, D.M. et al. Automated image processing method for the diagnosis and classification of malaria on thin blood smears. Medical and Biological Engineering and Computing, Vol. 44, pp. 427–436, 2006.

[Sab04] Sabino, D. M. U., da Fontoura Costa, L., Rizzatti, E. G., Zago, A. M. A texture approach to leukocyte recognition. Real-Time Imaging, Vol. 10, Iss. 4, pp. 205–216, 2004.

[Son02] Song, X. B., Abu-Mostafa, Y., Sill, J., Kasdan, H., Pavel, M. Robust image recognition by fusion of contextual information. Information Fusion, Vol. 3, Iss. 4, pp. 277–287, 2002.

[Ven13] Venkatalakshmi, B. and Thilagavathi, K. Automatic red blood cell counting using Hough transform. 2013 IEEE Conference on Information & Communication Technologies, Thuckalay, Tamil Nadu, India, pp. 267–271, 2013.

[Ver15] Verma, M., Raman, B. Center symmetric local binary co-occurrence pattern for texture, face and bio-medical image retrieval. Journal of Visual Communication and Image Represen-tation, No. 32, pp. 224–236, 2015.

[Wan17] Wang, S. et al. Texture analysis method based on Fractional Fourier Entropy and Fitness-scaling Adaptive Genetic Algorithm for detecting left-sided and right-sided sensorineural hearing loss. Fundamenta Informaticae, Vol. 151, Iss. 1–4, pp. 505–521, 2017.

[Zha09] Zhan, X., Xingbo, S., Yuerong L. Comparison of two gabor texture descriptor for texture classification. In: 2009 WASE International Conference on Information Engineering, pp. 52–56, Taiyuan, Chanxi, 2009.

Evaluation of Model-Based Tracking and its Application in a Robotic Production Line

Fabian Scheer
Daimler Protics GmbH
Fabian.scheer@daimler.com

Markus Neumann
Daimler Protics GmbH
Markus.n.neumann@daimler.com

Konrad Wirth
Daimler AG
Konrad.wirth@daimler.com

Marvin Ginader
Daimler AG
Marvin.ginader@daimler.com

Yavuz Oezkurt
Daimler AG
yavuz.oezkurt@daimler.com

Stefan Mueller
University of Koblenz-Landau
stefanm@uni-koblenz.de

ABSTRACT

The assessment of the accuracy and stability of model-based tracking approaches in the literature is a challenging task. Many of the presented methods lack a detailed evaluation or comparison to accurate ground truth data. Considering real world applications, it is hard to estimate the applicability of the respective method. In this paper we present methods to evaluate the accuracy and stability of model-based tracking approaches. Thereby, we focus on an industrial use and the robustness of the approach considering error sources like the influence of lighting over time, noise and the mechanical camera setup in general. The second part of this paper deals with the evaluation and application of model-based tracking in a robotic production line for gap measurements in the automotive industry. We present a method for synchronization with a conveyor setup and an operating robot. In this complex setup various error sources influence the output accuracy of the system. A further accuracy evaluation method is presented that uses the robot as a measurement device of the total error. The paper closes with the application of a model-based tracking system for robotic measurement tasks in a production line.

Keywords
evaluation, model-based tracking, evaluation methods, synchronization to conveyor, robotic production line

1. INTRODUCTION

While doing the research in the literature of model-based tracking (MBT) approaches we found that most presented papers lack a detailed evaluation concerning the achieved accuracy in comparison to ground truth data. The influence of different camera resolutions, camera types or the influence of interferences like lighting, noise or mechanical setup is often poorly evaluated or different non comparable evaluation methods are used. With this in mind we present a tool set of methods in this paper to evaluate MBT techniques. For the research community and the industry this toolset can be of special interest to assess the accuracy, stability and feasibility of a MBT approach for their desired use case. Correlations between different error sources and their influence to the results become clearer and are easier to evaluate. To allow researchers the evaluation of their own MBT approach we plan to publish a CAD model of a test specimen, camera intrinsic parameters, a video sequence of shifts of the specimen and the associated ground truth data recorded with a precise measurement device.

Automated industrial vision based solutions in cooperation with robots that work on a conveyor become more and more important in the context of future industrial manufacturing and are accompanied by several international initiatives like industry 4.0, smart factories or the industrial internet consortium. In this regard MBT can play an important role but several challenges have to be mastered. Based on this motivation this paper presents a method to synchronize the vision based MBT with a conveyor and robots. The synchronization is evaluated in an application scenario of automatic measuring of gaps on body shells with a light weight robot.

2. RELATED WORK

Garon et al. [Gar18] gives a good overview over the current state of the art of frameworks and datasets for the evaluation of six degrees of freedom object trackers. Derived from the drawbacks of existing methods they propose a new method based on ground truth data captured with a Vicon motion capture system. They use complex error-prone transformation chains for their setup with a Microsoft Kinect. It is known that vision systems in combination with

	X	Y	Z	rotX	rotY	rotZ
Span	166,67	74,151	38,4653	14,565119	20,282	23,1848
std.dev.(σ)	0,30	0,24	0,19	0,03	0,02	0,12
mean(µ)	0,29	0,27	0,21	0,03	0,02	0,07
Quantity	149		Camera		4110 x 3006 Color	
	X	Y	Z	rotX	rotY	rotZ
Span	52,426	71,67	33,131	1,249	15,002	23,123
std.dev.(σ)	0,12	0,25	0,24	0,04	0,02	0,03
mean(µ)	0,13	0,30	0,19	0,02	0,03	0,04
Quantity	77		Camera		4110 x 3006 B/W	
	X	Y	Z	rotX	rotY	rotZ
Span	110,572	66,155	96,461	4,13	7,35	3,16
std.dev.(σ)	0,34	0,54	0,51	0,02	0,02	0,02
mean(µ)	0,30	0,44	0,51	0,03	0,04	0,02
Quantity	41		Camera		2560 x 1920 Color	

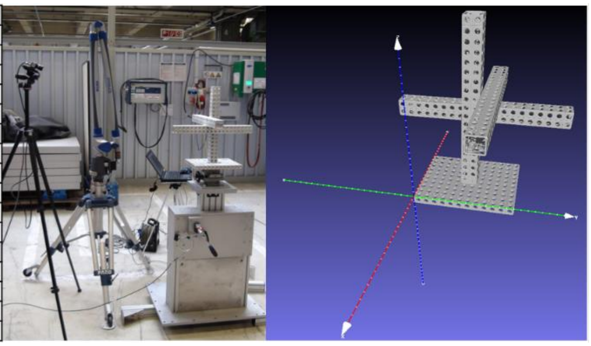


Figure 1: Results for Faro arm and test specimen. 3 different camera resolutions are used. Translations are in millimeters and rotations in degrees. Right: Experimental setup and scanned model of test object.

capturing systems like Vicon need calibration procedures that afflict the outcome results. Due to that, the results are mostly in the millimeter (mm) range. Devices of the measurement technology achieve better results and are in the submillimeter range. An independent measurement solution regarding the sensor technology for the ground truth data would be more appropriate.

Several evaluation techniques use fiducials like [Hin11] or a checker board [Wu17]. [Hod17] also gives a good overview of the state of the art regarding datasets for the evaluation of 6D pose estimation for texture less objects. Unfortunately they also use fiducials and manually aligned poses to define ground truth data.

The existing methods try to combine a lot of test cases for an evaluation, like textured or untextured objects, robustness to occlusion or clutter, detection rates, training of artificial intelligence methods and so on. As already mentioned in [Gar18] most evaluation datasets and frameworks lack high precise ground truth data which is comparable to the results of professional measurement technology. Therefore we present methods that rely on ground truth data captured with an independent measurement technology. The setup is kept simple and designed in a way that the results can be compared directly.

For our evaluation we use the model-based tracking approach of [Wue07] that is based on [Com06] and [Vac04]. The method was further developed by the authors and has reached a high degree of maturity. This was important in our selection process since we are focused on an industrial use case. We only considered one MBT method because of the great expense for all evaluations presented in this paper. For initialization, [Wue07] rely on a real camera perspective that closely matches a given CAD perspective. They use a real-time image space line model generation method as detailed in [Nie03]. After the matching in the initial position a continuous tracking can be achieved. In Figure 2 the right image shows an augmentation of an image with the MBT.

3. EXPERIMENTAL SETUP AND EVALUATION RESULTS

Our evaluation consists of different methods to determine the accuracy of a MBT approach. We used precise measurement devices like a Faro arm [Far20] and a Faro laser tracker [Far20] to obtain ground truth data. Various test objects are used in the tests: a test specimen constructed with standard mechanical engineering elements of aluminum, car body shells and assembled coated cars. To keep things simple and avoid complex transformation chains we focused on the evaluation of a simple transform of the test object. Thereby, the pose of the test object was measured in an initial position. Then the object was moved and measured again. The transformation between the two poses is our evaluation criterion. The MBT delivers results in the coordinate system of the CAD object. The same CAD model of the test object was used in the calibration of the ground truth measurement system to the test object. With this setting, we avoid complex coordinate system calibrations. That makes the transformation of the ground truth and the MBT data directly comparable.

	X	Y	Z	rotX	rotY	rotZ
Span	278,805	76,07	12,11	0,85	0,48	1,84
std.dev.(σ)	0,77	0,51	0,50	0,01	0,04	0,05
mean(µ)	0,73	0,57	0,56	0,01	0,03	0,02
Quantity	139		Camera		4110 x 3006 Color	
	X	Y	Z	rotX	rotY	rotZ
Span	135,5	67,23	9,82	0,82	0,27	1,8
std.dev.(σ)	0,51	0,72	0,37	0,01	0,01	0,01
mean(µ)	0,65	0,82	0,38	0,01	0,01	0,01
Quantity	87		Camera		2560 x 1920 Color	

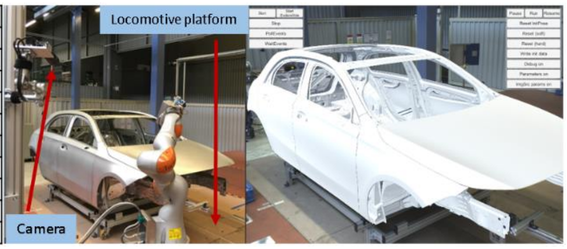


Figure 2: Evaluation results with Faro arm, Faro laser tracker and body shell. Two camera resolutions are tested. The measurements are in millimeter for the translation and in degrees for the rotation.

We tested the MBT with a camera resolution of 4110 x 3006 (12 megapixels (MP)) and 2560 x 1920 (4 MP) pixels. A uEye UI-3000SE-C-HQ USB 3.1 camera with global shutter and 12 mm lens, as well as the same camera as black and white version was used. For the 2560x1920 pixel resolution a UI-3580CP-C-HQ Rev.2 USB 3.0 camera with a 6 mm lens was used.

3.1 Faro arm with test specimen

The first experiment was conducted with a test specimen assembled of robustly connected standard aluminum elements from the mechanical engineering sector. For ground truth measurements a Faro arm was used.

3.1.1 Test setup

To avoid errors related to the CAD modeling process, the test specimen was scanned with a high precision GOM 3D scanner [Gom20]. The scanned CAD model was used for the MBT and for the calibration of the faro arm to the test specimen. The extent of the specimen was about 300 mm on the longest side and the temperature during the measurements was in the range of 20 to 30 degrees Celsius. Therefore linear expansions caused by temperature fluctuations lie in the range of approximately 0.07 mm and could be neglected for the evaluation. As described above, the test specimen was measured in an initial position by the Faro arm and the MBT. Then the object was moved and measured again with both systems. Thus, the transformation between the two poses of both systems can be compared directly.

We calculated the mean and standard deviation (std. dev.) of the absolute values for every axis of the translation and the rotation. The span of all movements was also recorded and is shown in Figure 1. To assess the influence of resolution and bayer color interpolation on the quality of the MBT, three different cameras were tested. The first one is a USB 3.1 12 MP global shutter color camera with a 12mm lens. The second is the same camera as a black and white version. The third camera is a USB 3.0 4 MP global shutter color camera with a 6mm lens.

3.1.2 Results

For the evaluation with the 12 MP color camera 149 measurements were conducted, for the 12 MP B/W camera 77 measurements and for the 4 MP color camera 41 measurements. The span of the specimen movements were in the range of up to 166 mm for the translation and up to 23 degrees for the rotation. We limited our tests to these small distances due to our industrial use case. This is explained in more detail down below. The distance from the specimen to the camera varied roughly between 1,5 to 3 meters. The coordinate system of the specimen sets the z-axis as up axis and uses a right handed coordinate system. Compared to the transformation between the two measured poses of the ground truth data, the MBT achieved results in the submillimeter range for all three camera types and for both camera resolutions. Rotation values are accurate to the second decimal place for nearly all values. It can be seen that with a higher resolution the results of the MBT improve in comparison to the ground truth data. These improvements apply to the mean and to the standard deviation of the absolute values of the difference between the two poses as well. To investigate the impact of Bayer demosaicing a measurement series with a B/W camera was conducted. Differences in the results are hardly noticeable and in most cases only concern the second decimal place. Considering the results in total we decided to use the 12 MP color camera for our further experiments and the industrial use case. The improvements of the B/W camera in the results were small and the usage of color information gives us further opportunities in our industrial use case considering coated car bodies.

3.2 Faro arm and Faro laser tracker with car body shell

The second experiment was conducted with a car body shell on a locomotive platform. The platform could be moved along one axis in each of its directions. For ground truth measurements a Faro arm was used. Since the range of the Faro arm is limited we also used a Faro laser tracker for contactless measurements of larger platform displacements.

3.2.1 Test setup

The test setup is similar to the one before. In an initial position the body shell was measured by one of the Faro devices and then by the MBT system. Then the locomotive platform was moved along the x-axis and the measurement of both systems was repeated. The transformation between the two measured poses for both systems were calculated and the mean and standard deviation of the absolute values of the transformation were determined per axis. To overcome the limitation of moving the body shell in the x axis only, we also moved the body shell in the y direction by hand. To consider movements on the z axis (up), we put objects under the body shell in order to lift it up. The span of all movements was recorded again. For the evaluation of the 12 MP and the 4 MP color camera we used the same setting as described above. The CAD model of the body shell was delivered by the planning department and can slightly differ to the real body shell up to a range of ± 0.5 mm.

3.2.2 Results

For the evaluation with the 12 MP color camera, 139 measurements were conducted and for the 4 MP color camera there were 87 measurements. The span of the movements of the specimen was in the range of up to 278 mm in the translation and up to 2 degrees in the rotation. The rotational movements were also adjusted by hand and due to the weight of the body shell, these

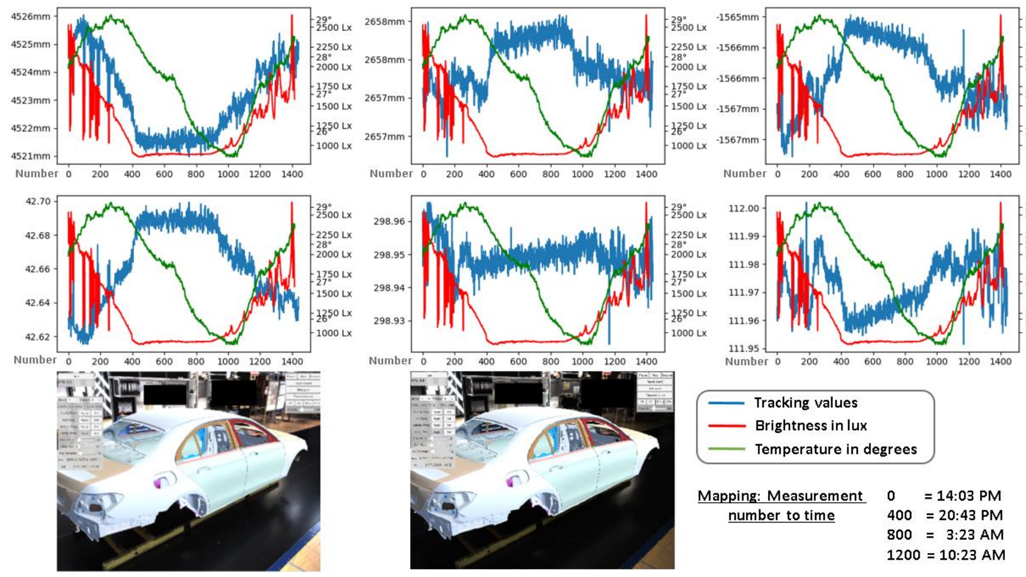


Figure 3: Lighting test results over 24 hours. Horizontal axis: measurement number. Vertical axis left: top row, translation in X, Y and Z in millimeters and mid row, rotation in x, y and z in degrees. Vertical axis right: Brightness in lux and temperature in degrees

rotational values are small. The right handed coordinate system of the body shell sets the z-axis as up axis. The x axis points along the car main axis and the y axis points through the mid of the front wheel on the right. Compared to the ground truth data for the translation, the results of the MBT lie in the submillimeter range and are accurate to the second decimal place for the rotation. This applies to both cameras and resolutions as shown in Figure 2. The usage of the 4 MP camera seems to have a slight advantage considering the accuracy but both results are close together. The total amount of measurements captured with the 4 MP camera is smaller and may look different if the same amount of measurements would have been conducted. Nevertheless we can assess a high accuracy for the MBT compared to the ground truth data for the considered span of movements in our tests. Since the results of the 12 MP camera were slightly better than the results of the 4 MP camera in the tests with the test specimen, we decided to use the 12 MP for our further experiments.

3.3 Lighting and temperature over time

The third experiment investigates the influence of lighting and temperature changes over time to the MBT. The tests were conducted under real operational conditions of our industrial use case. We set a body shell on a conveyor in a production hall and triggered a measurement with the MBT system every minute for 24 hours. We also measured the brightness of the scene in lux and the temperature in degrees Celsius. Figure 3 shows the results and correlations of the particular data for a 24 hour test with 1440 measurements. Besides, the outside lighting conditions at the test location had a direct effect to the indoor lighting due to sloped roof windows with milk glass. This caused some abrupt peaks in the results that can be seen in Figure 3. We started our tests at 2 p.m.. The mapping of measurement numbers to time stamps is depicted in Figure 3 for chosen values. As expected, the lighting conditions correlate directly to the results of the MBT. This can be seen best for the translation in x direction in Figure 3. Comparing the temperature and tracking results, a correlation is not as obvious. In Figure 3 the translation in z may show a trend that with sinking temperature the z values also decrease. The same is observed for the rotation around the z axis.

	X	Y	Z	RotX	RotY	RotZ
Span	3.28	1.66	1.68	0.04	0.03	0.03
μ	4508.6	2668.8	-1571.2	42.80	299.14	112.23
σ	0.35	0.35	0.28	0.006	0.006	0.007

Table 1. Results of the 24 hour test with span, mean (μ) and standard deviation(σ) values. Translations are in mm and rotations in degrees.

We tried different camera settings and achieved the best and most stable results considering the span and standard deviation of the results with an automatic exposure. The results of a 24 hour test is shown in Table 1. The MBT lies in the 2σ range of ± 0.7 mm for the translation and ±0.014° for the rotation. Regarding the total span, we found a total range of up to 3.28 mm for the translation that is caused by some outliers.

3.4 Noise

The fourth experiment aims towards the investigation of the jitter/noise behavior of the MBT. The setting for this test was the same given in the third experiment where a body shell in a production hall was used. Therefore 30 measurements of the MBT were taken successively. Table 2 shows results for one of these tests with 30 measurements. We repeated these tests

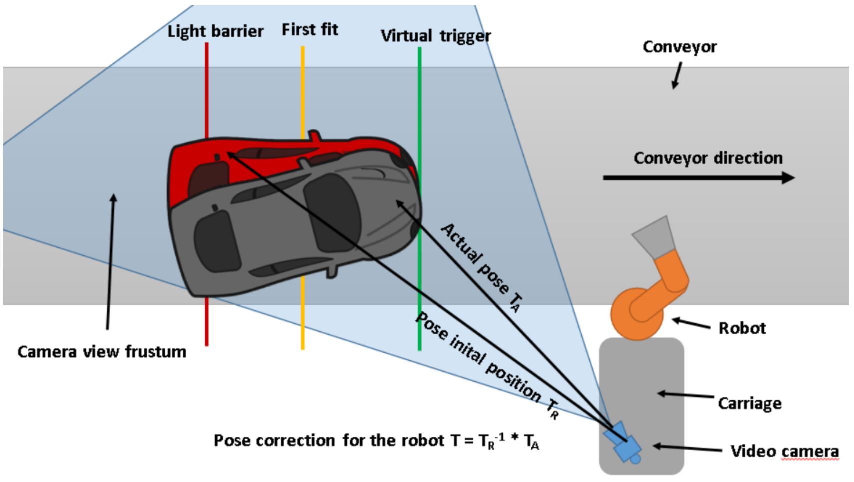


Figure 4: System setup for the robot measurement evaluation.

various times throughout our evaluation. It can be stated that the std. dev. throughout all tests only varied in the second decimal place for the translation and in the third decimal place for all rotation values. Table 2 also shows the column "dist". It contains the maximum distance to the mean value and is helpful to get an overview of the peaks of the values. Considering the fact that the values are given in mm and degrees, we found the MBT to be very stable and it achieves good results concerning jitter and noise.

	X	Y	Z	RotX	RotY	RotZ
dist	0.12	0.16	0.21	0.003	0.003	0.005
μ	4521.21	2659.38	-1565.02	42.67	298.93	111.99
σ	0.04	0.07	0.08	0.001	0.001	0.001

Table 2: Results of the noise test with maximum distance to the mean, mean (μ) and standard deviation (σ) values.

3.5 Setup and dismantling of the camera

The fifth experiment deals with the question if a camera can be physically dismantled and set up again while achieving similar tracking results afterwards. Therefore a special camera mount was needed. We

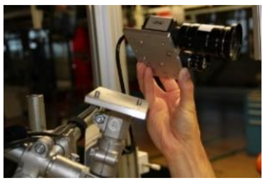


Figure 5: Camera dismantling with pinned fitting.

mounted the camera on a steel plate that was mounted on another steel plate with a connection of a pinned fitting (Figure 5). For the tests we measured a body shell with the MBT. Then we removed the alignment pins and the camera and then put the camera and the pins back on the second plate. Afterwards, a MBT measurement was triggered. We repeated this procedure six times.

Table 3 shows the mean and the std. dev. values of the absolute difference of the values per axis. The worst result is a submillimeter mean value of 0.8 mm on the z axis. The rotation only varies in the second decimal place. For applications requiring a dismantling of cameras in a calibrated setting this may be of interest to assess the impact on the accuracy.

	X	Y	Z	RotX	RotY	RotZ
μ	0.086	-0.046	0.822	-0.010	0.067	0.049
σ	0.208	0.244	0.166	0.004	0.014	0.014

Table 3: Results of camera setup and dismantling with mean (μ) and standard deviation(σ) values.

4. APPLICATION AND EVALUATION IN A PRODUCTION LINE

After the general investigations concerning the accuracy and stability of the MBT we show the application and evaluation in an industrial scenario. Therefore the MBT is used to track body shells on a conveyor. The results are send to a robot in order to navigate to points where gaps should be measured. Collisions with the body shell should be avoided.

4.1 Motivation of the scenario

Most industrial robot applications rely on an installation and configuration step in which all components are put into operation. For our use case the points for the gap measurements are taught to a robot on a body shell in an initial position. Thereby the conveyor stands still. The body shells are mounted on skids. During system operation the conveyor is moving and skids with the body shells are automatically placed on the conveyor by machines. Afterwards the position of the skids can deviate from the initial position. To avoid these deviations that can cause a collision between a robot and the body shells, the deviations have to be measured to correct the robots movements. To provide accuracy, the conveyor, the systems engineering and the imaging device for the MBT have to be synchronized.

4.2 System setup and general procedure

Figure 6 shows the setup of our industrial use case.

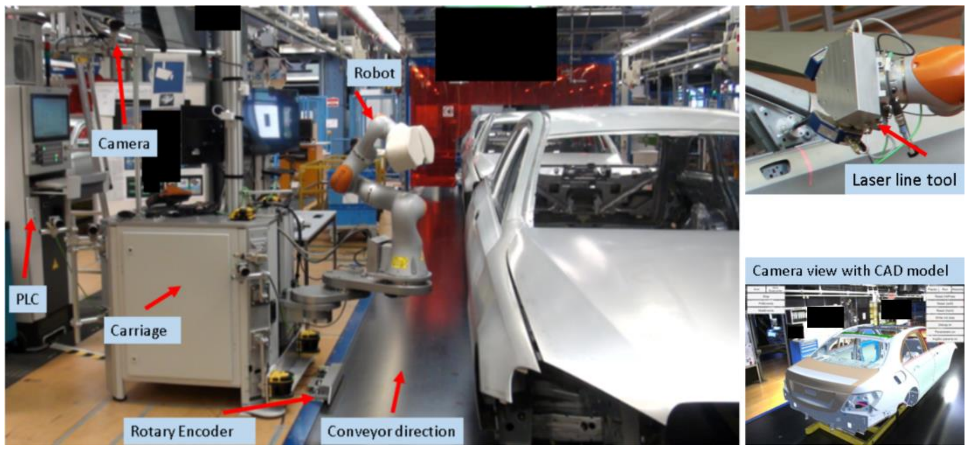


Figure 6: Real setup for the robot gap measurement evaluation. Critical information is blackened.

The MBT runs on an industrial PC in a carriage. In addition a light weight robot is attached to the carriage. The carriage and the robot are connected with a programmable logic controller (PLC) that controls the production station. When a body shell enters the station boundary, our system gets pieces of information about the car type and loads the corresponding CAD model in the MBT system. In the setup phase an initial position of the car is defined and the robot is calibrated for that position. The pose of the body shell is measured with the MBT and stored as reference pose T_R. Later on, this position on the conveyor is used as measurement position for the MBT and denoted as “virtual trigger”. From this virtual trigger we calculate an earlier position in the opposite conveyor direction. This position is denoted as “first fit” (see Figure 4) and marks the position were the MBT matches the edges of the CAD model and video data to obtain a continuous tracking for the following frames. To account for the conveyor position and possible conveyor backlashes the increment of the conveyor is stored as $Inc_{R.}$ Within the production process, cars come along the conveyor and are shifted or twisted in comparison to the reference pose. At the virtual trigger the actual pose T_A is measured with the MBT and the transformation between T_R and T_A is calculated with $T = T_R^{-1} * T_A$.

4.3 Synchronization between conveyor and model-based tracking

To synchronize the MBT with the conveyor and the robot we present a method that does not require external time severs. The conveyor increment is determined by a separate rotary encoder that uses a gear to attach to the conveyor. With a conversion factor the covered distance of the conveyor can be calculated. When a car enters the station, a light barrier detects it (see Figure 4) and sets a conveyor increment counter to zero. After this, the PLC sends the actual conveyor increment Inc_A every 3 milliseconds (ms) to our system. If Inc_A is greater than Inc_R we know that the virtual trigger position is reached and a timestamp t_{act} is set in our system. Then the last pose P_{t1} before the timestamp and the first pose P_{t2} after the timestamp are used in combination with the time difference t_{diff} to estimate the interpolated pose P_{VT} at the virtual trigger. With the help of the relation $P_{VT}/t_{diff} = factor$ we can use the slerp function to interpolate between the poses P_{t1} and P_{t2}. Doing so, the correct pose T_A at time t_{act} for the virtual trigger position can be calculated. T_A is directly comparable to the pose T_R of the reference position. Since the PLC sends the increments every 3 ms to the MBT system an additional maximum error of 0.225 mm can occur. The conveyor speed was 75 mm per second in our experiments.

4.4 Error sources

In this complex application scenario various error sources influence the accuracy. Below the most important ones under an assumption of 2 σ are listed:

- MBT errors: 0.7 mm ± 1.5 mm
- MBT-conveyor synchronization:1.7 ± 1.5 mm
- Robot to conveyor synchronization: ± 0.5 mm
- Robot calibration of gap points: ± 0.5 mm
- Production tolerance of test object ± 0.5 mm
- Calibration conveyor direction
- Calibration robot to test object

4.5 Evaluation of the synchronization

To test our synchronization method we put a coated car on a conveyor that can be moved forward and backward. The reference pose was measured in an initial position with the MBT system and the conveyor increments were recorded. Then the conveyor with the car was moved in the opposite conveyor direction in a start position before the virtual trigger

For the tests the conveyor was started and the measurements with the MBT were triggered by reaching the increment values of the virtual trigger. For every measured pose the transformation to the reference pose was calculated. We repeated this procedure 56 times. Ideally, the mean of the pose differences should be zero if no error would occur. The mean, standard deviation and span of the results is depicted in Table 4. In this test the MBT errors and the production tolerances have an influence on the results. As expected, the biggest error occurs on the x-axis that corresponds to the conveyor direction. We found that coated cars are a bigger challenge for the MBT than body shells. This is caused by light reflections in the coating that can lead to edges being detected falsely and thus can corrupt the matching process. Unfortunately we have no comparison to ground truth data for coated cars. Therefore we rely on the assessment of the total error including all error sources described above. The rotational values only vary in the second decimal place. The translation values only vary in the submillimeter space except the x-axis.

	X	Y	Z	RotX	RotY	RotZ
μ	-1.72	-0.19	0.08	0.00	-0.02	0.03
σ	0.78	0.24	0.20	0.01	0.01	0.01
span	3.11	1.17	0.83	0.04	0.05	0.04

Table 4: Evaluation results of the synchronization method with mean (μ), std. dev. (σ) and span.

To be able to assess and quantify the total error of our system with a more independent measurement method a further evaluation is presented in the next section.

5. Robot measurements

In section 4.4 the error sources of the complete system are described. In a complex setup described so far, various correlations between the error sources exist. Thus a distinguished assessment of the influence of a single error source is hard to determine. We tried to quantify single error sources in this paper to allow an easier assessment of single components and the behavior of the MBT. Finally, the total error is essential to assess the accuracy of the whole system. Therefore we present a method to measure the total error. This is done by using the light weight robot of the gap measurement use case. On the robots end effector a tool for gap measurements is attached, denoted as laser line tool (LLT). It consists of two laser line scanners that are arranged in a mount in the way that a large laser scan line is generated. The LLT is shown in Figure 6, once with and once without housing. If the robot approaches a measurement point at a car the deviation of a gap mid point to the LLT mid point can be measured. This gives us the total error in 2D LLT coordinates. With this criterion it can at least be stated how precise a measurement point of an actual car can be measured with the robot LLT and gives us an overview of how the synchronized MBT is suited for this kind of application.

As described in section 4.1 all necessary measurement points on the body shell are taught to the robot in an initial position. The LLT is thereby aligned perpendicularly to the normal of a gap, e.g. the gap between the hood and the car wing (see Figure 8). To assure a precise alignment a template for the alignment of the laser line is used (see Figure 8).

The template has little bridges on the backside that fit into a gap and ensure a tight application of the template. The laser line is then aligned orthogonally to the template by hand and the robots position is saved. Afterwards a software offset is calculated on the basis of the measurements that corrects small deviations and ensures a tight orthogonal alignment of the laser line to the gap.

During the system setup in the initial position the conveyor direction and the transformation between the robot base and the body shell is measured with a high precision measurement device, e.g. a Faro laser tracker. In the conveyor synchronous mode the robot uses these values to predict the position of the body shell at various conveyor increments. Together with the pose correction values of the MBT and the calibrated measurement points this makes an accurate approach of the robot to each measurement point possible. For the evaluation we used three measurement points on the body shells. Looking at the side of a car, the first is a horizontal gap between the trunk lid and the car body. On this gap our measurement point P1 is approximately placed in the mid. The second gap is vertical and between the back driver door and the car body. The measurement point P2 lies on the left side of the right wheel housing at the lower end. The third gap is also vertical and between the driver door and the back driver door. The measurement point P3 is placed in the upper area.

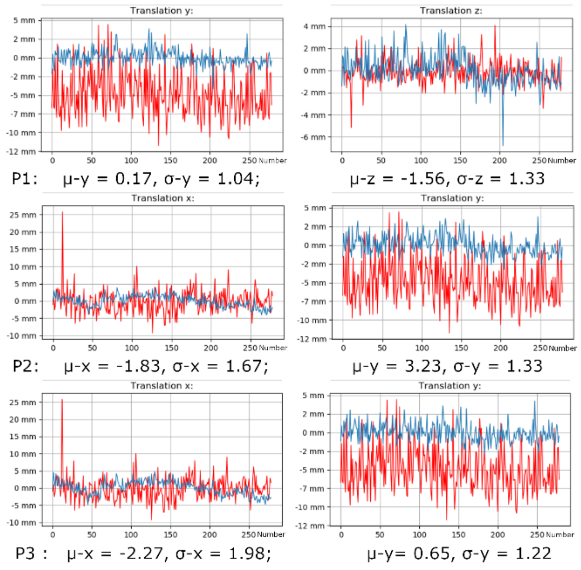


Figure 7: Results of robot measurements for three gap points. MBT results are marked in red for the respective axis and robot results in blue.

In our evaluation the MBT measures the body shell at the virtual trigger and sends the pose correction to the robot. The robot uses the values and his prediction of the body shell to correct the calibrated measurement positions in his coordinate system. Then the robot measures three points successively as the body shell on the conveyor passes. The robot approaches each gap in a way that the laser line is orthogonal to the gap. In our experiment, three gap points were measured 278 times. The results are shown in Figure 7. For each measurement point the axis is denoted in analogy to the car coordinate system (see Figure 7). The origin of the car coordinate system is in the mid of the front suspension. The x-axis is along the cars main axis and directed to the back wheels, the y-axis is directed to the right front wheel and z is the up-axis.

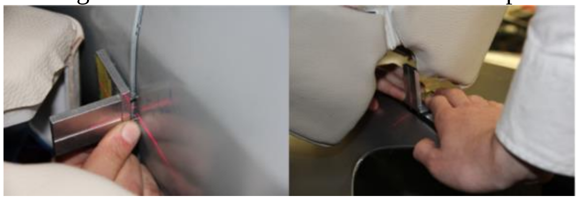


Figure 8 Laser line tool calibration to gaps.

The robot started each of the 278 measurement series with P1 then measured P2 and P3. As shown in Figure 6 the cars back enters the station first. The body shells length is approximately 4.5 meters. For an ideal situation, the blue robot measurements in Figure 7 would be nearly zero, indicating that the MBT results in red correct the robot movements perfectly. It can be seen in Figure 7 that the mean for the vertical gaps increase over the length of the body shells. Regarding the std. dev., a slight increase can be observed too. The further away the measurement points are in x-direction the more uncertainty can be found in the results. This is mainly caused by the leverage of rotational errors of the MBT, of the conveyor direction calibration and of the transformation between the body shell and the robot. Under the assumption of a gauss distribution of 2 σ, it can be stated that P1 can be measured with an accuracy of up to 4.22 mm, given the worst case. P2 with an accuracy of up to 5.89 mm and P3 with an accuracy of up to 6.23 mm in the worst case. The laser line has a length of several centimeters (see Figure 8). Thus, gaps can be measured with the presented method. To further improve our method, we plan to introduce a second virtual trigger that is shifted about a half or one-third of a body shell after the virtual trigger. This will reduce deviations caused by leverage effects of rotational errors or rising uncertainties of the matching pose by the MBT over larger distances.

6. CONCLUSION

In this paper we presented several methods to evaluate the accuracy and stability of model-based tracking approaches. We showed tests to determine the accuracy with different camera resolutions or camera types, the influence of light and temperature changes and the robustness to noise. For one particular approach we found that with precise CAD models an accuracy in the submillimeter range for the translation and up to the second decimal place for the rotation can be achieved. Distances of up to 300 mm and rotations of up to 23° were evaluated. Furthermore a method was presented to synchronize the vision based tracking to a conveyor and robot. This synchronization was also evaluated. Finally, we presented the application of model-based tracking in an industrial gap measurement use case with a light weight robot. This complex system was evaluated and the total error determined. In future we plan to introduce a second virtual trigger to further improve the accuracy of the system and to evaluate the system for coated cars. The presented methods help to assess the applicability of model based tracking approaches for even complex industrial application scenarios.

7. REFERENCES

[Wue07] Wuest H., Wientapper F., Stricker D. (2007) Adaptable Model-Based Tracking Using Analysis-by-Synthesis Techniques. In: Kropatsch W.G., Kampel M., Hanbury A. (eds) Computer Analysis of Images and Patterns. CAIP 2007.

[Com06] Comport, A., Marchand, E., Pressigout, M., Chaumette, F.: Real-time markerless tracking for augmented reality: the virtual visual servoing framework. IEEE Trans. on Visualization and Computer Graphics 12(4), 615–628 (2006)

[Vac04] Vacchetti, L., Lepetit, V., Fua, P.: Combining edge and texture information for realtime accurate 3d camera tracking. In: Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR) (2004)

[Hin11] Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N.,Lepetit, V.: Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes. IEEE International Conference on Computer Vision (ICCV), 2011

[Nie03] Nienhaus, M., Doellner, J.: Edge-enhancement - an algorithm for real-time nonphotorealistic rendering. In: WSCG (2003)

[Gar18] Garon,M., Laurendeau, D., Lalonde, J-F.: A Framework for Evaluating 6-DOF Object Trackers. In: Computer Vision – ECCV 2018.

[Wu17] Wu, P-C., Lee, Y-Y., Tseng, H-Y., Ho, H-I., Yang, M-H., Chien, S-Y: [Poster] A Benchmark Dataset for 6DoF Object Pose Tracking. ISMAR-Adjunct, 2017.

[Hod17] Hodaň,T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., Zabulis, X.: T-LESS An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects, IEEE Winter Conference on Applications of Computer Vision (WACV), 2017

[Far20] FARO Europe GmbH & Co. KG, Faro arm, Faro laser tracker: https://www.faro.com, 2020

[Gom20] GOM GmbH, ATOS, https://www.gom.com, 2020.

Foreground-aware Dense Depth Estimation for 360 Images

Qi Feng
Waseda University
Tokyo, Japan
fengqi@ruri.waseda.jp

Hubert P. H. Shum
Northumbria University
Newcastle upon Tyne, UK
hubert.shum@northumbria.ac.uk

Ryo Shimamura
Waseda University
Tokyo, Japan
s-ryo@akane.waseda.jp

Shigeo Morishima
Waseda Research Institute for Science and Engineering
Tokyo, Japan
shigeo@waseda.jp

ABSTRACT

With 360 imaging devices becoming widely accessible, omnidirectional content has gained popularity in multiple fields. The ability to estimate depth from a single omnidirectional image can benefit applications such as robotics navigation and virtual reality. However, existing depth estimation approaches produce sub-optimal results on real-world omnidirectional images with dynamic foreground objects. On the one hand, capture-based methods cannot obtain the foreground due to the limitations of the scanning and stitching schemes. On the other hand, it is challenging for synthesis-based methods to generate highly-realistic virtual foreground objects that are comparable to the real-world ones. In this paper, we propose to augment datasets with realistic foreground objects using an image-based approach, which produces a foreground-aware photorealistic dataset for machine learning algorithms. By exploiting a novel scale-invariant RGB-D correspondence in the spherical domain, we repurpose abundant non-omnidirectional datasets to include realistic foreground objects with correct distortions. We further propose a novel auxiliary deep neural network to estimate both the depth of the omnidirectional images and the mask of the foreground objects, where the two tasks facilitate each other. A new local depth loss considers small regions of interests and ensures that their depth estimations are not smoothed out during the global gradient's optimization. We demonstrate the system using human as the foreground due to its complexity and contextual importance, while the framework can be generalized to any other foreground objects. Experimental results demonstrate more consistent global estimations and more accurate local estimations compared with state-of-the-arts.

Keywords
Depth Estimation, Scene Understanding, Data Augmentation, 360 images

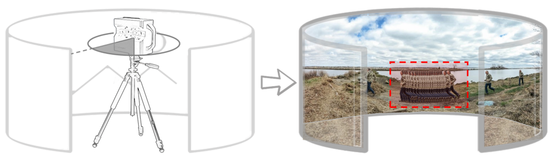


Figure 1: A demonstration of incorrect representations of dynamic objects (e.g. a running person) captured with an omnidirectional RGB-D scanning device.

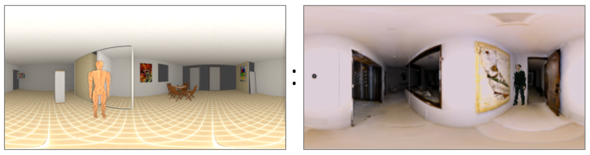


Figure 2: The previous approach of inserting human models introduces the problem of severe domain bias. This is demonstrated by comparing synthetic data (left) with captured data (right).

1 INTRODUCTION

As 360 cameras have become more popular and efficient, the need for image processing algorithms applicable to omnidirectional images increases. The ability to estimate depth from a monocular omnidirectional image can greatly benefit a wide range of applications including navigation in robotics [7], stereoscopic rendering in graphics [10], augmenting virtual objects [14].

Existing omnidirectional approaches produce sub-optimal estimations on real-world scenarios due to their lack of consideration of dynamic foreground objects. For captured-based approaches, using a stereo setup of two 360 cameras will inevitably include the other camera in the captured data [4]. While recent 360-capable scanning devices [3] can acquire paired RGB and ground truth depth of scenes with improved quality, they are incapable of including any dynamic object as a result of scanning and stitching scheme (Figure 1) [1]. For synthesis-based approaches [25], although researchers attempt to solve this problem by inserting 3D models into the scene to improve the

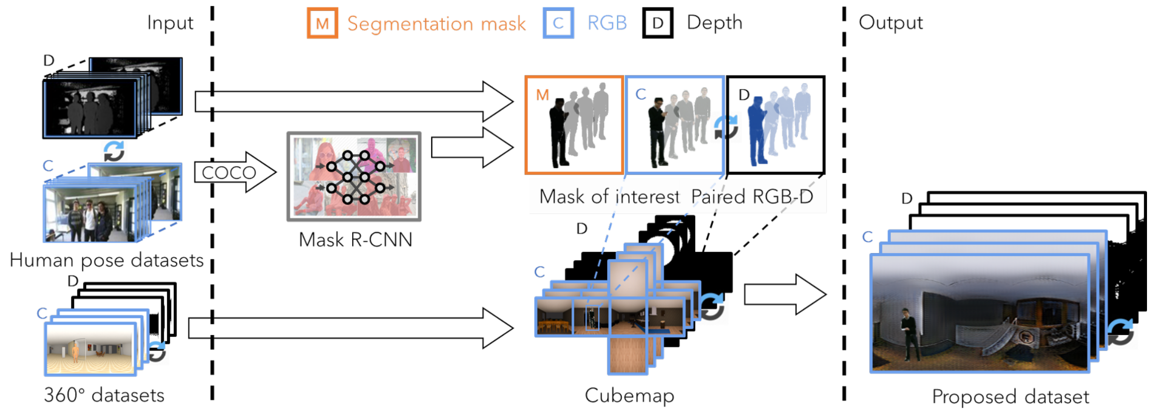


Figure 3: The pipeline of the proposed data synthesizing system. The left section shows the input datasets, the middle section shows the intermediate results, and the right section shows the output. We generate masks of the interested region with mask R-CNN and corresponding RGB-D batches from the input 2D dataset. The batches are then composited to the input 360 dataset with regard to the depth information.

prediction (Figure 2), it is challenging to efficiently generate highly-realistic virtual foreground objects that resemble real-world ones [2], and non-photorealistic data often lead to undesirable and inaccurate outputs.

In this paper, we tackle the problem of foreground by first augmenting datasets with realistic foreground representations. We observe that given the same object with a determined distance, its scale in spherical images should remain consistent. Taking advantage of it, we effectively composite color data of abundant and easily obtainable 2D datasets and rendered omnidirectional images according to ground truth depth maps to ensure correct occlusion representations. To preserve correct distortions in equirectangular images, we project the data to cube maps before and after compositions.

We then propose a novel auxiliary deep neural network that estimates both the mask of the foreground objects and regresses the depth of the omnidirectional images. With the depth and segmentation estimations, we design a new local depth loss of dynamic foreground objects to achieve more consistent depth predictions. This solves the problem that small areas with steep local gradients often got minimized when regressing the global gradient of the prediction, resulting in areas of interest that are frequently smoothed out in existing work.

In this paper, we choose humans as the dynamic foreground object to show the efficacy of our approach. As a foreground object, human shares both a high complexity in deformation and non-uniform depths, and great importance being one of the most interested and common subjects to deliver the context of the image. By showcasing accurate estimations of human, we demonstrate the ability of our method to be generalized to other foreground objects.

Experimental results show that the proposed method yields more consistent global estimations and more accurate local estimations against contemporary state-of-the-art models quantitatively and qualitatively. This research is best applied in fields including occlusion-aware augment reality, stereoscopic rendering.

Our contributions are summarized as follows:

1. We propose a method to synthesize an RGB-D omidirectional dataset with dynamic foreground objects to tackle the challenge of estimating the depth of them in the context of spherical images. The dataset is offered to promote future research.

2. We employ the proposed auxiliary network that estimates depth and segmentation masks to calculate a new local depth loss of dynamic foreground objects. This can resolve the issue of steep local gradient getting smoothed out during optimization and improve the estimation results of local regions. The source code is publicly offered online.

The rest of the paper is organized as follows: we revisit learning-based monocular depth estimation methods and methods for synthesizing training data in Section 2. In Section 3, we explain the novelty of our dataset and describe the generation framework. In Section 4, we describe the network architecture and the proposed loss function to leverage the dataset. Details of experiments are presented in Section 5 along with qualitative and quantitative evaluations. Finally, Section 6 concludes this work.

2 RELATED WORK

2.1 Learning-based Monocular Depth Estimation for Omnidirectional Images

Estimating the depth given a monocular RGB image is one of the most fundamental capabilities in understanding the 3D geometry of the scene [13]. A wide range of applications in robotics, graphics, virtual reality, etc.

can benefit from more accurate depth predictions. Owing to more established machine learning algorithms, learning an implicit relation between color and depth has seen significant progress recently.

A variety of algorithms [19] [17] have been proposed by training a model with collected color and ground truth depth images in a supervised fashion. Lately, numerous strategies have been proposed to achieve a more coherent and accurate monocular depth estimation. Multi-scale networks [5] make coarse global depth prediction and refine the local prediction. Multitask learning [15] [23] with multiple regression and classification objectives is also prevalent in understanding scene geometry and semantics due to their complementarity. A fully convolutional network architecture [17] that endows novel up-sampling blocks achieved impressive accuracy and efficiency.

Unsupervised methods focused on a stereo correspondence framework to cope with the need for an expensive secondary supervisory signal. This is either accomplished by synthesizing stereoviews with left/right consistency [10] to produce intermediary disparity map [6] [27], or multi-view consistency with structure-from-motion (SfM) [28] to learn a dense disparity prediction.

To yield accurate estimations of both global and local objects in the context of the omnidirectional domain, lacking paired data with dynamic foreground objects and distortion introduced by equirectangular projection will result in poor outputs for supervised approaches. On the other hand, while some unsupervised approaches do not explicitly require paired datasets, issues like distortions and occlusions still persists.

Therefore, predicting the depth of 360 contents with the aforementioned 2D approaches often yields suboptimal results [29]. Failing to learn feature representations in the equirectangular domain inevitably leads to inferior accuracy and coherency. To improve the performance of prediction in 360 contents, cubemap projection is one of the most popular choices. By projecting spherical signals onto faces of a cube, six non-distorted square patches can still be processed with existing convolution techniques. However, while such an issue may not be critical in certain tasks such as stylization and classification, the lack of consistency between the output of each patch is more pronounced in depth regression. Recently, methods for enabling rotation-equivariance in CNNs were proposed by Cohen [4]. However, since such equivariant architectures provide a lower network capacity, only single variable regression problems were demonstrated. Inspired by [26], the state-of-the-art method [29] incorporated distorted CNN filters to improve the performance of fully convolutional networks with skip connections and showed impressive predictions of equirectangular images. However, without any consideration on foreground objects, the network will penalize small areas with a steep local gradient when regressing the global gradient of the prediction, resulting in areas of interest such as humans are frequently missing in the output.

2.2 Synthesizing Omnidirectional Datasets

Since the standard method to approach monocular depth estimation is to train a model directly from paired RGB images and ground truth depth, the performance of such supervised approaches cannot produce better results than the limits of its training data. With advanced imaging devices and depth sensors, high-quality datasets consisting of traditional perspective images are easily obtainable, for instance, KITTI [8], NYUv2 [24], Make3D [22], etc. However, obtaining paired 360 data is not as straightforward as using traditional imaging devices with calibrated color and depth sensors such as Kinect to capture 2D contents. Using a stereo setup of two 360 cameras to calculate disparity is challenging due to the presence of occluded regions [20]. In the case of omnidirectional images, both cameras will inevitably include the other camera in the captured data. Recent scanning devices are capable of acquiring datasets that consist of paired 360 RGB and ground truth depth of static scenes with improved quality, such as Stanford 2D-3D [1] and Matterport3D [3].

However, existing methods fail to include any dynamic object in the scene. As a result of a scanning and stitching scheme, trying to include dynamic foreground objects in the captured data [1] [3] will lead to distorted and incorrectly composited images, as shown in Figure 1. [29] repurposed 3D model datasets, SunCG [25] and SceneNet [11], to render 360 synthesis-based RGB-D images with virtual cameras. However, a model trained with synthetic data does not necessarily generalize well to real-world scenarios, due to dataset bias. As observable in Figure 2, a previous attempt to resolve this issue by inserting human models into the existing synthetic dataset suffers from a severe domain bias from the real-world scenarios. However, because of the inability to include realistic human representation in the existing omnidirectional RGB-D dataset, the performance of all previous methods is greatly limited when applied to real-world scenarios with humans.

3 FOREGROUND-AWARE PHOTOREALISTIC 360 DATASET

To produce a foreground-aware photorealistic dataset for machine learning algorithms, we explain our method of augmenting datasets with realistic foreground objects using an image-based approach in this section. The pipeline of our method is visualized in

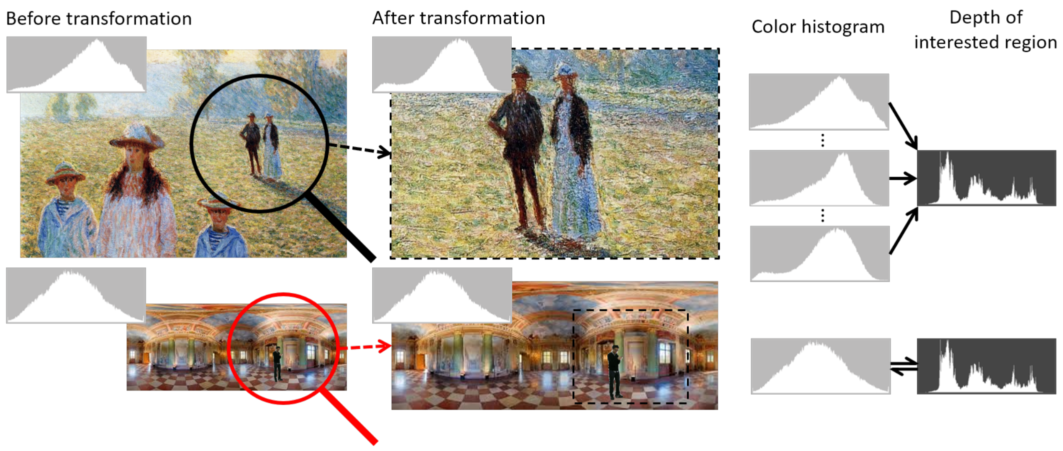


Figure 4: Since traditional 2D images can be processed with perspective transformations, it is difficult to establish a correspondence between color and depth information. In comparison, 360 images cannot be cropped or zoomed and hence have a scale-invariant RGB-D correspondence. As shown on the right side, in the 2D plane, different color representations map to the same depth map of interested regions. One the other hand, 360 images share a one-to-one mapping between color and depth, and every object has a fixed scale.

Figure 3. As shown in Figure 4, based on the observation that 360 images can circumvent challenges brought by perspective transformations in the traditional 2D plane, we effectively composite color data of abundant and easily obtainable 2D datasets and rendered omnidirectional images with z-buffer. We employ a Mask R-CNN network to predict pixel-perfect masks of the dynamic foreground objects. With the acquired masks of interest, we can obtain perspective paired color and depth batches. With cubemap projections done before and after compositions, we can composite with correct occlusions and distortions.

3.1 Scale-invariant RGB-D Correspondence in 360 Images

In this section, we explain the novelty and feasibility of compositing existing 2D RGB-D datasets onto equirectangular images.

We observe that it is difficult to establish a correspondence between color and depth in the traditional 2D domain. We take perspective transformations as an example and demonstrate with Figure 4. During the process of "zooming in" onto the target region (dashed box), the global color data changes continuously while the depth of the target area stays the same, forming a many-to-one mapping. It is particularly true in the real-world: when we use binoculars to observe the same object, even given the prior knowledge of an object's average size, it is inherently harder to estimate the distance without knowing the magnification.

One the other hand, the relation between color and depth in 360 images is scale-invariant. While some perspective transformations such as cropping will make 360 images no longer spherical, rotation and zoom will not affect the global color representation of the original image after down-scaling. Therefore, given the same object with a determined distance, the appearance of the target region in 360 images should remain consistent.

Based on this observation, we exploit such an advantage of omnidirectional images by inversely composite local regions onto them with regard to the depth information. In this work, we choose z-buffer to composite owing to its simple implementation, high efficiency and compatibility of occlusions.

3.2 Synthesizing RGB-D Foreground Batches

3.2.1 *General Foreground Synthesis*

To automatically acquire paired color and depth maps of a dynamic foreground object, we can either capture with sophisticated RGB-D sensors or take advantage of abundant and easily obtainable existing datasets in the traditional 2D domain. In order to efficiently acquire highly accurate segmentation masks of the input data, we adopt a Mask R-CNN model with a backbone of ResNet-101, trained with the COCO dataset to predict per-pixel label masks. The strengths of per-instance prediction and less complex post-processing are the main reason we choose Mask R-CNN over a simpler U-Net network. During prediction, our implementation predicts per-person-instance masks in a near-real-time speed (5 fps) with high accuracy. Some examples are shown in Figure 10 and Figure 11. With acquired masks for areas of interest, we crop batches from the input RGB-D data accordingly.

3.2.2 *Human Batch Synthesis*

Since human as a dynamic foreground object shares both a high complexity in deformation and non-uniform

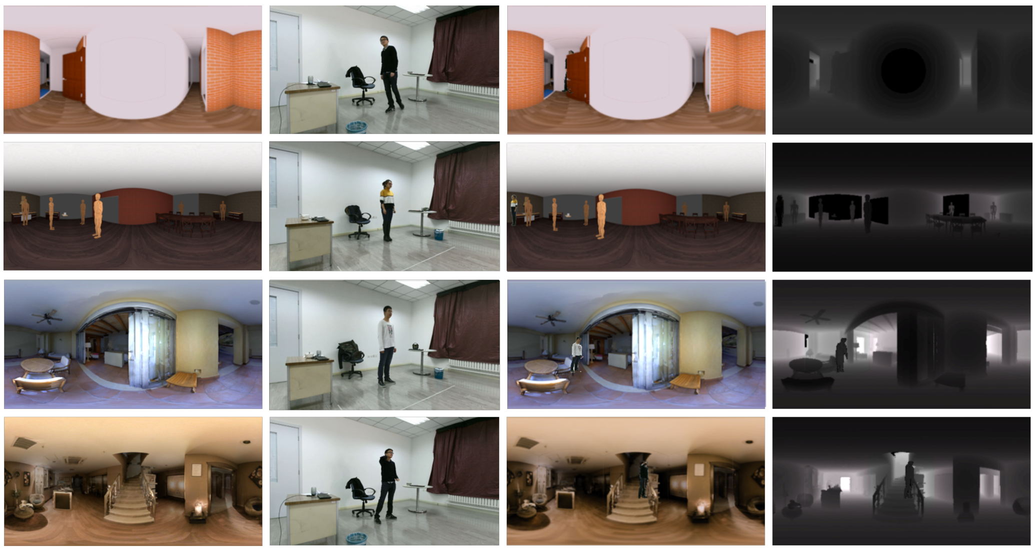


Figure 5: Generated examples with the proposed method. From left to right: rendered color images with original omnidirectional datasets, samples from an input human pose dataset, generated omnidirectional images with humans, and corresponding depth maps.

depths, we choose humans to show the efficacy of our approach. At the same time, humans have great importance being one of the most interested and common subjects to deliver the context of the image. By showcasing accurate estimations of human, we demonstrate the ability of our method to be generalized to other foreground objects. In this work, we repurpose the PKU-MMD dataset [18], which contains calibrated and synchronized RGB-D video sequences. This large-scale dataset includes motions of 51 categories performed by 66 distinct subjects. It contains different views, sufficient intra-class variations and adequate classes of motions to ensure a robustness prediction result.

3.3 Synthesizing RGB-D Omnidirectional Data with Foregrounds

3.3.1 *Omnidirectional Background Synthesis*

Since paired real-world 360 RGB-D datasets with humans are not available to our knowledge, to alleviate the difficulty of evaluating the accuracy between ours and the-state-of-the-art approaches, we use a similar strategy matching with [29] to render paired and realistic omnidirectional RGB-D images from the Stanford 2D-3D dataset and the Matterport3D dataset captured with professional 360-capable scanning devices. Specifically, a path tracing renderer with a virtual omnidirectional camera is used to generate the samples. The light source is positioned identically with the virtual camera. Omnidirectional depth maps with linear distances of each pixel are generated with Z depth. To show the effectiveness of our method across different domains and to benchmark the accuracy with synthetic 360 datasets, identical processes are brought out with the SunCG [26] and the SceneNet [11] as well.

3.3.2 *Compositing Foregrounds and Backgrounds*

Since the RGB-D local batches are captured in the traditional 2D domain, a direct composition will lead to distorted and unrealistic appearances in the 360 context. To cope with this challenge, both RGB and depth map of each rendered omnidirectional sample is projected onto a cube map through cubic projection. With ground truth depth information of both foreground batches and background faces, the composition is done through highly efficient and effective Z-buffer, preserving correct depth annotations and in-scene occlusions. To simulate real-world scenarios, batches are randomly composited to lower halves of 4 surrounding cube faces, while faces of the ceiling and the floor are not used during composition. Finally, a reverse cubemap projection is done to generate high-quality RGB-D equirectangular samples with dynamic foreground objects.

In this work, our proposed dataset consists of 25,000 realistic and 25,000 synthetic equirectangular samples with synchronized color information and depth annotations. Abundant variation is achieved through a sufficiently wide range of indoor scenes as backgrounds, and a large human batch pool acquired in the previous step as foregrounds.

4 DEPTH ESTIMATION FOR OMNIDIRECTIONAL IMAGES

This section presents our proposed end-to-end learning model to estimate a depth map from an equirectangular image. As shown in Figure 6, We use two fully-convolutional encoder-decoder structured networks,

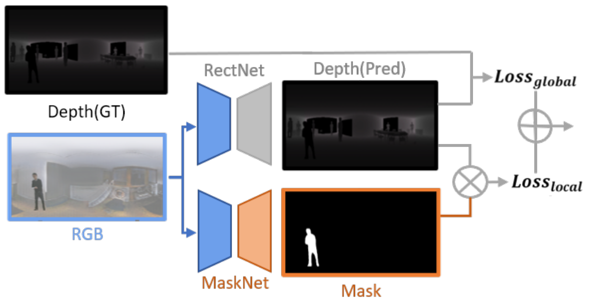


Figure 6: An overview of the proposed depth estimation network. The weight of the auxiliary MaskNet is fixed when training the depth estimation model RectNet [29].

RectNet and MaskNet to regress depth and predict masks of local regions respectively from a given RGB input. The RectNet that resembles the design in the literature can regress depth with changing filters in an omnidirectional context. To take advantage of the generated dataset with dynamic foreground objects, we leverage the generated masks of interested areas to train the auxiliary MaskNet. By calculating both local depth loss and global loss, our network further improves the consistency in local predictions.

4.1 Network Structure

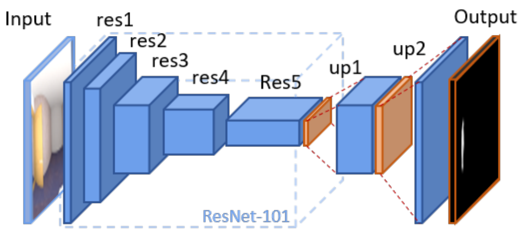


Figure 7: The architecture of the fully convolutional auxiliary MaskNet. The encoder of our network shares the same structure of ResNet-101 [12], followed by a decoding process with two upsampling layers to predict the mask of the target object.

The proposed network approaches dense depth estimation from monocular RGB images shares an encoder-decoder design that progressively downscales and upscales to the target representation through regression. Skip connections similar to ResNet structures can help to preserve the information from a higher level during regression while preventing vanishing gradient. When applied to equirectangular images, inspired by [26], we incorporate rectangular filters with changing sizes according to rows of the input to cope with the characteristic that the density of information, or namely the distortion level changes along the vertical axis but invariant along the horizontal axis. In addition to L2 depth loss to regress the prediction, a neighborhood smoothness regularization term [29] is also calculated to improve the global consistency of the output.

However, small regions with steep gradient changes usually got smoothed out during the regression and missing in the prediction. This can be observed in Figure 10. Predictions of human severely suffer from this issue. To tackle this limitation, we introduce an auxiliary network, MaskNet, to calculate the local depth loss of humans. The MaskNet network that predicts masks of foreground objects from equirectangular RGB inputs has the architecture shown in Figure 7. It is trained with the COCO dataset and finetuned with generated equirectangular RGB images with foreground objects and corresponding segmentation masks to minimize a cross-entropy loss. The weight is fixed during training the depth estimation model.

4.2 Loss Function

We train the depth estimating network in a compeletly supervised fashion with input of the generated foreground-aware RGB-D dataset. To address the problem of vanishing local gradients for areas of interest while keeping the desirable properties of the original RectNet like consistent global predictions, the total loss of our model consists of three different terms:

$$L_{total} = \sum_i (\alpha_i L_{depth} + \beta_i L_{smooth} + \gamma L_{local}),$$

while the α, β and γ are the weights for each loss term. Since the loss is calculated under different scales i, the estimations of lower scales are interpolated with nearest neighbors are concatenated together to form the final output. The depth loss L_{depth} is regressed by minimizing the least square errors between the groudtruth depth maps D_{gt} and the predicted depth maps D_{pred}:

$$L_{depth} = \|D_{gt} - D_{pred}\|^2.$$

The smoothness loss is calculated by $\|\nabla D_{pred}\|^2$ to minimize the gradient of the prediction. In order to calculate the local depth loss, we pass the equirectangular color image C_{input} through the trained auxiliary network $\mathbf{M}$ to obtain the mask of human instances $M_{human} = \mathbf{M}(C_{input})$, so we can calculate the local depth loss with

$$L_{local} = \|D_{pred} \otimes M_{human}\|^2.$$

By minimizing the local depth loss, we can ensure that spatially closer pixels within the same area of interest would have closer depth values.

5 EXPERIMENTS

In this section, we first evaluate our data augmentation method by presenting quantitative comparisons between models trained with existing omnidirectional datasets and our generated datasets. We then verify the performance of the proposed network by comparing it to the state-of-the-art omnidirectional depth estimation algorithm. Finally, to evaluate the effectiveness of our method in real-world scenarios with human objects,

Table 1: Quantitative results of different training datasets. Error metrics are calculated on a global basis.

Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Synthetic	0.4918	0.4133	0.8944	0.6550	0.4083	0.6806	0.8212
Proposed Synthetic	**0.3789**	**0.2893**	**0.6878**	**0.5225**	**0.4245**	**0.7926**	**0.9257**
Realistic	0.3765	0.3540	0.8864	0.5230	0.5907	0.7500	0.8926
Proposed Realistic	**0.3190**	**0.2180**	**0.5993**	**0.4788**	**0.6988**	**0.8454**	**0.9150**

For four error metrics, absolute relative difference (Abs Rel), squared relative difference (Sq Rel), root mean square error (RMSE) and RMSE log, lower values are better. For percentage of inliers under threshold $\delta < 1.25$, $\delta < 1.25^2$ and $\delta < 1.25^3$, higher values are better. Same for tables below.

Table 2: Quantitative evaluation against other models. Error metrics are calculated on a global basis.

Model	Training Set	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
RectNet [29]	Proposed Syn	0.3789	0.2893	0.6878	0.5225	0.4245	0.7926	0.9257
Proposed	Proposed Syn	**0.2895**	**0.2354**	**0.5957**	**0.4272**	**0.7440**	**0.8805**	**0.9284**
RectNet [29]	Proposed Real	0.3190	0.2180	0.5993	0.4788	0.6988	0.8454	0.9150
Proposed	Proposed Real	**0.1984**	**0.0817**	**0.3286**	**0.2608**	**0.7298**	**0.8984**	**0.9727**

we offer comparative qualitative results of estimating unseen images by different methods.

5.1 Training Details

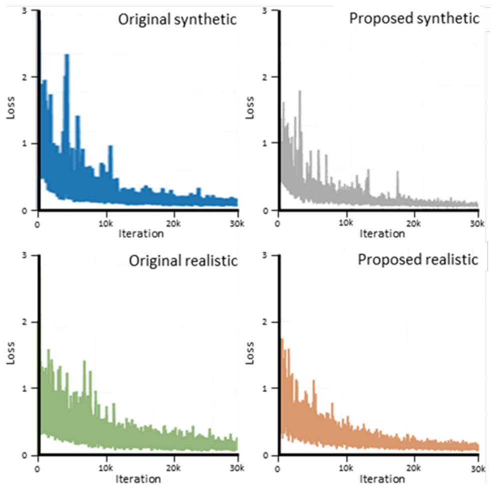


Figure 8: Learning curves of models respectively trained with original synthetic, original realistic, proposed synthetic and proposed realistic datasets.

For fair comparisons, we randomly acquired 25,000 samples from existing synthetic omnidirectional datasets to train models as the existing synthetic dataset, and then we acquired 25,000 samples from existing realistic omnidirectional datasets to train models as the existing realistic dataset. We respectively generate 25,000 synthetic samples and realistic samples augmented with human objects to train models as our proposed datasets. Each 512 x 256 sample has color information and corresponding ground truth depth annotation. We randomly split samples from each dataset into training and validation datasets with a ratio of 80% and 20%. All networks in this paper are implemented with PyTorch [21] on an Nvidia RTX 2080Ti graphic card and trained with Adam optimizer [16], Xavier initialization [9], and a learning rate of 2e-4. Training parameters of our networks are $[\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma] = [0.482, 0.245, 0.121, 0.061, 0.090]$, while parameters of training previous RectNet models are $[\alpha_1, \alpha_2, \beta_1, \beta_2] = [0.535, 0.272, 0.134, 0.068]$. The same quantitative metrics from the literature [10] [29] are used for evaluation. During experiments, predicting a single image approximately costs 100 ms with the same setup.

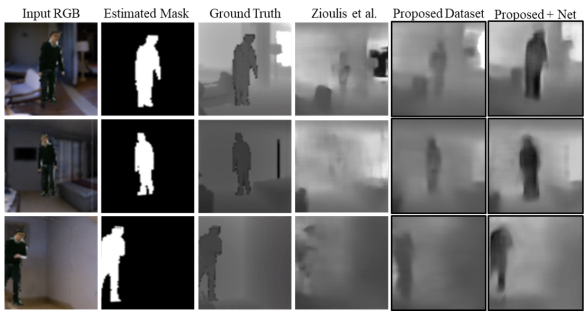


Figure 9: Estimated depth information of local regions with different configurations. An ablation study shows that using our augmented dataset can improve the accuracy of local regions, and the proposed network shows an improved consistency with clearer boundaries.

5.2 Quantitative Results

Table 1 presents the results of the state-of-the-art models respectively trained with existing synthetic and realistic datasets and our proposed datasets. We observe that when tested on unseen samples with human objects, networks trained with our proposed datasets outperform the existing ones. The increased performance in accuracy against previous methods attributes to more

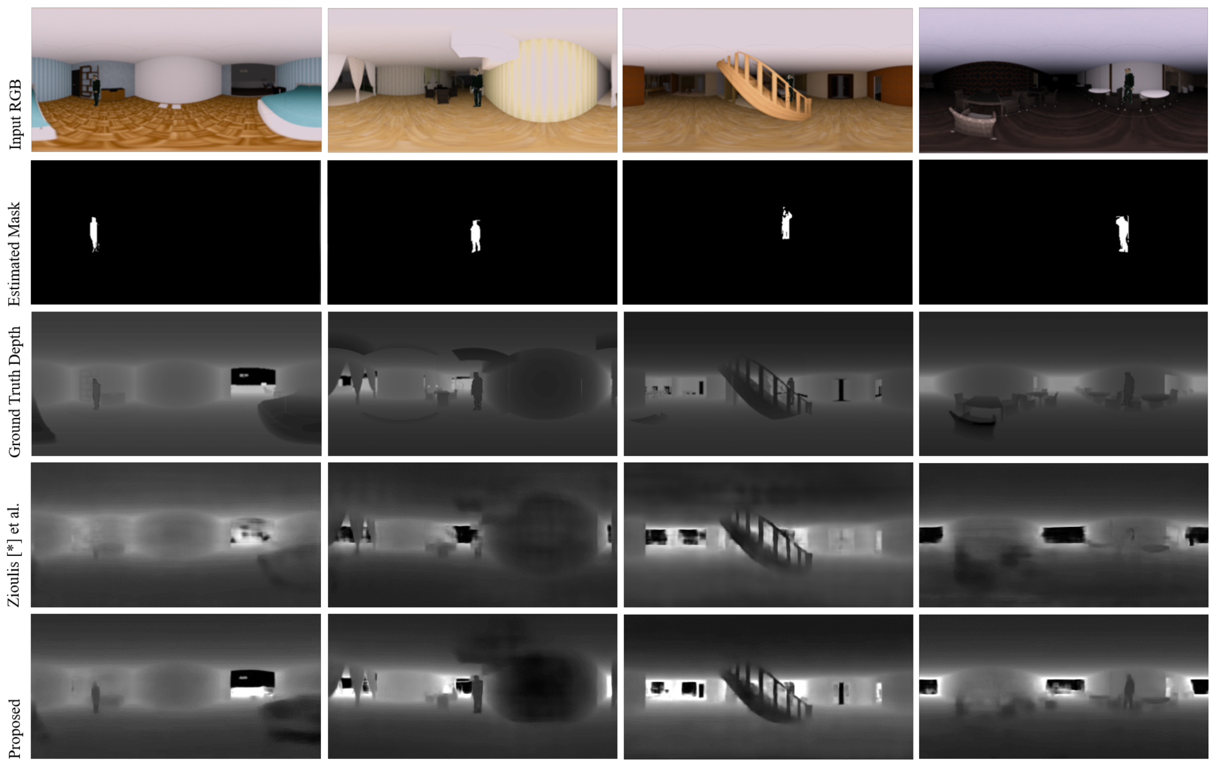


Figure 10: Qualitative comparison between each model when tested on synthetic images.

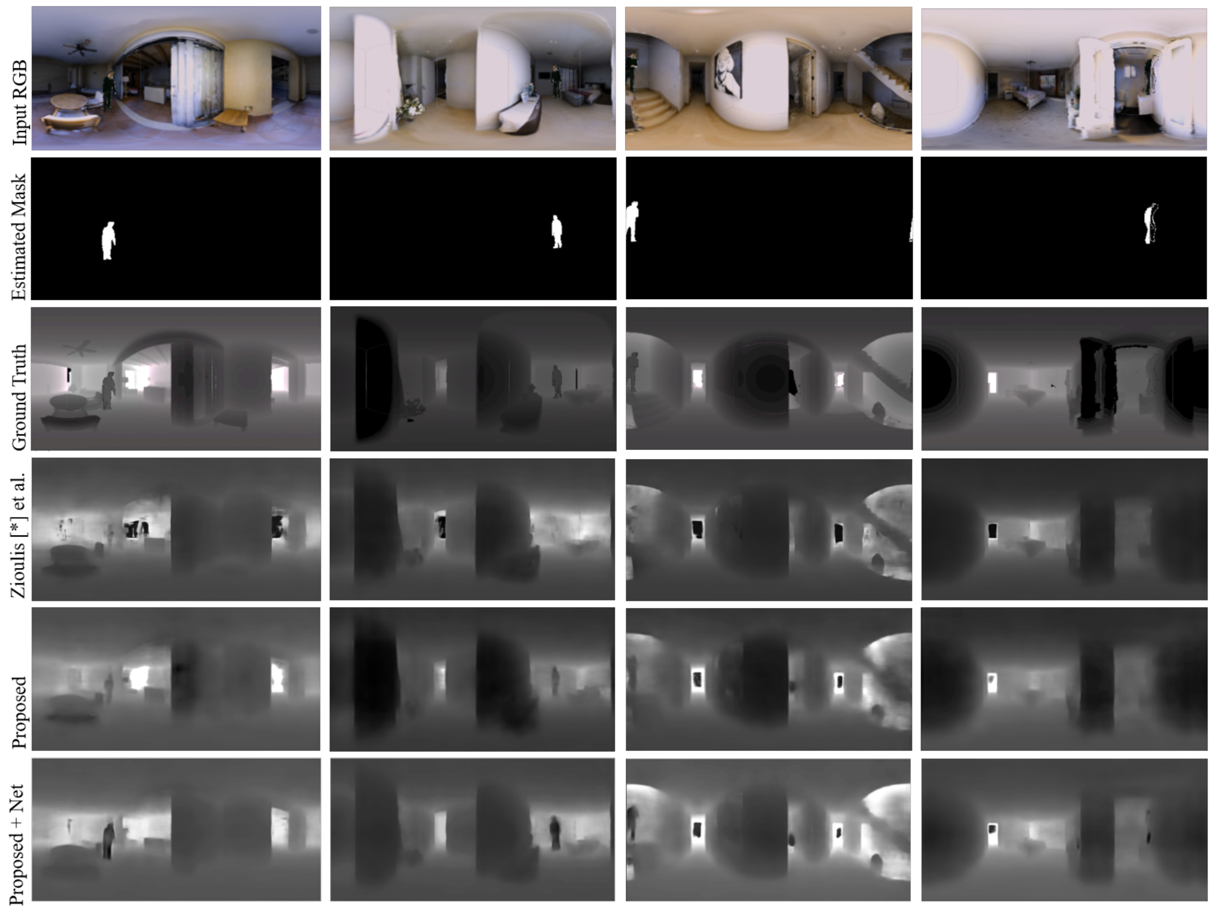


Figure 11: Qualitative comparison between each model when tested on realistic images.

accurate estimations of local human regions, as can be observed in Figure 9. By further quantitatively evaluating the accuracy of estimations between our proposed network and the state-of-the-art models, we can observe the inferior performance of previous approaches as expected in Table 2. Depth estimations of local human regions are further refined with our proposed network.

5.3 Qualitative Results

To qualitatively evaluate our models' ability to generalize to unseen data, we further acquire and augment samples from the SunCG and the Matterport3D that come from other locations different from training datasets. As we can observe in Figure 10 and Figure 11, our models perform better to estimate the depth of both synthetic and realistic scenes with a human. While previous models yield human depth estimations that are blended with the background and have a blurred edge, our models can predict much clearer and human-shaped results. It is worth mentioning that although all omnidirectional samples used in the experiment only cover indoor settings, our method works with outdoor cases as well.

After observing generated samples, we believe there are many challenges left to overcome. First, even our method can augment foreground objects, we do not take lighting into consideration during the process. This unnaturalness may lead to less robust estimation in certain scenarios (e.g. scenes with very high brightness).

5.4 Ablation Study

In Figure 9, we compare the accuracy of depth estimations for local regions under different configurations. Specifically, we compare using original data and proposed data to train only the depth estimation network without the auxiliary MaskNet at first to validate the effectiveness of our data generation method. We then use augmented data to train depth estimation networks with the auxiliary MaskNet, and verified that the local depth loss can successfully improve the consistency of estimated depth within areas of interest. As we can observe in Figure 9, our method significantly outperforms the state-of-the-art in local depth estimation.

6 CONCLUSION

We have presented a data augmentation method to generate high-quality equirectangular datasets with paired color and ground-truth depth annotations by repurposing abundant and easily obtainable 2D RGB-D datasets. With this dataset, we further introduced and implemented an auxiliary network that calculates local depth loss to resolve an issue that small regions of interest are frequently smoothed out during optimizing global gradients. We take human, a crucial subject in 360-degree contents, as an example to show the efficacy of our approach. We showed improved accuracy of our approach compared to the state-of-the-art technique. We believe that the ability to estimate depth for foreground objects in 360 images can benefit a wide range of applications such as navigation in robotics and augmenting virtual objects with occlusions.

Currently, our data augmentation method is based on the premise that both 2D and 360 data are captured with similar extrinsic parameters (e.g. cameras are aligned horizontally, positioned at average eye-level height) and lighting conditions, while it is true for most data captured in lab conditions, its application for in-the-wild images is limited. Furthermore, our approach works for both indoor and outdoor settings. Nevertheless, for outdoor settings, a higher dynamic range of luminosity and sunlight's ambient IR will render capturing RGB and depth information inherently difficult. For future work, we aim to explore generating samples with different lighting conditions with GANs to improve the robustness of depth estimation.

ACKNOWLEDGEMENT

We would like to thank Dr. Yuki Koyama, Dr. Satoru Fukayama and Prof. Masahiro Hamasaki from National Institute of Advanced Industrial Science and Technology (AIST) for their kind guidance and support in this research. The project was supported in part by the JST ACCEL (JPMJAC1602), JST-Mirai Program (JPMJMI19B2), JSPS KAKENHI (JP19H01129), the Royal Society (Ref: IES\R2\181024) and the Defence and Security Accelerator (Ref: DSTLX-1000140725).

7 REFERENCES

[1] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[2] A. Atapour-Abarghouei and T. P. Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2800–2810, 2018.

[3] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[4] T. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical cnns. *International Conference on Learning Representations (ICLR)*, 2018.

[5] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pp. 2650–2658, 2015.

[6] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pp. 740–756. Springer, 2016.

[7] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transactions on robotics and automation*, 16(6):890–898, 2000.

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361. IEEE, 2012.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[10] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 270–279, 2017.

[11] A. Handa, V. Pătrăucean, S. Stent, and R. Cipolla. Scenenet: An annotated model generator for indoor scene understanding. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5737–5743. IEEE, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[13] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1, pp. 654–661. IEEE, 2005.

[14] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-dof vr videos with a single 360-camera. In *2017 IEEE Virtual Reality (VR)*, pp. 37–44. IEEE, 2017.

[15] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491, 2018.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pp. 239–248. IEEE, 2016.

[18] C. Liu, Y. Hu, Y. Li, S. Song, and J. Liu. Pku-mmd: A large scale benchmark for continuous multi-modal human action understanding. *arXiv preprint arXiv:1703.07475*, 2017.

[19] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5162–5170, 2015.

[20] K. Matzen, M. F. Cohen, B. Evans, J. Kopf, and R. Szeliski. Low-cost 360 stereo photography and video capture. *ACM Transactions on Graphics (TOG)*, 36(4):148, 2017.

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[22] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2008.

[23] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[24] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pp. 746–760. Springer, 2012.

[25] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1746–1754, 2017.

[26] Y.-C. Su and K. Grauman. Learning spherical convolution for fast features from 360 imagery. In *Advances in Neural Information Processing Systems*, pp. 529–539, 2017.

[27] J. Xie, R. Girshick, and A. Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pp. 842–857. Springer, 2016.

[28] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1983–1992, 2018.

[29] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras. Omnidepth: Dense depth estimation for indoors spherical panoramas. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 448–465, 2018.

ISSN 1213-6972
Journal of WSCG
http://www.wscg.eu
Vol.28, No.1-2, 2020

Ensembles and Cascading of Embedded Prototype Subspace Classifiers

Anders Hast and Mats Lind
Department of Information Technology
Uppsala University
SE-751 05 Uppsala, Sweden
anders.hast@it.uu.se;
mats.lind@it.uu.se

ABSTRACT

Deep learning approaches suffer from the so called interpretability problem and can therefore be very hard to visualise. Embedded Prototype Subspace Classifiers is one attempt in the field of explainable AI, which is both fast and efficient since it does not require repeated learning epochs and has no hidden layers. In this paper we investigate how ensembles and cascades of ensembles perform on some popular datasets. The focus is on handwritten data such as digits, letters and signs. It is shown how cascading can be efficiently implemented in order to both increase accuracy as well as speed up the classification.

Keywords
Subspaces, Ensembles, Cascading, Embedded Prototypes, Neural Networks, Deep Learning.

1 INTRODUCTION

Recently, deep learning based methods [Sha18], have shown tremendous performance in the area of handwritten text recognition [KDJ18, DKMJ18, SF16]. Nevertheless, there are a few problems and challenges that still need to be addressed. First of all these methods usually require powerful GPU resources in the training process, not at least because of the back propagation in numerous epochs, which makes them very time consuming. Secondly, the learning process of a deep neural network can be regarded as a black-box [CPC19]. Furthermore, the learning mechanism is not easy to comprehend nor to visualise, because of the many hidden layers. This is know as the interpretability problem [Kri19, CPC19] and in order to find a solution the field of explainable AI (XAI)[ADRS*19, GSC*19, CPC19] has emerged.

A new kind of learning process together with a well researched classification method [KLR*77] was recently proposed [HLV19] called *Embedded Prototype Subspace Classification* (EPSC), which go beyond the black-box deep neural network. It is a mathematically well-defined neural net based on subspaces (or manifolds) aimed at classification of handwritten letters, which is both easy to interpret, explain and visualise. This method is an especially interesting alternative when speed is crucial since it is not based on back propagation and therefore does not require an iterative training procedure. Even if it not always beats the state of the art, it comes close enough to be an interesting alternative due to its interpretability and compactness, having just one input layer and one output layer with no hidden layers.

In this paper we present how ensembles of EPSC can be set up in such a way that classification can be done in an efficient manner. Cascading is used to progressively exclude nodes in the resulting neural network that cannot contribute to the correct classification. The idea is to make classification faster without compromising accuracy.

2 BACKGROUND

The first application of subspaces in pattern recognition was proposed by Watanabe et al. [WP73] in 1967, and later further developed [WLK*67], by Kohonen and others [KLR*77, KO76, KRMV76, OK88]. The idea of the later approaches of subspace learning was to choose some group of prototypes for the construction of each subspace. This was done by searching for the k nearest neighbors in feature space, which is a rather time consuming process. In the following subsections a more efficient and accurate procedure is explained.

2.1 Embedded Prototype Subspace Classification

The idea of EPSC [HLV19] was to use t-distributed stochastic neighbour embedding (t-SNE) [MH08] to find representative clusters in 2D image space by applying kernel density estimation (KDE) [CHTT96] and a so called inverse watershed transform (IWS) [RM00, VS91], which is simply the watershed transform on an inverse image. Therefore, *valleys* between mountain tops are found in the image rather than the *drainage divide* that separates adjacent drainage basins, also known as watersheds.

The t-SNE is a machine learning technique that reduces the number of dimensions of high dimensional data to 2 or 3 dimensions. Clusters are formed since it strives to maintain a representation of similarities among features, so that similar features are represented as points closer to each other and dissimilar points further away from each other.

PCA [WEG87] is used both as an intelligent start guess for t-SNE in order to make the process deterministic, but also for generating each subspace [Laa07]. By computing the norm of the projected feature vector to be classified into each subspace, the process can be regarded as a two layer neural network [OK88, Laa07], where the weights are mathematically defined through PCA. Another important advantage is that the learning process can easily be visualised, which makes it easy to understand, interpret and explain compared to most state of the art deep learning approaches.

2.2 Clustering

The main idea behind clustering, or cluster analysis, is to group data that are in some sense similar into separate clusters. Usually some kind of distance function is used to group similar data points together [JMF99]. Clustering is a common technique for exploratory data mining and analysis, and is used in many fields, including machine learning, pattern recognition, image analysis, machine vision and its applications [JMF99].

The selection of an appropriate clustering approach is indispensable for the performance of a machine vision task [XW05]. Clustering can be done in many ways, and a variety clustering algorithms have been proposed in literature, such as k-means [HW79] and DBSCAN [EKSX96]. These algorithms require the user to set the number of clusters to be found, while others, like Mean-Shift [CM02, FH75] finds the number of clusters depending on the size of the Gaussian Kernel chosen.

Just as in [HLV19], it was chosen to perform clustering in image space in order to easily be able to visualise it as the learning process proceeds. Nonetheless, the method used is producing a similar result as Mean-Shift.

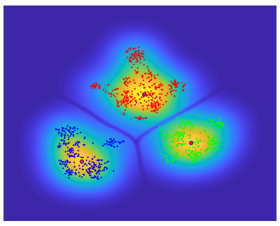
2.3 Kernel Density Estimation

Kernel Density Estimation (KDE) [CHTT96] can be performed by splatting Guassian discs onto an image, for each point in the dataset. The size d of each disc can be set in a similar way as the kernel size for Mean-Shift. However, since KDE and IWS work in image space, the result can easily be visualised. It was chosen in [HLV19] to use the Silverman's rule of thumb as the base level for computing the bandwidth h of the clustering,

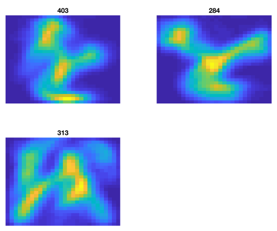
$$h = \left(\frac{4\sigma^5}{3n}\right)^{1/5} \tag{1}$$

where σ is the standard deviation of n samples.

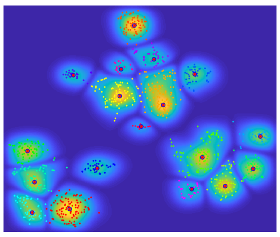
The KDE surface can be used in a visualisation as a visual aid to identify clusters, but is also used as the basis for the IWS. By multiplying h by a scale factor, it is possible to vary the number of clusters.



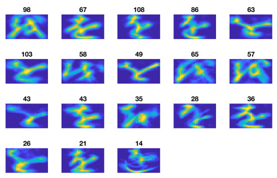
(a) Clusters, $h = 3.0$.



(b) Heatmaps for the 3 resulting clusters.



(c) Clusters, $h = 1.0$.

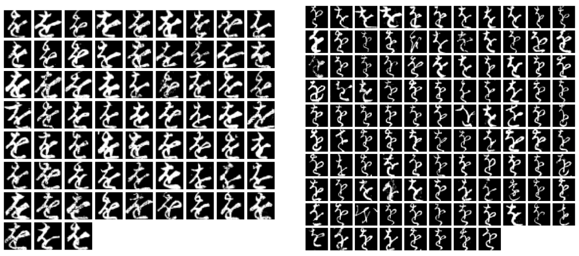


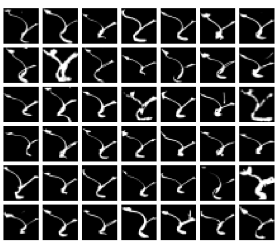
(d) Heatmaps for the 18 resulting clusters.

Figure 1: Visualisation of a subset using one character from the Kuzushiji-MNIST dataset. (a) Visualisation using low resolution. (b) Heatmaps from each cluster. (b) Visualisation using high resolution. (c) Heatmaps from each cluster. The number above the heatmaps tells how many images are superimposed to make the heatmap. Figure best viewed in color.

Figure 1 shows the result of clustering using the character を from the Kuzushiji-MNIST dataset. In Figure 1a) a rather large scale factor $h = 3.0$ is chosen and three main clusters appears representing the three main ways of writing the character を . By decreasing the scale factor to $h = 1.0$ more clusters are found as shown in Figure 1c) and the heat maps in Figure 1d) reveal more variation among the individual characters.

In Figure 2 all characters from four different clusters are are depicted. Each cluster contains rather similar ways


(a) Two different sub-clusters from the upper main cluster.


(b) Another variant of the character from the lower left cluster.

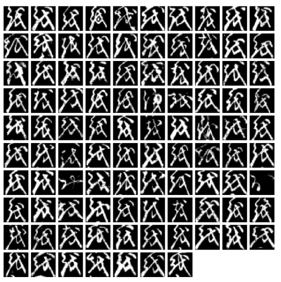

(c) Yet another variant from the lower right cluster.

Figure 2: Visualisation of the actual content of some clusters. (a) Two different clusters belonging to the first heatmap in Fig. 1b, which are represented in Fig. 1d as the second, third and fourteenth heatmap respectively. (b) This cluster belongs to the second heatmap in Fig. 1b, and is the eleventh in Fig. 1d. (c) This cluster belongs to the third heatmap in 1b, and is the first in Fig. 1d.

of writing each character. This is an important feature of the clustering since it makes it possible to create subspaces that correspond to each character variation, i.e. it makes it possible to classify similar looking characters by each such subspace.

2.4 Subspace Definition

We follow the same definition of subspaces as in [HLV19, Laa07, OK88]. Every image to be classified is represented by a feature vector $\mathbf{x}$ with m real-valued elements $\mathbf{x}_j = \{x_1, z_2 ... x_m\}, \in \mathbb{R}$, such that operations take place in a m-dimensional vector space $\mathbb{R}^m$. Any set of n linearly independent basis vectors $\{\mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_n\}$, where $\mathbf{u}_i = \{w_{1,j}, w_{2,j} ... w_{m,j}\}, w_{i,j} \in \mathbb{R}$, which can be combined into an $m \times n$ matrix $\mathbf{U} \in \mathbb{R}^{m\times n}$, span a subspace $\mathscr{L}_\mathbf{U}$

$$\mathscr{L}_\mathbf{U} = \{\mathbf{x} | \mathbf{x} = \sum_{i=1}^{n} \rho_i \mathbf{u}_i, \rho_i \in \mathbb{R}\} \tag{2}$$

where,

$$\rho_i = \mathbf{x}^T \mathbf{u}_i = \sum_{j=1}^{m} x_j w_{i,j} \tag{3}$$

By projecting the vector $\mathbf{x}$ into each and every subspace $\mathscr{L}_{\mathbf{U}_k}$ classification is performed. The vector $\hat{\mathbf{x}}$ will therefore be a reconstruction of the input vector $\mathbf{x}$, using all vectors in the subspace through

$$\hat{\mathbf{x}} = \sum_{i=1}^{n} (\mathbf{x}^T \mathbf{u}_i)\mathbf{u}_i \tag{4}$$

$$= \sum_{i=1}^{n} \rho_i \mathbf{u}_i \tag{5}$$

$$= \mathbf{U}^T \mathbf{U} x^T \tag{6}$$

Since all the vectors in $\mathbf{U}$ are normalised, the norm of the projected vector can be simplified as

$$||\hat{\mathbf{x}}||^2 = (\mathbf{U}x^T) \cdot (\mathbf{U}x^T) \tag{7}$$

$$= (\mathbf{U}x^T)^2 \tag{8}$$

$$= \sum_{i=1}^{n} {\rho_i}^2 \tag{9}$$

Therefore, the feature vector $\mathbf{x}$, which is most similar to the feature vectors that were used to construct the subspace in question $\mathscr{L}_{\mathbf{U}_k}$ will subsequently also have the largest norm $||\hat{\mathbf{x}}||^2$.

3 CASCADING AND ENSEMBLES

Cascading [GB00], first introduce by Viola and Jones for face detection [VJ01] can be regarded as a special case of ensemble learning [OM99, Rok10]. It is based on the concatenation of several classifiers rather than combining the result of multiple classifiers in order to obtain a better result. The idea is that further processing is done only on tentative classes and non-relevant classes are progressively excluded. By implementing cascading for a neural network, the overall workload will not only decrease, but in some sense the network will mimic what humans do. When being presented to a classification problem a person often instantly see the correct answer, i,e. when being sure enough, no further examination is necessary and therefore we sometimes naturally make use of what is known as *early termination*. However, sometimes a person have to take a closer look and choose from two or more tentative answers by scrutinising the images further, e.g. "is the number a sloppily written '5' or a '6'?". Hence, the decision has already been made that it cannot be any other number. Such closer examination helps in making a more accurate classification and therefore humans also deploy the strategy of cascading in such cases.

3.1 Ensembles

Ensembles can be setup in several ways, and here was chosen to use four EPSC networks working on the very same test data images but using four different variants of features that are extracted from the images. Hence, the t-SNE will create a bit different looking clusters depending on what kind of image features the different features extractors are focusing on.

3.2 Features

The features used herein are handcrafted in order to be faster than learned features from CNN's. They were made from different combinations of HoG [DT05] and combinations of some of the most significant elements of the magnitude of the FFT (subsequently referred to as mFFT), i.e. low pass filters, [MHWS17] of different parts of the image and also in combination with down scaled versions of the image itself. These were made ad hoc through experiments in order to find good features yielding both high accuracy used alone, but also used together in an ensemble. In the paper introducing EPSC [HLV19] a similar approach, using a combination of HoG and a down scaled version of the image, achieved an accuracy of 99.2%, i.e. error of 0.8%. As shown in table 1 using mFFT helped lowering the error below that error for all features.

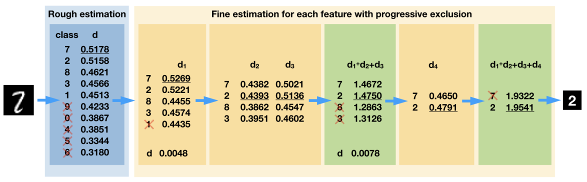
Feature 1 was setup by combining two HoG features with different cell size and five mFFT features from five different partitions of the image. The second Feature is a combination of one HoG and elevn mFFT of eleven different partitions. The third feature is a combination of one HoG, five mFFT and the image itself down scaled to a quarter of its size. The fourth feature is a combination of one HoG and five mFFT.

3.3 Cascading

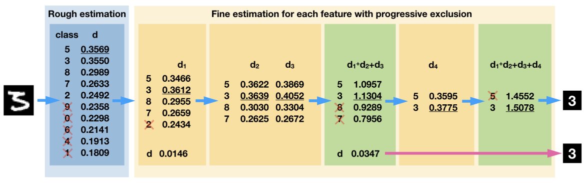
The cascading was setup in six stages as visualised in Figure 3. In the first stage a rough estimation on feature 1 is done by using a rather large scale parameter $h = 2.5$ for clustering, which will produce a quite small neural net that produces good enough results to distinguish with a high probability the (obvious) false positives from the tentative true positives.

In the second stage, one net (still using the same feature) evaluates only the top five candidates in a more accurate but larger net that was produced by using a smaller scale parameter $h = 0.5$ in order to get smaller clusters. Hence, only the parts of the larger net that corresponds to these five candidates need to be evaluated. The lowest ranking candidate is then excluded from further processing. The advantage here is that only one step of t-SNE is used for this particular feature, used in the first two stages, while two different KDE's are produced. The latter is many times faster than the t-SNE so the overhead is small.

In the third stage, the nets corresponding to the four candidates, evaluates the second and third features. These are summed together in the fourth stage, together with the output of the second stage, and a new ranking is done based on the accumulated results from these three finer nets. The two lowest score candidates are then discarded and in the fifth stage the remaining two candidates are evaluated using the fourth feature and their corresponding nets. Finally, a total accumulated score from all four stages is used to finally determine the class.



(a) This time a '2' is fed into the network. Both the rough estimation and the first network regards it to be a '7'. However, the others conclude that it must be a '2'.



(b) An image of the number '3' is fed into the network. Note that the rough estimation regards it to be a '5' but the finer estimations concludes it its a '3'. Early termination can be done in this case since the difference between the top two classes is large enough.

Figure 3: Two examples of how the rough classification finds 5 candidates for further investigation in the cascading. Next one net evaluates the candidates using a more accurate but larger net and removes the lowest ranking candidate. Next two nets evaluates the remaining four and a ranking is done based on the accumulated results from the three finer nets. Finally only the top two candidates are evaluated by the fourth net, and a total accumulated result is used to determine the winner. Figure best viewed in color. Note that the top candidate is underlined for each step.

The whole procedure is visualised for two examples of non trivial images in Figure 3. Note how the pipeline eventually comes up with the correct answer even if the top candidate is wrong in the beginning of the pipeline.

3.4 Early Termination

If the difference in score between the top candidates at stage 2 or 4 exceeds a certain threshold θ, it can in most cases be safely concluded that the top candidate is already found, since the uncertainty is considered low. If the difference in score on the other hand is low, it means that the nets cannot easily tell which class it is, i.e. the uncertainty is high, and further processing is necessary. Experimentally it was found that a threshold of $\theta = 0.03$ in both stages worked well. However, since stage 4 is the sum of three scores it means that the values compared are higher and the difference generally is higher, which in its turn means that the threshold is in fact set more tight in this stage.

4 METHOD AND DATASETS

This section describes the method used in the experiments and the individual datasets.

Three important questions needed to be addressed. First of all, how much would an ensemble of EPSC improve the classification results? The second is, how much would cascading improve speed and how close to the accuracy of the ensemble could be achieved? Finally, how much would an early termination mechanism affect both speed and accuracy?

First different variants of features where run on MNIST to find good candidates. A combination of HoG and mFFT on the whole image (or partitions of the image) turned out to give better results than just using HoG and a down scaled version of the image. As explained earlier they were made ad hoc in a trial and error process. This means that other features might do better for other types of input data, and we propose this for future research.

Next combinations of four features where run in an ensemble to find what four features actually give the most different classifications in order to diminish the total error in the ensemble.

Finally, the experiments where repeated on the other data sets without changing any parameters or features. This was done in order to determine whether the cascading and ensembles still improves on these datasets.

4.1 Datasets

The proposed EPSC framework has been tested on the images from the following datasets:

- The MNIST database of handwritten digits [LCB10], which is the standard benchmark dataset within the machine learning community. It consists of a training set of 60,000 examples, and a test set of 10,000 examples.
- The E-MNIST (MNIST) dataset has the same example and test set sizes as MNIST [CATvS17], but the conversion process tried to optimise the size of each digit.
- L-EMNIST has the same example and test set sizes as MNIST and contains letters (26 in total).
- B-MNIST has the same example and test set sizes as MNIST and contains both letters and digits (47 different in total).
- The Kuzushiji-MNIST (K-MNIST) dataset has the same example and test set sizes as MNIST [CBK*18] and contains 10 differennt Japanese Hentaigana, where each character may possess rather different forms. The letters present in the dataset are あきすつなはまやれを.
- The Fashion MNIST (F-MNIST) [XRV17] dataset has the same example and test set sizes as MNIST and contains miniature images of clothes and bags.

These were chosen for several reasons. First of all most of them contain handwritten numbers and letters, which is of our main interest, except for the F-MNIST dataset that contains miniature images of clothes and bags. Secondly, they all have similar format, making them easy to evaluate for algorithms that have already been tested on the MNIST dataset, which is very well researched for machine learning. Nevertheless, it is a good starting point for understanding how well classification of handwritten text can perform.

5 RESULTS

Table 1 shows the classification errors on the datasets using different approaches and features. In all cases the four features used on ordinary EPSC (denoted $F1$, $F2$, $F3$ and $F4$) perform a little better than using the feature for EPSC reported in [HLV19].

More importantly, cascading on all features (C-ESCP) as well as a full ensemble without cascading (E-EPSC) perform much better than any of the single networks using any of the features. One can also note that cascading performs almost as well as the full ensemble, but being much faster. Furthermore, the cascading on just one feature (C-F1), which corresponds to the first stage in Figure 3 also works very well. It is, not surprisingly, the fastest of all methods as can be seen in Table 3, but is of course not the most accurate since it does not make use of all the features and the full cascading pipeline.

The result of using C-EPSC compared to what was presented in different papers introducing each of the datasets are shown in Table 2. Certainly, better accuracy have been achieved by much more complicated deep learning architectures since the datasets where introduced. However, the point here is not to beat the state of the art by using the C-EPSC, but rather to show that it is possible to come very close to a much lesser cost, both for learning but also for the classification.

The cascading with early termination, denoted C_{et}, comes close to what the cascading does when it comes to accuracy, but is quite a lot faster. In fact, it comes close to what cascading using only one feature (C-F1) does. The reason is that the whole cascading is seldom necessary for an accurate classification. Table 4 shows the percentage of classifications that goes on beyond stage 2 and 4. Hence, the lower percentage, the more cases can be terminated early.

It can be seen in Table 4 that the percentage vary quite a lot among the datasets and one reason is probably the fact that the same threshold of $\theta = 0.03$ for both stages were used for all datasets. This implies that this is one parameter that could be learned for an implementation of word classification.

Table 1: Prediction error (%) using different features, cascading (C), with early termination (C_{et}) and the full ensemble (E)

Dataset	EPSC	E-EPSC	C-EPSC	C_{et}-EPSC	C-F1	F1	F2	F3	F4
MNIST	0.80	0.56	0.56	0.56	0.69	0.69	0.75	0.70	0.74
E-MNIST	0.78	0.58	0.59	0.60	0.66	0.65	0.63	0.66	0.61
L-MNIST	7.60	6.38	6.43	6.43	6.90	6.89	6.97	6.84	6.98
B-MNIST	14.13	12.62	12.65	12.66	13.35	13.32	13.28	13.52	13.66
K-MNIST	3.08	2.17	2.20	2.19	2.52	2.51	2.15	2.35	2.58
F-MNIST	11.88	8.73	8.73	8.73	10.11	10.11	10.05	9.82	10.30

Table 2: Classification accuracy (%) for some popular datasets and comparison with other learning methods. Note that we have taken the results from each paper proposing these methods and therefore the table is not covering the use of all methods on all datasets.

Learning method	MNIST	E-MNIST	L-MNIST	B-MNIST	K-MNIST	F-MNIST
# classes	10	10	26	47	10	10
C-EPSC	99.44	**99.41**	**93.57**	**87.35**	97.80	**91.23**
EPSC [HLV19]	99.20	99.22	92.40	85.87	96.92	88.12
OPIUM [CATvS17]	-	96.22	85.15	78.02	-	-
MLP [XRV17]	97.20	-	-	-	-	87.1
Keras CNN[CBK*18]	99.06	-	-	-	95.12	-
PreActResNet-18 [CBK*18]	99.56	-	-	-	97.82	-
PreActResNet-18 + Manifold Mixup [CBK*18]	**99.54**	-	-	-	**98.83**	-

Table 3: Wall clock time of classification in seconds of using different features, cascading (C), with early termination (C_{et}) and the full ensemble (E).

Dataset	E-EPSC	C-EPSC	C_{et}-EPSC	C-F1	F1	F2	F3	F4
MNIST	825	381	192	187	250	247	178	157
E-MNIST	797	379	193	194	248	228	177	153
L-MNIST	3667	834	592	509	1088	1050	736	696
B-MNIST	4296	691	565	478	1321	1198	888	786
K-MNIST	614	306	172	159	186	183	131	118
F-MNIST	684	326	257	164	202	202	147	134

Table 4: Percentage of how many are not terminated early in step 2 and 4 respectively, but passes to the next step in the cascading with respect to the total amount of images to be classified.

Dataset	ET-2	ET-4
MNIST	6.23%	1.04%
K-MNIST	11.52%	2.75%
F-MNIST	63.20%	33.73%
E-MNIST	4.01%	0.77%
L-MNIST	45.30%	21.22%
B-MNIST	30.20%	12.15%

6 CONCLUSION

Deep learning architectures and the learning and classification process are not easy to visualise, nor to explain. The EPSC on the other hand is based on clustering of embedded features. Both the clusters and corresponding images as well as the resulting neural nets can easily be visualised and comprehended. Therefore, EPSC is an interesting alternative in the domain of XAI. It was experimentally shown that both ensembles and cascading can improve the speed and accuracy. Furthermore, early termination bring down the cost of classification quite a lot, without compromising the accuracy.

For future work we propose to use EPSC for word images rather on individual letters. Moreover, it is necessary to find some plausible explanation why HoG and features based on the magnitude of FFT works so well together.

7 REFERENCES

[ADRS*19] Arrieta A. B., D'iaz-Rodr'iguez N., Ser J. D., Bennetot A., Tabik S., Barbado A., Garc'ia S., Gil-L'opez S., Molina D., Benjamins R., Chatila R., Herrera F.: Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *ArXiv abs/1910.10045* (2019).

[CATvS17] Cohen G., Afshar S., Tapson J., van Schaik A.: EMNIST: an extension of MNIST to handwritten letters. *CoRR abs/1702.05373* (2017).

[CBK*18] Clanuwat T., Bober-Irizar M., Kitamoto A., Lamb A., Yamamoto K., Ha D.: Deep learning for classical japanese literature. *CoRR abs/1812.01718* (2018).

[CHTT96] Carbon M., Hallin M., Tat Tran L.: Kernel density estimation for random fields: the l 1 theory. *Journal of nonparametric Statistics 6*, 2-3 (1996), 157–170.

[CM02] Comaniciu D., Meer P.: Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 5 (May 2002), 603–619.

[CPC19] Carvalho D. V., Pereira E. M., Cardoso J. S.: Machine learning interpretability: A survey on methods and metrics. *Electronics 8*, 8 (Jul 2019), 832.

[DKMJ18] Dutta K., Krishnan P., Mathew M., Jawahar C.: Improving cnn-rnn hybrid networks for handwriting recognition. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)* (2018), IEEE, pp. 80–85.

[DT05] Dalal N., Triggs B.: Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (June 2005), vol. 1, pp. 886–893 vol. 1.

[EKSX96] Ester M., Kriegel H.-P., Sander J., Xu X.: A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (1996), KDD96, AAAI Press, pp. 226–231.

[FH75] Fukunaga K., Hostetler L.: The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory 21*, 1 (January 1975), 32–40.

[GB00] Gama J., Brazdil P.: Cascade generalization. *Machine Learning 41*, 3 (Dec 2000), 315–343.

[GSC*19] Gunning D., Stefik M., Choi J., Miller T., Stumpf S., Yang G.-Z.: Xai—explainable artificial intelligence. *Science Robotics 4*, 37 (2019).

[HLV19] Hast A., Lind M., Vats E.: Embedded prototype subspace classification : A subspace learning framework. In *The 18th International Conference on Computer Analysis of Images and Patterns (CAIP)* (2019), Lecture Notes in Computer Science, pp. 581–592.

[HW79] Hartigan J. A., Wong M. A.: Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 28*, 1 (1979), 100–108.

[JMF99] Jain A. K., Murty M. N., Flynn P. J.: Data clustering: a review. *ACM computing surveys (CSUR) 31*, 3 (1999), 264–323.

[KDJ18] Krishnan P., Dutta K., Jawahar C.: Word spotting and recognition using deep embedding. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)* (2018), IEEE, pp. 1–6.

[KLR*77] Kohonen T., Lehtiö P., Rovamo J., Hyvärinen J., Bry K., Vainio L.: A principle of neural associative memory. *Neuroscience 2*, 6 (1977), 1065 – 1076.

[KO76] Kohonen T., Oja E.: Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. *Biological Cybernetics 21*, 2 (Jun 1976), 85–95.

[Kri19] Krishnan M.: Against interpretability: a critical examination of the interpretability problem in machine learning. *Philosophy & Technology* (2019).

[KRMV76] Kohonen T., Reuhkala E., Mäkisara K., Vainio L.: Associative recall of images. *Biological Cybernetics 22*, 3 (Sep 1976), 159–168.

[Laa07] Laaksonen J.: *Subspace classifiers in recognition of handwritten digits*. G4 monografiaväitöskirja, Helsinki University of Technology, 1997-05-07.

[LCB10] LeCun Y., Cortes C., Burges C.: Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist 2* (2010).

[MH08] Maaten L. v. d., Hinton G.: Visualizing data using t-sne. *Journal of machine learning research 9*, Nov (2008), 2579–2605.

[MHWS17] Matuszewski D. J., Hast A., Wählby C., Sintorn I.-M.: A short feature vector for image matching: The log-polar magnitude feature descriptor. *PLOS ONE 12*, 11 (11 2017), 1–21.

[OK88] Oja E., Kohonen T.: The subspace learning algorithm as a formalism for pattern recognition and neural networks. In *IEEE 1988 International Conference on Neural Networks* (July 1988), vol. 1, pp. 277–284.

[OM99] Opitz D., Maclin R.: Popular ensemble methods: An empirical study. *J. Artif. Int. Res. 11*, 1 (July 1999), 169–198.

[RM00] Roerdink J. B., Meijster A.: The watershed transform: Definitions, algorithms and parallelization strategies. *Fundam. Inf. 41*, 1,2 (Apr. 2000), 187–228.

[Rok10] Rokach L.: Ensemble-based classifiers. *Artificial Intelligence Review 33*, 1 (2010), 1–39.

[SF16] Sudholt S., Fink G. A.: Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *ICFHR* (2016), IEEE Computer Society, pp. 277–282.

[Sha18] Shapshak P.: Artificial intelligence and brain. *Bioinformation 14*, 1 (2018), 38.

[VJ01] Viola P., Jones M.: Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (Dec 2001), vol. 1, pp. I–I.

[VS91] Vincent L., Soille P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 6 (June 1991), 583–598.

[WEG87] Wold S., Esbensen K., Geladi P.: Principal component analysis. *Chemometrics and intelligent laboratory systems 2*, 1-3 (1987), 37–52.

[WLK*67] Watanabe W., Lambert P. F., Kulikowski C. A., Buxto J. L., Walker R.: Evaluation and selection of variables in pattern recognition. In *Computer and Information Sciences* (1967), Tou J., (Ed.), vol. 2, New York: Academic Press, pp. 91–122.

[WP73] Watanabe S., Pakvasa N.: Subspace method in pattern recognition. In *1st Int. J. Conference on Pattern Recognition, Washington DC* (1973), pp. 25–32.

[XRV17] Xiao H., Rasul K., Vollgraf R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR abs/1708.07747* (2017).

[XW05] Xu R., Wunsch D.: Survey of clustering algorithms. *IEEE Transactions on neural networks 16*, 3 (2005), 645–678.

Journal of WSCG
ISSN 1213-6972 http://www.wscg.eu Vol.28, No.1-2, 2020

Regions Based Semi-fragile Watermarking Scheme for Video Authentication

Amal Hammami
Research Groups in Intelligent Machines
University of Sfax
National Engineering
School of Sfax
Sfax, 3038, Tunisia
amal.hammami@enis.tn

Amal Ben Hamida
Research Groups in Intelligent Machines
University of Sfax
National Engineering
School of Sfax
Sfax, 3038, Tunisia
amal.benhamida@enis.tn

Chokri Ben Amar
Research Groups in Intelligent Machines
University of Sfax
National Engineering
School of Sfax
Sfax, 3038, Tunisia
chokri.benamar@ieee.org

Henri Nicolas
Bordeaux Computer Science Research Laboratory
University of Bordeaux 1
Talence, 33405, France
henri.nicolas@u-bordeaux.fr

ABSTRACT

In this paper, we propose a new semi-fragile watermarking scheme in the frequency domain for surveillance videos authentication. Our system starts operating by generating a binary watermark based on a novel watermark construction process. This latter combines Speeded Up Robust Features (SURF) and Maximally Stable Extremal Regions (MSER) detectors to extract frames relevant features that can resist common attacks while being fragile to intentional manipulations. Furthermore, the watermark security is improved using torus automorphism mapping. For the embedding process, Regions of Interest (ROI) are detected and then used as watermark holders. These regions are decomposed into different frequency sub-bands using Singular Value Decomposition (SVD) as well as Discrete Wavelet Transform (DWT). Then, the watermark is embedded in selected bands following an additive method. A blind detection is conducted to extract the hidden signature from the watermarked video. Evaluation results show that the proposed scheme is suitable for authentication purpose since it efficiently discriminates malicious manipulations from non-malicious ones. Besides, it preserves a high level of perceptual quality.

Keywords

Video authentication, semi-fragile watermarking, discrete wavelet transform, singular value decomposition.

1. INTRODUCTION

Recently, the ever-evolving multimedia technology and communication makes surveillance system installation further flexible as well as more cost effective. Accordingly, video surveillance cameras are broadly used for security purposes by recording daily activities in several places such as airports, hospitals and train stations [Kha15, Abe14, Ben14].

However, the progress in digital technologies field leads also to the development of powerful video processing tools that facilitate the recorded videos content illegal modification by fraud person. As a result, content-authentication becomes an increasingly important requirement for video security protecting especially in surveillance context where video can be involved as a legal proof in forensic investigations.

Video watermarking has emerged as an appropriate technique to tackle this issue. Fundamentally, this authentication approach consists in embedding a secret information, which is referred as a watermark in the host video [Bpa14, Jos17, Mko12]. When needed, the embedded information is extracted from the watermarked video to check whether it has been tampered or not, thereby to verify its content integrity and authenticity. An efficient watermarking technique should provide a trade-off between several requirements mainly robustness, imperceptibility and capacity [Aga19, Mch14].

In literature, several video watermarking schemes are proposed. Depending on the watermark selection holders, these video watermarking approaches are dived into two categories. The first one is frame by frame watermarking where the authentication signature is hidden in all regions in each video frame. Conversely, in the schemes dedicated for the second class only regions of interest (ROI) are watermarked. These ROI are identified based on specific criteria in such way to ensure that their modification will lead to a detectable change. ROI based techniques are recognized as very compression resilient approaches that provide a high level of imperceptibility [Ker17]. Moreover, watermarking schemes can be classified into two sets according to the insertion domain. The first category represents the spatial domain techniques in which the watermark embedding is performed by directly modifying frame pixels values. These techniques exhibit a low complexity while preserving a poor robustness [Ara19]. The second type is the frequency domain based techniques. In this case, the video frame is firstly transformed to a new domain. Then, the watermark bits are hidden into the transformed coefficients. Frequency domain based watermarking techniques are more efficient and ensure better robustness and higher invisibility compared to the spatial domain ones [Gup16, Mas17].

SVD and DWT as well as Lifting Wavelet Transform (LWT) and Discrete cosine transform (DCT) are the most used transform domain methods. Indeed, in [Bha18] a LWT based watermarking video scheme is introduced. In this scheme, the watermark bits are hidden in the quantified LH3 coefficients resulting from the application of the LWT to a selective set of frames. Simulation results prove the robustness and the imperceptibility of this technique. In [Him18], a watermarking technique combining DWT and SVD is presented. After being ciphered by a chaotic logistic function, the watermark is embedded only within key-frames. This scheme is robust against various attacks and it exhibits good imperceptibility level. A content-based authentication technique using DCT is proposed in [Far16]. In this scheme, frame index and invariant features extracted from intra macroblocks form the authentication code. Next, this latter is inserted into Quantized coefficients of the DCT performed on an arbitrary chosen Group of Pictures. This watermarking scheme is immune to non-malicious attacks but sensitive to content changing ones. Likewise, another semi-fragile watermarking scheme for surveillance video authentication is presented in [Man16]. In this approach, only scenes with major changes are implied in the watermarking process. The signature is generated from the highest informative block of the DCT coefficients and inserted in the lowest one.

Taking into account the above highlighted advantages and drawbacks of each watermarking category, we propose in this paper a ROI based semi-fragile watermarking scheme in the frequency domain using SVD and DWT for video authentication. The rest of this paper arrangement is as follows: section 2 provides a detailed description of the proposed watermarking scheme. Next, we present and discuss the obtained results in section 3. Finally, section 4 concludes this work.

2. PROPOSED SCHEME

The proposed video watermarking scheme involves three main components namely the watermark construction, the watermark embedding and the watermark extraction. These processes mechanisms are discussed in the following subsections. The originality of this work includes:

1) The proposed watermark construction strategy that allows to obtain a to-be-embedded watermark that can resist common attacks while being fragile to intentional manipulations. This is achieved thanks to the use of two efficient features detectors Speeded Up Robust Feature (SURF) and Maximally Stable Extremal Regions (MSER).

2) The suitable choice of the more appropriate regions ROI for signature insertion since they are the most targeted regions by malicious attacks in a video frame and each forgery on their content will be detectable.

3) The proposed frequency domain embedding process used to conceal the signature into the host video that involves two different domain transform techniques namely the SVD and the DWT. This watermark insertion method is designed upon the balance amongst the imperceptibility and the semi-fragility requirements consideration.

2.1 Watermark Construction Process

In the proposed scheme, the host video is initially divided into sequences of N frames. N is used as a first watermarking key and set to the number of frames per second (FPS) in each video. Then, as illustrated in Figure 1, an authentication watermark is generated from the first frame in the considered sequence using relevant features extracted from video frames. Next, regions of Interest (ROI) are detected. Indeed, moving objects are identified as the most important regions in the frame especially in video surveillance context [Ker17, Tbo12]. A technique that involves an adaptive improved version of Gaussian Mixture Model (GMM) [Cst19] for background subtraction and several morphological filters is used to accurately isolate the moving objects in every video frame [Abe13, Ben13]. Key points are further extracted from these ROI using SURF detector which is invariant to geometrical transformations and additive noise [Hba08]. To enhance the watermark robustness, we opt for combining the SURF detector and the MSER one [Jma04]. MSER detector allows to determine connected pixels with intensity values that exhibit a negligible variation over several thresholds ranges. MSER regions are immune to illumination variation, rotation, scaling and translation [Ali18]. After performing MSER, only SURF key points hold in MSER sets are selected for the further steps and the remainder points are discarded. Considered points coordinates are converted to binary codes and then concatenated in one sequence. To increase the scheme security, this latter is scrambled using one dimensional Torus isomorphic mapping [Fzh19] which is a typical chaotic map operating following the formula:

$$X' = (K \times X) \bmod (L) + 1 \quad (1)$$

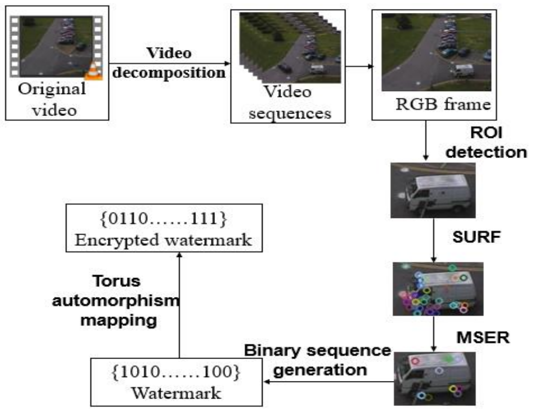


Figure 1. The proposed watermark construction process schematic framework.

Where mod denotes the modulus function. X' and X are the original bit position and the encrypted one. K is the second watermarking key, which should be a prime integer, and L is the binary sequence length.

Finally, the encrypted binary sequence is used as a watermark.

2.2 Watermark Embedding Process

The embedding procedure framework is presented in Figure 2. As already mentioned, the host video is decomposed in sequences containing N frames and a watermark is constructed for every sequence and embedded within each frame of the given sequence following the described procedure below. To start with, the RGB frame is converted to the YUV space color and its three constituent planes namely Y, U and V are separated. Only Y component is selected for the watermarking since it is more difficult to perceptually notice changes on luminance component as compared to chrominance ones. Following, the watermark is hidden in each ROI, recognized as already described in the previous subsection, which is an efficient choice for the authentication purpose as these regions hold on the most relevant information in the frames and every forgery on their content will be detectable; thus, each ROI is split into non-overlapping blocks.

To enhance the watermarking capacity, we adopt the 4x4 size instead of the generally used 8x8 size since one watermark bit is hidden in each block.

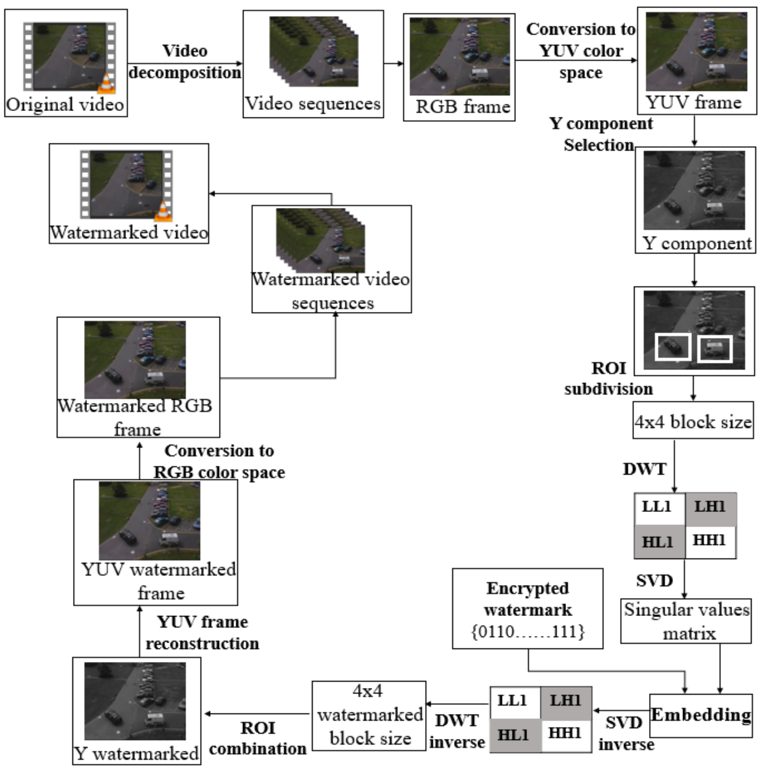


Figure 2. The proposed watermark construction process schematic framework.

As the proposed watermarking scheme operates in the frequency domain, each block undergoes one level DWT. This to avoid the visual degradation and the complexity cost yielded by multilevel DWT using. Providing the best compromise between the robustness and the imperceptibility makes the mid frequency sub bands the most appropriate for the watermark insertion. Hence, SVD is applied to these sub-bands. Due to its stable nature, the singular values matrix S is the most convenient location for watermarking. Finally, the watermark embedding is achieved by exploiting an additive blind method [Ham19]. This method allows the watermark strength control and optimization in order to well establish the trade-off between the robustness and the imperceptibility using the following equations:

If W=0

$$\begin{cases} S'(0,0) = S(0,0) + k_\alpha \\ S'(1,1) = S(0,0) \end{cases} \quad (2)$$

Otherwise

$$\begin{cases} S'(0,0) = S(1,1) + k_\beta \\ S'(1,1) = S(1,1) \end{cases} \quad (3)$$

With

$$k_\alpha = \frac{S(0,0)+S(1,1)}{\alpha} \quad (4)$$

$$k_\beta = \frac{S(0,0)+S(1,1)}{\beta} \quad (5)$$

With W is the to-be-embedded bit, S and S’ are the original and the watermarked versions of S matrix. α and β are two scaling factors. Their values are fixed based on several experiments which demonstrate that the couple (2, 4) gives the best compromise between the imperceptibility and the robustness [Ham19]. Hence α=2 and β=4 are the used values in this work.

Finally, the inverse of each used function is performed to obtain the watermarked frame**.**

2.3 Watermark Extraction Process

The general architecture of the blind detection process is depicted in Figure 3. It starts operating with the same steps of the embedding process. After decomposing the watermarked video in sequences using the same watermarking key N, the frame is converted to YUV space and the Y plane is extracted. Once the ROI are detected and divided into 4x4 blocks, DWT and SVD are applied. The resulting singular values matrix coefficients are then scanned to extract the embedded information according to the following equation:

If S”(0,0)-S”(1,1) > $\frac{k_\alpha + k_\beta}{2}$

W” = 0

Otherwise (6)

W” = 1

Where S” and W” denoted the extracted singular values matrix and the extracted watermark bit. Hence, from each frame F_j in a given sequence a watermark denoted W_{Fj} will be extracted.

Let denote $\Omega_i = \{W^i_{F1}, W^i_{F2}, \ldots, W^i_{FN}\}$ a sample space. With W^i_{Fj} is the i-th bit in W_{Fj}. A global watermark W_E is built by determining each of its bits W^i_E as follows:

If p (W^i_{Fj} =1) > p (W^i_{Fj} =0)

W^i_E = 1

Otherwise (7)

W^i_E = 0

With p the probability function.
Besides, the generation process is performed under the first watermarked frame in the given sequence in order to obtain the reconstructed watermark W_R.
W_E and W_R are compared to verify the watermarked video authenticity.

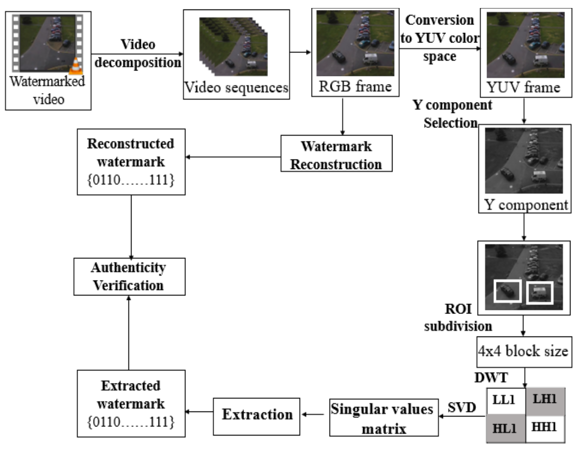


Figure 3. The proposed watermark extraction process schematic framework.

3. EVALUATION RESULTS

The proposed scheme performance is investigated in terms of imperceptibility and robustness. Various surveillance as well as common videos that comprise at least one moving object are used. These videos include test.avi, camera2.avi, video1.avi, akiyo.avi, news.avi and coastguard.avi. The first three sequences belong to PETS benchmark datasets. The last three videos are often employed to evaluate previous existing approaches.

3.1 Imperceptibility Assessment

To evaluate the visual distortion yielded by the watermarking scheme, the Peak Signal to Noise Ratio (PSNR) is used (Nta18, Kad16). The PSNR values corresponding to every test video are calculated using equation (8) and presented in Figure 4.

$$\mathrm{PSNR} = 10\log\left(\frac{255^2}{\mathrm{MSE}}\right) \tag{8}$$

Where MSE designed the Mean Square Error.

As clear from Figure 4, the obtained PSNR values vary between 48.7431 dB and 73.425 dB which indicates the high similarity between the original videos and the watermarked ones. Moreover, the original frames and the corresponding watermarked ones, illustrated in Figure 5, confirm that the two frames versions are perceptually identical. In fact, the suitable choice of the ROI as well as the selection of the singular values matrix of the SVD transform allow guarantying this high invisibility level.

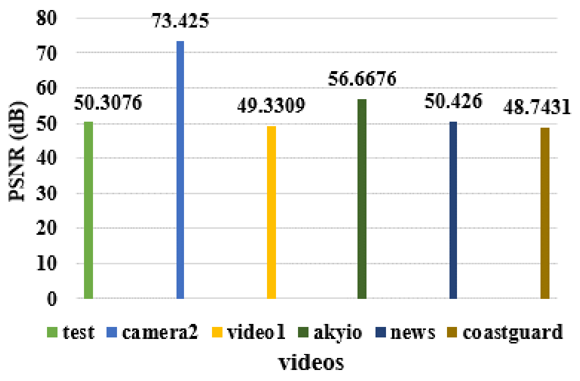


Figure 4. PSNR values for several watermarked videos.

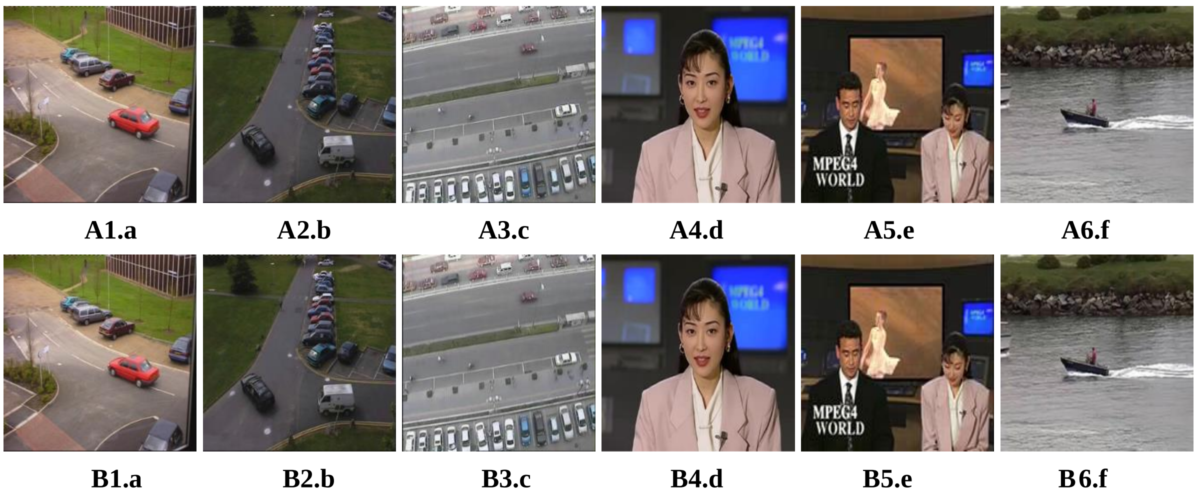


Figure 5. Samples of (A) Original frames and their (B) watermarked versions from different used videos. (a) test.avi, (b) camera2.avi, (c) video1.avi, (d) akiyo.avi, (e) news.avi, (f) coastguard.avi.

Videos / Attacks	Test	Camera2	Video1	Akiyo	News	Coastguard
Salt & Pepper	0.0217	0.0018	0.0412	0.0315	0.0106	0.0330
Gaussian noise (0.01)	0.0091	0.0008	0.0414	0.0330	0.0053	0.0329
Gaussian noise(0.02)	0.0105	0.0008	0.0404	0.0340	0.0061	0.0372
Median Filter (3x3)	0.0158	0.0011	0.0390	0.0268	0.0058	0.0310
Median Filter (5x5)	0.0153	0.0004	0.0254	0.0192	0.0088	0.0326
MJPEG compression	0.0184	0.0010	0.0398	0.0009	0.0009	0.0329
Rotation (10°)	0.0215	0.0006	0.0312	0.0244	0.0155	0.0363
Rotation (20°)	0.0209	0.0012	0.0310	0.0328	0.0087	0.0537
Rotation (45°)	0.0149	0.0010	0.0408	0.0307	0.0056	0.0485
Brightness (+10%)	0.0140	0.0019	0.0440	0.0335	0.0061	0.0331
Brightness (+20%)	0.0181	0.0012	0.0392	0.0322	0.0062	0.0329
Brightness (-10%)	0.0237	0.0008	0.0396	0.0326	0.0061	0.0329
Brightness (-20%)	0.0189	0.0013	0.0398	0.0327	0.0065	0.0328
Contrast (x2)	0.0080	0.0108	0.0291	0.0325	0.0093	0.0341
Contrast (x0.5)	0.0122	0.0007	0.0416	0.0314	0.0046	0.0346

Table 1. The BER values under various unintentional attacks

3.2 Robustness Assessment

Bit error rate (BER), which is calculated via equation (9), is the metric used to evaluate the robustness of the technique and its ability to verify the content authentication (Asi15).

$$\text{BER} = \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{W_R(i,j) \oplus W_E(i,j)}{m \times n} \qquad (9)$$

With W_E and W_R are the extracted watermark and the reconstructed one.

The proposed scheme is designed to be capable to discriminate between unintentional attacks and intentional ones. To achieve this purpose, a threshold T is required. Since small BER values reflect the high immunity to a considered attack, the threshold T value is empirically fixed to 0.1.

The authentication decision is done based on the following assumptions:

If BER < T

The video is authentic.

Otherwise

The video is intentionally tampered.

As already mentioned, the scheme robustness is tested against two sets of attacks i.e. unintentional attacks and intentional ones. Table 1 exhibits the BER values obtained under unintentional attacks including (noise adding, filtering, rotation, MJPEG compression and brightness and contrast variation). From this table, it is evident that the proposed approach is immune to Gaussian noise as well as salt and pepper attacks since the obtained BER values are below the preset threshold T=0.1. The combination of SURF and MSER detectors as well as using the DWT, which are noises immune, guarantee this high resilience.

Furthermore, our watermarking scheme exhibits a good robustness against rotation attack. In fact, the obtained BER values vary between 0.0006 and 0.0537 for different rotation degrees. This robustness is achieved due to the utilization of SURF detector when designing the watermark and the involvement of SVD during the embedding process which are both rotation invariant. For median filter attack, the resulting BER values are ranged between 0.0004 and 0.0390 thereby inferior to T. Thus, the proposed scheme can withstand this kind of attack.

According to the BER values tabulated in Table 1, the embedded information is successfully extracted after applying MJPEG compression attack. The watermark insertion in the DWT mid frequency sub bands, which allows to avoid information loss during compression process, explains this high robustness. The last non-malicious considered attack is the brightness and contrast adjustment. Again, the BER values obtained by varying the brightness as well as the contrast ratios prove the robustness of our watermarking scheme since they are inferior to threshold T. This resilience is reached thanks to the use of MSER detector, which is immune to illuminance changes.

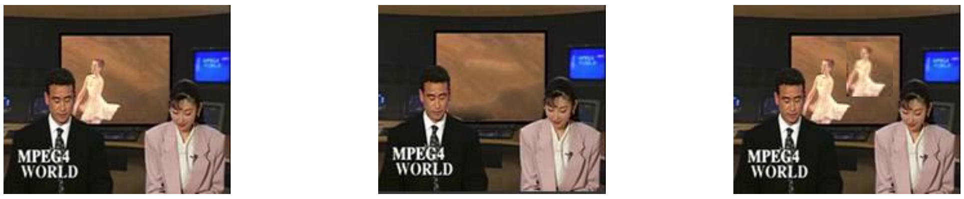
Similarly, the proposed scheme robustness is investigated under malicious attacks including (cropping, objects deletion and objects insertion attacks). For cropping attack, resulting BER values are displayed in Table 2. As can be seen from this table, The BER values, which are superior to the threshold T=0.1, demonstrate that the watermarked video is deliberately manipulated.

Concerning object manipulations attacks, we intentionally delete or add an object to the frames content of randomly chosen sequences as shown in Figure 6. BER values relative to object deletion and object insertion are respectively presented in Figure 7 and Figure 8. According to these figures, resulting BER values are significantly higher than the threshold 0.1. Therefore, the proposed scheme can effectively detect these two malicious attacks. This efficient ability is provided owing to the choice of the moving object as watermarking best location.

From all results, it can be concluded that our proposed scheme is suitable for authentication purpose where the discrimination between content preserving manipulations and content changing attacks is a prominent requirement.

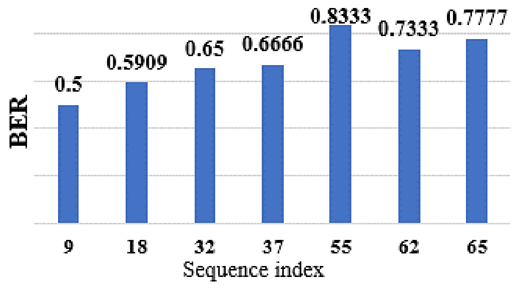
Video / Window size	Test	Camera2	Video1	Akiyo	News	Coastguard
[20*20]	0.3484	0.2656	0.3840	0.1066	0.2502	0.1203
[40*40]	0.3608	0.3626	0.3840	0.1133	0.1984	0.1240
[60*60]	0.3884	0.4019	0.3840	0.1305	0.2233	0.1223
[80*80]	0.3531	0.4159	0.3840	0.1590	0.1378	0.1245
[100*100]	0.2712	0.41104	0.3840	0.1814	0.2700	0.1325

Table 2. The BER values under cropping attack with different window sizes

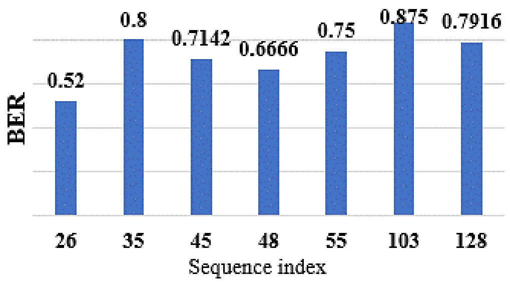


(a) (b) (c)

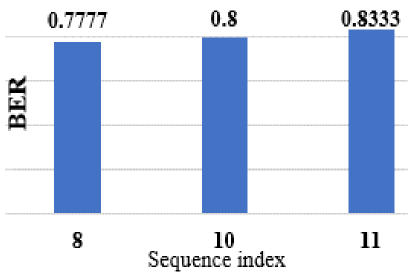
Figure 6. Sample of (a) original frame and its intentionally attacked versions by (b) object deletion and (c) object insertion.



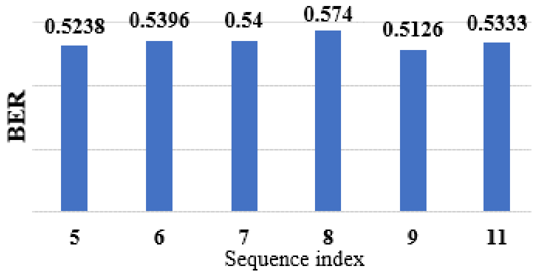
(a)



(b)

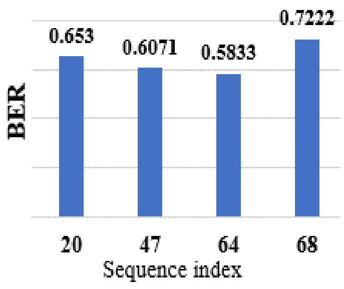


(c)

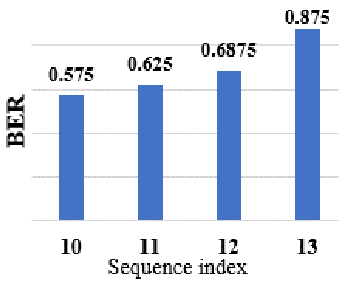


(d)

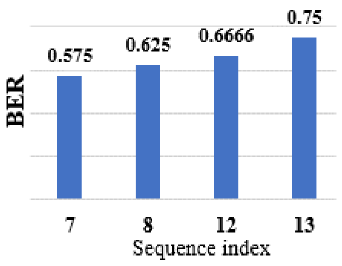
Figure 7. BER under object deletion attack for (a) test.avi, (b) camera2.avi, (c) video1.avi, (d) news.avi.



(a)



(b)



(c)

Figure 8. BER under object insertion attack for (a) test.avi, (b) akiyo.avi, (c) coastguard.avi.

3.3 Comparative Study

This section provides a comparison between the proposed watermarking scheme performance and the techniques proposed in Refs [Far16] and [Bha18] under different attacks. The technique presented in [Far16] operates using DCT by inserting the watermark into DCT coefficients. In [Bha18], a LWT based scheme is proposed where the watermark bits are concealed in the LH3 coefficients. Table 3 depicts BER values obtained after carrying out various attacks on the two watermarked videos versions resulting from the application of our approach and the technique presented in [Far16] to news.avi. Table 3 analysis reveals that the proposed system outperforms the technique in [Far16] in terms of robustness against all attacks listed in this table. The reason behind the proposed scheme robustness is the utilization of two domain transform techniques DWT and SVD instead of one unique transformation which allows to better strengthen the resilience to common manipulations. Furthermore, the proposed watermark construction process which involves two noises immune features detectors namely SURF and MSER yields a signature characterized by a high survival level against several attacks. Performances comparison outcomes with [Bha18] are displayed in Table 4. Based on BER values reported in this table, it can be inferred that our technique exhibits better robustness compared to the work of [Bha18]. Concerning the imperceptibility performances, our watermarking scheme exceeds the method in [Bha18] as indicate the PSNR values given in Table 4. The main reason for this higher perceptual quality is the suitable choice of the watermark holders. In fact, the exploitation of SVD characteristics and DWT sub-bands ones to conceal watermark bits into the frame areas where human eye is less sensitive to the change preserves better output watermarked video quality.

Attacks	Ref [Far16]	Proposed scheme
Gaussian noise(0.01)	0.0581	0.0053
Gaussian noise(0.02)	0.0598	0.0061
MJPEG compression	0.0244	0.0009
Brightness (+10%)	0.0466	0.0061
Brightness (+20%)	0.0628	0.0062

Table 3. Robustness comparison with Ref [Far16]

	News		Akiyo	
	Ref [Bha18] **PSNR**=41.5(dB)	Proposed scheme **PSNR**=50.42(dB)	Ref [Bha18] **PSNR**=41.2(dB)	Proposed scheme **PSNR**=56.66(dB)
Gaussian noise (0.01)	0.0791	0.0053	0.0742	0.0330
Salt & pepper	0.0303	0.0106	0.0322	0.0315
Median Filter (3x3)	0.0117	0.0058	0.0098	0.0268
Median Filter (5x5)	0.0371	0.0088	0.0371	0.0192
MJPEG compression	0.0009	0.0009	0.0009	0.0009

Table 4. Imperceptibility and robustness comparison with Ref [Bha18]

4. CONCLUSIONS AND FUTURE SCOPE

In this paper, a semi-fragile watermarking approach is proposed for video content authentication. It initially involves a content based authentication signature generation process using both of SURF and MSER detectors. After that, the watermark bits are hidden in the host video frames following a ROI-SVD-DWT based embedding method. The proposed technique ensures a high imperceptivity level. Moreover, the simulation results prove that the proposed semi-fragile scheme appropriately suits the content authentication goal. In fact, it successfully detects content changes attacks while tolerating incidental manipulations. In coming future works, we will extend the proposed approach to handle temporal attacks and forgery localization.

5. ACKNOWLEDGEMENTS

The research leading to these results received funding from the Tunisian Ministry of Higher Education and Scientific Research under the grant agreement number LR11ES48.

6. REFERENCES

[Abe13] A. Ben Hamida, M. Koubaa, H. Nicolas and C. Ben Amar, "Video pre-analyzing and coding in the context of video surveillance applications," In: IEEE International Conference on Multimedia and Expo Workshops, pp. 1-4, 2013.

[Abe14] A. Ben Hamida, M. Koubaa, C. Ben Amar and H. Nicolas, "Toward scalable application-oriented video surveillance systems," In: Science and Information Conference, pp. 384-388, 2014.

[Aga19] Agarwal, N., Singh, A.K. and Singh, P.K., "Survey of robust and imperceptible watermarking," Multimedia Tools and Applications, 78(07), pp. 8603-8633, 2019.

[Ali18] Ali I.H. and Salman S., "A performance analysis of various feature detectors and their descriptors for panorama image stitching," International Journal of Pure and Applied Mathematics, 119(15), pp. 147-161, 2018.

[Ara19] Arab F., Zamani M., Poger S., Manigault C. and Yu S., "A Framework to Evaluate the Performance of Video Watermarking Techniques," In: Proceeding of the International Conference on Information and Computer Technologies, pp. 114-117, 2019.

[Asi15] Asim N., Yasir S., Nisar A. and Aasia R., "Performance evaluation and watermark security assessment of digital watermarking techniques," Science International, 27(2), pp.1271-1276, 2015.

[Ben13] Ben Hamida A., Koubaa M., Nicolas H. and Ben Amar C, "Spatio-temporal video filtering for video surveillance applications," In: IEEE International Conference on Multimedia and Expo Workshops, pp. 1-6, 2013.

[Ben14] Ben Hamida A., Koubaa M. and Ben Amar C Nicolas H, "Parallelepipedic shape modeling for moving objects in video surveillance systems," In: Science and Information Conference, pp. 379-383, 2014.

[Bha18] Bhardwaj, A., Verma, V.S. and Jha, R.K., "Robust video watermarking using significant frame selection based on coefficient difference of lifting wavelet transform," Multimedia Tools and Applications, 77(15):19659-19678, 2018.

[Bpa14] B. P. Aditya, U. G. K. Avaneesh, K. Adithya, Akshay Murthy, R. Sandeep and B. Kavyashree, "Invisible semi fragile watermarking and steganography of digital videos for content authentication and data hiding," International Journal of Image and Graphics, 19(03),pp. 1950015-1-19, 2019.

[Cst19] C. Stauffer C and W.E.L Grimson, "Adaptive background mixture models for real-time tracking," In: Computer Society Conference on Computer Vision and Pattern Recognition, pp. 246-252, 1999.

[Far16] Farfoura M.E., Horng S.J. and Guo J.M., "Low complexity semi-fragile watermarking scheme for H.264/AVC authentication," Multimedia Tools and Applications, 75(13), pp. 7465–7493, 2016.

[Fzh19] F. Zhang and Y. Zou, "Reducible mapping class of the canonical Heegaard splitting in a mapping torus," Topology and its Applications, 258, pp. 425-432, 2019.

[Gup16] Gupta G., Gupta V.K and Chandra M., "Review on Video Watermarking Techniques in Spatial and Transform Domain," In: International Conference on Information Systems Design and Intelligent Applications, pp. 686-691, 2016.

[Ham19] Hammami A., Ben Hamida A. and Ben Amar C., "A Robust Blind Video Watermarking Scheme Based on Discrete Wavelet Transform and Singular Value Decomposition," In: International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Application, pp. 597-604, 2019.

[Hba08] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," Computer Vision and Image Understanding, 110(3), pp. 346-359, 2008.

[Him18] Himeur Y. and Boukabou A., "A robust and secure key-frame based video watermarking system using chaotic encryption," Multimedia Tools and Applications, 77(7), pp. 8603-8627, 2018.

[Jma04] J. Matas, O. Chum, M. Urban and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," Image and Vision Computing, 22(10), pp. 761-767, 2004.

[Jos17] Joshi A.M., Gupta S., Girdhar M., Agarwal P. and Sarker R., "Combined DWT–DCT-Based Video Watermarking Algorithm Using Arnold Transform Technique," In: Proceedings of the International Conference on Data Engineering and Communication Technology, pp. 455–463, 2017.

[Kad16] Kadu S., Naveen C., Satpute V. R. and Keskar A.G., "Discrete wavelet transform based video watermarking technique," In: International Conference International Conference on Microelectronics, Computing and Communications (MicroCom), pp. 1-6, 2016.

[Ker17] Kerbiche A., Ben Jabra S., Zagrouba E. and Charvillat V., "Robust Video Watermarking Approach Based on Crowdsourcing and Hybrid Insertion," In: International Conference on Digital Image Computing: Techniques and Applications, pp. 1–8, 2017.

[Kha15] Khalid Tahboub, Neeraj Gadgil, Javier Ribera, Blanca Delgado, and Edward J. Delp. "An intelligent crowdsourcing system for forensic analysis of surveillance video", In: Proceeding of the 27th IS&T/SPIE on Video Surveillance and Transportation Imaging Applications, pp. 94070-I-19, 2015.

[Man16] Manikandan V.M., Masilamani V., "Real-Time Scene Change Detection and Entropy Based Semi-Fragile Watermarking Scheme for Surveillance Video Authentication," In: International Symposium on Big Data and Cloud Computing Challenges, pp. 101-110, 2016.

[Mas17] M. Asikuzzaman and M. R. Pickering, "An overview of digital video watermarking," IEEE Transactions on Circuits and Systems for Video Technology, 28(9), pp. 2131-2153, 2017.

[Mch14] M. Charfeddine, M. El'Arbi and C. Ben Amar, "A new DCT audio watermarking scheme based on preliminary MP3 study," Multimedia Tools and Applications, 70(3), pp. 1521-1557, 2014.

[Mko12] M. Koubaa, M. Elarbi, C. Ben Amar and H. Nicolas, "Collusion, MPEG4 compression and frame dropping resistant video watermarking," Multimedia Tools and Applications, 56(2), pp. 281-301, 2012.

[Nta18] N. Tarhouni, M. Charfeddine and C. Ben Amar, "Discrete wavelet transform based video watermarking technique," In: International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 78-86, 2018.

[Tbo12] T. Bouchrika, M. Zaied, O. Jemai and C. Ben Amar, "Ordering computers by hand gestures recognition based on wavelet networks," In: 2nd International Conference on Communications Computing and Control Applications, pp. 1–6, 2012.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.28, No.1-2, 2020

Improving RGB Image Consistency for Depth-Camera based Reconstruction through Image Warping

Fernando Reyes-Aviles, Philipp Fleck, Dieter Schmalstieg, Clemens Arth
Institute for Computer Graphics and Vision (ICG)
Graz University of Technology
Inffeldgasse 16/2, 8010 Graz
[fernando.reyes-aviles, philipp.fleck, schmalstieg, arth]@icg.tugraz.at

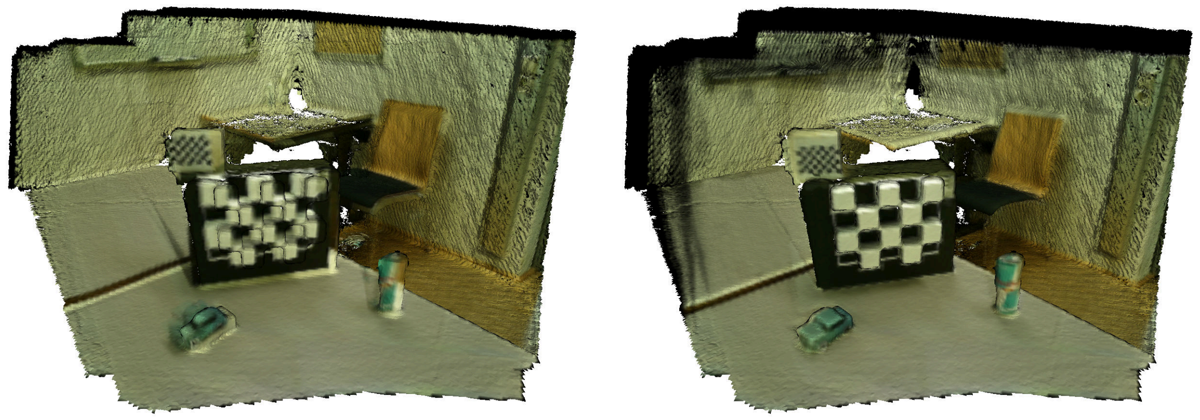


Figure 1: A partial room reconstruction using InfiniTAMv3. Left: result without applying a warping function. Right: result using our warping method.

ABSTRACT

Indoor reconstruction using depth camera algorithms (*e.g.,* InfiniTAMv3) is becoming increasingly popular. Simple reconstruction methods solely use the frames of the depth camera, leaving any imagery from the adjunct RGB camera untouched. Recent approaches also incorporate color camera information to improve consistency. However, the results heavily depend on the accuracy of the rig calibration, which can strongly vary in quality. Unfortunately, any errors in the rig calibration result in apparent visual discrepancies when it comes to colorization of the 3D reconstruction. We propose an easy approach to fix this issue for the purpose of image-based rendering. We show that a relatively simple warping function can be calculated from a 3D checkerboard pattern for a rig with poor calibration between cameras. The warping is applied to the RGB images online during reconstruction, leading to a significantly improved visual result.

Keywords

Depth camera, image warping, calibration, image-based rendering.

1 INTRODUCTION

Indoor reconstruction algorithms for depth cameras, like InfiniTAMv3, are becoming increasingly popular. Such an algorithm performs camera pose estimation based on the depth maps and feature extraction from the color images to build 3D models from image observations. In order to provide textured or colored 3D reconstructions, a large amount of approaches were developed adopting RGB-D cameras as input devices.

One of the practical issues encountered in using such approaches is the need for accurate calibration of the camera rig. The pose information from the color camera and the depth camera have to be correct in order to enable proper color and texture projection onto the 3D point cloud or mesh. RGB-D devices are often calibrated in the factory. The calibration parameters are stored inside a nonvolatile memory [BMP18]. The depth-to-color registration process uses the intrinsic and extrinsic parameters stored in the nonvolatile memory, but the parameters themselves are inaccessible due to the closed nature of the sensor design.

Only a fraction of the large variety of depth camera types is shipped with an accurate internal calibration (*e.g.,* Microsoft Kinect v2). Some products provide inaccurate calibrations; custom rigs always require manual calibration. In any case, a noticeable misalignment between the depth maps and the color images makes a full calibration of the RGB-D sensor necessary. A survey of calibration literature suggests that creating a proper calibration for RGB-D rigs is surprisingly difficult because of the heterogeneous nature of the involves cameras[1].

Projection-based depth cameras use a projector casting infrared (IR) patterns observed by an IR cameras (or at least a camera fitted with IR filters). The depth maps accessible through the device driver or middleware like OpenNI[2] are generated using closed algorithms, which use a nonlinear, depth-dependent mapping that cannot be explained with a rigid transformation between the two sensors. One could compare the raw IR image delivered by the depth camera to the RGB image [BF15]. However, this assumes that the IR image and the depth image use the same projection, which is not the case for the sensors we have examined. Indeed, there are pairwise nonlinear mappings between color image, IR image and depth image.

This work was motivated by a problem encountered during the scanning of real environments for a Virtual Reality application. When reconstructing model geometry with InfiniTAMv3 [KPR$^+$15] for an image-based rendering (IBR) system [EHH$^+$19], we noted a significant misalignment between depth and color information (Figure 1, left).

Diving deeper into this issue (see Figure 2) makes the discrepancies more apparent. The color-depth offset is non-linearly decreasing with the distance from the camera center and increasing with distance from the optical axis of the sensor (see Figure 3). Attempts to calibrate the RGB-D sensor with standard camera calibration technique using a checkerboard pattern failed to produce usable results. We also tested the depth-sensor calibration algorithm by Ferstl *et al.* [FRR$^+$15], but neither of those calibration methods led to a reduction of the depth-color offset.

In this work, we devise a method to calibrate RGB-D sensors using a polynomial warping function. The warping function registers the RGB images to the depth

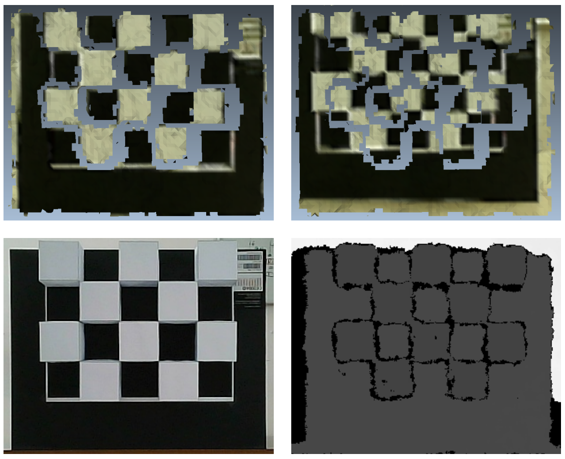


Figure 2: Visualizing depth discontinuities for Orbbec Astra Pro sensor. First row: Left, example of expected result. Right, the actual result. Second row: a corresponding pair of color image and depth map.

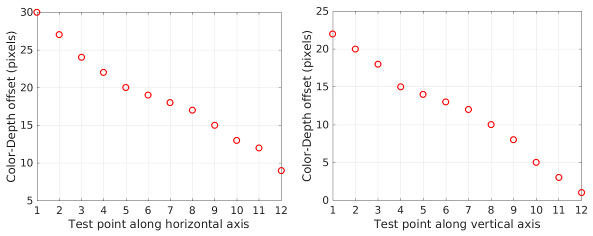


Figure 3: Non-linear offset between depth maps and color images, varying over the distance from the optical axis. These offsets were obtained by moving a known object in a static scene and measuring the depth to color offset. We moved it along the horizontal (left plot) and vertical (right plot) axis of the image plane.

maps, which produces a 1:1 correspondence between pixels in the depth map and the color image. We use a 3D equivalent of a conventional checkerboard calibration pattern, which has its white squares raised and turned into small cubes. The resulting 3D target can be reliably detected with a depth sensor. After computing the translational offset between corresponding corners in the RGB and depth images, we fit an *n*th degree polynomial to the resulting differences and use it to compute the warped image $RGB'(x,y) = RGB(x_w, y_w)$, with (x,y) being pixel coordinates and (x_w, y_w) denoting the coordinates retrieved from the warping function. Thus, the primary contributions of this paper can be summarized as:

1. The implementation of a calibration algorithm to retrieve translational offsets
2. An image warping registration algorithm which achieves 1:1 correspondence of color to depth
3. The demonstration of the plausibility of the calibration approach on the popular Orbbec sensors

[1] For the rest of the discussion, we focus on projector-based depth cameras only.
[2] OpenNI Github Repository

We apply the proposed algorithm to an IBR algorithm [EHH$^+$19] and provide some experimental results (see *e.g.,* right of Figure 1).

2 RELATED WORK

The problem of proper calibration of RGB-D cameras has been mentioned multiple times in the literature. The factory calibration of commercial RGB-D sensors, such as Kinect and Structure Sensors, is not suitable for applications requiring high quality 3D data, *i.e.*, 3D building models of centimeter-level accuracy [DTLC17]. Thus, several studies have extensively analyzed the sources of errors of such sensors [GVS18, SPB04, KBR14, HHEM16].

Simultaneous localization and mapping (SLAM) using depth sensors has become immensely popular after the publication of the KinectFusion [NIH$^+$11] algorithm. Later work improved or extended the original capabilities [KPR$^+$15, WSF18]. However, these methods generally assume a properly calibrated RGB-D rig.

Other work uses IBR to create high-quality results despite poor alignment of color images, essentially correcting the problem after acquisition is completed. This usually involves generating new views from recorded video frames [HRDB16, DH18, EHH$^+$19]. The rig calibration problem was encountered in these works, but, again, no direct solutions were proposed. For the rest of the discussion, we thereby focus on the calibration problem itself.

2.1 RGB to IR camera calibration

Basic camera calibration is usually done through well-known algorithms, such as Heikkila and Silven's calibration method [HS97], Zhang's algorithm [Zha00], Bouguet's Matlab camera calibration toolbox [Bou01], or more recent work, such as Rojtberg and Kuijper's method [RK17].

Chen *et al.* [CYS$^+$18] and Darwish *et al.* [DLT$^+$19] calibrate RGB-D sensors with a flat checkerboard pattern via images from the RGB and IR cameras. The former calibrates multiple RGB-D sensors into a single coordinate system, while the latter improves the depth measurement accuracy.

Geiger *et al.* [GMCS12] developed a method for fully automatic camera-to-camera calibration that uses images of planar checkerboard patterns as calibration targets. They detect checkerboard corners in an image by computing a corner likelihood at each pixel in the image using corner prototypes. To produce a list of corner candidates, they apply non-maxima suppression, followed by a scoring based on gradient statistics that they threshold to obtain a final list of corners.

2.2 RGB to depth map calibration

Like most approaches, Zhang and Zhang [ZZ11] use a checkerboard pattern, but they do not calibrate the depth camera through the detection of checkerboard points in the images from the IR camera. They propose a maximum-likelihood method based on the fact that points on the checkerboard in the depth map shall be co-planar, and the plane is known from color camera calibration. They use point correspondences between the depth and color images that may be manually specified or automatically established.

Herrera C. *et al.* [HKH12] present an algorithm that simultaneously calibrates the intrinsics and extrinsics of two color cameras and a depth camera. They use a planar checkerboard pattern for calibration. In the color images, the checkerboard corners are extracted, and, in the depth maps, the four corners of the checkerboard plane are located. For the depth camera calibration, the user selects the corners of the calibration plane in the depth maps.

Jin *et al.* [JLG14] proposed an intrinsic calibration of depth sensors in which they use a set of cuboids with known sizes. Their approach consists of an iterative plane fitting and non-linear optimization that takes around five minutes to accomplish the calibration. The objective function for calibration is based on the length, width, and height of cuboids and its angle between the neighboring surfaces.

Staranowicz *et al.* [SBMM15] developed an RGB-D camera calibration algorithm for the estimation of intrinsic and extrinsic parameters. They use a recorded sequence of a moving sphere to calibrate the sensor. Ellipse and sphere-fitting algorithms are used to detect the sphere in the RGB images, and the center of the fitted sphere is reprojected onto the depth map. A great advantage is that no a-priori knowledge of the sphere's size is required.

Basso *et al.* [BMP18] developed a method to calibrate RGB-D sensors by estimating the rigid body transformation that relates the two sensors, while inferring the depth error correction function. Their method requires the depth sensor to be coupled with a calibrated RGB camera that frames approximately the same scene. First, they use the pre-calibrated RGB camera to detect a checkerboard on a wall. Then, they infer the location of the latter based on the pose of the checkerboard in the RGB images, and finally they estimate the rigid body transformation that relates the two sensors.

2.3 Learning-based approaches

Ferstl *et al.* [FRR$^+$15] use a random regression forest to optimize the manufacturer supplied depth measurements. They detect feature points in both the RGB and the depth maps using a planar target with known dimensions.

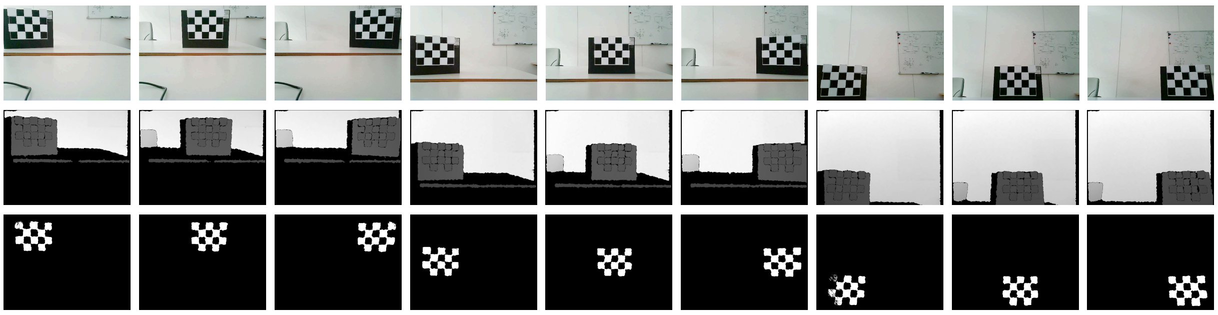


Figure 4: Sample images. RGB images I_C (top). Depth maps I_D (middle). Binary images I_T (bottom).

Teichman *et al.* [TMT13] propose an unsupervised intrinsic calibration approach built on top of a robust SLAM system. The RGB-D sensors they used to test their system gave them correct extrinsic parameters, *i.e.*, a correct depth-to-color registration. The main advantage is that no specialized hardware, calibration target, or hand measurement is required. However, their optimization approach takes several hours to converge.

2.4 Discussion

Calculating a correct differential pose of the depth map and the RGB camera is not possible directly. At least, a calculated geometric pose correction only works for a dedicated physical distance examined, but does not hold throughout the entire range of distances observed in a practical scenario.

Our approach aims at creating suitable imagery for IBR. We do not aim for improving the reconstruction directly through the use of image observations. However, while our algorithm has similarities to existing approaches, it is much more straight-forward to apply, only requires a small amount of images and works exclusively in the image domain.

3 CALIBRATION ALGORITHM

In order to describe our calibration algorithm, we first introduce some basic notation here. An RGB camera provides an RGB image denoted by I_C; a depth sensor provides a depth image denoted by I_D. From an RGB image of a scene that contains a checkerboard, it is possible to extract the checkerboard corners ${}^{I_C}B$, where the superscript I_C denotes the fact that the corners are expressed in two-dimensional pixel coordinates, *i.e.*, ${}^{I_C}B = (x,y)_1, ..., (x,y)_n$.

We do not use images from the IR sensor to calibrate the RGB-D camera for the reasons explained earlier. Instead, we directly go for a method to relate RGB images and the corresponding depth maps. For our algorithm, we require a 3D checkerboard target, which can be detected in both the RGB image and the depth map. Bamji *et al.* [BOE+15] reported a depth resolution of $5cm$ at $7.5m$ and a precision of $2.7cm$ over a range of $0.2-6m$ for commercial RGB-D sensors. Thus, we chose a minimum cube size of $125cm^3$ as features to detect in the depth image.

We record color and depth images of the 3D checkerboard in different positions (see Figure 4). We need to automatically detect as many points as possible to obtain dense 3D-to-3D correspondences. This requirement deviates from the method of Zhang and Zhang [ZZ11], who only need a few points because they estimate a rigid transformation with only six degrees of freedom. In contrast, we estimate a warping function to achieve a 1:1 correspondence of color and depth.

From the imagery, we extract the checkerboard points in all the k color-depth image pairs, with $k = 1,2,...,K$. Once we have the coordinates of the checkerboard points ${}^{I_C}B$ and ${}^{I_D}B$ in both color and depth images, we calculate the offsets (in pixels). Two warping functions $S^{\mathbf{x}}$ and $S^{\mathbf{y}}$ are estimated from these measurements, which warp the image in **x** and **y**, respectively.

3.1 Corner detection

We use the algorithm of Geiger *et al.* [GMCS12] to detect the checkerboard corner points in both color and depth. Corner detection in RGB images is straightforward, but depth frames need pre-processing before checkerboard detection works.

First, we estimate surface normals from the raw depth image using the surface normal determination algorithm of Ückermann *et al.* [UEHR12]. This procedure results in edges becoming more distinct. The method uses the 2D image coordinates and depth value to yield valid three-dimensional vectors for every image point. The surface normals are calculated from the plane spanned by three image points in a vicinity of 5×5 pixels. Using the cross product

$$\vec{n} = (\vec{b} - \vec{a}) \times (\vec{c} - \vec{a}) \tag{1}$$

and the normal to the image plane $\vec{n}_{xy}$, we obtain the angle $\angle(\vec{n}_i, \vec{n}_{xy})$ which gives us the degree of deviation of the $\vec{n}_i$ normals from the optical center of the RGB-D sensor. We heuristically determined a threshold value of $\theta_{max} = 1^\circ$ to remove the edge pixels in the raw depth map.

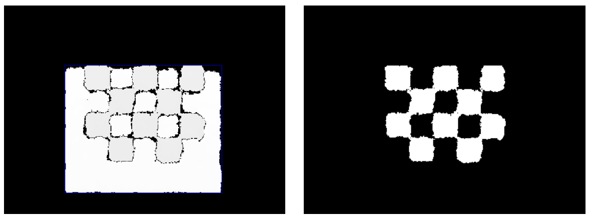


Figure 5: Left: 3D checkerboard target blob. Right: Binary image I_T.

We use the output of the edge detection step to detect sets of connected components. As with most checkerboard detection algorithms, we require to board to be non-square, *i.e.*, one board dimension should be even, the other, odd. Consequently, blobs should be nearly rectangular in screen space, and we can reject outliers based on their aspect ratio.

From this rejection test, we obtain a bounding box E that delimits the blob containing the 3D checkerboard target. We use its boundary to remove all the pixels in the depth map that lie outside of E and get an image that only contains the 3D target (see left of Figure 5).

We threshold the depth map containing only the bounding box E (see left of Figure 5) to obtain the binary images I_T (see right of Figure 5 and bottom of Figure 4). To do so, we determine a plane $P(x,y)$ with three non-colinear points (see Figure 6) which allows us to remove the board's background (checkerboard black squares).

The 3D checkerboard position cannot be parallel to the image plane because, when it is moved too far away from the optical axis, the side faces of the cubes interfere with the checkerboard detection algorithm [GMCS12]. Thus, the board has to be slightly rotated when moving away from the center. Hence, we cannot use a mask which is parallel to the image plane.

Given the center E_C (red point in Figure 6) of the bounding box E, the plane points are determined as

$$\begin{aligned} P_1 &= \left(E_C^x - d,\quad E_C^y - d,\quad G_1^{min}\right) \\ P_2 &= \left(E_C^x,\quad E_C^y + d,\quad G_2^{min}\right) \\ P_3 &= \left(E_C^x + d,\quad E_C^y,\quad G_3^{min}\right) \\ d &= E_H/5, \quad e = E_H/5 \\ G_i^{min} &= min\{D_j > 0\} + \theta_T, \end{aligned} \tag{2}$$

with E_H being the height of the bounding box E, and D_j, all the depth values inside the green square G_i in Figure 6. The threshold θ_T lets us decide the depth values the plane $P(x,y)$ masks out. We experimentally determined a threshold $\theta_T = 4cm$, since we know that the height of the cube is $5cm$ (given a cube size of $125cm^3$).

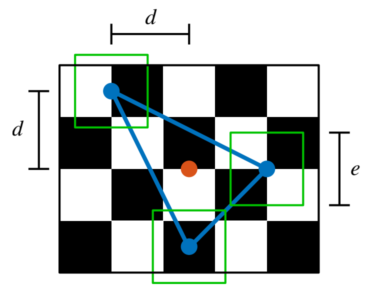


Figure 6: Points (blue) defined around the bounding box center (red) to determine the plane used as a filter to remove the board's background (checkerboard black squares).

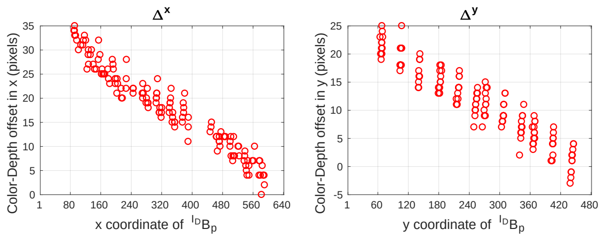


Figure 7: Sorted lists of points $\Delta_s^{x,y}$ (non-linear offset between ${}^{I_C}B$ and ${}^{I_D}B$ checkerboard corner points) obtained from the sample data set of Figure 4.

We analyze the depth value of all the pixels inside the bounding box E and binarize them to obtain I_T. Pixels with a depth value greater than $P(x,y)$ are discarded.

$$I_T(x,y) = \begin{cases} 0 & \text{if } E(x,y) \geq P(x,y) \\ 1 & \text{if } E(x,y) < P(x,y) \end{cases} \tag{3}$$

After this, we obtain binary images containing only the checkerboard (see right of Figure 5 and bottom of Figure 4) that we use as input for the detection algorithm [GMCS12].

3.2 Color-to-depth offset

From a board with $M \times N$ corner points, we obtain a set of p points, with $p = 1,2,...,M \cdot N$. Once we have the coordinates of the checkerboard points ${}^{I_C}B$ and ${}^{I_D}B$ in both color and depth images, we calculate the offset (in pixels) between each corner point ${}^{I_C}B_p$ in the I_{Ck} image and ${}^{I_D}B_p$ in the I_{Dk} image.

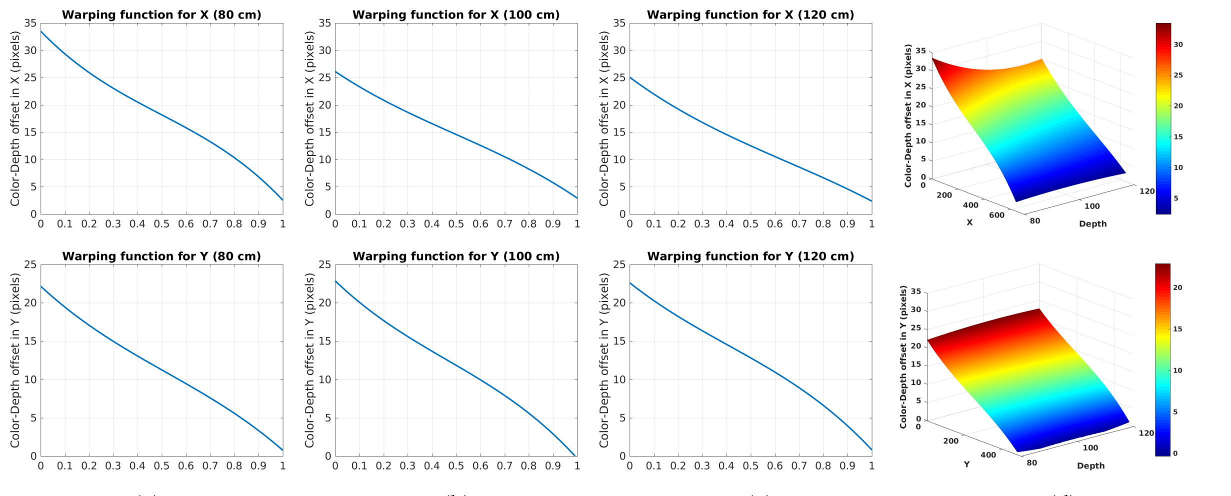
$$\Delta_p^x = {}^{I_D}B_p^x - {}^{I_C}B_p^x \quad \text{with} \quad x = 1,2,...,n \tag{4}$$

$$\Delta_p^y = {}^{I_D}B_p^y - {}^{I_C}B_p^y \quad \text{with} \quad y = 1,2,...,m \tag{5}$$

We sort the points $\Delta_p^{x,y}$ according to the coordinates of its associated point ${}^{I_D}B_p$ (see Figure 7).

3.3 Image warping

From the sorted lists of points $\Delta_s^{x,y}$, we estimate the warping functions $W^{\mathbf{x}}$ and $W^{\mathbf{y}}$. We fit cubic polynomials to Δ_s^x and Δ_s^y to obtain the warping functions $W^{\mathbf{x}}$



(a) (b) (c) (d)

Figure 8: Warping functions $W^{\mathbf{x}}$ and $W^{\mathbf{y}}$ for different distances for the Orbbec Astra Pro sensor. (a): 80cm calibration distance. (b): 100cm calibration distance. (c): 120cm calibration distance. (d): Interpolated warping functions $S^{\mathbf{x}}$ (top) and $S^{\mathbf{y}}$ (bottom).

and $W^{\mathbf{y}}$, respectively (after the investigation of multiple sensors and setups, a cubic polynomial turned out to be sufficient). We solve the polynomial regression problem using the least squares method

$$\mathbf{A^TAc} = \mathbf{A^Tb}, \tag{6}$$

For $W^{\mathbf{x}}$, the vector $\mathbf{b} = \Delta_s^x$ and the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & {}^{I_D}B_1^{x^1} & \dots & {}^{I_D}B_1^{x^n} \\ 1 & {}^{I_D}B_2^{x^1} & \dots & {}^{I_D}B_2^{x^n} \\ \vdots & & \ddots & \vdots \\ 1 & {}^{I_D}B_s^{x^1} & \dots & {}^{I_D}B_s^{x^n} \end{bmatrix} \tag{7}$$

The functions $W^{\mathbf{x}}$ and $W^{\mathbf{y}}$ can only be correctly approximated, when the 3D target in all the I_C and I_D images is at the same distance d from the image plane of the RGB-D sensor. Hence, it is required to estimate warping functions $W_d^{\mathbf{x}}$ and $W_d^{\mathbf{y}}$ for multiple distances d, respectively.

We interpolate all the available warping functions $W_d^{\mathbf{x}}$ and $W_d^{\mathbf{y}}$ to model the warping of $\mathbf{x}$ and $\mathbf{y}$ as a function of (x,D) and (y,D), respectively. We use cubic spline interpolation to obtain the functions $S^{\mathbf{x}}$ and $S^{\mathbf{y}}$ which warp all (x,y) pixels over all the depth distances D.

The function $S^{\mathbf{x,y}}$ is used to compute the warped image $I'_C(x,y) = I_C(x_w,y_w)$, such that

$$x_w = x + S^{\mathbf{x}}(x,D) \tag{8}$$
$$y_w = y + S^{\mathbf{y}}(y,D) \tag{9}$$

where D is the depth distance value at the pixel (x,y) of the I_D associated to I_C.

4 EXPERIMENTS

We conducted a set of experiments with different RGB-D cameras (Orbbec Astra Pro, Orbbec Astra Embedded S, *etc.*). An exhaustive overview about individual products is beyond the scope of this paper. However, in the following we discuss our findings with respect to the Orbbec Astra Pro sensor. Applying the algorithm to others led to results showing the same trends, however.

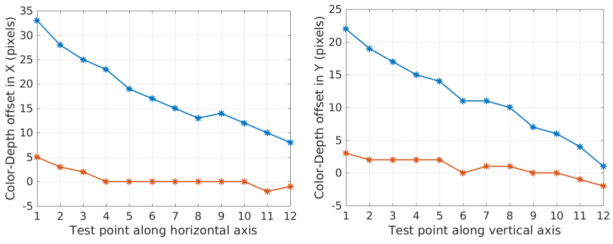


Figure 9: Offset error between depth and color images. Blue points represent the offset error, for 12 different points along the horizontal and vertical axes, before the calibration. Red points represent the error for the same 12 points after the calibration.

4.1 Calibration results

In Figure 8, the calculated warping functions for different target distances, 80cm, 100cm and 120cm, are depicted. With increasing distance from the sensor, the warping is observed to become slightly more linear. In Figure 9, the offset between ${}^{I_C}B$ and ${}^{I_D}B$ before the calibration (blue), and the offset after using the warping function on the RGB images (red). The error is significantly reduced in both the x and y dimension.

At the top of Figure 10, the offset for a test data set of 12 images is depicted. We took twelve images in different positions. The red circles are the ${}^{I_C}B$ board corners in RGB. The blue stars are the ${}^{I_D}B$ board corners in depth. The figure at the top shows the offset before applying

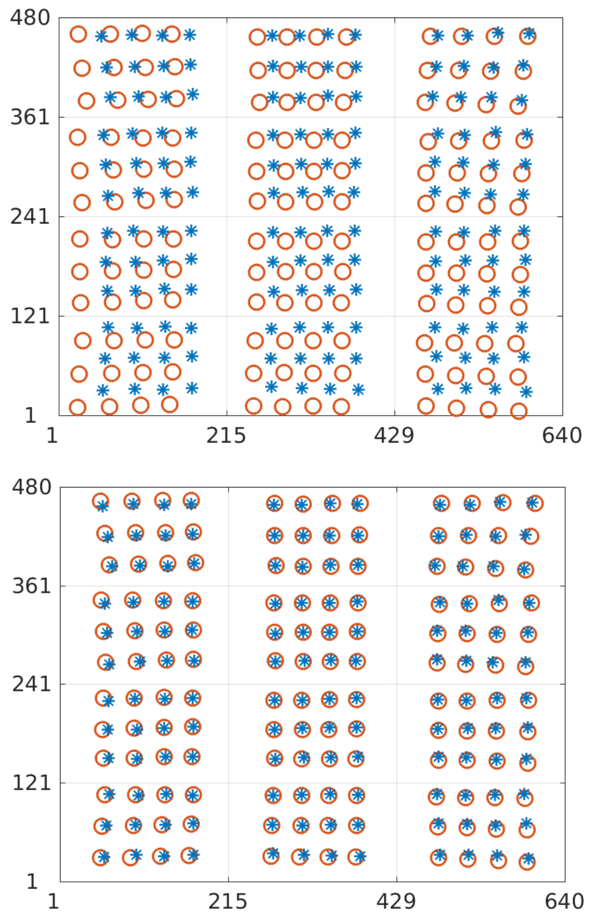


Figure 10: Scatter plot of the depth-to-color checkerboard corners offset for two data sets of 12 images. Red circles represent the ${}^{I_C}B$ corners, and blue stars, the ${}^{I_D}B$ ones. Top: offset before calibration. Bottom: offset after calibration.

the $S^{\mathbf{x}}$ and $S^{\mathbf{y}}$ functions. Similarly, for a second testset, the result is shown at the bottom of Figure 10.

4.2 Colored point clouds

The warping can be done online or offline. The $S^{\mathbf{x},\mathbf{y}}$ functions are used to create a lookup table that can be consulted at runtime in order to apply the correction. We added this functionality to InfiniTAMv3 without any noticeable performance drop. We compute $I'_C(x,y) = I_C(x_w, y_w)$ for every input frame I_C. An example of a colorized point cloud is showed in Figure 1, without and leveraging our warping method. Another example is shown in the top row of Figure 2, denoting the result after (expected) and before (original) warping.

Another example is shown in Figure 11. The borders of the tables in the scene prominently stick out, as well as the partial erroneous texturing of parts of the mobile computer, the mouse next to it, and the break of the lamp in the back. Using the warping method, the overall visual quality is significantly improved.

4.3 Image-based rendering

Erat *et al.* proposed an algorithm for generating an unstructured lumigraph in real-time from an image stream. Despite calculating the texture beforehand, the texture mapping onto a 3D model is performed online based on the actual view point onto the scene. For a more explicit description of the approach, the interested reader is referred to [EHH$^+$19].

The overall quality of the approach is heavily dependent on the accuracy of the RGB poses given to the algorithm. Passing the original RGB images from the sensor to the system results in clear visual artefacts and inaccuracies, as depicted on the left of Figure 12, while the results of our warping-based approach for the same scenes are depicted on the right.

For the scene depicted in Figure 12, the most obvious improvements can be observed at the 3D checkerboard pattern itself. Another visible inaccuracy is present at the left boundary of the monitor, which is completely cropped in the original approach, but nicely preserved in our new method.

5 CONCLUSION

In this paper, we presented a method to calibrate RGB-D sensors using a polynomial warping function. The warping function registers the color images to the depth maps producing a 1:1 correspondence between pixels in the former and the latter. This method is particularly useful when the camera extrinsics determination, through both color and depth intrinsics, does not give a satisfactory result.

We presented a set of experiments using the popular Orbbec Astra Pro RGB-D sensor along reconstruction and IBR systems, showing the quality improvement obtained after performing the warping.

The main drawback of our approach is the need to take multiple images of the same 3D target, at different distances from the sensor, to obtain a dense enough 3D warping function. The more 2D warping functions it contains the better the results the warping produces, though, a substantial improvement is noticeable with only a few warping entries.

The main advantage of the proposed method is that a depth-to-color correspondence between unrelated (or factory attached) RGB cameras and depth sensors could be achieved. For example, it could be used to build a camera rig made up of multiple depth sensors and a high definition color camera. This is subject to further research and development in the future.

6 ACKNOWLEDGMENTS

The authors wish to thank Okan Erat. This work was supported by FFG grant 859208.

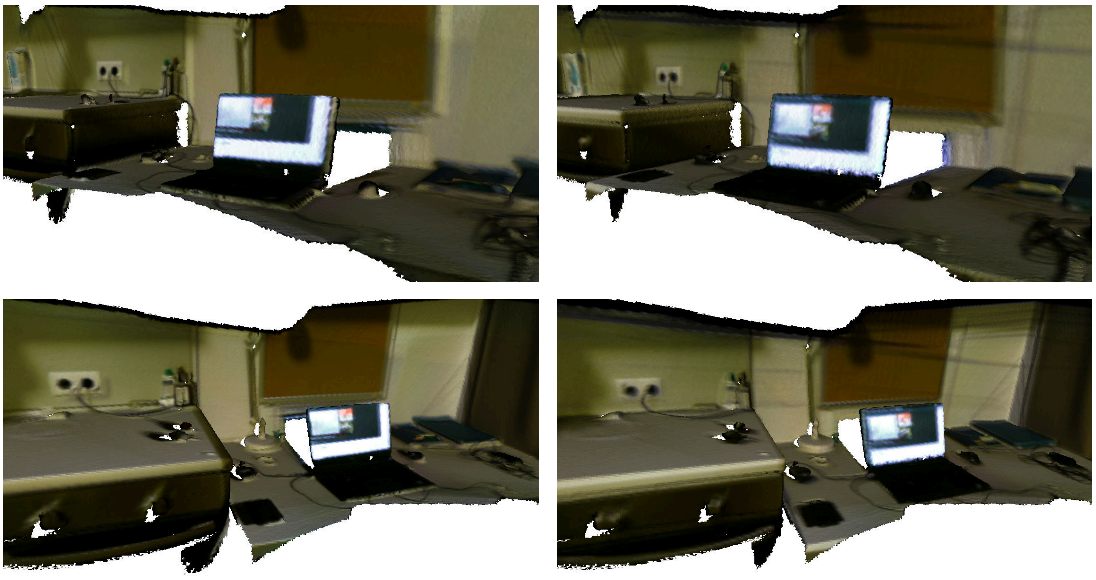


Figure 11: Results from a reconstruction recorded with InfiniTAMv3. Left: original results. Right: results generated with the proper warping functions.

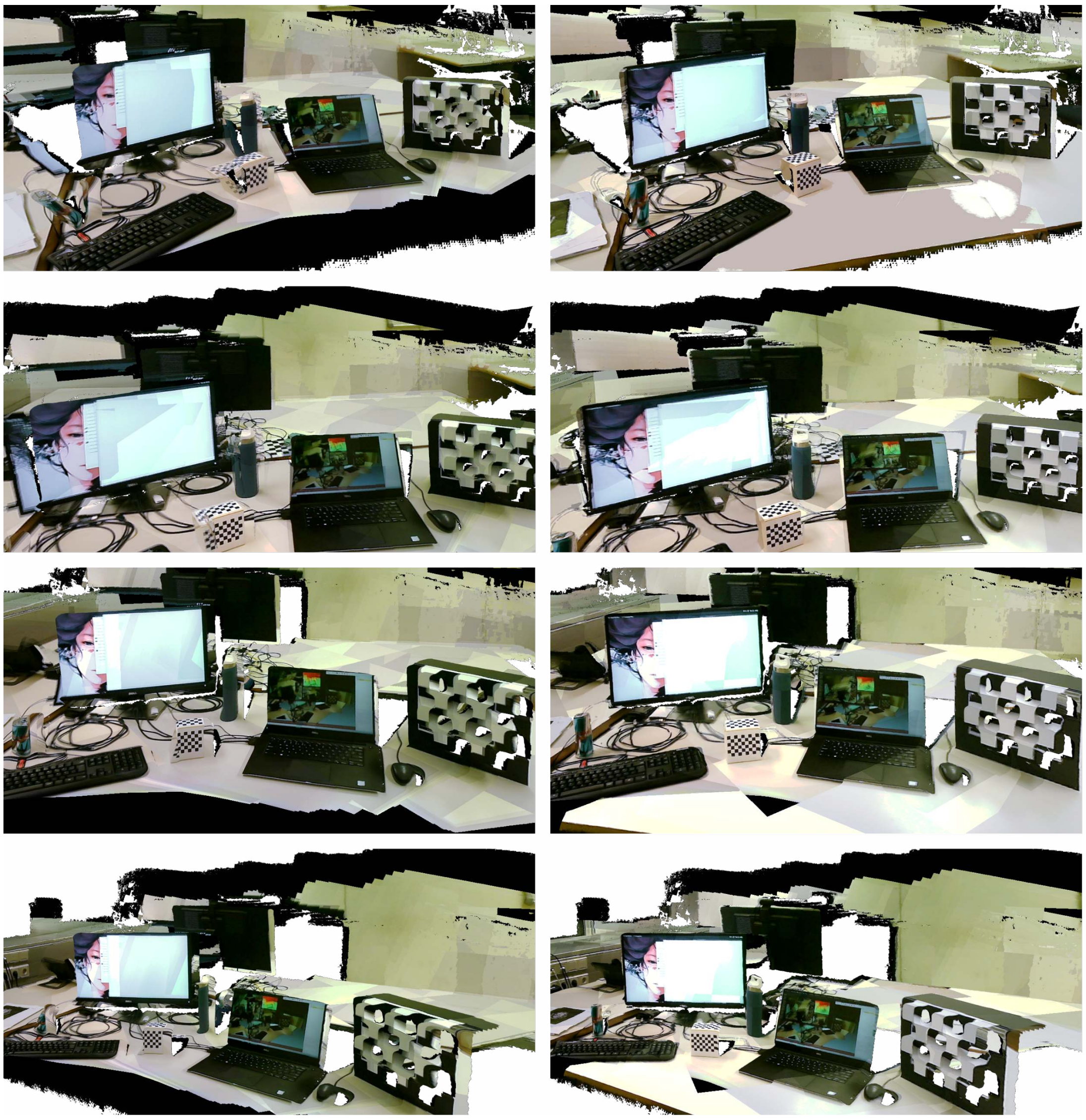


Figure 12: Results from the IBR pipeline of [EHH+19]. Left: original results. Right: results generated with the proper warping functions.

7 REFERENCES

[BF15] S. Beck and B. Froehlich. Volumetric calibration and registration of multiple rgbd-sensors into a joint coordinate system. In *3DUI*, pages 89–96, March 2015.

[BMP18] F. Basso, E. Menegatti, and A. Pretto. Robust intrinsic and extrinsic calibration of rgb-d cameras. *IEEE Trans. on Robotics*, 34(5):1315–1332, Oct 2018.

[BOE$^+$15] C. S. Bamji, P. O'Connor, T. Elkhatib, S. Mehta, B. Thompson, L. A. Prather, D. Snow, O. C. Akkaya, A. Daniel, A. D. Payne, T. Perry, M. Fenton, and V. Chan. A 0.13 μm cmos system-on-chip for a 512 x 424 time-of-flight image sensor with multi-frequency photo-demodulation up to 130 mhz and 2 gs/s adc. *IEEE Journal of Solid-State Circuits*, 50(1):303–319, Jan 2015.

[Bou01] J. Y. Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, 2001.

[CYS$^+$18] C. Chen, B. Yang, S. Song, M. Tian, J. Li, W. Dai, and L. Fang. Calibrate multiple consumer rgb-d cameras for low-cost and efficient 3d indoor mapping. *Remote Sensing*, 10(2), 2018.

[DH18] S. Dong and T. Höllerer. Real-time re-textured geometry modeling using microsoft hololens. In *VR*, pages 231–237, 2018.

[DLT$^+$19] W. Darwish, W. Li, S. Tang, B. Wu, and W. Chen. A robust calibration method for consumer grade rgb-d sensors for precise indoor reconstruction. *IEEE Access*, 7:8824–8833, 2019.

[DTLC17] W. Darwish, S. Tang, W. Li, and W. Chen. A new calibration method for commercial rgb-d sensors. *Sensors*, 17(6), 2017.

[EHH$^+$19] O. Erat, M. Hoell, K. Haubenwallner, C. Pirchheim, and D. Schmalstieg. Real-time view planning for unstructured lumigraph modeling. *TVCG*, 25(11):3063–3072, Nov 2019.

[FRR$^+$15] D. Ferstl, C. Reinbacher, G. Riegler, M. Rüther, and H. Bischof. Learning depth calibration of time-of-flight cameras. In *BMVC*, pages 102.1–102.12. BMVA Press, September 2015.

[GMCS12] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster. Automatic camera and range sensor calibration using a single shot. In *ICRA*, pages 3936–3943, May 2012.

[GVS18] S. Giancola, M. Valenti, and R. Sala. *A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopy Technologies*. SpringerBriefs in Computer Science. Springer Int. Pub., 2018.

[HHEM16] R. Horaud, M. Hansard, G. Evangelidis, and C. Ménier. An overview of depth cameras and range scanners based on time-of-flight technologies. *Machine Vision and Applications*, 27(7):1005–1020, Oct 2016.

[HKH12] D. Herrera C., J. Kannala, and J. Heikkila. Joint depth and color camera calibration with distortion correction. *PAMI*, 34(10):2058–2064, 2012.

[HRDB16] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow. Scalable Inside-Out Image-Based Rendering. *ACM Trans. Graph.*, 35(6):231:1–231:11, 2016.

[HS97] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *CVPR*, pages 1106–1112, June 1997.

[JLG14] B. Jin, H. Lei, and W. Geng. Accurate intrinsic calibration of depth camera with cuboids. In *ECCV*, pages 788–803, Cham, 2014. Springer International Publishing.

[KBR14] A. Kadambi, A. Bhandari, and R. Raskar. *3D Depth Cameras in Vision: Benefits and Limitations of the Hardware*, pages 3–26. Springer International Publishing, Cham, 2014.

[KPR$^+$15] O. Kähler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *TVCG*, 21(11):1241–1250, 2015.

[NIH$^+$11] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011.

[RK17] P. Rojtberg and A. Kuijper. [poster] efficient pose selection for interactive camera calibration. In *ISMAR (adj. material)*, pages 182–183, 2017.

[SBMM15] A. N. Staranowicz, G. R. Brown, F. Morbidi, and G. Mariottini. Practical and accurate calibration of rgb-d cameras using spheres. *CVIU*, 137:102 – 114, 2015.

[SPB04] J. Salvi, J. Pages, and J. Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827 – 849, 2004. Agent Based Computer Vision.

[TMT13] A. Teichman, S. Miller, and S. Thrun. Unsupervised intrinsic calibration of depth sensors via slam. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[UEHR12] A. Uckermann, C. Elbrechter, R. Haschke, and H. Ritter. 3d scene segmentation for autonomous robot grasping. In *IROS*, pages 1734–1740, Oct 2012.

[WSF18] R. Weilharter, F. Schenk, and F. Fraundorfer. Globally consistent dense real-time 3d reconstruction from rgbd data. In *Proceedings of the OAGM Workshop 2018*, pages 121–127, 2018.

[Zha00] Z. Zhang. A flexible new technique for camera calibration. *PAMI*, 22(11):1330–1334, 2000.

[ZZ11] C. Zhang and Z. Zhang. Calibration between depth and color sensors for commodity depth cameras. In *2011 IEEE Int. Conference on Multimedia and Expo*, pages 1–6, 2011.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.28, No.1-2, 2020

Reconstruction of Photon Distributions in Liquid Scattering Media

Benjamin T. Cecchetto
Queen's University
25 Union Street
K7L 2N8
Kingston, ON, Canada
cecchett@cs.queensu.ca

James Stewart
Queen's University
25 Union Street
K7L 2N8
Kingston, ON, Canada
jstewart@cs.queensu.ca

ABSTRACT

We present an experimental setup to capture photon distributions from a liquid scattering medium and to reconstruct the flux density throughout the medium. The capture mechanism moves an occluder through the medium, which is illuminated by a laser. For each position of the occluder, the exiting light intensity is recorded at selected positions on the boundary of the medium; this produces a measured distribution of intensities blocked by the occluder. The differences between blocked and unblocked intensities are used to solve for sets of photon paths that match the observed exiting light intensity. From these photon paths, we calculate the photon distribution within the medium. This work is validated with experiments using two occluder sizes in different concentrations of milk-water solutions, as well as different incoming and outgoing light poses.

Keywords
Scattering, Physics, Photon Distributions, Optimization, Simulation, Computer Graphics, Rendering, Physically-based Modeling, Imaging

1 INTRODUCTION

This paper presents an experimental setup and algorithm to determine the distribution of photons within a physical scattering medium. In what follows, an input pose (i.e. a position and direction) and an output pose, both on the boundary of the medium, will be fixed. Then the paths of only those photons that enter the medium from the input pose and exit the medium from the output pose will be considered.

In a purely transmissive medium, the photon path is a straight line with all of the light travelling along the line. In a scattering medium, there are many photon paths for a given pair of input and output poses, so the flux through each position will vary across the medium.

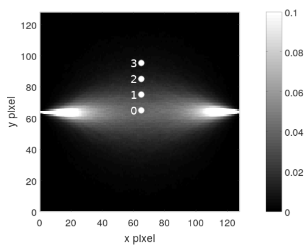
The output of this algorithm is, in two dimensions, a *photon distribution image* in which each pixel stores the average flux density through points in the medium corresponding to that pixel, normalized by the output magnitude for a given entry and exit pose pair.

In addition to a fixed entry and exit pose, we consider an additional constraint in order to further study the photon paths. We fix an internal point that the light must travel through, and examine the distribution of photon paths with this additional constraint. Simulated examples of these distributions are depicted in Figure 1.

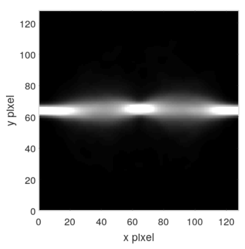
Knowledge of photon distributions may be of importance to rendering and reconstruction, because the distribution provides probabilistic weightings in the path matrix used in *ART* or *SART* reconstruction [1, 2]. However, it is difficult to capture these distributions using a simple, low-cost setup. Our goal is to capture a photon distribution image of desired resolution so that it may be used in the aforementioned reconstruction techniques.

To obtain the photon distribution image, a laser is directed into the medium and a camera records the output intensity on the other side of the medium. An occluder is then moved through the medium. At each position of the occluder, the measured difference between the output intensity with the occluder and that without the occluder will provide the power of the photons intersecting the occluder, provided the occluder has an albedo of zero. But the occluder must be much larger than a single pixel to obtain a sufficient signal-to-noise ratio in the difference of intensity. To obtain the power at individual pixels, further processing is required.

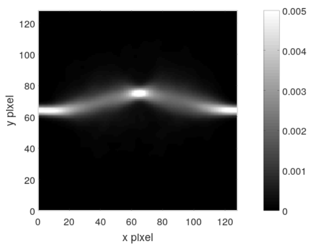
Photon paths are computed that produce the same differences of output intensity as were measured by the



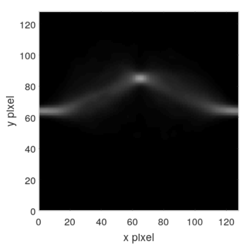
(a) The full photon distribution



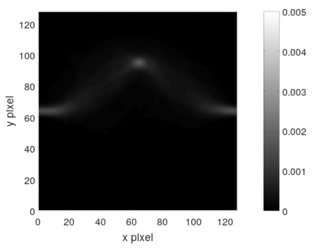
(b) Through point 0



(c) Through point 1



(d) Through point 2



(e) Through point 3

Figure 1: Photon distributions through different points in the medium.

large occluder. These paths can be used to provide an estimated power through any region of the medium, including a pixel region, which is smaller than the occluder.

The use of such paths was motivated by the paths analytically derived by Feng *et al.* [3] for a diffusion approximation. Each of their paths travels from an input pose to an output pose that is a varying distance away from a central "most probable path". Their derivation also provides an "approximate boundary" of the distribution. Therefore, we propose a similar numerical approach.

We use an active contour-based method, which starts with an initial set of template photon paths originating at the input pose (i.e. the laser position) and terminating at the output pose (i.e. a pixel in a camera image). The simulated output from these photon paths is compared to the physically captured output, and the paths are warped to correct the error between the two. The use of photon paths permits arbitrary distribution shapes to be fit to the captured output. The density of the paths at a point within the medium is proportional to the flux density at that point. Compared to Monte Carlo methods, this approach provides a very efficient simulation of light transport.

2 RELATED WORK

Our work builds on the distribution derived by Feng *et al.* [3]. The derivation depended on a diffusion model [4] for photon migration. A similar experimental setup has captured photon distributions, but was limited by the pixel resolution [5]. Feng *et al.*built on their work to perform an analysis for an object blocking the photon paths [6] as well. The diffusion model has been used to perform backprojection [7] to solve for the interior of a scattering region. With the actual photon distributions, it may be possible to perform optical tomography using a variety of techniques [8].

There are many different scattering approximations that can be used to render a scene. Real-time systems need further approximations to obtain the necessary performance [9], as ground truth Monte-Carlo simulations cannot yield real-time performance. Real-time systems use depth of materials, or approximate scattering in texture space. One of the more important contributions is by Jensen *et al.* [10], in which a diffusion approximation is used to derive a local model for incoming and outgoing light.

Light can be sampled through a scattering medium along paths to determine whether the light paths are occluded [11]. Sampling methods use parts of physical models to render pleasing, but physically inaccurate images. Gkioulekas *et al.* have done much work with reconstruction and simulation of scattering media [12, 13]. They have also shown the types of subtle variations possible through changing the phase function, which isn't captured in approximated models [14].

3 APPROACH

3.1 Experimental Setup

The experimental setup is shown as a schematic in Figure 2 and as a photograph in Figure 3.

A glass container with a square cross-section holds the medium. Light from a laser enters at one edge. A two-dimensional plotter robot moves an occluder through the medium. For each position of the occluder, a camera image is captured (see Figure 4) at another edge of the medium. The whole setup is draped with light-blocking fabric.

For a fixed laser pose and a fixed camera pose, each pixel, $p=(u,v)$, in the camera image corresponds to an output pose at the boundary of the medium. The radiance recorded at p varies with the position (x,y) of the occluder. Let $I_p^w(x,y)$ be the radiance at p when the occluder of width w is at position (x,y). Let I_p^{absent} be the radiance of p without the occluder present. Then, the

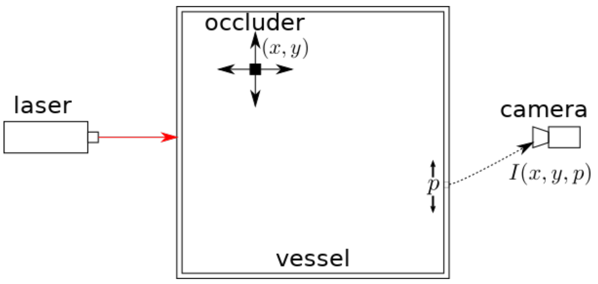


Figure 2: Photon distribution scanner setup.

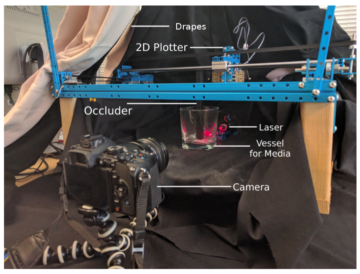


Figure 3: Experimental setup

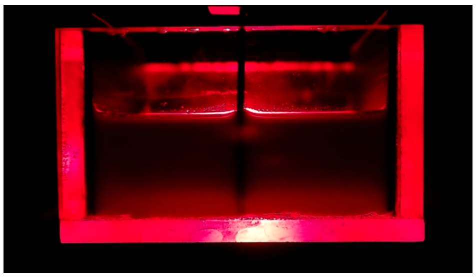


Figure 4: A camera image with the occluder in the front middle of the glass container.

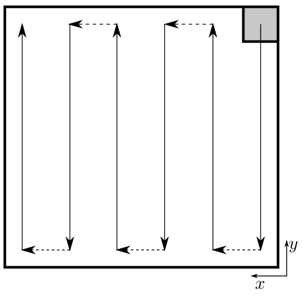


Figure 5: The path the two-dimensional plotter robot takes over the medium. The *on* positions for the laser are marked with a solid line, the *off* positions are marked with a dotted line. The occluder is represented by the grey box following the arrow path. There are a total of 128 vertical paths, sampled at 128 positions along each vertical line to generate a 128 × 128 image.

photon distribution *for a given pixel p* with an occluder of width w at position (x,y) is

$$D_w^p(x,y) = I_p^{\text{absent}} - I_p^w(x,y).$$

While $D_w^p(x,y)$ is a difference of radiances, it is proportional to the difference of fluxes through p for every (x,y), as the pixel area and its solid angle are constant. Thus, $D_w^p(x,y)$ is proportional to the flux passing through the occluder at position (x,y).

The approximate flux density at (x,y) can be obtained by dividing by half of the occluder's surface area inside the volume, excluding the bottom surface of the occluder, which scrapes along the bottom of the container. As this surface area is constant, $D_w^p(x,y)$ is also approximately proportional to the flux density at (x,y).

We will discuss using the raw data $I_p^w(x,y)$ to obtain the most probable photon paths in Section 3.2. Our path-based approach has the novelty of determining the photon distributions even for regions of the medium that are smaller than the occluder used in the experiment, yielding more accuracy in reconstruction or rendering. Example distribution images are shown in Section 4.1.

3.1.1 Scan Process

To accelerate the data acquisition, the camera captures a video during the experiment. While the camera records, the plotter moves continuously in the y direction from the start boundary to the end boundary. Then the plotter moves in the x axis by a small amount depending on the desired capture resolution, and the process repeats. This is illustrated in Figure 5. The laser is turned off between each continuous movement. Doing so facilitates the later (automatic) processing of the video.

3.2 Iterative Path Algorithm

Scanning the medium as described above allows us to use various occluder widths. However, a better signal-to-noise ratio is obtained with a larger occluder, since the measured difference between the occluder and no occluder is farther from the noise floor of the camera. The size of the occluder that maximizes the signal-to-noise is experiment-dependent. Performing a scan using a small occluder may obtain a better resolution, but we wish to obtain the same resolution with larger occluders. Thus, we propose a novel algorithm that obtains the most probable photon paths, and is relatively insensitive to the occluder size.

From these paths we can generate a simulated **absorption image** $I_p^w(x,y)$ for any occluder width w. To clarify, the absorption image $I_p^w(x,y)$ is not an image captured by the camera, nor is it a photon distribution image, but is an array of values of the intensity difference seen by the camera through a particular pixel, p, for different occluder locations, (x,y).

Let the entry and exit poses be denoted by $\Pi_0 = (\boldsymbol{x}_0, \boldsymbol{d}_0)$ and $\Pi_1 = (\boldsymbol{x}_1, \boldsymbol{d}_1)$ with positions $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, and directions $\boldsymbol{d}_0$ and $\boldsymbol{d}_1$. Photons travel in many piecewise linear paths from pose Π_0 to Π_1, which can be simulated using the radiative transfer equation. However, for better performance, we assume that the most probable photon paths that match the light's overall physical behaviour are smooth. This approximation yields reasonable results, as discussed in Section 4.2. We propose an iterative approach, where we start with an initial set of paths and generate a simulated absorption image. Then we compare the simulated absorption image with the corresponding physically measured absorption image, and update the paths to reduce the error between the two. This process is repeated until we converge on a solution. This approach is similar to active contour models, but solves for many contours at once [15].

3.2.1 Template Paths

We generate an initial set of smooth template paths with boundary points $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ as follows. Let the midpoint of the line connecting $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ be $\boldsymbol{M} = \frac{1}{2}(\boldsymbol{x}_0 + \boldsymbol{x}_1)$. Let $\boldsymbol{n}$ be the unit normal of this line. Let d_{min} be the distance from the midpoint to the closest medium boundary along $\boldsymbol{n}$.

The template paths are generated by fitting a quadratic curve between $\boldsymbol{x}_0$, a middle point $\boldsymbol{M}_i = d_i \boldsymbol{n} + \boldsymbol{M}$, and endpoint $\boldsymbol{x}_1$ for each path i and $d^i_{\text{path}} \in [-d_{\text{min}}, d_{\text{min}}]$. If there are paths generated that leave the medium, we remove the same number of paths on both sides such that they are contained in the medium. The distances, d^i_{path}, are sampled using a Gaussian distribution so that they are more dense near zero. The analytical distribution curves are also quadratic, of the form

$$y \approx (2\ln 2)\frac{x(r_{\text{prob}} - x)}{\kappa\, r_{\text{prob}}} \tag{1}$$

for a given y position, radius of the distribution probability r_{prob}, and the inverse diffusive absorption distance $\kappa = \sqrt{3\sigma_a \sigma_s (1-g)}$ [3]. Since The material parameters are unknown, so we set the quadratic curves to be initially non-intersecting in order to allow the warp to move the curves into the correct positions.

This is depicted in Figure 6. These paths do not intersect except at the fixed entry and exit points. This is necessary to prevent degenerate cases where overlapping paths cannot be separated with a warp. These template paths will be non-linearly warped at each path point such that the absorption image simulated from these paths matches the captured absorption image, as described in the following sections.

3.2.2 Distribution Image Generation

Let the simulated absorption image for the set of paths $\mathbb{P}$ and known occluder width w be $G^w_{\mathbb{P}}(x,y)$. Given a

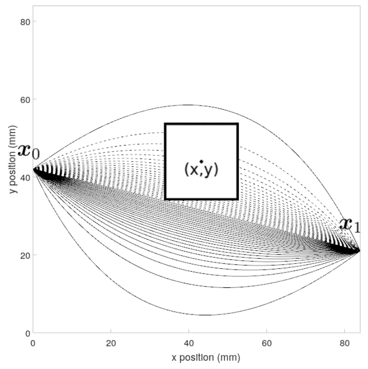


Figure 6: The template paths chosen for our iterative path algorithm given the start and end points $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ respectively. The dashed lines indicate the paths that do not contribute to the simulated output intensity at $\boldsymbol{x}_1$, as they get absorbed by the occluder.

set of paths, to generate an absorption image we calculate the percentage of paths that are occluded by an occluder of width w centered at each position (x,y) in the medium. These positions match the distribution scanner positions as discussed earlier. This is a very efficient way to approximate the radiative transfer simulation, as it is simply an intersection test between every path and a square. We will compare this simulated absorption image with a captured absorption image with the same occluder width. This approach allows us to create photon distributions with a finer resolution than the original captured occluder width.

3.2.3 Warp Calculation

Given a target absorption image $I^w_p(x,y)$ for a known camera pixel, p, and occluder width, w, and set of paths, $\mathbb{P}$, we minimize $||I^w_p(x,y) - G^w_{\mathbb{P}}(x,y)||$ through warping $\mathbb{P}$. A warp field is defined as a vector field $\boldsymbol{v}(x,y) = (s(x,y), t(x,y))$ that encodes a displacement vector $\boldsymbol{v}$ for every (x,y) position in the image. We evaluated three different approaches to generate a vector field between two absorption images: the gradient of the difference image $\nabla(I^w_p(x,y) - G^w_{\mathbb{P}}(x,y))$, the Lucas-Kanade optical flow field from $G^w_{\mathbb{P}}(x,y)$ to $I^w_p(x,y)$, and a combination of the two by alternating every iteration.

The convergence for the three methods is shown in Figure 7. The optical flow technique does not perform very well. The combination optical flow/gradient technique converges faster, but converges to a result with a higher error. The gradient approach performs the best, so we use this approach for the remainder of the paper.

3.2.4 Path Warping

For each path point $\boldsymbol{x}_k$ over all paths, bilinear interpolation is applied to the four closest vectors in the warp field to determine the vector $\boldsymbol{v}_k = (s_k, t_k)$ at the given point. Then $\boldsymbol{v}_k$ is projected onto the path's normal $\boldsymbol{n}_k$ at point $\boldsymbol{x}_k$, and update the point

$$\boldsymbol{x}_k \leftarrow \boldsymbol{x}_k + h \cdot proj_{\boldsymbol{n}_k} \boldsymbol{v}_k, \tag{2}$$

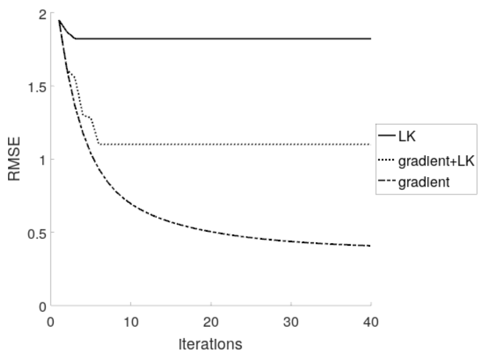


Figure 7: The root mean squared error (*RMSE*) between simulated absorption image $G_{\mathbb{P}}^{w}(x,y)$ at iteration i and target absorption image $I_{p}^{w}(x,y)$ using a gradient approach, using Lucas-Kanade optical flow (LK), and using an alternating gradient and Lukas-Kanade optical flow approach. The *RMSE* is unitless as it is a difference of percentages.

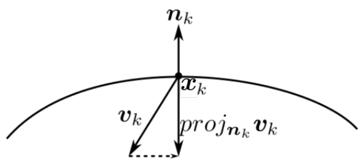


Figure 8: Sampling the vector field $\boldsymbol{v}$ to apply a warp to the path point $\boldsymbol{x}_k$. Then $\boldsymbol{v}_k$ is projected onto the path's normal $\boldsymbol{n}_k$.

for a small $h > 0$. For a gradient field, the displacement equals a fraction of the directional derivative in the direction $\boldsymbol{n}_k$. This step is visualized in Figure 8.

3.2.5 Constraints

Applying the warp may introduce undesirable properties such as intersecting paths, non-smoothness, or broken boundary conditions. We introduce constraints to remove each of these properties as follows.

To force the paths not to intersect, the intersection points are projected outwards from the middle path, as follows. The middle path is compared to one of its neighbouring paths. The intersection points between the two paths is determined (if it exists). If any intersection points exist, the path points of the outer path are projected away from the middle path by some small distance. This process is repeated outwards until the boundary path for that side is reached. The same process is repeated for the other neighbouring side.

To obtain the correct direction at the entrance and exit poses, the pose positions are set, then nearby points are forced in line with the pose direction.

Paths are smoothed using a Gaussian filter. Finally, the path points are resampled at regular intervals using cubic splines. This ensures a good sampling for the next iteration of the warp algorithm. The whole process is repeated multiple times. This is similar to constraints using Verlet integration and other physics solvers [16].

3.2.6 Convergence

The above steps are repeated until the difference in error between consecutive iterations falls below a threshold. The algorithm outputs the set of paths from the last iteration. Finally, a *photon distribution image* is generated by counting the paths that intersect each pixel in that image.

4 RESULTS

4.1 Raw Capture Results for Milk-Water Solutions

Images were captured as video with an Olympus E-M1 Camera at 30 frames per second. The camera used a lens at 60mm focal length, f/2.8 aperture, and 1600 *ISO*. In order to capture these dark images, a high *ISO* and open aperture were requried. The camera was focussed at the boundary between the solution and the glass. We assumed the amount of light arriving at the camera from the medium due to refraction and reflection events was minimal.

The distribution scan was performed for multiple different milk-water solutions. Figure 9, shows results from a 75% milk-water solution (that is: 75% milk and 25% water). The figure shows different scans for different output positions on the left edge and a fixed input position on the right edge. Figure 10 shows a similar image, except the camera is now on the bottom edge. Different distributions are shown for varying output pixel positions. Figure 11 shows the result for varying milk-water dilution ratios. All ratios use the same output pixel position on the left of each distribution. The intensity at each pixel is the amount of light absorbed by the occluder for the position in physical space. The darkest two regions occur where the occluder is blocking all the light. One position is in front of the pixel in question which blocks all exiting light. The other position is where the laser enters the solution, blocking all incoming light.

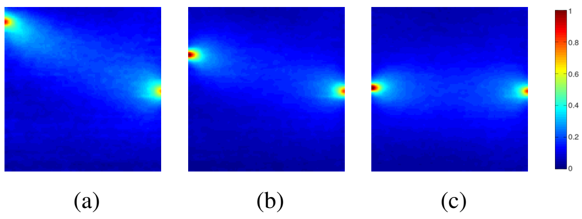


Figure 9: Distribution scans for varying pixel positions at 75% milk-water solution with the laser (right side) on the opposite side as the camera (left side).

4.2 Validation of Algorithm

The algorithm was valided through a simulated absorption image as well as captured absorption images from the scanner. For each run of the algorithm, 512 photon

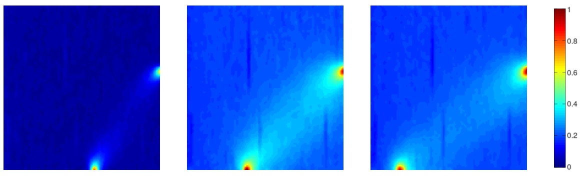

(a) (b) (c)

Figure 10: Distribution scans for varying pixel positions at 75% milk-water solution with the laser (right side) on the adjacent side to the camera (bottom side).

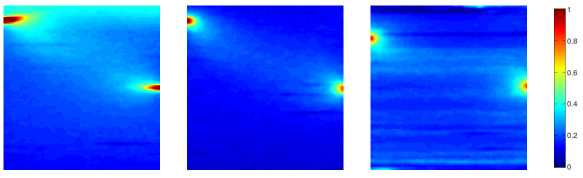

(a) 12.5% milk-water dilution (b) 50% milk-water dilution (c) 100% milk-water dilution

Figure 11: Distribution scans for a single output pixel position with varying milk-water solutions. The laser (right side) is on the opposite side of the camera (left side).

paths were used in the initial and subsequent iterations. For visualization here, only a subset of these paths are shown.

Figure 12 depicts the output of the algorithm on a *virtual* set of target paths using a 10mm occluder. These target paths have a nonlinear pinching closer towards the end points to approximate a narrow band phase function. This provides a set of paths that incorporates pose orientation, as well as position. The algorithm performs quite well when given an image generated by a set of these known paths. Even though the error using the template paths is large, it converges on a valid result, as the final simulated absorption image is similar to the "ground truth" captured absorption image. Also, the root mean squared error (*RMSE*) shows a large improvement, from 0.114 with an image generated from the template paths to 0.014 with the final warped paths. This test was repeated with a 2mm occluder and obtained a similar *RMSE* improvement, from 0.18 to 0.03.

Figures 13 depict examples of the output of the algorithm with captured absorption images *from the physical experiment*. The three datasets are a 50% milk-water solution with output pose straight across from the laser, output pose at an offset from the laser on the opposite side, and a 75% milk-water solution with the output pose on the adjacent side relative to the laser. We denote these as the *straight*, *offset*, and *side* experiments respectively. Note that for each of the three experiments, the 2mm occluder captured images has a lower background intensity than using the 10mm occluder. This suggests that the 10mm captured images yields a higher signal-to-noise ratio since there should be a uniform intensity outside the photon distribution boundary.

The errors between the reconstructed images and the captured ones are depicted in the fourth row of Figure 13. The template and final errors are shown in Table 1. The template errors correspond to the starting template paths. The algorithm works well for the virtual captures with both occluders. For physical captures, the algorithm performs better using the 10mm than the 2mm occluder captures. This is expected as the 2mm occluder capture images have more noise, hence the active contour algorithm could not find a better solution.

Other deviations may be caused by stray light, background noise, reflections and other physical phenomena not captured by our model. The diffusion model used also assumes that the medium is semi-infinite or infinite. Thus the paths derived do not interfere with the boundaries.

The fifth row of Figure 13 depicts the final resulting photon distributions generated with a 1mm occluder for a 128×128 resolution image. The scale is modified to be between 0 and 0.2 to enhance the darker areas.

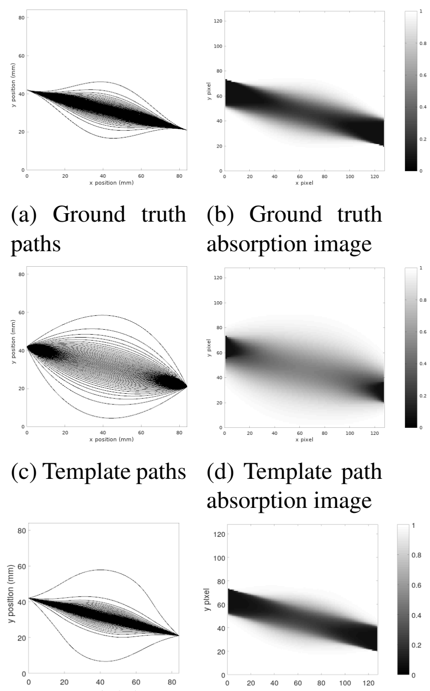
Table 1: The final root mean squared error (*RMSE*) between capture and simulated reconstruction images. The *RMSE* is unitless, corresponding to the fraction of the flux over the medium arriving at the output pose.

Experiment Type	**Occluder Size (mm)**	**Template *RMSE***	**Final *RMSE***
Virtual Offset	2	0.180	0.030
Virtual Offset	10	0.114	0.014
Straight	2	0.078	0.078
Offset	2	0.070	0.069
Side	2	0.101	0.100
Straight	10	0.062	0.023
Offset	10	0.125	0.043
Side	10	0.063	0.016

5 CONCLUSION

In this experimental setup, we have shown that it is possible to scan a scattering medium with a relatively large occluder, in order to obtain photon distributions through the medium. We have also presented an iterative path algorithm that can efficiently determine a set of photon paths that produce a simulated absorption image that matches the captured absorption image.

It may be interesting to use multiple occluder sizes and different volumes of liquid. This may be useful to determine the limit for the flux density as the occluder size and liquid depth approach zero. Furthermore, a translucent occluder also might be a better posed problem in



(a) Ground truth paths (b) Ground truth absorption image

(c) Template paths (d) Template path absorption image

(e) Final computed paths (f) Final computed absorption image

Figure 12: Reconstruction from a *virtual* experiment. (a) the "ground truth" photon paths in the virtual medium. (b) the simulated absorption image from the ground truth paths. (c) the initial template paths. (d) the absorption image from the initial template paths. (e) the final computed paths. (f) the absorption image from the final computed paths.

terms of reconstruction, as each path may be able to encode the percentage of intensity absorbed instead of a binary amount.

Photon distributions are a generalization of the matrix used for *ART* and *SIRT*. It would be interesting to see how well the distributions perform using the linear reconstruction techniques for a non-linear simulated phenomena such as scattering. In addition to reconstruction, it would be useful to determine the accuracy of using the photon paths for rendering simulation. An example would be to use them to do efficient approximations for occluders in scattering media. The path shapes could determine the type and amount of scattering whereas the path attenuations could depend on the medium in question. This is an approximation of the Monte-Carlo process, but it would be beneficial to validate the performance versus accuracy tradeoffs.

6 REFERENCES

[1] R. Gordon, R. Bender, and G. T. Herman, "Algebraic reconstruction techniques (art) for three-dimensional electron microscopy and x-ray photography," *Journal of theoretical Biology*, vol. 29, no. 3, pp. 471–481, 1970.

[2] A. H. Andersen and A. C. Kak, "Simultaneous algebraic reconstruction technique (sart): a superior implementation of the art algorithm," *Ultrasonic imaging*, vol. 6, no. 1, pp. 81–94, 1984.

[3] S. C. Feng, F. Zeng, and B. Chance, "Monte Carlo simulations of photon migration path distributions in multiple scattering media," in *Photon Migration and Imaging in Random Media and Tissues*, vol. 1888, pp. 78–90, International Society for Optics and Photonics, 1993.

[4] M. S. Patterson, B. Chance, and B. C. Wilson, "Time resolved reflectance and transmittance for the noninvasive measurement of tissue optical properties," *Applied optics*, vol. 28, no. 12, pp. 2331–2336, 1989.

[5] W. Cui, C. Kumar, and B. Chance, "Experimental study of migration depth for the photons measured at sample surface," in *Time-Resolved Spectroscopy and Imaging of Tissues*, vol. 1431, pp. 180–191, International Society for Optics and Photonics, 1991.

[6] S. Feng, F.-A. Zeng, and B. Chance, "Photon migration in the presence of a single defect: a perturbation analysis," *Applied optics*, vol. 34, no. 19, pp. 3826–3837, 1995.

[7] S. Colak, D. Papaioannou, M. van der Mark, H. Schomberg, J. Paasschens, J. Melissen, N. Van Asten, *et al.*, "Tomographic image reconstruction from optical projections in light-diffusing media," *Applied optics*, vol. 36, no. 1, pp. 180–213, 1997.

[8] S. R. Arridge, "Optical tomography in medical imaging," *Inverse problems*, vol. 15, no. 2, p. R41, 1999.

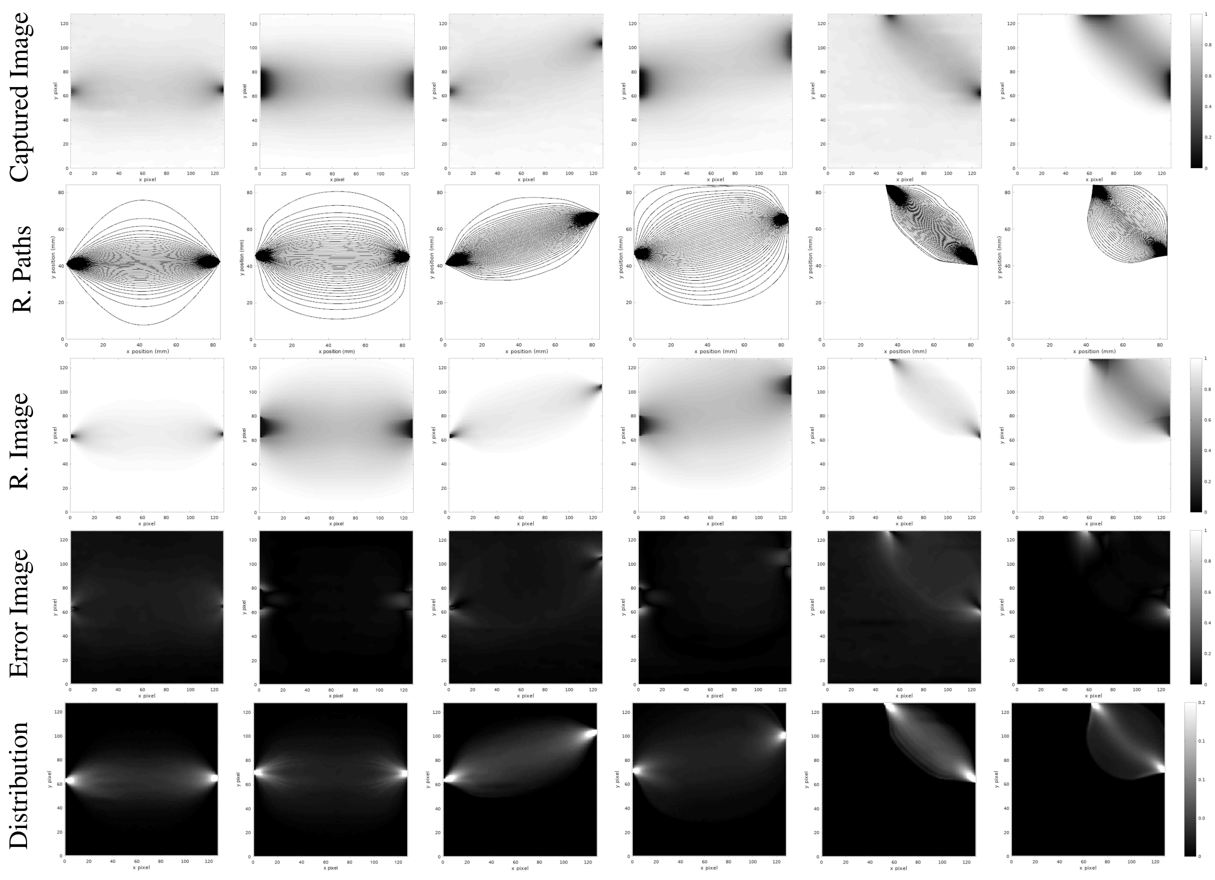
[9] S. Green, "Real-time approximations to subsurface scattering," *GPU Gems*, vol. 1, pp. 263–278, 2004.

[10] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan, "A practical model for subsurface light transport," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 511–518, 2001.

[11] B. Langsdorf, "Gpu programming exposed: The naked truth behind NVIDIA's demos," tech. rep., Technical report, NVIDIA Corporation, 2005.

[12] I. Gkioulekas, S. Zhao, K. Bala, T. Zickler, and A. Levin, "Inverse volume rendering with material dictionaries," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, pp. 1–13, 2013.

[13] I. Gkioulekas, A. Levin, and T. Zickler, "An evaluation of computational imaging techniques for



(a) 2mm Straight (b) 10mm Straight (c) 2mm Offset (d) 10mm Offset (e) 2mm Side (f) 10mm Side

Figure 13: The *straight*, *offset*, and *side* experiments for 2mm and 10mm occluders. Depicted are the raw captured absorption images, reconstructed paths, simulated absorption images, error absorption images, and final photon distribution images. The light enters the left side and exits the right side for the straight and offset experiments. The light enters the right side and exits the top side for the side experiments.

heterogeneous inverse scattering," in *European Conference on Computer Vision*, pp. 685–701, Springer, 2016.

[14] I. Gkioulekas, B. Xiao, S. Zhao, E. H. Adelson, T. Zickler, and K. Bala, "Understanding the role of phase function in translucent appearance," *ACM Transactions on graphics (TOG)*, vol. 32, no. 5, pp. 1–19, 2013.

[15] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[16] L. Verlet, "Computer" experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules," *Physical Review*, vol. 159, no. 1, p. 98, 1967.

Real-time Service-based Stream-processing of High-resolution Videos

Florian T. Wagner
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam, Germany
florian.wagner@student.hpi.de

Jürgen Döllner
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam, Germany
juergen.doellner@hpi.de

Matthias Trapp
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam, Germany
matthias.trapp@hpi.de

ABSTRACT

This paper reports on a service-based approach to enable real-time stream-processing of high-resolution videos. It presents a concept for integrating black-box image and video processing operations into a streaming framework. It further describes approaches to optimize data flow between the processing implementation and the framework to increase throughput and decrease latency. This enables the composition of streaming services to allow scaling for further throughput increase. We demonstrate the effectiveness of our approach by means of two real-time streaming-processing application examples.

Keywords: stream processing, high-resolution video, real-time

1 INTRODUCTION

1.1 Motivation and Challenges

This work aims at executing fast video processing operations, such as color enhancement or stylization, in a service-based environment using streaming. Currently, such operations are performed on the user's own mobile or fixed device using specialized software and hardware, e.g., Graphics Processing Units (GPUs). However, the increasing complexity of such operations coupled with the increase of spatial and temporal resolution of the input data requires more powerful hardware. Additionally, more complex operations need to be optimized for each possible target device to achieve good performance, requiring high investments in developer time even after the initial development. As the user expects low latency feedback of the chosen combination of video, operation, and parameters, extensive online preprocessing approaches becomes unfeasible. The preprocessing cannot be performed in an offline fashion, as the video data may be read-only and transient, e.g., in a live-streaming scenario. Therefore a suitable approach has the following challenges/constraints:

Low Latency (C.1): In order for the user to effectively choose an operation/parameter combination, the latency to process videos should be in the sub-second range.

High Throughput (C.2): An approach should be able to process high-resolution videos with state-of-the-art operations with at least the same frame rate as the input video.

Hardware Independence (C.3): The user should be able to use the approach on arbitrary devices. This especially includes mobile devices such as smart phones or tablets.

Low Integration Costs (C.4): The overhead to integrate a given operation into the processing approach should be low. This is necessary to provide the user with new and updated operations in a short time frame, e.g., to facilitate short time-to-market.

Challenges C.1 and C.2 represent the requirement for an efficient end-to-end solution. This means that the system is required not only to introduce a small performance overhead, but also needs to improve the performance of the offered video processing operations.

1.2 Optimization Approaches

There are multiple combinable approaches to improve the performance of video processing operations. Each of these approaches is based on a trade-off: some sacrifice quality, others development time. The major approaches and their trade-offs are as follows:

Preprocessing (A.1): This approach speeds up computation by preprocessing input into a format that is advantageous for further processing. This often comprises an increase of data locality or indexing operations. While preprocessing can deliver significant improvements, it always introduces additional latency. On top of this there is a trade-off between generality of the preprocessing and the obtained speedup.

Manual Optimization (A.2): This approach increases hardware utilization through operations and hardware specific optimizations. The speedup gained is proportional to the time invested and the skill of the optimizing engineer. This approach can not improve the performance of operations that are already sufficiently optimized. Additionally, this approach requires a thorough understanding of the implementation to be optimized and thus further introduces development costs for new operations.

Approximate Computing (AC) (A.3): This approach sacrifices some quality to speed up the computation. It is often not possible to do this without making modifications to the operation implementation, due to the tight coupling between the implementations and the underlying graphics libraries and hardware drivers. This means that additional developer time is required to either introduce approximations into the operation implementation itself or bring it into a form where it can be automatically approximated.

Domain-specific Language (DSL) (A.4): The above optimizations can often be made in an automated fashion, if the operation is described in a DSL. These automated optimizations often outperform manual optimizations both in achieved performance as well as performance to development overhead ratio. However, most implementations are not developed using these DSLs and translating/transpiling them will still incur a per-operation overhead.

Task Parallelization (A.5): While data parallelism is utilized in current video processing operations, their scope is too limited to implement task parallelism. Task parallelism can instead be added on a higher systemic level to fully utilize all available hardware. Because task parallelism is provided at a higher level, it can be implemented only once and then reused across multiple video processing operations and hardware architectures. However, it still requires the user to provide enough hardware to meet their performance requirements.

Off-device Processing (A.6): By performing the computation on a different device, the user is freed from the cost of operations and ownership of current processing hardware. Additionally they can easily scale the hardware to their performance requirements. This approach incurs additional latency when transferring the initial video to the computation device and the processed video to the consumption device. However, if the initial video is not already present on the consumption device, then the latency for transferring the initial video might even be lower with this operation, due to higher bandwidth of the processing hardware.

As can be seen, none of these approaches alone is capable of solving all the challenges. Additionally, some approaches require access to the operation's source code to perform effectively, rendering them impossible in scenarios where the implementations are provided by a third party. Therefore, a combination of approaches seems promising.

1.3 Approach and Contributions

Our approach combines parallelization (A.5) with off-device processing (A.6) to alleviate the challenges associated with current video processing systems. Off-device processing allows usage of high-powered processing hardware without the need to actually own this hardware. The added transmission latency is low and reduces with fast connections, increasingly. Additionally, processing latency is lowered due to more powerful processing hardware, yielding a low total latency. Further, parallelization is used to gain not just the benefit of more powerful but also more plentiful processing devices to be had with off-device processing. This further improves throughput and potentially can also lower the processing latency even further.

To summarize, this paper makes the following contributions: (1) it presents a concept for integrating black-box software components for image and video processing operations into a streaming framework, (2) it describes approaches to optimize data flow between the processing implementation and the framework to increase throughput and decrease latency, (3) it enables composition of streaming services to allow scaling for further throughput increase.

The remainder of this paper is structured as follows. Sec. 2 reviews related and previous work w.r.t. various optimization approaches of image and video processing operations and and service-based processing in general. Sec. 3 introduces the concept for enabling real-time service-based stream processing of high-resolution videos. Sec. 4 briefly describes implementation aspects of our proposed concept. Sec. 5 discusses the presented approach and implementation by means of different applications examples and a performance evaluation. Finally, Sec. 6 concludes this paper and outlines future research directions.

2 RELATED WORK

There is a vast body of related work w.r.t. video streaming and processing. This section focus on optimization approaches and specifics of service-based provisioning for video transformation operations.

2.1 Domain-specific Languages

DSLs are very appealing as they allow division of optimizations from algorithm description. Additionally they offer a degree of platform independence, as they abstract away the hardware specifics without sacrificing performance. One of the simplest optimizations is improving the locality of data and the parallelism of computation for a given operation [28]. Further domain knowledge helps with optimizing for specific hardware may be directly captured by the language [17]. Alternatively it may be specified separately from the algorithm, thus an optimization expert can quickly tune new operations for specific hardware. Some frameworks enable optimization that operates over all operations, therefore using all available information about a given operation [21]. Additionally, the abstraction provided by a

DSL can enable higher developer productivity without sacrificing performance [29]. In the same vein, DSLs can enable optimizations that are outside the usual domain expertise of developers [20]. They can enable developers to automatically optimize for distributed systems [1]. On the other end of the scale, they can be used to use more powerful but specialized hardware than is usual for image and video processing operations, such as ASICs and FPGAs [26].

2.2 Approximate Computing

A different approach is found in AC, which trades quality for performance. One major aspect of AC are the singular operations that speed-up computation, such as skipping samples in an image [15] or modifying a kernel [16]. These have been accumulated into collections that can be automatically applied to existing kernels and optimized in a subsequent tuning phase, either by selecting between different kernels [31] or by generating variations of the original kernel [30] based on a quality target. Further approaches use alternative representations of the input to efficiently sample the approximation-parameter space [13]. This tuning has been extended and refined for video processing [41]. All these operations have in common that they *intrusively modify* the algorithm and trade quality for performance. In contrast thereto, our approach does not rely on preprocessing of algorithms and avoids quality control directed by the user. However, the above operations can be used in combination with our approach.

2.3 Visual Media Processing Services

Several software architectural patterns are feasible for implementing service-based image-processing [5, 22]. However, one prominent style of building a web-based processing system for any data is the service-oriented architecture [35]. It enables server developers to set up various processing endpoints, each providing a specific functionality and covering a different use case. These endpoints are accessible as a single entity to the client, i.e., the implementation is hidden for the requesting clients, but can be implemented through an arbitrary number of self-contained services. Since web services are usually designed to maximize their reusability, their functionality should be simple and atomic. Therefore, the composition of services [9] is critical for fulfilling more complex use cases [14]. The two most prominent patterns for implementing such composition are *choreography* and *orchestration* [23]. The choreography pattern describes decentralized collaboration directly between modules without a central component. The orchestration pattern describes collaboration through a central module, which requests the different web services and passes the intermediate results between them [27].

In the field of image analysis, Wursch *et al.* [39, 40] present a web-based tool that enables users to perform various image analysis methods, such as text-line extraction, binarization, and layout analysis. It is implemented using a number of Representational State Transfer (REST) web services and application examples include multiple web-based applications for different use cases. Further, the viability of implementing image-processing web services using REST has been demonstrated by Winkler *et al.* [37], including the ease of combination of endpoints. Another example for service-based image-processing is Leadtools (https://www.leadtools.com), which provides a fixed set of approx. 200 image-processing functions with a fixed configuration set via a web Application Programming Interface (API). In this work, however, a similar approach using REST is chosen, although with a different focus in terms of granularity of services.

Applications with respect to medical image processing are presented by Yuan *et al.* [43] as well as Moulick and Gosh [18]. Both propose a web-based platform to present and process medical images by using server-side computing for a series of image processing algorithms. Further, in the field of geodata, the Open Geospatial Consortium (OGC) set standards for a complete server-client ecosystem. As part of this specification, different web services for geodata are introduced [19]. Each web service is defined through specific input and output data and the ability to self-describe its functionality[42]. In contrast, in the domain of general image-processing there is no such standardization yet. However, it is possible to transfer concepts from the OGC standard, such as unified data models. These data models are implemented using a platform-independent operation format [4]. In the future, it is possible to transfer even more concepts set by the OGC to the general image-processing domain, such as the standardized self-description of services.

2.4 Distributed Processing Approaches

Over the last decade, a number of generalized approaches to distributed processing of streaming data have been developed. The most important ones, as identified by Karimov *et al.* [10] are Apache Storm [34], Spark [44], and Flink [3]. While these frameworks are developed for moving data between processing nodes, they do not address processing of image data on GPUs, which is required for high-throughput video processing applications. With respect to this, Scanner [25] is a system that is optimized for these kinds of applications. However, it does require preprocessing of the input data and does not work on a stream-based abstraction.

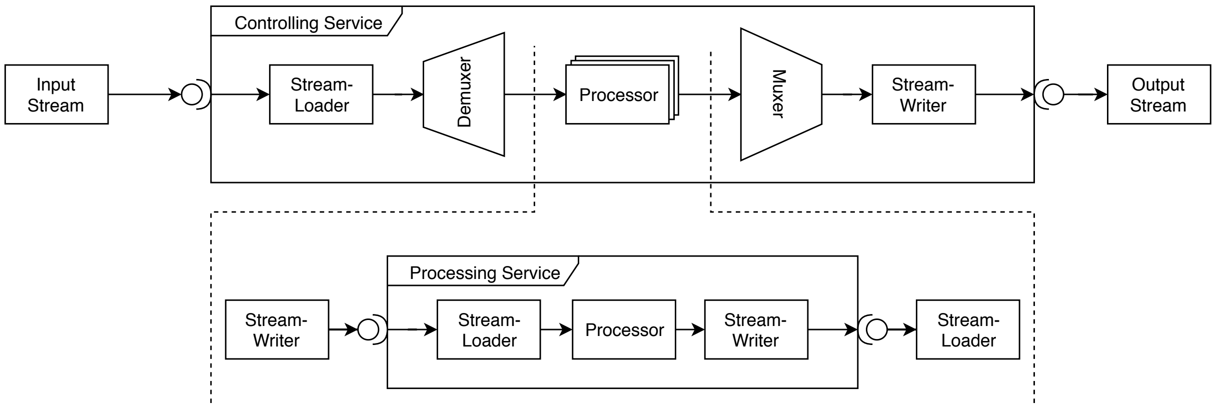


Figure 1: Conceptual overview comprising the components as well as data and control flow of the processing pipeline.

2.5 Visual Media Processing Operations

In this work, we focus on edge-aware and content-preserving image-processing as a fundamental tool in computational photography and non-photorealistic rendering for abstraction and artistic stylization for application and testing purposes. Typical approaches that operate in the spatial domain for abstraction use a kind of anisotropic diffusion [24, 36] and are designed for parallel execution, such as approximated by the bilateral filter [33] and guided filter [8].

A plenitude of stylization operations exist using these filters as building blocks to simulate traditional painting media and effects [12], such as cartoon [38] and oil paint [32]. However, these may become computationally expensive when applied in an iterative multi-stage process. This particularly applies to operations using global optimizations to separate detail from base information, e.g., based on weighted least squares [6] or locally weighted histograms [11], and recent operations that separate style from content using neural networks [7]. Because of their global optimization scheme, they are typically not suited for real-time application, in particular not on mobile devices. To this end, we implemented a variety of these operations using the proposed image-processing service including stylization, High Dynamic Range (HDR) tone mapping and compression, JPEG artifact removal and colorization, to demonstrate its versatile application. We used a representative subset of these operations for performance evaluation.

3 CONCEPTUAL OVERVIEW

This section describes our approach to real-time service-based stream processing of high-resolution videos. Based on preliminaries and assumptions, a conceptual overview of our system is given (Sec. 3.1), and components are described (Secs. 3.2 to 3.4).

3.1 Preliminaries & Assumptions

We base our work on software previously developed. For the handling of video data we use a framework designed to load videos and process them frame by frame, based on a pipeline concept. For the processing of the video frames, we use a software component comprising state-of-the art image abstraction operations that is called using a library interface.

Additionally, we make assumptions about the processing hardware and operation implementations used by our approach. We do not assume that the video is on the local host or has even been fully recorded when processing starts, as is the case in live-streaming contexts. Offline processing is therefore not possible. Additionally we assume that all operations work on a locality of the time and space domains. We also assume that the operation is being callable through a library interface and therefore resides in the same process space as our video processing pipeline. This means that memory and other resources (such as GPU handles) can be easily shared between the operation processor and the video pipeline, if the processor exposes appropriate interfaces. Finally, we assume the software component to be a black-box implementation, i.e., we cannot change or rely on the implementation details of the operation apart from the assumptions made above.

Fig. 1 shows a conceptual overview comprising the components as well as data and control flow of the processing pipeline of the proposed approach. It basically comprises the following components, which are described in greater detail in the remainder of this section; components (1) for receiving and sending video streams, (2) for tiling video in the spatial or temporal domain, and (3) that wrap the video processing operation.

3.2 Stream Separation (De-Muxing)

There are two domains, in which we can separate the data into tiles for parallel processing: spatial and temporal. If the applied processing for a target sample considers source samples at multiple points in a domain, then an additional border around the original tile is provided by the split. This prevents quality degradation due to missing source samples. The size of this border depends on the range of source samples in the given domain and is therefore dependent on the operation and its parameters. Because the software component is a black box, is is not possible to exempt the border from processing. Computational overhead is therefore introduced by each sample in the border. Because of this, the relative size of the border compared to the content should be minimized to minimize overhead.

Spatial Stream Separation. Spatial Separation splits the video in the spatial domain. Each spatially split frame produces exactly one frame-tile for each parallel strand of execution. If a border is required, it is added to the frame-tile dimensions. As an optimization the separation can be performed in only one dimension of the spatial domain. This eliminates borders in the other dimensions and is especially effective for large frame-tiles and operations that necessitate large borders. Spatial separation also facilitates computing operations using limited resources; While computation time is potentially unbounded, Video Random Access Memory (VRAM) and other hardware resources are limited. The usage of these resources is often coupled to the size of the input and output images. Spatial separation enables the processing of complex and resource intensive operations on hardware, which is otherwise not capable of handling these on the requested input complexity.

Temporal Stream Separation. Temporal separation is of advantage whenever the operation is *time-local*. In this case, no border frames need to be created and therefore the amount of data that is transferred between the distributed stages is minimized. Additionally, this method is the only possible method when the technique is global in the spatial domain. If the technique is also non-local in the time domain, border frames need to be introduced. To ensure a good ratio of content vs. border, multiple consecutive frames should be sent to each node for processing. Depending on the throughput of the nodes, this will introduce a significant increase in latency, as the border frames need to be processed before the content frames.

3.3 Per-Frame Video Processing

The actual video processing module is accessed via an API. This necessitates wrapping the module within a stage to embed it in our pipeline. The wrapper requires a way to pass data into and out of this processing module. To this end, we implement two approaches to pass data between the wrapper and the processing module, both exhibiting different computational overhead and flexibility. The first approach passed data via shared memory. If the processing is performed on the GPU, the overhead might significantly deteriorate the throughput. Additionally, it can not be guaranteed that the module does not first copy the data for processing, leading to potentially more overhead. The second approach is passing the data as a texture in a shared GPU context. To enable this, a GPU context can be obtained from the module. This context is then used to allocate the respective data memory in VRAM. The handles to this memory are then passed into the module for processing. Even if the data is copied, the asynchronous and parallel nature of GPU processing allows progress to be made on other work items while the data is copied.

3.4 Stream Compositing (Muxing)

Stream muxing combines the previously separated data into a final composition result. It receives the data, which is not defined as border values and stitches them to create the final video stream. Therefore, it is required to work in the same domain as the separation operation. If the separation is performed in the spatial domain, then the frame tiles are copied into a final frame representation. If the separation is performed in the temporal domain, then sufficient frames need to be buffered to unblock the processing nodes. These frames are then aligned into the same order as their originating frames, discarding border frames if they exist. This stage also maintains the synchronization with the input audio stream.

4 IMPLEMENTATION ASPECTS

This section describes implementation specifics for our approach. Based on the concept, the implementations of the codec handling (Sec. 4.1), separation and recombination (Sec. 4.2), as well as the service-based specifics (Sec. 4.3) are covered.

4.1 GPU-based En-/Decoding

Modern GPUs often have dedicated units for de- and encoding of videos in commonly occurring formats. Especially the encoders offer a significantly higher performance than the software implementations run on Central Processing Unit (CPU). Because of this our framework allows to utilize GPU de- and encoders where supported by the hardware. If the processing of the frames is also implemented on the GPU then an additional advantage may be gained by forgoing the uploading and downloading of data to and from the GPU. However, the interface of the GPU coding units might not always be exposed in the same GPU abstraction as the processing is required.

To bridge this gap, we support translation of the data between different GPU abstractions. This happens transparently to the processing module. Due to driver limitations, registering a resource from one GPU abstraction in another might introduce an additional allocation of memory. This might lead to performance degradation or even memory shortage if performed every frame. To this end we implement the adapter as a texture that is registered once at the initialization of the pipeline and then copied to and from. While this introduces additional data copies on the GPU, it does not degrade performance and allows processing of large data even on drivers where registering resources allocates memory.

4.2 Tiling Implementation

We use different implementations for the tiling and compositing stages of the pipeline, depending on the domain of the division and the location of the data. If data separation is performed on the spatial domain, it is implemented using a copy operation, depending on the location where the data resides. Data residing on the GPU is separated using texture copies, while data residing on the CPU is separated using *memcpy*. This means, that at least twice the memory footprint of a single frame is consumed during spatial separation - more if a border is present (e.g., required for neighborhood filtering). In effect, this overhead is often negligible since it only occurs during the separation itself, i.e., only one frame at a time consumes twice the amount of memory. Such additional use of resources is significantly smaller than the use of resources required to have multiple frames in flight.

If the separation should be performed in the temporal domain, it can be implemented by routing individual or chunks of frames to different processing stages. If a temporal border is required, all frames that are within a border need to be routed to multiple processing stages. Because the processing stages do not consume the memory of the frames, no copies need to be made during this routing. Copies are often made during transmission to different nodes, such that memory overhead might occur in this case as well. In a worst case scenario, each frame consumes $(1 + \mathit{numberOfInstances}) \cdot \mathit{memoryOfSingleFrame}$ memory at any point in time. Similar to spatial separation, this only occurs for at most one frame at a time, thus the total memory needed is bounded and near constant in a high-throughput pipeline.

4.3 Provisioning and Streaming

Service Provisioning. In order to rapidly deploy the service to new hosts, it is bundled as a Docker image [2]. This allows instancing up the respective image on different hosts without rebuilding the complete system. However, due to the required tight coupling to the GPU, all required processing libraries and APIs (such as Open Graphics Library (OpenGL) or CUDA) need to be present and supported on the host. This is a limitation of Docker with respect to GPUs. Additionally, as GPUs are often limited regarding the number of physical en/decoding units, using more than one instance per graphics card can lead to a degradation of performance.

Application of Streaming Protocols. There are three ways in which data may be streamed by the system: (1) ingesting streams from outside the system, (2) streaming between system instances, and streaming outside the system to consumers. All three cases raise different requirements on the streaming protocols used: For ingestion, common existing streaming protocols should be supported to enable easy interaction with existing stream sources, such as YouTube or similar. For streaming inside the system, low latency and high throughput is desired, so as to not degrade performance when scaling over smaller nodes. For streaming to the consumer, the protocol should be supported by web-browsers for easy consumption. For consumption, it is further desired to allow for some degree of asynchronicity, as the user might drop-out of the session and return to it at a later point in time.

Because of these differing requirements, we choose to support different protocols as follows. For stream ingestion and streaming inside the system, Real-Time Messaging Protocol (RTMP) was chosen. This protocol allows for low overhead, low latency streaming and is used by popular stream ingestion APIs such as Twitch. For streaming to the consumer we use Hypertext Transfer Protocol (HTTP) Live Streaming (HLS). This protocol allows to use readily available technologies on both ends of the connection. The server can be a HTTP driven and does not require prior knowledge about the nature of the video. On the client side the stream can be consumed using a standard video player in the web browser. Additionally, this allows for asynchronicity as the processed stream can be stored for as long as desired.

5 RESULTS & DISCUSSION

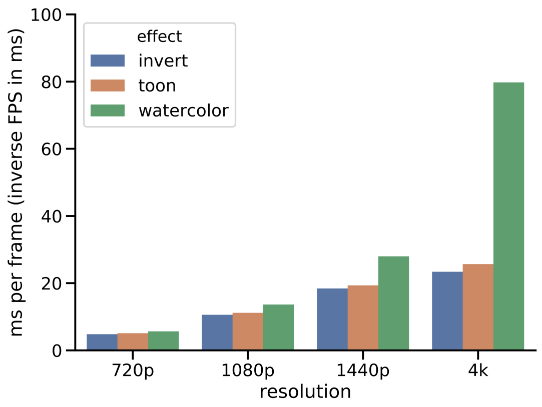
This section demonstrates our approach by means of different application examples (Sec. 5.1) and discusses its runtime performance as well as technical and conceptual limitations (Sec. 5.2).

5.1 Application Examples

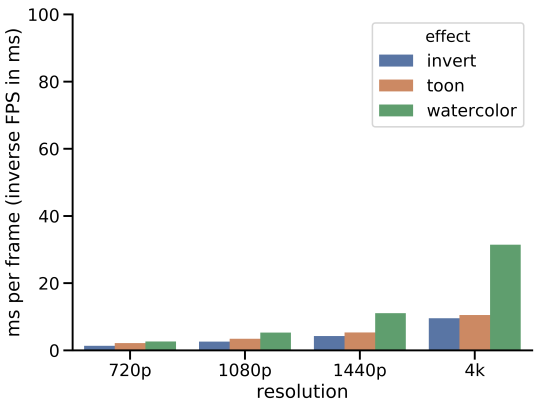
We tested our approach and its service-based integrations by means of different applications in the domain of web-based video processing as follows (please refer also to the accompanying video).

Interactive Web-based Video Editing. This application example demonstrate how the streaming can be used to provide visual feedback of chosen operations and parameters.The user first selects a video, specific processing operations parameters and triggers processing. Subsequently, a preview of the video with the operations applied is presented to the user.

The user can easily chain multiple operations without an impact on the responsiveness of the application. This

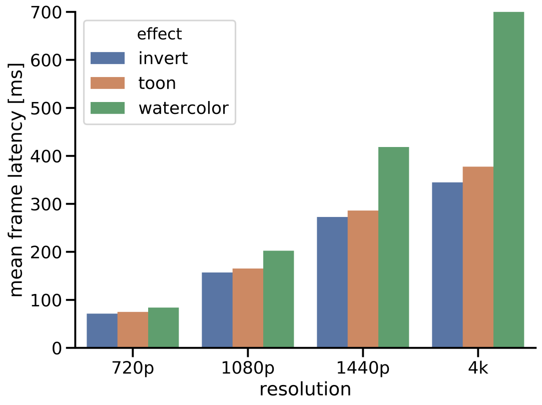


(a) NVIDIA GTX 1050.

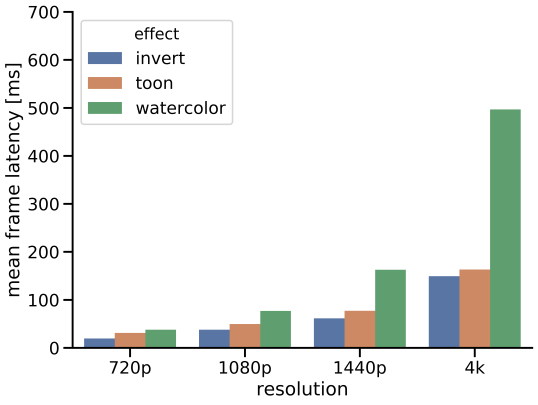


(b) NVIDIA GTX1080.

Figure 2: Throughput measurements in seconds using a consumer desktop (a) and a dedicated server (b).



(a) Latencies for different operations.



(b) Latencies for different spatial resolutions.

Figure 3: End to end frame latencies on a consumer desktop (a) and a dedicated server (b)..

is possible even if the underlying processing framework does not support chaining per se. Once the user has decided on a set of operations and parameters, they can download the result.

Real-time Stream-processing of Videos. This application example demonstrates how a consumer can take existing videos hosted online and process them using our system. The user provides the link to a video hosted on a video streaming platform, chooses an operation to apply to this video, and starts the pipeline to consume the processed video. As the video is processed in a streaming fashion, the user can start watching the video before it is fully processed.

5.2 Performance Evaluation

With respect to the application examples, a performance evaluation using different test data sets and operations are performed.

Test Data and Operations. Different image resolutions were tested with different image-processing techniques to estimate the run-time performance regarding the spatial resolution of a video as well as the complexity of processing techniques. The following common spatial resolutions (in pixels) were chosen: 1280×720 (HD), 1920×1080 (FHD), 2560×1440 (QHD), and 3840×2160 (4K). The source videos had a frame rate of 24 Frames-per-Second (FPS) and a total duration of 150 s each.

In addition to various spatial resolutions, different image processing techniques were tested in order to cover a broad spectrum and obtain variant performance estimates and behavior with respect to different types of processing tasks. These operations range in complexity from simple point based operations (invert) to complex stylization operations (watercolor).

Test Hardware and Setup. To cover a common hardware spectrum, the performance test are conducted on two different test hosts: (1) a desktop PC with commodity hardware with an Intel i5-4690 CPU (4 cores) at 3.5 GHz, 16 GB Random Access Memory (RAM), and

GeForce GTX 1050 Ti GPU 4 GB VRAM as well as a (2) dedicated GPU server with an Intel i7-7700 CPU (4 cores) at 3.6 GHz, 64 GB RAM, and a GTX 1080 GPU 24 GB VRAM.

The videos were streamed in and out of the processor by separate processes running on the same host. All configurations are run 20 times to eliminate external factors from the measurements.

Performance Results and Discussion. As depicted in Fig. 2, the system is capable of processing the given effects in real-time on the consumer device. For high-resolution videos processed with compute-intensive effects, more powerful hardware is required.

As depicted in Fig. 3, the presented approach achieves sub-second latencies for the processing of all given effects. Most of this latency is incurred by the batching of frames on the codec, as is evidenced by the comparatively similar latency over the different effects per resolution. Only the watercolor effect introduces significant latency.

Inspection of the running process shows that we achieve close to 100 % GPU-utilization using our system. This means that our system is capable of fully utilizing the GPU with processing the operations. Whether this utilization represents a relatively high or low throughput depends (1) on the input data, (2) the chosen operations and parameters, and (3) the implementation of the operations in the given processing system. However, the presented approach can not overcome bandwidth limitations inside or between processing nodes, since frames are not preprocessed. This means that the connection to the video source must be sufficiently fast to deliver the video data in real-time. With respect to processing hardware, we assume that each node has the capabilities to process two encoded streams of the desired tile resolution. This includes the encoder and decoder on this node being capable of processing the stream at a frame rate that is greater or equal to the frame rate of the input video.

6 CONCLUSIONS

This paper presents a concept and prototypical implementation to integrate 3rd-party image and video processing software components into services cable of real-time streaming. The system achieves this without the modification or optimization of the specific processing algorithm's implementations. It supports scaling-up and scaling-out to cover different input and processing complexities. This paper further presents use cases in which such an approach is required to facilitate new applications or features for users.

For future work, this system could be improved to be more adaptive to the kind of workload that is imposed. This includes automated provisioning of new nodes to scale-out as well as easy re-use of provisioned nodes to handle multiple tasks without incurring start-up and tear-down costs. In addition thereto, the system's behavior can be runtime-adaptive, thus the appropriate scale can be computed automatically – even for unknown operations or input data complexities. To support this, research with respect to tuning the parallelization process to balance the overhead associated with tiling and the gain in performance must be conducted. To lower latencies in complex processing pipelines, a transitioning from traditional video streaming protocols to new data exchange protocols such as a messaging queue, can be considered.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable feedback. This work was funded by the Federal Ministry of Education and Research (BMBF), Germany, for the "mdViPro" project (01IS15041).

REFERENCES

[1] Riyadh Baghdadi, Jessica Ray, Malek Ben Romdhane, Emanuele Del Sozzo, Abdurrahman Akkas, Yunming Zhang, Patricia Suriana, Shoaib Kamil, and Saman Amarasinghe. Tiramisu: A Polyhedral Compiler for Expressing Fast and Portable Code. *arXiv:1804.10694 [cs]*, 2018.

[2] Carl Boettiger. An Introduction to Docker for Reproducible Research. *SIGOPS Oper. Syst. Rev.*, 49(1):71–79, January 2015.

[3] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink^{TM}: Stream and batch processing in a single engine. *IEEE Data Eng. Bull.*, 38(4):28–38, 2015.

[4] Tobias Dürschmid, Maximilian Söchting, Amir Semmo, Matthias Trapp, and Jürgen Döllner. ProsumerFX: Mobile Design of Image Stylization Components. In *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*, SA '17, pages 1:1–1:8, New York, NY, USA, 2017. ACM.

[5] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[6] Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. Edge-preserving Decompositions for Multi-scale Tone and Detail Manipulation. *ACM Transactions on Graphics*, 27(3):67:1–67:10, 2008.

[7] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image Style Transfer Using Convolutional Neural Networks. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, Los Alamitos, 2016. IEEE Computer Society.

[8] Kaiming He, Jian Sun, and Xiaoou Tang. Guided Image Filtering. In *Proc. European Conference on Computer Vision (ECCV)*, pages 1–14. Springer, 2010.

[9] Alexander Jungmann and Bernd Kleinjohann. Automatic Composition of Service-based Image Processing Applications. In *Proc. IEEE International Conference on Services Computing (SCC)*, pages 106–113. IEEE Computer Society, 2016.

[10] Jeyhun Karimov, Tilmann Rabl, Asterios Katsifodimos, Roman Samarev, Henri Heiskanen, and Volker Markl. Benchmarking distributed stream data processing systems. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1507–1518. IEEE.

[11] Michael Kass and Justin Solomon. Smoothed Local Histogram Filters. *ACM Transactions on Graphics*, 29(4):100:1–100:10, 2010.

[12] Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. State of the 'Art': A Taxonomy of Artistic Stylization Techniques for Images and Video. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):866–885, 2013.

[13] Michael A. Laurenzano, Parker Hill, Mehrzad Samadi, Scott Mahlke, Jason Mars, and Lingjia Tang. Input Responsiveness: Using Canary Inputs to Dynamically Steer Approximation. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '16, pages 161–176. ACM, 2016.

[14] Angel Lagares Lemos, Florian Daniel, and Boualem Benatallah. Web Service Composition: A Survey of Techniques and Tools. *ACM Computing Surveys*, 48(3):33:1–33:41, 2015.

[15] Liming Lou, Paul Nguyen, Jason Lawrence, and Connelly Barnes. Image Perforation: Automatically Accelerating Image Pipelines by Intelligently Skipping Samples. *ACM Trans. Graph.*, 35(5):153:1–153:14, 2016.

[16] Daniel Maier, Biagio Cosenza, and Ben Juurlink. Local Memory-aware Kernel Perforation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, CGO 2018, pages 278–287. ACM, 2018.

[17] Richard Membarth, Oliver Reiche, Frank Hannig, Jurgen Teich, Mario Korner, and Wieland Eckert. HIPA^{cc}: A domain-specific language and compiler for image processing. *IEEE Transactions on Parallel and Distributed Systems*, 27(1):210–224, 2016.

[18] Himadri Nath Moulick and Moumita Ghosh. Medical image processing using a service oriented Architecture and Distributed Environment. *American Journal of Engineering Research (AJER)*, 2(10):52–62, 2013.

[19] Matthias Mueller and Benjamin Pross. *OGC WPS 2.0.2 Interface Standard*. Open Geospatial Consortium, 2015. http://docs.opengeospatial.org/is/14-065/14-065.html.

[20] Ravi Teja Mullapudi, Andrew Adams, Dillon Sharlet, Jonathan Ragan-Kelley, and Kayvon Fatahalian. Automatically Scheduling Halide Image Processing Pipelines. *ACM Transactions on Graphics*, 35(4):83:1–83:11, July 2016.

[21] Ravi Teja Mullapudi, Vinay Vasista, and Uday Bondhugula. PolyMage: Automatic Optimization for Image Processing Pipelines. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, pages 429–443. ACM, 2015. event-place: Istanbul, Turkey.

[22] Mike P. Papazoglou and Willem-Jan Heuvel. Service Oriented Architectures: Approaches, Technologies and Research Issues. *The VLDB Journal*, 16(3):389–415, 2007.

[23] Chris Peltz. Web Services Orchestration and Choreography. *Computer*, 36(10):46–52, October 2003.

[24] Pietro Perona and Jitendra Malik. Scale-space and Edge Detection using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[25] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. Scanner: Efficient video analysis at scale. *ACM Trans. Graph.*, 37(4), July 2018.

[26] Jing Pu, Steven Bell, Xuan Yang, Jeff Setter, Stephen Richardson, Jonathan Ragan-Kelley, and Mark Horowitz. Programming Heterogeneous Systems from an Image Processing DSL. *ACM Trans. Archit. Code Optim.*, 14(3):26:1–26:25, 2017.

[27] Ricardo Queirós and Alberto Simões. SOS - Simple Orchestration of Services. In Ricardo Queirós, Mário Pinto, Alberto Simões, José Paulo Leal, and Maria João Varanda, editors, *6th Symposium on Languages, Applications and Technologies (SLATE 2017)*, volume 56 of *OpenAccess Series in Informatics (OASIcs)*, pages 13:1–13:8, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[28] Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, FrÃ©do Durand, and Saman Amarasinghe. Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation in Image Processing

Pipelines. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '13, pages 519–530. ACM, 2013. event-place: Seattle, Washington, USA.

[29] Mahesh Ravishankar, Justin Holewinski, and Vinod Grover. Forma: A DSL for Image Processing Applications to Target GPUs and Multi-core CPUs. In *Proceedings of the 8th Workshop on General Purpose Processing Using GPUs*, GPGPU-8, pages 109–120. ACM, 2015. event-place: San Francisco, CA, USA.

[30] Mehrzad Samadi, Davoud Anoushe Jamshidi, Janghaeng Lee, and Scott Mahlke. Paraprox: Pattern-based Approximation for Data Parallel Applications. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, pages 35–50. ACM, 2014.

[31] Mehrzad Samadi, Janghaeng Lee, D. Anoushe Jamshidi, Amir Hormati, and Scott Mahlke. SAGE: Self-tuning Approximation for Graphics Engines. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 13–24. ACM, 2013.

[32] Amir Semmo, Daniel Limberger, Jan Eric Kyprianidis, and Jürgen Döllner. Image Stylization by Interactive Oil Paint Filtering. *Computers & Graphics*, 55:157–171, 2016.

[33] Carlo Tomasi and Roberto Manduchi. Bilateral Filtering for Gray and Color Images. In *Proc. International Conference on Computer Vision (ICCV)*, pages 839–846. IEEE, 1998.

[34] Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M. Patel, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, Nikunj Bhagat, Sailesh Mittal, and Dmitriy Ryaboy. Storm@twitter. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 147–156. ACM. event-place: Snowbird, Utah, USA.

[35] Mircea-Florin Vaida, Valeriu Todica, and Marcel Cremene. Service oriented Architecture for Medical Image Processing. *International Journal of Computer Assisted Radiology and Surgery*, 3(3):363–369, 2008.

[36] Joachim Weickert. *Anisotropic Diffusion in Image Processing*, volume 1. Teubner Stuttgart, 1998.

[37] Robert P. Winkler and Chris Schlesiger. Image Processing REST Web Services. Technical Report ARL-TR-6393, Army Research Laboraty, Adelphi, MD 20783-119, 2013.

[38] Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. Real-Time Video Abstraction. *ACM Transactions on Graphics*, 25(3):1221–1226, 2006.

[39] M. Würsch, R. Ingold, and M. Liwicki. SDK Reinvented: Document Image Analysis Methods as RESTful Web Services. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 90–95, 2016.

[40] Marcel Würsch, Rolf Ingold, and Marcus Liwicki. DivaServices - A RESTful web service for Document Image Analysis methods. *Digital Scholarship in the Humanities*, 32(1):i150–i156, 2017.

[41] Ran Xu, Jinkyu Koo, Rakesh Kumar, Peter Bai, Subrata Mitra, Sasa Misailovic, and Saurabh Bagchi. VideoChef: Efficient Approximation for Streaming Video Processing Pipelines. In *2018 {USENIX} Annual Technical Conference ({USENIX} {ATC} 18)*, pages 43–56, 2018.

[42] Xiaoxia Yang. Remotely Sensed Image Processing Service Automatic Composition. State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, 2009.

[43] Rong Yuan, Ming Luo, Zhi Sun, Shuyue Shi, Peng Xiao, and Qingguo Xie. Rayplus: a web-based platform for medical image processing. *J. Digital Imaging*, 30(2):197–203, 2017.

[44] Matei Zaharia, Tathagata Das, Haoyuan Li, Scott Shenker, and Ion Stoica. Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters. In *Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Ccomputing*, HotCloud 2012, page 10, USA, 2012. USENIX Association.

Chebyshev's Method on Projective Fluids

Alexander Sommer[1] alexander.sommer@hs-rm.de

Ulrich Schwanecke[1] ulrich.schwanecke@hs-rm.de

Elmar Schoemer[2] schoemer@uni-mainz.de

[1] Computer Vision and Mixed Reality Group, RheinMain University of Applied Sciences Wiesbaden Rüsselsheim, Germany

[2] Institute of Computer Science, Johannes Gutenberg University Mainz, Germany

ABSTRACT

We demonstrate the acceleration potential of the Chebyshev semi-iterative approach for fluid simulations in Projective Dynamics. The Chebyshev approach has been successfully tested for deformable bodies, where the dynamical system behaves relatively linearly, even though Projective Dynamics, in general, is fundamentally nonlinear. The results for more complex constraints, like fluids, with a particular nonlinear dynamical system, remained unknown so far. We follow a method describing particle-based fluids in Projective Dynamics while replacing the Conjugate Gradient solver with Chebyshev's method. Our results show that Chebyshev's method can be successfully applied to fluids and potentially other complex constraints to accelerate simulations.

Keywords: fluid simulation, constraint-based simulation, projective dynamics, nonlinear optimization, animation

1 INTRODUCTION

The physically plausible simulation of the behavior of various objects has been an important research topic in computer graphics for the past decades. In the early days, methods were adapted from simulations in computational physics, like the Finite-Element-Methods (FEM) [14]. These methods are derived from continuum mechanical principles and therefore offer high accuracy. This high accuracy is needed because the goal of simulations in computational physics or chemistry is to replace real-world experiments. In computer graphics, however, physical simulations are mostly used for special effects in film or for interactive applications. High numerical accuracy is therefore often not needed as long as the objects behave plausibly. This results in new methods that are specially tailored to the needs of computer graphics simulation.

A common simulation method in computer graphics is Position-Based-Dynamics (PBD) [13]. Classical approaches use forces to apply a change of momentum in a physical system, which leads to a position change through numerical integration of velocities and accelerations. In PBD there are specially designed constraints, that enforce a change of position in a quasi-static fashion. Constraints in PBD can be used for simulating a broad variety of objects such as rigid-bodies, soft-bodies, cloth, hair, muscles and even fluids. Unfortunately, this technique has limited accuracy as the constraints are not directly derived from continuum mechanical principles. In addition, a large number of iterations are often required to solve the entire system, as the constraints are projected onto each other.

Projective Dynamics (PD) [3] is a relatively new technique that also uses constraints to enforce position changes. But in contrast to PBD, these constraints are highly nonlinear energy potentials directly derived from deformation energies in continuum mechanics. PD bridges the gap between the physically correct FEM and the fast, reliable and robust PBD. Since the emergence of PD, research has continued to expand the number of objects that can be simulated, improve convergence rate and increase speed. The Chebyshev Semi-Iterative approach [17] is one method of accelerating PD by using concepts from linear algebra to efficiently solve the fundamentally nonlinear system in PD. This approach has been successfully applied to deformable bodies, which have a relatively linear behavior.

In this work, we demonstrate the acceleration potential of Chebyshev's method on fluids, which in contrast to rigid or even deformable bodies are particularly complex and result in far more nonlinear dynamic systems. To the best of our knowledge, we are the first to successfully accelerate fluid simulation with Chebyshev's method. We use the approach proposed in [18] for simulating fluids in PD, borrowing Smoothed Particle Hydrodynamics (SPH) methods. Our results also show the potential of Chebyshev's method for other complex constraints in PD (e.g. hair or cloth).

2 RELATED WORK

In the following, we briefly introduce the concept of Projective Dynamics and its extension to fluids. In particular, we also present the notation used, which is required in the section discussing the Chebyshev method.

2.1 Projective Dynamics

PD is a constraint-based simulation method using an implicit integration scheme while projecting the constraints in an alternating Gauss-Seidel fashion onto a common solution space. An object can be described as a mesh consisting of m vertices. Each vertex $(\mathbf{x}_i, \mathbf{v}_i), i \in \{1 \dots m\}$ is characterized by a position $\mathbf{x}_i$ and a velocity $\mathbf{v}_i$. All positions and velocities of the system can be stored inside two matrices $\mathbf{X} \in \mathbb{R}^{m\times 3}$ and $\mathbf{V} \in \mathbb{R}^{m\times 3}$. The implicit time integration from time t_n to t_{n+1} of these characteristics leads to:

$$\begin{aligned}\mathbf{X}_{n+1} &= \mathbf{X}_n + h \cdot \mathbf{V}_{n+1} \\ \mathbf{V}_{n+1} &= \mathbf{V}_n + h \cdot \mathbf{M}^{-1}\mathbf{F}(\mathbf{X}_{n+1})\end{aligned} \tag{1}$$

with h being the simulation time-step and $\mathbf{M} \in \mathbb{R}^{m\times m}$ a constant mass matrix. Forces $\mathbf{F}$ acting on the system can be characterized as internal or external. External forces $\mathbf{f}_{ext}$, like gravity, are held constant during the simulation. Internal forces can be understood as material properties and are expressed by strictly position-depended scalar energy potentials $\mathbf{f}_{int}(\mathbf{X}) = -\sum_i \nabla W_i(\mathbf{X})$. As shown in [11], for finding the unknown $\mathbf{X}_{n+1}$, the coupled system (1) can be converted into the optimization problem

$$\min_{\mathbf{X}_{n+1}} \left(\frac{1}{2h^2} \left\| \mathbf{M}^{\frac{1}{2}} (\mathbf{X}_{n+1} - (\mathbf{X}_n + h\mathbf{V}_n + h^2\mathbf{M}^{-1}\mathbf{f}_{ext})) \right\|_F^2 + \sum_i W_i(\mathbf{X}_{n+1}) \right). \tag{2}$$

One of the key points in PD is the replacement of the generally highly nonlinear energy function $W_i(\mathbf{X})$ by specially designed constraints. A fulfilled constraint describes the rest state or undeformed configuration. [3] showed that each constraint can be solved independently with an auxiliary position variable $\mathbf{P}$ by projection to a common constraint manifold minimizing the distance between the current and the projected positions, i.e.

$$W(\mathbf{X}_{n+1}) = \min_{\mathbf{P}} \frac{w}{2} \left\| \mathbf{C}\mathbf{X}_{n+1} - \mathbf{D}\mathbf{P} \right\|_F^2 + \delta_C(\mathbf{P}),$$

with constraint depended constant matrices $\mathbf{C}$ and $\mathbf{D}$, an indicator function δ_C that evaluates if the constraint is fulfilled and a weighting factor w.

For solving the whole problem, two minimization steps are necessary: a local and a global solve. During the local solve the minimization is executed over the auxiliary variables $\mathbf{P}_i$, while the positions $\mathbf{X}_{n+1}$ are kept fixed. This means that vertex positions are projected to their closest positions on the constraint manifold. Since the auxiliary variables $\mathbf{P}_i$ are independent for each constraint this can be done in parallel. In the second minimization step the already projected auxiliary variables $\mathbf{P}_i$ are fixed and (2) is minimized over the vertex positions $\mathbf{X}_{n+1}$. This finds the configuration with the best compromise between all constraints. Since the unknown $\mathbf{X}_{n+1}$ is quadratic in all terms of (2), the minimization is equivalent to solving the linear system

$$\left(\frac{\mathbf{M}}{h^2} + \sum_i w_i \mathbf{C}_i^T \mathbf{C}_i \right) \mathbf{X}_{n+1} = \frac{\mathbf{M}}{h^2} (\mathbf{X}_n + h\mathbf{V}_n) + \sum_i w_i \mathbf{C}_i^T \mathbf{D}_i \mathbf{P}_i + \mathbf{f}_{ext}. \tag{3}$$

Since the objective function is bounded for an arbitrary choice of constraints, each iteration is guaranteed to weakly decrease energies. Therefore the vertex positions $\mathbf{X}_{n+1}$ converge iteratively towards the state of minimal energy, without the need for any safeguards.

2.2 Projective Fluids

For the simulation of fluids in a constraint-based framework, a fluid density constraint can be used, which was first proposed for position based simulations in [9]. Later, [18] translated the methodology for the use in PD. They defined the (incompressible) fluid constraint as the state in which internal pressures force the continuum into a resting state of constant density ρ_0, i.e.

$$W_i(\mathbf{X}) = \frac{1}{\rho_0} \left(\sum_k m_k K(\mathbf{x}_i - \mathbf{x}_k, l) \right) - 1, \tag{4}$$

where local densities are calculated by the standard SPH density estimator as an interpolation of neighboring particles by an SPH Kernel function K with a smoothing kernel length l.

The local solve is done by finding a position correction vector $\Delta\mathbf{P}$ in the projected positions $\mathbf{P}$ that satisfies the constraint $W(\mathbf{P} + \Delta\mathbf{P}) = 0$. A first-order approximation of the constraint with a restriction of $\Delta\mathbf{P}$ in the direction of the constraint gradient that conserves the linear and angular momenta according to D'Alembert's principle leads to exactly one scalar Lagrange multiplier λ that solves the system (see [2])

$$W(\mathbf{P} + \Delta\mathbf{P}) \approx W(\mathbf{P}) + \lambda \nabla W^T \nabla W.$$

The gradient of the SPH density estimator is just defined by the gradient of the kernel function. For an arbitrary particle k two cases need to be distinguished, i.e. whether k is a neighboring particle or not and thus one gets

$$\nabla_{\mathbf{p}_k} W_i = \frac{1}{\rho_0} \begin{cases} \sum_j \nabla_{\mathbf{p}_k} K(\mathbf{p}_i - \mathbf{p}_j, l) & \text{if} \quad k = i \\ -\nabla_{\mathbf{p}_k} K(\mathbf{p}_i - \mathbf{p}_j, l) & \text{if} \quad k = j \end{cases}$$

Since the result is only a first order approximation the fluid constraint (4) in general will not vanish. Therefore iterative updates $\mathbf{P}_{\zeta+1} = \mathbf{P}_\zeta + \Delta\mathbf{P}_\zeta$ are used to converge $\mathbf{P}_\zeta$ towards the correct projected positions $\mathbf{P}$, where the (4) vanishes. The iteration can be stopped when the constraint is smaller than a predefined constant, i.e. $W(\mathbf{P}_\zeta + \Delta\mathbf{P}_\zeta) < \varepsilon$. Usually after three to four iterations $\varepsilon < 10^{-14}$ is reached, which is sufficient [18].

3 CHEBYSHEV'S METHOD

The Chebyshev method is a semi-iterative approach to solve a linear system $\mathbf{Ax} = \mathbf{b}$, like the global solve (3) in PD. Contrary to the Conjugate Gradient method (CG) proposed in [18], it doesn't depend on inner products, which offers more potential for parallelization [17], since less safeguards are needed to prevent race conditions. The key idea behind the method is to obtain better results from an iterative solution, by blending in previous results as a convex combination with blending weights $\nu_{i,k} > 0$, $\sum_{i=0}^{k} \nu_{i,k} = 1$. These blending coefficients $\nu_{i,k}$ can be determined by minimizing the error

$$\mathbf{e}_k = \mathbf{x}'_k - \mathbf{x} = \sum_{i=0}^{k} \nu_{i,k}(\mathbf{x}_i - \mathbf{x}) \tag{5}$$

where $\mathbf{x}'_k$ is the total iterative solution after k iterations, $\mathbf{x}$ is the exact solution and

$$\mathbf{x}_i = \mathbf{Q}^{-1}\left(\mathbf{R}\mathbf{x}_{i-1} + \mathbf{b}\right) \tag{6}$$

is the result of the i-th iteration, where $\mathbf{A} = \mathbf{Q} - \mathbf{R}$ and $\mathbf{Q}$ being an easily invertible matrix (e.g. a diagonal matrix). Substituting (6) into (5) results in

$$\begin{aligned}\mathbf{e}_k &= \sum_{i=0}^{k} \nu_{i,k}\left(\mathbf{Q}^{-1}(\mathbf{R}\mathbf{x}_{k-1} + \mathbf{b}) - \mathbf{x}\right) = \sum_{i=0}^{k} \nu_{i,k}\mathbf{Q}^{-1}\mathbf{R}\mathbf{e}_{k-1} \\ &= \sum_{i=0}^{k} \nu_{i,k}\left(\mathbf{Q}^{-1}\mathbf{R}\right)^i \mathbf{e}_0 = p_k(\mathbf{Q}^{-1}\mathbf{R})\mathbf{e}_0\end{aligned}$$

with $p_k(\mathbf{Y}) = \sum_i \nu_{i,k}\mathbf{Y}^i$ being a polynomial.

A polynomial of a matrix $\mathbf{Y}$ can be minimized by minimizing $\|p_k(\mathbf{Y})\|_2 = \max_{\lambda_i} |p_k(\lambda_i)|$, the analog polynomial function for the scalar Eigenvalues λ_i of matrix $\mathbf{Y}$. Finding the eigenvalues of a large linear system is not practical, but the spectral radius ρ gives a measure of the range in which the eigenvalues must lie. This means t hat p_k can be minimized over a range $x \in [-\rho, \rho]$, i.e.

$$p_k(x) = \arg\min\left(\max_{-\rho \le x \le \rho} |p_k(x)|\right). \tag{7}$$

[17] showed that $p_k(x) = T_k\left(\frac{x}{\rho}\right) / T_k\left(\frac{1}{\rho}\right)$ with the Chebyshev polynomials of the first kind $T_k(x)$ (for more details on Chebyshev Polynomials see e.g. [7]) is a solution to (7). Furthermore, [17] showed in detail that, based on relations for these polynomials, an iterative solution with a minimized error term only depending on the range limiter ρ is given by

$$\mathbf{x}'_{k+1} = \omega_{k+1}\left(\mathbf{Q}^{-1}(\mathbf{R}\mathbf{x}'_k + \mathbf{b}) - \mathbf{x}'_{k-1}\right) + \mathbf{x}'_{k-1} \tag{8}$$

with a recursive resulting factor $\omega_{k+1} = \frac{4}{4-\rho^2\omega_k}$ $\forall i \ge 1$ and $\omega_1 = 1$.

The idea of using the Chebyshev approach for PD is to replace the global solve with the Chebyshev Formula (8) using the results of the previous iterations for $\mathbf{x}'_k$ and $\mathbf{x}'_{k-1}$. Furthermore, the matrix $\mathbf{Q}$ is chosen as the diagonal matrix of the projective system matrix. For a linear system, ρ is well defined by the spectral radius of $\mathbf{Q}^{-1}\mathbf{R}$. For PD where the combination of local and global solves forms a fundamentally nonlinear system this definition of ρ can't be used any longer and an alternative approach is needed. [17] proposed to use a fixed ρ for the whole simulation. This ρ can be found in two steps before the actual simulation starts. First it is initialized by the error rate in the last iteration K of a test run, i.e. $\rho = \frac{\|\mathbf{e}_K\|_2}{\|\mathbf{e}_{K-1}\|_2}$. After that, ρ is modified in a couple of further test runs to minimize the convergence rate. When ρ is overestimated the results begin to oscillate and when ρ is underestimated the results converge slower.

4 IMPLEMENTATION

Algorithm 1 PD solve from t = $n \to$ t = $n+1$

```
1:  $\mathbf{X}^{(0)} \leftarrow$ InitialPositionGuess($\mathbf{X}_n$, $\mathbf{V}_n$)
2:  for iterations $k = 0 \dots K$-1 do
3:      for all constraints $i$ do
4:          $\mathbf{P}_i \leftarrow$ ProjectConstraint($\mathbf{C}_i$, $\mathbf{X}^{(k)}$)    //(4)
5:      end for
6:      $\tilde{\mathbf{X}}^{(k+1)} \leftarrow$ SolveLinearSystem($\mathbf{X}_n$, $\mathbf{V}_n$, $\mathbf{P}_i$) //(3)
7:      $\mathbf{X}^{(k+1)} = \omega_{k+1}\left(\tilde{\mathbf{X}}^{(k+1)} - \mathbf{X}^{(k-1)}\right) + \mathbf{X}^{(k-1)}$ //(8)
8:  end for
9:  $\mathbf{X}_{n+1} = \mathbf{X}^{(K)}$
10: $\mathbf{V}_{n+1} = (\mathbf{X}_{n+1} - \mathbf{X}_n)/h$
```

We implemented a fluid simulation framework in C++ 11, using OpenMP for parallelization and the Eigen library for mathematical operations. The parallelization is used, among other things, to project fluid constraints in parallel onto their manifolds, while updating the solution vector atomically to prevent race conditions, as described in [18]. We used parallel Spatial Hashing by [8] for finding particle neighbors (see also [15]) and a cubic b-spline kernel [12] as the SPH kernel for interpolating field quantities from neighboring particles. Boundary and object collision handling is done by surface sampled boundary particles as proposed in [1]. The particle sampling on arbitrary surfaces is done with an improved parallelized version

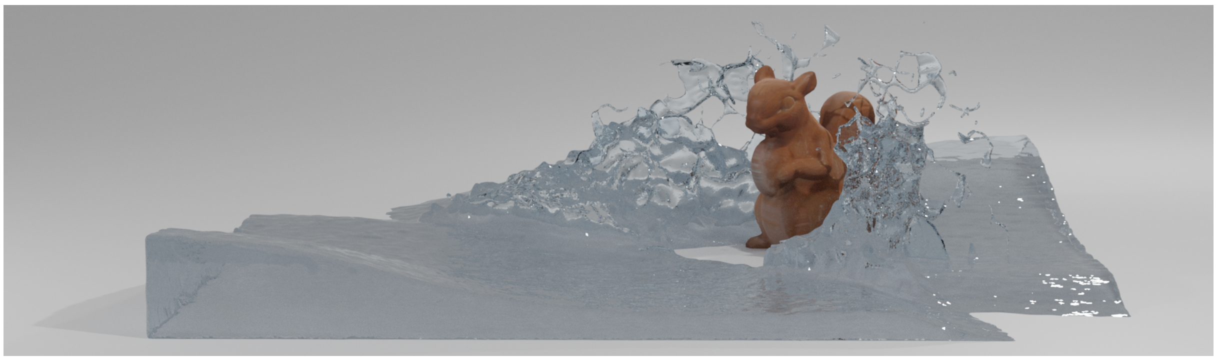


Figure 1: A diagonal double dam break in a squared bounding box with one collision object and 298k fluid particles.

Solver	*Dam Break* 30k	*Dam Break* 88k	*2Dam Break* 128k	*Squirrel* 298k
CG 1 It.	73ms	277ms	484ms	883ms
CG	94ms	616ms	889ms	2558ms
LDLT	**88ms**	306ms	486ms	1057ms
Jacobi 1 It.	89ms	**276ms**	476ms	1163ms
Jacobi	101ms	381ms	555ms	1225ms
Chebyshev	89ms	295ms	**459ms**	**995ms**

Table 1: Timing results for different scenarios. Fastest runtimes without artifacts in bold.

of Poisson disk sampling [4]. The simulation time-step size is restricted by the CFL condition [5]. The pseudo-code for one PD solve, also referred to as one time-step from a time t = n to t = $n+1$ is shown in Algorithm 1. The corresponding formulas from this paper are added in parentheses behind code lines.

5 RESULTS

In this section, we present the simulation results and discuss the time advantage over previously used solving methods. We use four different simulation scenarios to emphasize our statements. The first one is a dam break, which is a classical test case in fluid simulation. In this setup, a block of water containing 30k particles flows under the influence of gravity in a rectangular bounding box. For the second scenario we used the same setup (see Fig. 2), but this time with 88k particles. The third setup, the double dam break, involves two blocks of water in a rectangular bounding box on opposite sides containing a total of 128k particles (see Fig. 3). In the last setup, two blocks of water with each 149k particles in opposing corners of a squared bounding box are set loose, while a squirrel as a complex collision object stands inside the bounding box (see Fig. 1).

As mentioned in section 3 the first thing that needs to be done is finding an equivalent to the spectral radius ρ that minimizes the convergence rate. Figure 4 exemplary displays the averaged runtime of the first 80 simulation time-steps for different values of ρ, for the second scenario. The minimum is found in $\rho = 0.925$. Furthermore, it shows that for an overestimated ρ the convergence rate slows significantly, probably due to oscillating results, as proposed in section 3.

Figure 2: A dam break scenario in a rectangular bounding box with 88k particles.

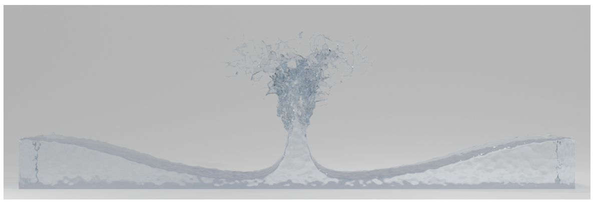


Figure 3: A double dam break scenario in a rectangular bounding box with two bodies of water on opposite sides with a total of 128k particles.

For evaluating the quality of the solver we took the average runtime during one time-step. Since all compared solvers have different convergence behavior, we used a common criteria for stopping the local/global iteration. It is stopped when the maximum Euclidean distance between two corresponding particle locations in the current and the previous iteration is bellow a fixed threshold. We compared our solver to other commonly used solvers (see Table 1). For the Conjugate Gradients (CG) solver, we used the matrix-free implementation proposed in [18]. As an iterative method, we implemented a standard Jacobi method [16]. We also compared our results to a direct solving method the LDLT Cholesky decomposition [6]. For this method, we used Eigen's `SimplicialLDLT` with sparse matrices. We tested the CG and the Jacobi method until convergence for the global solve (as stated in [18]) and for a single global solve iteration only. While the CG method with only one iteration was often the fastest in our tests,

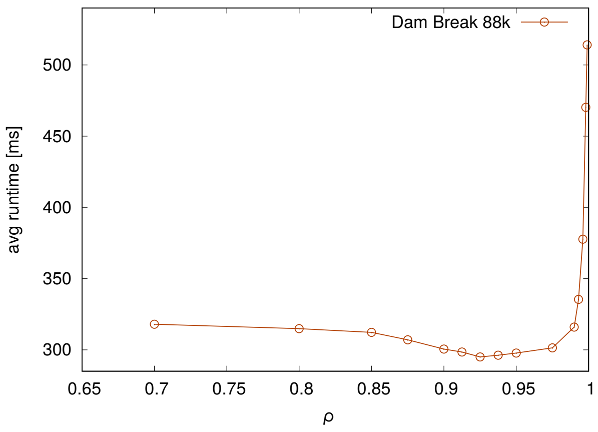


Figure 4: Average runtime for the dam break scenario with 88k particles for different values for the equivalent to the spectral radius ρ.

it produced artifacts and therefore isn't feasible. The Chebyshev method doesn't generate any artifacts and is the fastest method of all tested fully working methods for higher particle counts. This shows, that even for complex and highly nonlinear constraints, like the fluid constraint, the Chebyshev method accelerates the simulation. Compared to [18]'s CG solver (until convergence), we were able to reduce solving times between 41% and 52% without creating undesired visual side effects. All results have been produced on a MacBookPro with an Intel Core i7-4980HQ processor with 4 cores (8 threads), clocked at 2.80GHz and 16GB of RAM.

6 FUTURE WORK

In the future, we want to further investigate the acceleration potential of our solving approach on massively parallel hardware like the GPU. Since the approach does not depend on inner products we believe that there is a strong potential of increasing the performance even more. Furthermore, we plan on testing the solver in more complex simulation setups with different non-fluid constraints. Additionally, it would be interesting to see if the small substep method from [10] for extended position-based dynamics (XPBD), could also further improve Chebyshev's method in PD.

REFERENCES

[1] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.*, 31(4):62:1–62:8, July 2012.

[2] Jan Bender, Matthias Müller, Miguel A. Otaduy, Matthias Teschner, and Miles Macklin. A survey on position-based simulation methods in computer graphics. *Comput. Graph. Forum*, 33(6):228–251, September 2014.

[3] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4):154:1–154:11, July 2014.

[4] John Bowers, Rui Wang, Li-Yi Wei, and David Maletz. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph.*, 29(6):166:1–166:10, December 2010.

[5] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100:32–74, 1928.

[6] Dariusz Dereniowski and Marek Kubale. Cholesky factorization of matrices in parallel and ranking of graphs. pages 985–992, 01 2003.

[7] Gene H. Golub and Richard S. Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order richardson iterative methods. *Numerische Mathematik*, 3(1):147–156, December 1961.

[8] Markus Ihmsen, Nadir Akinci, Markus Becker, and Matthias Teschner. A parallel sph implementation on multi-core cpus. *Comput. Graph. Forum*, 30:99–112, 03 2011.

[9] Miles Macklin and Matthias Müller. Position based fluids. *ACM Trans. Graph.*, 32(4):104:1–104:12, July 2013.

[10] Miles Macklin, Kier Storey, Michelle Lu, Pierre Terdiman, Nuttapong Chentanez, Stefan Jeschke, and Matthias Müller. Small steps in physics simulation. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '19, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. Example-based elastic materials. *ACM Trans. Graph.*, 30(4):72:1–72:8, July 2011.

[12] J. Monaghan and C. Lattanzio. A refined method for astrophysical problems. *Astronomy and Astrophysics*, 149:135–143, 07 1985.

[13] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109 – 118, 2007.

[14] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 137–146, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[15] Juraj Onderik. Efficient neighbor search for particle-based fluids. *Journal of the Applied Mathematics Statistics and Informatics (JAMSI)*, 2, 01 2007.

[16] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, USA, 2nd edition, 2003.

[17] Huamin Wang. A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph.*, 34(6):246:1–246:9, October 2015.

[18] Marcel Weiler, Dan Koschier, and Jan Bender. Projective fluids. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, pages 79–84, New York, NY, USA, 2016. ACM.

Topological-based roof modeling from 3D point clouds

Dobrina Boltcheva[1], Justine Basselin[1,2], Clément Poull[1], Hervé Barthélemy[2], Dmitry Sokolov[1]
[1]Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France
[2]RhinoTerrain, Nancy, F-54000, France

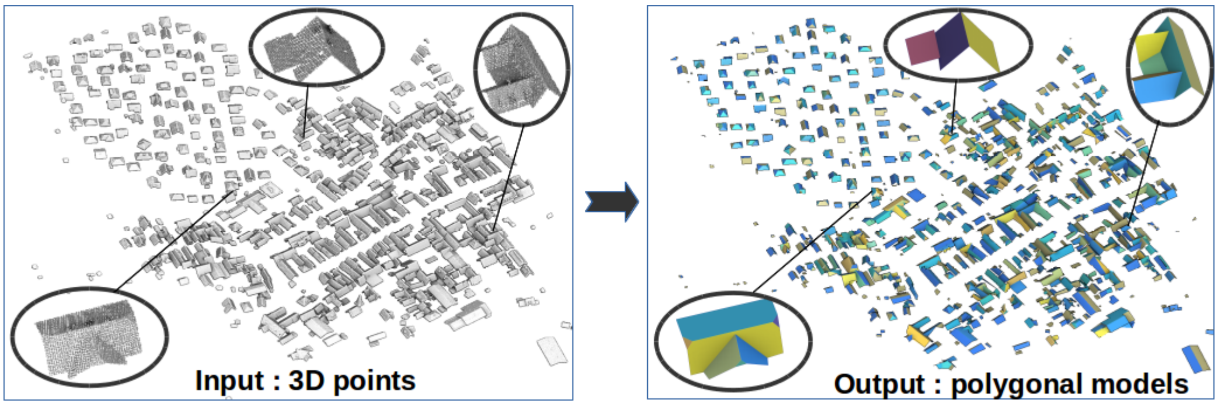


Figure 1: We address the problem of constructing polygonal 3D roof models from previously classified LIDAR point data.

Abstract

Automatic extraction of building roofs from remote sensing data is important for many applications including 3D city modeling, urban planning, disaster management, and simulations. In this paper, we propose an automatic workflow for roof reconstruction by polygonal models from classified high-density LIDAR data. Roof planes are initially delineated by a segmentation algorithm combining a robust Hough-based normal estimator and a region growing strategy. Then, each roof is modeled by a 2D α-shape mesh which is used to discover not only building outline but also all ridges defined by intersecting roof planes, without any geometrical calculations. The mesh directly encodes the topological relations between neighboring planes which allows us to build the final polygonal model straightforwardly. This topological approach makes our solution more simple and robust than existing methods which mostly extract the intersection lines by means of geometrical computations. Experimental results show that the proposed workflow offers a high success rate for extraction at plane level (94% completeness, 92.7% correctness, 90.8% quality) when LIDAR point density is sufficiently high.

Keywords

automatic roof modeling, feature extraction, topology graph, polygonal model, 3D point cloud

1 INTRODUCTION

The digitization of real objects is increasingly used in fields such as urban planning, architecture, disaster management and homeland security. Acquisition tools such as airborne light detection and ranging (LiDAR) scanners make it possible to produce digital representations of entire cities in the form of 3D point clouds sampling the surfaces of the objects in the environment.

Despite the high degree of maturity achieved by digitization techniques, effective computational solutions for preprocessing and reconstruction from these measurements are rare and ill-adapted to the complexity of the environment (complex building structures and entire cities).

Today, the process of creating a digital model from such data is long, tedious and essentially manual. In this reverse engineering process, the human operator manually draws 3D model elements as close as possible to the point cloud.

Although significant effort has been put into the development of automatic and semi-automatic methods, which are currently appearing on the market, no solution proposed so far meets all industrial requirements in terms of precision, accuracy and efficiency. This is because the reconstruction of 3D building models is a complex task that requires a workflow of several processing steps such as classification, outline extraction, segmentation, feature recognition, hypothesis generation and verification, geometric modeling and construction, adjustment and refinement.

In addition, the reconstructed models must respect a number of structural constraints (planarity of roof segments, horizontal roof ridges, symmetry, etc.), which cannot be retrospectively integrated into existing solutions.

Despite the acquired knowledge, there is still a significant number of unsolved problems coming from: gaps in the data (due to occlusions or unwanted reflections and absorption); noise and outliers; limited resolution and variable point density; high variability and complexity of building shapes in urban areas, to name a few.

In this work, we address the particular problem of constructing (creating) polygonal 3D roof models from previously classified LIDAR point data. Our method follows a pure data-driven approach and is fully automatic, except for the parameters which must be chosen with care by a human operator. The proposed method strives for building roof models with a level of detail (LOD2), as defined in the CityGML standard [18], e.g. detailed roof structures but without superstructures (such as chimneys, dormers, *etc.*). Moreover, each individual roof is assumed to be a set of flat planes.

A typical workflow of data-driven reconstruction methods consists of the following three steps:

1. Building points are aggregated to planar patches (segments) which represent roof facets.
2. The resulting segments are then combined to extract building modeling features or cues such as intersection and step lines, building outlines, corners, simple surface primitives, *etc.*.
3. Finally, 3D building models are constructed based on the extracted modeling features and subsequently regularized.

We are following this pipeline, improving steps 2 and 3. Our originality comes from the fact that we fix in priority the topological properties of the resulting model, which are known to be difficult to discover using geometrical approaches.

In addition, this combinatorial approach makes it easier to handle the step edges corresponding to height jumps.

2 RELATED WORK

Numerous building reconstruction approaches have been proposed in the last two decades and are presented in various surveying articles [37, 19, 8].

The model-driven approaches try to fit certain shapes to the data, while the data-driven approaches try to extract shapes present in the data. Although the model-driven methods are robust, their performance is limited to known models [46, 51, 50]. The data-driven methods work, in theory, for any rectilinear building shapes.

Here, we provide a brief overview of some data-driven approaches that are appropriate to our work; the sections correspond to the above pipeline.

2.1 Segmentation of planar patches

We can classify the (data-driven) building reconstruction methods into four categories w.r.t the approach do detect planar regions in point clouds:

Hough transform [20] is one of the earliest methods used for building reconstruction [27], and it has been enhanced and tuned many times [21, 32, 28].

RANSAC is another method that is applied frequently [16, 9]. It tends to be more robust than Hough transform. A comparison between these methods is presented in [45] for automatic segmentation of planar areas from point clouds.

Region growing gains its place thanks to its simplicity and efficiency. In contrast to Hough transform and RANSAC, it is a local segmentation technique. In [2] surface growing is applied to a regularized raster of building points to construct planar segments by a cell aggregation technique. Further surface growing based segmentation methods in the context of 3D building reconstruction are, for example, used in [4, 1, 44, 52].

Another viable (but costly) option is to perform a *global optimization*. In [26], flat roof segments are determined by minimizing an energy function formulated as a multiphase level set.

2.2 Model generation

After a set of segments has been determined, modeling cues can be extracted for the subsequent construction of 3D building models. For this, *intersection lines*, *step lines*, and *building outlines* are frequently extracted. Although building outlines can be considered as a special type of step lines, they are often determined in a separate processing step. There are several approaches that focus on generating a construction draft and simplifying or regularizing it. These approaches are, for example, based on RANSAC [23], α-shape [36, 5, 38], structured grids [52, 44], or line simplification such as Douglas-Peucker [12].

For the detection of step edges, height discontinuities between adjacent segments are searched in [48]. In [41], step edges are determined based on statistical tests and robust estimation. A so-called compass line filter (CLF) is proposed in [17]. It determines the local edge orientation for each step edge.

The ridge lines are usually directly obtained by the intersection of two planes that are derived from a pair of adjacent segments. Other modeling cues are, for example, extracted in [33]. Here, segments are enlarged to roof faces by intersecting all segments, regardless of their adjacency to each other. An approach to detect modeling cues in form of ridge lines utilizing RANSAC is presented in [15].

Most data-driven reconstruction methods generate a 3D polyhedral model directly based on the modeling cues, [41, 32, 36, 52, 49]. For this, the extracted lines are extended and connected with each other so that each roof surface is bounded by a closed sequence of connected line segments. Further faces with a vertical orientation are then added at step edges so that each edge in the polyhedral model becomes part of two surfaces defining line segment sequences. The implementation of such a direct polyhedral model generation method often comes, however, with quite a few problems because there are usually some ambiguities how the line segments are to be connected to guarantee the planarity of each polyhedral face without any gaps in between. There are methods using 2.5D triangulation [11], but it does not guarantee the correct topology of the model (step edges).

2.3 Model regularization

Since building models derived from data-driven reconstruction approaches resemble very closely the (imperfect) input data, several regularization and optimization methods have been developed. For this, most of the methods described above incorporate the main orientation of the building and support orthogonal and parallel structures.

An adjustment model that considers the building topology to improve the shape of a building model is, for example, proposed in [40]. In [52], global regularities are incorporated during the construction. This includes orientation and placement regularities between two roof surfaces, parallelism and orthogonality between building outlines and the normal of their owner planes, and regularities between two boundary edges in terms of their height and position. In order to automatically determine independent and consistent constraints, a greedy algorithm is proposed in [6] while Grobner bases are used in [29]. In [43] and [24], a refinement of building models is proposed based on aerial images from which building edges are detected and incorporated in the reconstruction process. An implicit regularization process in the framework of MDL in combination with hypothesize and test (HAT) is, for example, proposed in [22].

3 PROPOSED METHOD

Our method takes as input a set of clusters of 3D points where each cluster corresponds to the roof of an individual building or a set of adjacent buildings. It provides a 3D geometric and topological roof model where each planar patch is represented by a flat polygon and these polygons are consistently sewn together. These models are suitable for the subsequent reconstruction of polyhedral buildings by geometric extrusion or as an initialization for model-based fitting approaches.

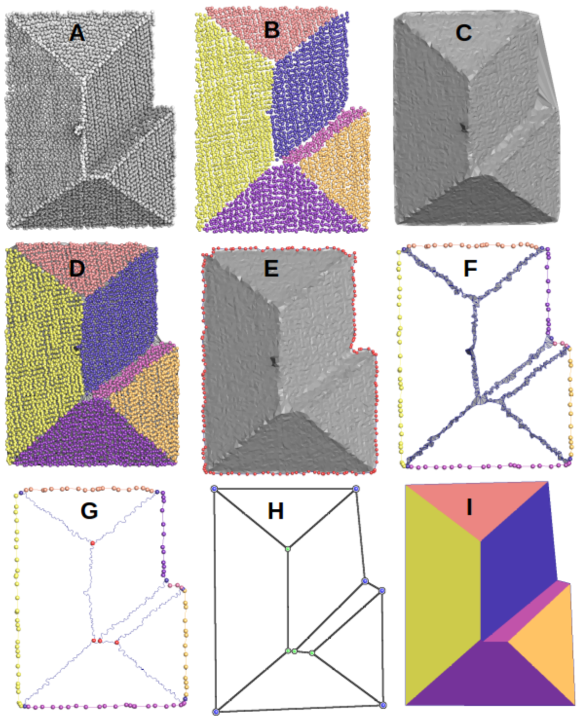


Figure 2: Detailed workflow. A: input points; B: delineated planar segments; C: Delaunay triangulation; D: α-shape; E: roof outline; F: ridge lines triangles; G: raw topological graph; H: simplified graph; I: polygonal mesh.

We follow the pipeline depicted in §1:

1. The planar roof segments are first extracted from each cluster using a robust Hough-based normal estimator and a region growing approach.

2. Then, each cluster of 3D points is projected to the xy-plan and its α-shape is built from its Delaunay triangulation in 2D. The initial building outline, in terms of edges, is extracted from the outmost triangles. We extract roof ridge lines and corners directly from the mesh using the set of triangles belonging to two or three segments. This approach allows us to build at first an accurate topological model (graph) where each vertex corresponds to the intersection of exactly three planar roof patches. Each edge corresponds to the intersection of adjacent planes and each facet corresponds to a roof plane.

3. Finally, we regularize the model by detecting the step edges and ensuring the polygons planarity.

The rest of the section is organized as follows: §3.1 describes the plane detection step, then §3.2 explains the model generation step, and, finally, §3.3 detailed the model regularization step. Figure 2 provides a visual representation of the method as a number of different entities computed during the processing (A to I).

3.1 Plane detection

Our method for plane detection processes each building cluster individually and takes into account the coordinates of the input points and the estimated surface normal vectors. We used a robust Hough-based algorithm provided by Boulch [7] which approximates the tangent plane at each point by considering its 30-neighborhoods. This method is particularly well suited to our data because it does not smooth the normal vectors at the intersection lines between adjacent planes.

Then, a *Region Growing* algorithm, provided by the Point Cloud Library [35], is used to merge the points that are close enough in terms of a *smoothness constraint* defined as the deviation between normals of the points. In order to compute the Cartesian equation of each detected planer patch, we compute the plane normal vector as the average of the normals of its points.

At the end of this step, each building cluster has its points labeled with a value corresponding to the planar patch to which it belongs, as shown in Fig.2-B. In addition, we keep the computed plane equations, as a text file, which will be used in a next processing step.

3.2 Model generation

Based on the planar patch segmentation performed in the previous step, this step deals with the construction of a roof *shape descriptor* in the form of a graph consisting of the *roof's outer edge*, the *ridge lines* and their mutual intersections, referred to as *corners*. It essentially involves a surface mesh generation algorithm allowing to build a triangular mesh connecting all the input points which can then be used to extract the coarse roof model. This initial model will be geometrically optimized during the last step, in order to deliver a valid polygonal 3D roof model.

First, we project the input 3D points to the xy-plane by ignoring the z coordinate. Then, we build the *2D Delaunay Triangulation* of the points which is a mesh of their *convex hull* and can be embedded in 3D, as shown in Fig.2-C.

Since the building footprints are rarely convex polygons, this initial mesh has to be *carved* from outside in order to remove the triangles spanning the concave area. This brings us exactly to the definition of the α-shape

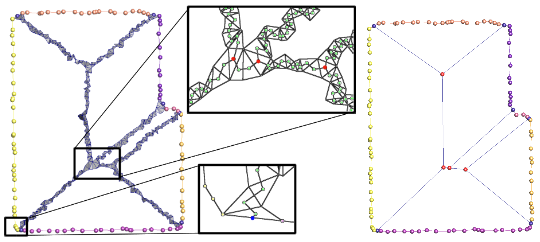


Figure 3: On the left, dual ridge edges. The *corner* vertices are in red, the *connection* vertex is in blue. On the right, simplified ridges by removing *ordinary* vertices.

of the points which is usually computed from the Delaunay mesh by removing the triangles having an edge longer than the parameter $\alpha > 0$, [14]. Indirectly we are performing a Delaunay sculpting approach as introduced in [13] and [30].

In practice, we have implemented an erosion and boundary discovery algorithm. It stats by constructing the convex roof boundary as the list of outmost edges (*i.e.* incident to only one triangle in the Delaunay mesh). Then, it considers the shortest edge and checks if it is smaller than the threshold value α. If yes, the edge is kept and the algorithm continues with one of its adjacent edges. Otherwise, the edge is removed from the boundary list and is replaced by the two other edges of the current triangle. The algorithm stops when all the edges of the boundary list have been checked.

The most critical aspect of this algorithm consists in the selection of an optimal value for the parameter α, since it depends directly on point density and the desired level of detail in the boundary extraction. Although there exist a recent attempt [38] to incorporate the density variation information in the algorithm in the particular context of building modeling.

At this stage of the processing, we have a dense triangular mesh which perfectly fits to the input data, as depicted in Fig.2-D. Note that we keep the labels at the vertices computed at the previous stage. They are shown in different colors and indicate to which plane each vertex belongs to. This information will be used to discover the ridge lines and the corners between planes of the polygonal model.

In addition, the mesh edges of the concave border are also tagged with a specific value, their vertices are shown in red in Fig.2-E. We assume that the building footprint (outline) has only one connected component. Otherwise, processing stops and the input building cluster must be refined in order to correctly separate the various connected components.

In deferential geometry, the ridge lines are defined as the locations of local extrema of the principal curvature of the surface, whereas their intersecting *corners* are locations where the principal curvature vanishes.

In our particular case, the ridge lines are defined by the intersection of two adjacent planar patches, while the corners corresponds to the intersections of exactly three of them. These topological features are easily accessible from the mesh as the set of triangles having more than two different vertex labels, as shown in Fig.2-F.

This simple observation, allows us to design a robust combinatorial algorithm in 2D which essentially consists in a simple mesh traversal and does not use any user-given parameters.

This is in sharp contrast with most of the concurrent methods which usually relay on 3D geometrical computations (adjacent planes searching, planes intersection infinite line computation, ridge line end points searching, *etc.*) which are known to be tricky to be robustly implement with floating point numbers [10].

In addition, by choosing to work as long as possible in 2D, we avoid many complex 3D configurations and defer the consideration of height jump edges to the last step of the workflow.

In practice, there is no technical difficulty to build the topological graph from the mesh as far as the mesh is encoded in an appropriate data structure, for example the half-edge data structure [31, 25].

The algorithm starts by adding to the graph the vertices and the edges of the outline border which have been previously tagged in the mesh. Then it adds the *dual edges* of the ridge triangles that have been previously tagged (spanning two or three planes). This consists in adding a vertex in the center of each triangle and connecting the barycenters of the triangles that share a common edge (see zoom in Fig.3). The algorithm also records, for each graph vertex, an attribute composed of the labels of the involved planes. In 2D, there are three possible configurations, as follows: If a vertex has exactly three labels, it is tagged as a *corner* and will be locked up and kept forever in the graph. If a vertex has exactly two labels, it is an *ordinary* vertex and could be removed during the optimization step. If a vertex has a unique label, it is a *connection* vertex and must be preserved. It is connected to the outline border by projecting it to the closest boundary edge (which is necessarily one of the edges of the triangle it belongs to) (see zoom in Fig.3).

The resulting raw topological graph is shown in Fig.2-G where the *corner* vertices are depicted in red, while the *connection* ones are in blue. The *ordinary* vertices are not highlighted for better visibility but the edges between them are drawn in blue.

Notice that, at this stage of the processing, the topological graph is composed of many small edges which have been extracted from the dense α-shape mesh. However, we are able to anticipate the model optimization by simply removing all *ordinary* vertices, which allows is to already simplify the internal ridge lines, as shown in Fig.3.

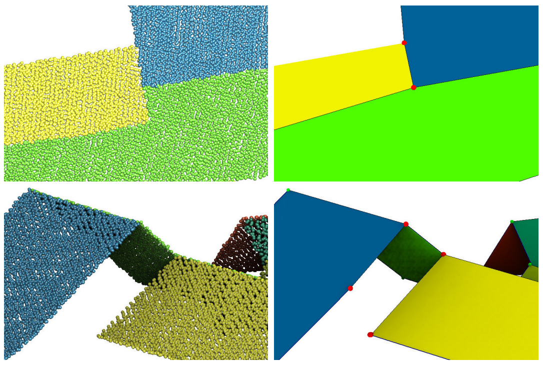


Figure 4: Model regularization step. **Top row:** input points as seen from above and corresponding 2D polygons after the simplification. **Bottom row:** side view of the original input points and the regularized model. Red dots show the vertices that were duplicated.

3.3 Model regularization

The model computed by the previous steps already has the correct topology but needs to be geometrically optimized to become an acceptable roof model. First we need to simplify the polygon boundaries, then we have to deal with the planarity constraint as well as the detection of height skip lines.

There are several methods in the literature to simplify polylines. While Douglas-Peucker's algorithm [12] is the best known, we have implemented Visvalingam's algorithm [47], because it is more efficient. The idea is very simple: we gradually remove points with the least-perceptible change, as in [34].

We have a polygonal (non-planar) mesh and the Cartesian plane equations. Each vertex of our model to be regularized is incident to one, two or three facets. According to the connectivity we move the vertices:

- *One incident facet:* we project it onto the corresponding plane.
- *Two incident facets:* we project it onto the intersection line.
- *Three incident facets:* we move it into the intersection of the corresponding planes.

While the projection is conceptually simple, care must be taken. Refer to the bottom left image of the Fig. 4 for an illustration. Yellow and blue planes do not intersect each other, yet the 2D α-shape is not aware of that.

Consider the vertices highlighted in red in top right image of the figure; if we project them onto both planes simultaneously, we get an inconsistent result. Fortunately, it is very easy to detect this inconsistency.

Gross density	Average density	Beam divergence	Accuracy x, y	Accuracy z	Mean distance between points
28 pts/sqm	30 pts/sqm	0,25 mrad	10 cm	6 cm	20 cm

Table 1: Lidar parameters.

Indeed, an inconsistency can be detected thanks to the distance between the original position of the point and its possible new position. If this distance exceeds a certain threshold then it's an inconsistency. If the inconsistent point has a valence of three, then among the adjacent planes, we identify a pair of planes that appear to be the most parallel (the ones that will generate the greatest inconsistency). And we duplicate the point so as to separate the two planes. If it has a valence of two then we simply duplicate the point and separate the two planes as shown in the bottom right image. So, the regularization step can be seen as the following loop:

1. project all vertices onto the corresponding planes;
2. if there are no inconsistent configurations, stop;
3. else duplicate the inconsistent vertices and go to the projection step.

The remarkable aspect here is the fact that our algorithm detects and recovers the inconsistencies that correspond to the step edges (height jump lines). The step edge detection is a hard problem and our combinatorial approach is more robust than the geometry-based algorithms like 3D α-shape or snapping outlines of individual roof panes. Fig. 5 presents some examples of the regularization algorithm.

4 EXPERIMENTAL RESULTS

Data set description

The data set used in our tests was captured over Breuschwickersheim of the Eurometropolis of Strasbourg in France using the LMS Q780 Riegl laser scanner (see Tab. 1). It covers an area of approximately $1000\,m \times 1000\,m$.

The classification of the point cloud is obtained during the acquisition by return wave analysis. From this classification, we were able to extract a set of 2.5M points corresponding to the building roofs of the entire area (Fig.1). This single point cloud has been further delineated into clusters corresponding to individual buildings. We obtained 445 clusters using the *Euclidean Cluster Extraction* algorithm provided by the Point Cloud Library [35]. Any other clustering algorithm could be used for this purpose. Each resulting cluster corresponds to the roof of a single building or a set of adjacent buildings (Fig.1). Note that, in very densely built-up areas, the latter configuration may be the norm and no automatic tool can determine a limit between neighboring buildings if there is no gap between them, without cadastral information. Since we are not interested in the problem of building detection, we assume that the building delineation is correct. We have kept the large clusters as they were delivered, without cutting them manually. This allows us to provide reasonably challenging inputs to our roof modeling method, presented here.

As a reference data, a human operator has built manually the polygonal models for every cluster shown in Fig.1 using the RhinoCapture software [39]. An example of a reference roof is given in Fig. 6.

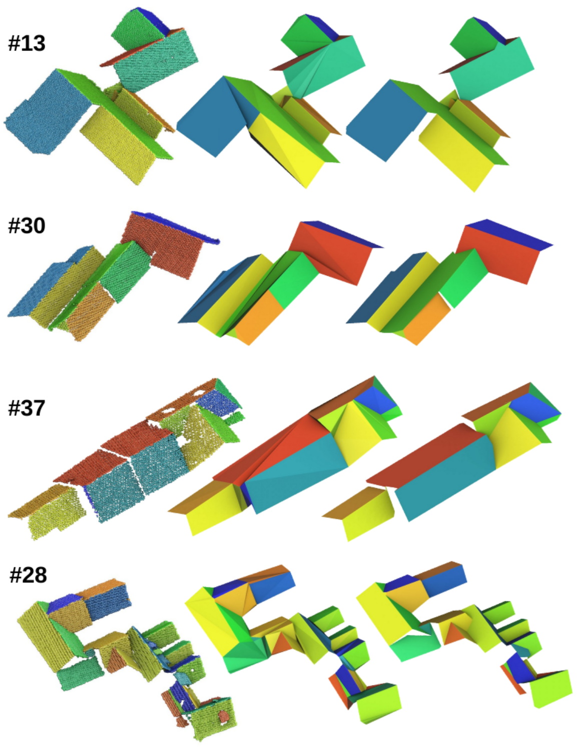


Figure 5: Models with step edges. First column: annotated points; Second column: topological graph; Third column: model after regularization.

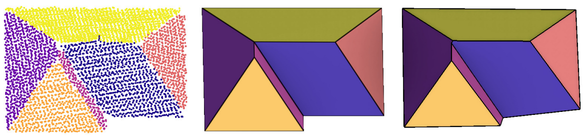


Figure 6: **Left:** labeled input point set; **Middle:** manually constructed roof model; **Right:** automatic reconstruction.

	C_m	C_r	Q_l
Mean	94.0	92.7	90.8
Median	100	100	100
First quartile	91.2	89.6	80.8

Table 2: Accuracy assessment of the plane detection over the entire data (445 building clusters).

Quantitative evaluation

We have used the evaluation system proposed in [42, 3] which assumes that the roofs are represented by polygonal models and each individual roof consists of a set of flat planes. We have carried out an *object-based evaluation at plane level* since, let us recall, our method addresses only the problem of roof modeling by detecting planes in already segmented building clusters.

In object-based evaluation, *completeness* (C_m), *correctness* (C_r), and *quality* (Q_l) are estimated by counting the number of true positive (TP), false positive (FP) and false negative (FN) planes in the extracted results, as follows : $C_m = TP/(TP+FN)$, $C_r = TP/(TP+FP)$ and $Q_l = TP/(TP+FN+FP)$. Ideally, these values should be maximum at 100%. The *completeness* indicates the detection rate and is the percentage of entities in the reference data that were detected. The *correctness* indicates how well the detected entities match the reference data and is closely linked to the false alarm rate. The *quality* provides a compound performance metric that balances *completeness* and *correctness*.

As depicted on Table 2, the average completeness, correctness and quality metrics, computed on the entire dataset (445 clusters of buildings), were respectively 94%, 92.7% and quality 90.8%.

In addition to these quantitative results, qualitative analysis is also presented via visualization (see Fig.7).

Table 3 gives detailed evaluation metrics for the clusters depicted in Fig.5, Fig.7 and Fig.8. All of the clusters corresponding to individual buildings were perfectly reconstructed (see Fig.7). Large clusters corresponding to adjacent buildings were also correctly handled (see Fig.7:#10,#23). In the case of clusters presenting step edges (Fig.5), some true planes were missed and some roof planes were wrongly constructed. The last four clusters (Fig.8) are considered as fail cases which are mainly due to poor plane segmentation.

Discussion

The proposed method strives for building roof models with a level of detail LOD2, as defined in the CityGML standard [18]. This means that we look for a detailed roof structures without superstructures (e.g., chimneys, dormers, etc.).

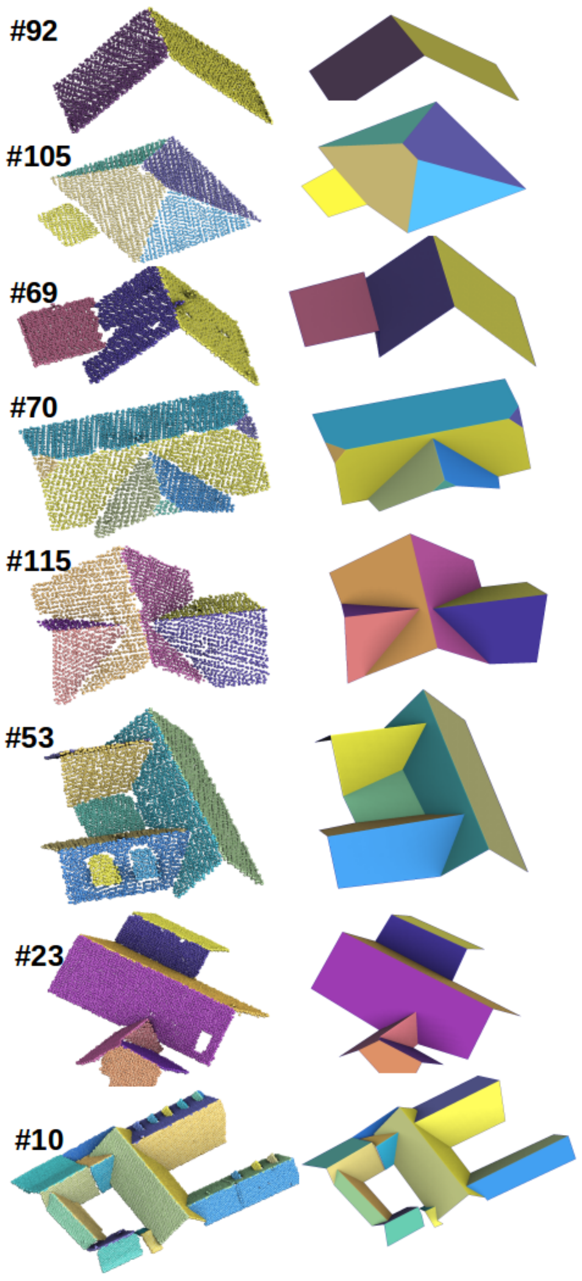


Figure 7: Resulting models for 8 clusters.

The small roof superstructures such as chimneys, antennas and various noisy components are mostly removed by the *plane segmentation algorithm* (step 1).

However, some small roof extensions and detail roof structure were correctly modeled as shown in Fig.7:#105.

The dormers are roof elements defining small planes which are topologically *inside* the primary roof planes. Large dormers are recognized and accurately delineated by our step 1 algorithm, as shown on Fig.7:#53. But these structures are usually removed by the *model generation algorithm* (step 2).

#	#ref	TP	FP	FN	C_m	C_r	Q_l
92	2	2	0	0	100	100	100
105	5	5	0	0	100	100	100
69	3	3	0	0	100	100	100
70	7	7	0	0	100	100	100
115	6	6	0	0	100	100	100
53	7	7	0	0	100	100	100
23	8	8	0	0	100	100	100
10	18	18	0	0	100	100	100
13	13	13	1	0	100	92.9	92.9
30	9	8	0	1	88.8	100	88.8
37	13	11	1	2	84.6	91.7	78.6
28	19	18	1	1	94.7	94.7	90
151	7	7	4	0	100	63.6	63.6
214	3	3	2	0	100	60	60
215	9	8	0	1	88.9	100	88.9
256	2	2	1	0	100	66.7	66.7

Table 3: Detailed evaluation metrics for the roofs in Fig.7, Fig.5 and Fig.8.

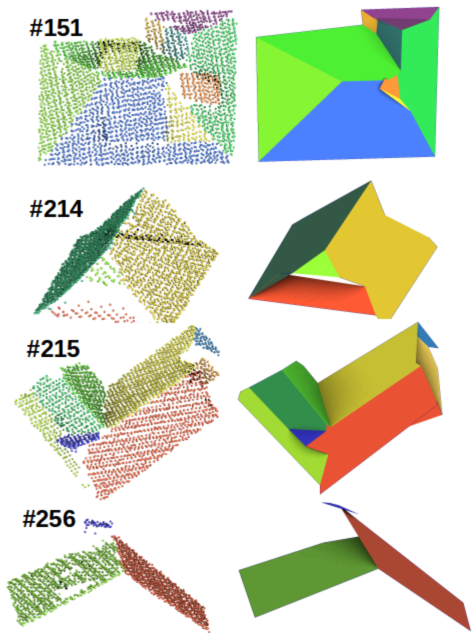


Figure 8: Examples of failure cases due to incorrect plane segmentation.

Let us recall that our *model generation algorithm* works in 2D, which allows us, on one hand, to design a robust combinatorial algorithm avoiding many 3D problems, on the other hand, to postpone the 3D embedding problems due to height jumps. This makes our method able to handle roofs with simple geometry quite efficiently. For simple flat, gamble and hip roofs, the algorithm achieved 100% completeness, correctness and quality. Our data contains 40% of this kind of structure. For cross-fipped, cross-gabled, hip and valley and intersecting roofs, the method achieved 98% of perfect models. These shapes constitute about 30% of our dataset (Fig.7). As show on Fig.5, most of the complex structures with height jumps are correctly optimized by our *model regularization* algorithm. The remaining 30% are complex structures, 20% of them are successfully handled and the resulting models are acceptable for further use (see Fig.7:#10,#23), while 10% of our data are real failures which are mostly due to poor plane segmentation, as shown in Fig.8.

5 CONCLUSION

In this paper we propose a new workflow for the reconstruction of building roofs from pre-classified LIDAR point data. The method takes as input a cluster of 3D points corresponding to the roof of an individual building. It delivers a 3D polygonal model where each planar roof patch is represented as a flat polygon and their ridge and corner intersections are topologically consistent. The main contribution is a combinatorial approach that is more robust than the geometrical methods trying to snap intersection lines. Having the combinatorial information, we easily reconstruct step edges during the regularization phase.

Experimental testing has shown that the proposed pipeline successfully deals with challenging input data. It achieves a plane detection rate of 94% and a quality rate of 90%.

We believe that the resulting models will be suitable for further polyhedral building reconstruction through geometric extrusions, for example. Moreover, they could also be useful for the initialization of model-driven methods for roof modeling.

6 ACKNOWLEDGMENTS

Our thanks to the company RhinoTerrain, for providing us with several datasets and for data modelling.

7 REFERENCES

[1] ABDULLAH, S., AWRANGJEB, M., AND LU, G. Lidar segmentation using suitable seed points for 3d building extraction. *Int. Arch. of the Phot., R. Sensing & S. I. Sciences* (2014).

[2] ALHARTHY, A., AND BETHEL, J. Detailed building reconstruction from airborne laser data using a moving surface method. *Inter. Archives of Photogrammetry and Remote Sensing 35* (01 2004).

[3] AWRANGJEB, M., AND FRASER, C. An automatic and threshold-free performance evaluation system for building extraction techniques from airborne lidar data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 7*, 10 (2014), 4184–4198.

[4] AWRANGJEB, M., AND FRASER, C. Automatic segmentation of raw lidar data for extraction of building roofs. *Remote Sensing 6* (05 2014), 3716–3751.

[5] B. ALBERS, M. KADA, A. W. Automatic extraction and regularization of building outlines from airborne lidar point clouds. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences 41* (2016).

[6] BOGE, M., MEIDOW, J., AND BULATOV, D. Extraction and refinement of building faces in 3d point clouds. *The Inter. Society for Optical Engineering 8892* (10 2013), 88920V.

[7] BOULCH, A., AND MARLET, R. Fast and robust normal estimation for point clouds with sharp features. *Computer Graphics Forum 31* (08 2012), 1765–1774.

[8] BRENNER, C. Building reconstruction from images and laser scanning. *International Journal of Applied Earth Observation and Geoinformation 6*, 3 (2005), 187 – 198.

[9] D. CHEN, L. ZHANG, J. L., AND LIU, R. Urban building roof segmentation from airborne lidar point clouds. *Inter. Journal of Remote Sensing 33* (10 2012), 6497–6515.

[10] DEVILLERS, O., AND GUIGUE, P. Finite Precision Elementary Geometric Constructions. Tech. Rep. RR-4559, INRIA, Sept. 2002.

[11] DONG, C., WANG, R., AND PEETHAMBARAN, J. Topologically aware building rooftop reconstruction from airborne laser scanning point clouds. *IEEE Transactions on Geoscience and Remote Sensing PP* (08 2017).

[12] DOUGLAS, D., AND PEUCKER, T. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Inter. J. for Geographic Information and Geovisualization 10*, 2 (1973), 112–222.

[13] DUCKHAM, M., KULIK, L., WORBOYS, M., AND GALTON, A. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition 41* (10 2008), 3224–3236.

[14] EDELSBRUNNER, H. Alpha shapes - a survey. *Tessellations in the Sciences* (01 2010).

[15] FAN, H., YAO, W., AND FU, Q. Segmentation of sloped roofs from airborne lidar point clouds using ridge-based hierarchical decomposition. *Remote Sensing 6* (04 2014), 3284–3301.

[16] G. FORLANI, C. NARDINOCCHI, M. S., AND ZINGARETTI, P. Complete classification of raw lidar data and 3d reconstruction of buildings. *Pattern analysis and appli. 8*, 4 (2006), 357–374.

[17] G. SOHN, X. H., AND TAO, V. Using a binary space partitioning tree for reconstruction polyhedral building models from airbome lidar data. *Photogrammetric Engineering and Remote Sensing 74* (08 2008), 1425–1440.

[18] GRÖGER, G., AND PLÜMER, L. Citygml–interoperable semantic 3d city models. *ISPRS Journal of Photogrammetry and Remote Sensing 71* (2012), 12–33.

[19] HAALA, N., AND KADA, M. An update on automatic 3d building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing 65*, 6 (2010), 570 – 580.

[20] HOUGH, P. Method and means for recognizing complex patterns. *US patent 3*, 6 (1962).

[21] HUANG, H., AND BRENNER, C. Rule-based roof plane detection and segmentation from laser point clouds. In *Joint Urban Remote Sensing Event* (2011), IEEE, pp. 293–296.

[22] JAEWOOK, J., YOONSEOK, J., AND GUNHO, S. Implicit regularization for reconstructing 3d building rooftop models using airborne lidar data. *Sensors 17* (03 2017).

[23] JARZABEK-RYCHARD, M. Reconstruction of building outlines in dense urban areas based on lidar data and address points. *Inter. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 39* (07 2012), 121–126.

[24] JARZABEK-RYCHARD, M., AND MAAS, H.-G. Geometric refinement of als-data derived building models using monoscopic aerial images. *Remote Sensing 9* (03 2017), 282.

[25] KETTNER, L. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry 13*, 1 (1999), 65–90.

[26] KIM, K., AND SHAN, J. Building roof modeling from airborne laser scanning data based on level set approach. *ISPRS Journal of Photogrammetry and Remote Sensing 66*, 4 (2011), 484–497.

[27] MAAS, H., AND VOSSELMAN, G. Two algorithms for extracting building models from raw laser altimetry data. *J. of Photogrammetry and Remote Sensing 54*, 2 (1999), 153 – 163.

[28] MALTEZOS, E., AND IOANNIDIS, C. Automatic extraction of building roof planes from airborne lidar data applying an extended 3d randomized hough transform. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences III-3* (06 2016), 209–216.

[29] MEIDOW, J., AND HAMMER, H. Algebraic reasoning for the enhancement of data-driven building reconstructions. *J. of Photogrammetry and Remote Sensing 114* (04 2016), 179–190.

[30] METHIRUMANGALATH, S., PARAKKAT, A., AND MUTHUGANAPATHY, R. A unified approach towards reconstruction of a planar point set. *Computers & Graphics 51* (05 2015).

[31] MULLER, D., AND PREPARATA, F. Finding the intersection of two convex polyhedra. *Theoretical Computer Science 7*, 2 (1978), 217 – 236.

[32] NOVACHEVA, A. Building roof reconstruction from lidar data and aerial images through plane extraction and colour edge detection. *Inter. Society for Photogrammetry and Remote Sensing* (01 2008), 51–56.

[33] OVERBY, J., BODUM, L., KJEMS, E., AND IISOE, P. Automatic 3d building reconstruction from airborne laser scanning and cadastral data using hough transform. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV* (01 2004).

[34] PARAKKAT, A., BONDI, U., AND MUTHUGANAPATHY, R. A delaunay triangulation based approach for cleaning rough sketches. *Computers & Graphics 74* (05 2018).

[35] PCL. Pcl. pointclouds.org.

[36] PETER, D., AND PFEIFER, N. A comprehensive automated 3d approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds. *Sensors 8* (11 2008).

[37] R. WANG, J. P., AND CHEN, D. Lidar point clouds to 3-d urban models : a review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 11*, 2 (2018), 606–627.

[38] R.C. DOS SANTOS, M. G., AND CARRILHO, A. Extraction of building roof boundaries from lidar data using an adaptive alpha-shape algorithm. *IEEE Geoscience and Remote Sensing Letters 16*, 8 (2019), 1289–1293.

[39] RHINOTERRAIN. Rhinocapture. www.rhinoterrain.com.

[40] ROTTENSTEINER, F. Consistent estimation of building parameters considering geometric regularities by soft constraints. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 34* (01 2006).

[41] ROTTENSTEINER, F., TRINDER, J., CLODE, S., AND KUBIK, K. Automated delineation of roof planes from lidar data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36* (05 2012).

[42] RUTZINGER, M., ROTTENSTEINER, F., AND PFEIFER, N. A comparison of evaluation techniques for building extraction from airborne laser scanning. *J. of Selected Topics in Applied Earth Observations and Remote Sensing 2*, 1 (2009), 11–20.

[43] SOHN, G., JUNG, J., JWA, Y., AND ARMENAKIS, C. Sequential modelling of building rooftops by integrating airborne lidar data and optical imagery: preliminary results. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences II-3/W1* (05 2013).

[44] SUN, S., AND SALVAGGIO, C. Aerial 3d building detection and modeling from airborne lidar point clouds. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of 6* (06 2013), 1440–1449.

[45] TARSHA-KURDI, F., LANDES, T., AND GRUSSENMEYER, P. Extended ransac algorithm for automatic detection of building roof planes from lidar data. *The Photogrammetric Journal of Finland 21* (01 2008), 97–109.

[46] V. VERMA, R. K., AND HSU, S. 3d building detection and modeling from aerial lidar data. *Computer Society Conference on Computer Vision and Pattern Recognition 2* (02 2006), 2213 – 2220.

[47] VISVALINGAM, M., AND WHYATT, J. *Line generalisation by repeated elimination of the smallest area.* Cartographic Information Systems Research Group, 1992.

[48] VOSSELMAN, G. Building reconstruction using planar faces in very high density height data. *Int. Arch. Photogramm. Remot. Sens 32* (12 2000).

[49] XIAO, Y., WANG, C., LI, J., ZHANG, W., XI, X., WANG, C., AND DONG, P. Building segmentation and modeling from airborne lidar data. *International Journal of Digital Earth 8*, 9 (2015), 694–709.

[50] XIONG, B., ELBERINK], S. O., AND VOSSELMAN, G. A graph edit dictionary for correcting errors in roof topology graphs reconstructed from point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing 93* (2014), 227 – 242.

[51] XU, B., JIANG, W., AND LI, L. Hrtt: A hierarchical roof topology structure for robust building roof reconstruction from point clouds. *Remote Sensing 9* (04 2017), 354.

[52] ZHOU, Q.-Y., AND NEUMANN, U. 2.5d building modeling by discovering global regularities. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (June 2012), pp. 326–333.

Mixing deep learning with classical vision for object recognition

Maciej Stefańczyk
Warsaw University of Technology
Institute of Control and Computation Eng.
Nowowiejska 15/19, 00-665 Warsaw, Poland
maciej.stefanczyk@pw.edu.pl

Tomasz Bocheński
Warsaw University of Technology
Institute of Control and Computation Eng.
Nowowiejska 15/19, 00-665 Warsaw, Poland
tbochens@gmail.com

ABSTRACT

Nowadays, when one needs a system for image recognition, it is mostly a matter of finding pre-trained CNN and, sometimes, adding additional training based on transferred knowledge. Accurate 6-DOF object localization in the image is a more laborious task and requires more complex training data to be available. On the other hand, if we know the model of the object, it is straightforward to acquire its pose from the image (RGB or RGB-D). In this paper, we try to show the advantages of mixing deep learning object recognition/detection with classical 6-DOF pose estimation algorithms, with a focus on applications in service robotics.

Keywords
CNN object detection, VGG16, ResNet50, 6-DOF pose estimation, RanSaC, ICP, RGB-D

1 INTRODUCTION

1.1 Motivation

It is hard to imagine life without robots. They have become an inseparable part of our reality and are applicable in almost every area of life. Robots are used in factories for transport, production, and quality control. Telemanipulators, controlled by doctors, are used to perform surgical procedures. Thanks to robots, hazardous environments (for example, underwater or in space) can be safely investigated. In the military, their primary use is to disarm bombs or take other dangerous actions, saving the life of the soldiers.

Robotics, as an industrial field, is developing rapidly. According to a report of the International Federation of Robotics, global robot sales in 2017 increased by 30% compared to the previous year [oR18]. The development in the field of industry is also accompanied by great progress in the field of research. Modern robots are able to carry out work that not long ago was only performed by people. One of the sources of progress is equipment robots with senses, making them more autonomous.

Eyesight is one of the most important senses of man as it allows us to perceive most of the information from the environment. It is estimated that 83% of human perception takes place through sight. For comparison, hearing, smell, touch, and taste are processing 11%, 3.5%, 1.5%, and 1% of information respectively [SK11]. Therefore, equipping robots with eyesight becomes an important research issue.

One of the tasks in service robotics is object manipulation, where robots should cope with everyday things from the human surrounding. For this task to be performed flawlessly, robots must be equipped with effective manipulators and grippers and a vision system capable of accurate object pose estimation. Deep learning approaches have proven their high quality and accuracy in object classification multiple times, and are capable of distinguishing hundreds or thousands of different classes with minimal impact on performance. Accurate object pose estimation, on the other hand, is straightforward to achieve if we know what object we see and we have its model. In this paper, we try to show the advantages of mixing very popular deep learning approaches for object classification/detection with classical 6-DOF pose estimation algorithms in service robotics applications.

1.2 Paper structure

The rest of the paper is structured as follows. The next section describes state of the art in object detection and pose estimation tasks. Section 3 presents proposed system structure, followed by the implementation details in sec. 4. Sample results for a single scenario are presented in sec. 5, and the last section concludes the paper.

2 STATE OF THE ART

2.1 Object classification and detection

Object classification answers the question *what is on this picture*. Object detection deals with a more complicated task of not only telling *what* is on the picture, but also *where*, for multiple objects at the same time. The simplest way of creating an object detector is to apply the classifier multiple times (e.g. sliding window) on different input scales, but there are also much more sophisticated techniques. Here the important algorithms for both are presented. The advent of object detection can be connected with the first Viola-Jones face detector [VJ01], which, although being still the sliding window detector, applied methods for quick rejection of candidate regions to achieve big performance gain. Next step in "classical" object detection is HOG [DT05] with further improvement of deformable part models [FMR08]. Those solutions were very good at their time, and to some extent, were able to generalize on objects different from the initial ones (faces and humans). As they rely mostly on shape, not texture, it is hard to use them to distinguish between different subtypes of similar objects.

Another branch of classification and detection algorithms is based on the image feature points (either classical SIFT [Low99] or more recent binary descriptors [RRKB11]). Those can be either compared directly with the object models (with RanSaC) or their statistics can be used (in the form of a bag of visual words). The latter can be further extended to create an object detector [VL09].

All aforementioned approaches were model-based, handcrafted to a smaller or bigger extent. With the advent of consumer-grade high power GPUs, the branch of data-driven approaches emerged, with neural networks and deep learning playing the main role. In classification task, the most widely used architectures are VGG [SZ14], Inception [SVI$^+$16] and ResNet [HZRS16], with their extensions and new ideas being published almost constantly [ZSGY19].

Currently, two types of detectors are used to solve the problem of object detection with the use of CNN, i.e. one-stage and two-stage. The way of operation of two-stage detectors consists of two steps: proposing the regions of interest and classification of these, together with the improvement of their bounding boxes. The single-stage detectors immediately determine the bounding boxes along with the classes of objects, without first detecting the regions of interest. The two-stage detectors are characterized by high performance in terms of accuracy but do not always operate in real time. The single-stage detectors operate in real-time, but the results are not always as good as those of the two-stage detectors. The most popular two-stage detector is Faster R-CNN [RHGS15], while the most commonly used single-stage detectors are SSD [LAE$^+$16] and YOLO [RDGF16].

2.2 Pose estimation

The goal of object pose estimation is to find the position and orientation of the query object in the scene (relative to some given coordinate frame, e.g. camera). This task can be performed either solely in RGB space or using additional depth information provided by current sensors (like Kinect or Intel RealSense). Contrary to object detection (or classification), in general, one needs an exact model of the object to find its pose.

A simple (yet effective) approach to pose estimation for textured objects relies on feature points. Those can be either extracted directly from RGB (like SIFT or ORB) and then matched against a set of reference views [Low01] or reprojected to 3D using available depth information and matched against reference sparse cloud of feature points. When depth (or 3D in general) information is available, features can be computed directly from 3D [MBO10, RBTH10]. To overcome problems from multiple matches between the feature points (in multi-object scenarios), additional hypothesis verification can be applied [HCL$^+$13, ATP$^+$13].

For texture-less models, LineMOD [HLI$^+$12] uses multiple object silhouettes, generated from the rendered 3D views. Part-based models, along with 3D CAD models, can also be efficiently used to retrieve the pose of challenging objects [AME$^+$14]. An interesting branch is also usage of procedural models [GFJ$^+$18], which, apart from the pose, estimates also other object parameters, like size, radius, etc.

The next advancement in the field was the application of trainable RGB-D feature detectors. It is either based on previous solutions (like extending LineMOD [TTKK14]) or learning features with the CNN approaches [GAGM15, KMT$^+$16].

Finally, end-to-end deep learning solutions were proposed. This problem can be simplified if only the grasp region candidates are required [TTS$^+$18]. PoseCNN [XSNF17] extracts full 6D pose of the 21 known objects. The authors subdivided the problem into three main steps: semantic labeling, 3D position estimation, and final rotation regression, implemented as the set of components in a multi-task network. The network was trained on the data generated from the YCB dataset [CSW$^+$15], providing textured 3D models of the objects and perfectly labeled test scenes. SSD-6D [KMT$^+$17] is an approach to extend the SSD network to work with 3D data. In order to facilitate the final 6D pose estimation, the authors trained the network to recognize one of the multiple discrete viewpoints. Final transformation refinement is done using classical vision algorithms (based on either RGB-only or depth-enriched data). One of the latest advancements in the field is YOLOff [GKMM20], which is also a pair of cooperating networks. First detects the 3D object patches, which are then passed through the 3D points regression module, detecting the corresponding points between the scene and the model. Final pose refinement is done using classical algorithms. Those kinds of solutions are, however, much harder to train than 2D object detectors and not as popular yet [HMB$^+$18]. As the authors of [XSNF17] state, sufficient and well-labeled training data is necessary. On the other hand, for classical approaches, only a simple model is enough, and no retraining is needed in case new models are required.

3 PROPOSED SYSTEM STRUCTURE

To take the best of two worlds, we propose the hybrid system, with CNN based object detector and feature-point based pose estimator (as a lot of the objects in human surroundings possess a rich texture). To further increase the system performance, we decided to use additional depth data (from the RGB-D sensor mounted on the robot). The system is composed of three main processing blocks (fig. 1) and an additional 3D model database.

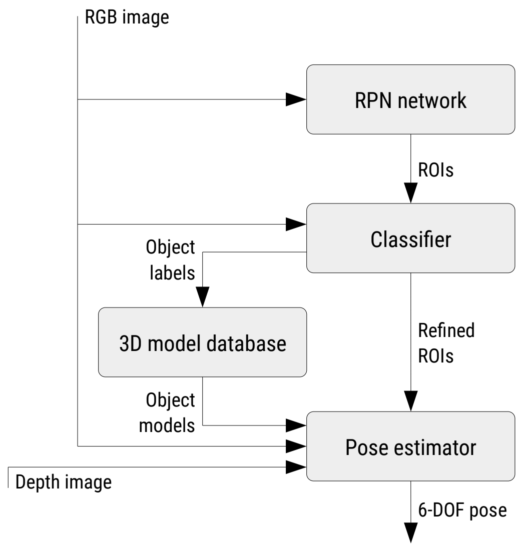


Figure 1: System structure

The first block is responsible for regions of interest detection in the RGB image. Service robots usually cope with a small number of objects visible in a single scene, and typically don't require high detection framerates. In that kind of scenario, separating RPN from classifier doesn't impact the overall system performance. Detected regions are passed to the classifier, which labels the ROIs with known classes and refines the bounding boxes. Finally, the object's pose is estimated using additional depth data and the 3D model retrieved from the database based on the label provided by the classifier.

4 IMPLEMENTATION DETAILS

4.1 Region proposal network

The region proposal network was based on simple yet popular VGG16 architecture. This architecture provides a good compromise between complexity (has only 16 trainable layers) and quality (won the ILSVRC competition in 2014). Simplified scheme of our RPN is presented in fig. 2.

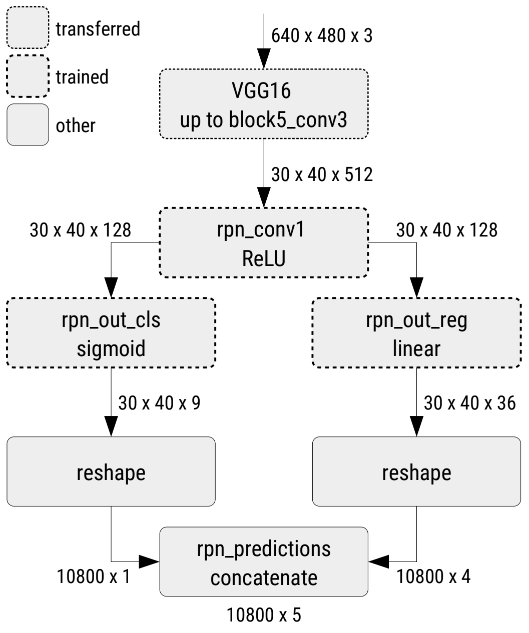


Figure 2: Region proposal network

The network takes VGA RGB images as the input. There were nine anchors defined for RPN. As the feature extracting backbone, the first 13 layers from VGG were utilized, with an additional convolution added. After that, the network splits into classification and regression parts. Classification part produces a label (object/background) for each anchor box (9 in total). The regression part produces offsets (for top left corner) and size corrections (width and height) for each anchor box.

4.2 Classification network

For the classification stage, the ResNet50 was used as a backbone (fig. 3). Object proposals, cropped to 224×224 pixels, were fed as an input. Two additional convolution layers further transformed features from the backbone, and the global average pooling layer produced final features. After that, network splits into the classification part (fully connected, all known classes plus one for background) and bounding box regression part (also fully connected, four values, like in RPN).

4.3 Pose estimation

After the object detection step, all recognized objects are cropped from the RGB image along with accompanying depth information. Object type information

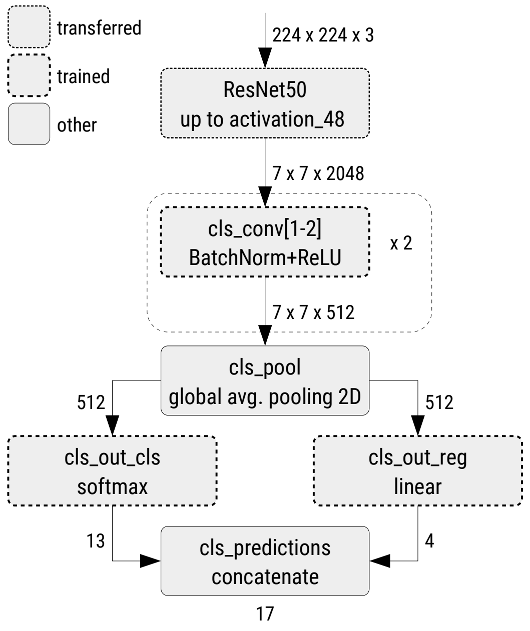


Figure 3: Classification and localization network

is used to retrieve the appropriate 3D model from the model database (both dense point cloud and sparse feature cloud). Pose estimation is divided into three main steps (fig. 4). The first one is responsible for feature extraction and matching. On the cropped RGB image, SIFT feature points are detected, which are then reprojected from 2D image coordinates to 3D world coordinates using available depth information. After that, those are matched with model features.

When the pairs of matched points are prepared, the RanSaC is used to estimate the initial rigid transform between the model and the query image. As this transform is based on a sparse cloud of features, and the query cloud itself is very low resolution, this transform often needs refinement. To do this, as the last step, ICP is used. It tries to find the final transformation between the full model point cloud and the query point cloud (built from RGB and depth crops). In the testing scenario, simple ICP with the point-to-plane metric was used.

5 RESULTS

5.1 Scenario, hardware and dataset

The prepared system was tested on everyday objects that can be found in the kitchen, mainly tea boxes and food cans. The assumed scenario was a robot tasked to find and bring a particular object from the cupboard. As the system should detect the same kind of objects that the robot works with, it was trained on some most popular ones. The set consists of 12 different objects, and, as there is no available dataset with those, the new one had to be created. For the training phase of the detector, there were around 30 images for each object,

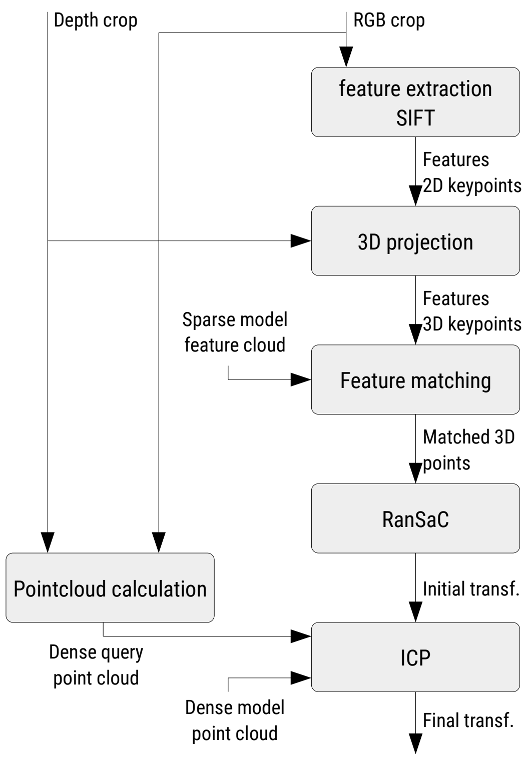


Figure 4: Pose estimation module

taken from two angles (front and top) around the object. For each image, there was also a mask prepared for further cropping and preparing augmented training dataset. Data acquisition setup and sample object cutout are presented in fig. 5. Pose estimation step used 3D object models prepared as cuboids or cylinders, with applied scanned textures (1 model per object). The whole process of data acquisition follows the one described in [KS17], and created datasets are publicly available [SLK16].

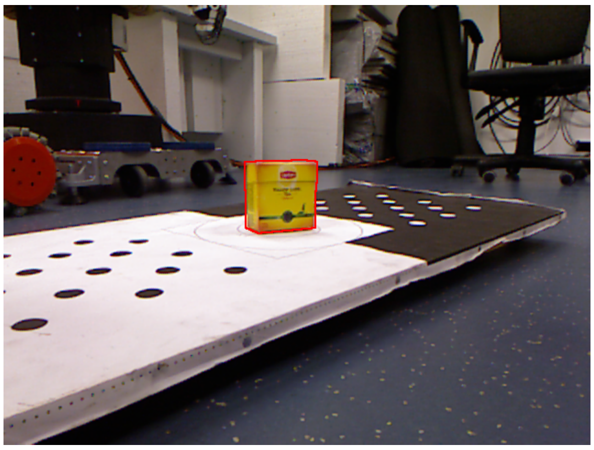


Figure 5: Training data acquisition – sample image with generated mask overlay

The data was acquired using the MS Kinect sensor, providing aligned RGB and depth images with VGA resolution. Test scenes were arranged with the objects placed inside the cupboard (fig. 6).

5.2 Detection

On the object detection stage, two parts of the network were evaluated. RPN network was trained using weights transferred from the ImageNet model, and only the added layers' weights were optimized using Adam optimizer with a learning rate of $10e^{-4}$. 10000 synthetic images were used for training. The final network achieved a precision of 0.63 and a recall of 0.98. High recall suggests that all interesting objects were properly selected. Low precision means that apart from the objects of interest (12 classes), others were selected (yellow boxes on fig. 6). This is a good result, as it makes the network robust to changing needs and easy to extend with new objects. The spurious RPN detections are filtered on the classification stage. On 4GB GTX 970 (rather mediocre in today's standards), training took 10 hours. This time is not very short, but the RPN doesn't have to be retrained if new objects are added to the system.

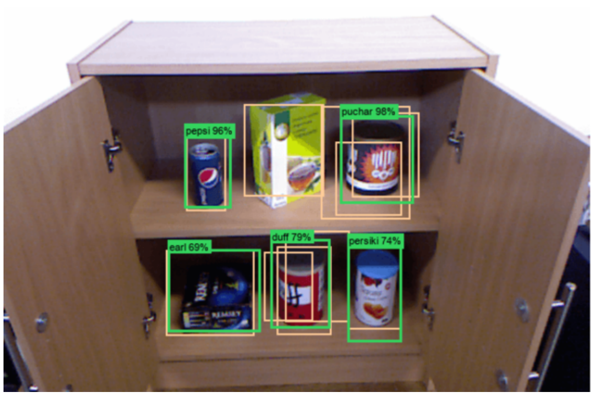


Figure 6: Sample result of object detection: yellow boxes – RPN result, green boxes – detected objects.

The classifier also had its initial weights transferred from the ImageNet model, and also only the top layers had updated weights. The training was done using 2000 synthetic images (taken from the same set as in RPN training), and the network achieved its final quality after only three epochs. This fact makes the process of adding new objects very fast. Classifier network achieved 0.99 precision and 0.98 recall. The training took 3.5 minutes on GTX 970.

Final tests were conducted for the network cascade, where the output of the RPN was tied to the classifier input. This time the precision was still very high (0.99), but recall dropped to 0.92 (mainly due to one class, which had the worst statistics in both the RPN and classifier stage). Sample detections are marked with green boxes on fig. 6. The top-middle object is not in the training set, so it was properly ignored by the classifier.

5.3 Pose estimation

The output of the detector is used to cut the single objects from the RGB and depth image. Due to the limited resolution of the Kinect sensor, the resulting object clouds contain only a small number of points (fig. 7a). It is, however, still possible to fit the 3D model to this cloud. For the fitting stage, the initial estimation is done using the RanSaC transformation estimation on the sparse cloud of SIFT points. Features are calculated on the RGB image, and their 3D coordinates are calculated based on the depth image. In 3D models, a sparse cloud of SIFT points is created on the model creation stage in a similar way [KL16].

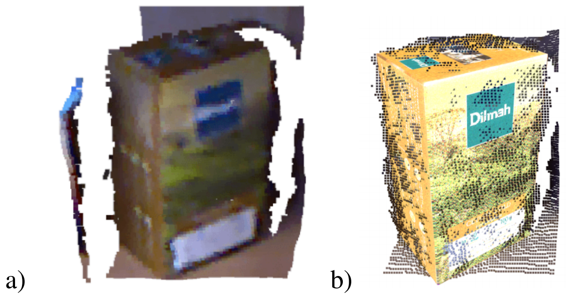


Figure 7: 6-DOF pose estimation: a – input, b – result

After the initial transformation is known, the full point clouds of the object cropped from the scene and the model are used in ICP refinement. Final transformation (fig. 7b), along with the known object parameters (e.g. its size) is then passed to robots grasping subsystem [SS16].

6 CONCLUSION

6.1 Results discussion

The presented system combines the advantages of commonly used detection and pose estimation algorithms. RPN network is trained in such a way, that it is possible to detect novel objects from similar classes (new boxes or cans), without the need for further training. Classification module, in order to correctly recognize particular instances, has to be trained on data as close to the target as possible, but as only a few layers are trained there, this process is rather fast. This makes the process of extending/modifying known objects dataset doable in an acceptable time (3.5 minutes in our experiments). Pose estimation part works with the already preselected data – object that is on the scene is already known and only the 6-DOF pose has to be fit. Adding new objects requires no retraining of this stage, as it is purely procedural.

Performance-wise, the object detection part takes 0.9 s per image and pose estimation another 1.0 s (on average, depending on the spatial resolution of the object). The whole process is definitely not real-time, but for the grasping purpose, it is more than enough. After the first phase (detection) coarse pose of the object can be calculated, and while the robot moves its arm to the pre-grasp location, the final object pose is calculated. The system was tested on the Velma service robot [WBS15], working in a safe environment (fig. 8).

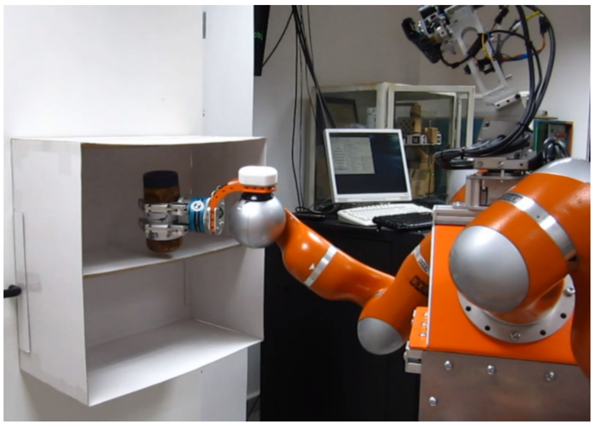


Figure 8: Robot grasping the object

It is hard to directly compare our solution with those presented in sec. 2. The most crucial difference, which makes the comparison unfeasible, is the format of the training dataset. PoseCNN, SSD-6D, and similar requires pose-labeled images as the input for the training. We decided that to make the retraining and extending the system easier, those are not required (and we don't have them). On the other hand, one of the strong assumptions in the presented solution is the rich texture in the objects. In datasets used by the aforementioned methods, those are a minority.

One comparison can be made in terms of the expected burden of system extension with new objects. Data-wise, in our system, one has to supply few (less than 20) pictures of the new object and scanned textures (as the many household objects can be modeled with simple shapes). In contrary, to train the end-to-end solutions, one has to supply pose-labeled pictures, and there must be a lot of them to cover most of the possible viewpoints. Time-wise, as it was described above, only the part of the pipeline needs to be retrained, with a smaller amount of data, which should take less time than retraining the full end-to-end pipelines.

6.2 Future works

There are multiple ways this system can evolve. For the training phase, instead of using the pictures of the object, one can use the rendered scenes solely [PZL+18], with perfect masks and no need for hand labeling.

Another possible direction for making the pose estimation better is changing the feature points detector to binary (if algorithm speed is crucial) or apply depth-based feature points rectification [Ste18]. ICP transformation can also be refined by using other ICP flavors (e.g. color information [LKS16]).

In its current form, the system is general-purpose and can be applied in multiple scenarios. As it is used in robotic applications, some extensions facilitating the hardware available can be made. The biggest impact on the pose estimation accuracy can be obtained using the active vision. After the first estimation, when the arm moves towards the object, additional pictures from the wrist-mounted sensor can be passed to the pose refinement subsystem, effectively implementing the multi-camera visual servoing [KZ15]. As the initial pose is already known, only a few ICP iterations could make the estimation better.

7 ACKNOWLEDGEMENTS

M. Stefańczyk is funded by the National Science Centre, Preludium grant no. UMO-2017/25/N/ST6/02358.

8 REFERENCES

[AME+14] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of cad models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3762–3769, 2014.

[ATP+13] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. Di Stefano, and M. Vincze. Multimodal cue integration through hypotheses verification for RGB-D object recognition and 6DOF pose estimation. In *2013 IEEE international conference on robotics and automation*, pages 2104–2111. IEEE, 2013.

[CSW+15] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[FMR08] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[GAGM15] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. Aligning 3D models to RGB-D images of cluttered scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4731–4740, 2015.

[GFJ$^+$18] R. Getto, K. Fina, L. Jarms, A. Kuijper, and D. W. Fellner. 3D object classification and parameter estimation based on parametric procedural models. *WSCG*, 2018.

[GKMM20] M. Gonzalez, A. Kacete, A. Murienne, and E. Marchand. YOLOff: You Only Learn Offsets for robust 6DoF object pose estimation. *arXiv preprint arXiv:2002.00911*, 2020.

[HCL$^+$13] Q. Hao, R. Cai, Z. Li, L. Zhang, Y. Pang, F. Wu, and Y. Rui. Efficient 2D-to-3D correspondence filtering for scalable 3D object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 899–906, 2013.

[HLI$^+$12] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[HMB$^+$18] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, et al. Bop: Benchmark for 6d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[HZRS16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[KL16] T. Kornuta and M. Laszkowski. Perception subsystem for object recognition and pose estimation in RGB-D images. In *Recent Advances in Automation, Robotics and Measuring Techniques*, volume 440 of *Advances in Intelligent Systems and Computing*, pages 597–607. Springer, 2016.

[KMT$^+$16] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep learning of local RGB-D patches for 3d object detection and 6d pose estimation. In *European conference on computer vision*, pages 205–220. Springer, 2016.

[KMT$^+$17] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1521–1529, 2017.

[KS17] T. Kornuta and M. Stefańczyk. ModReg: a modular framework for RGB-D image acquisition and 3D object model registration. *Foundations of Computing and Decision Sciences*, 42(3):183–201, 2017.

[KZ15] T. Kornuta and C. Zieliński. Robot control system design exemplified by multi-camera visual servoing. *Journal of Intelligent & Robotic Systems*, 77(3–4):499–524, 2015.

[LAE$^+$16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[LKS16] M. Łępicka, T. Kornuta, and M. Stefańczyk. Utilization of colour in ICP-based point cloud registration. In *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015*, Advances in Intelligent Systems and Computing, pages 821–830. Springer, 2016.

[Low99] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[Low01] D. G. Lowe. Local feature view clustering for 3D object recognition. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[MBO10] A. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2-3):348–361, 2010.

[oR18] International Federation of Robotics. Executive summary world robotics 2018 industrial robots, 2018.

[PZL$^+$18] J. Peng, C. Zheng, P. Lv, T. Cui, Y. Cheng, and S. Lingyu. Using images rendered by PBRT to train Faster R-CNN for UAV detection. *WSCG*, 2018.

[RBTH10] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3D recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162. IEEE, 2010.

[RDGF16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[RHGS15] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[RRKB11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[SK11] H. D. Stolovitch and E. J. Keeps. *Telling ain't training*. American Society for Training and Development, 2011.

[SLK16] M. Stefańczyk, M. Laszkowski, and T. Kornuta. WUT visual perception dataset: a dataset for registration and recognition of objects. In *International Conference on Automation*, pages 635–645. Springer, 2016.

[SS16] D. Seredyński and W. Szynkiewicz. Fast Grasp Learning for Novel Objects. In *Recent Advances in Automation, Robotics and Measuring Techniques*, volume 440 of *Advances in Intelligent Systems and Computing*, pages 681–692. Springer, 2016.

[Ste18] M. Stefańczyk. Improving RGB descriptors using depth cues. In *Computer Vision and Graphics*, volume 11114 of *Lecture Notes in Computer Science*, pages 251–262. Springer, 2018.

[SVI$^+$16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[SZ14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[TTKK14] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim. Latent-class hough forests for 3D object detection and pose estimation. In *European Conference on Computer Vision*, pages 462–477. Springer, 2014.

[TTS$^+$18] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[VL09] V. Viitaniemi and J. Laaksonen. Spatial extensions to bag of visual words. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, pages 1–8, 2009.

[WBS15] T. Winiarski, K. Banachowicz, and D. Seredyński. Two mode impedance control of Velma service robot redundant arm. In *Progress in Automation, Robotics and Measuring Techniques. Vol. 2 Robotics.*, volume 351 of *Advances in Intelligent Systems and Computing*, pages 319–328. Springer, 2015.

[XSNF17] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[ZSGY19] Z. Zou, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

The Effects of Different Triangulation Techniques for Cage Based Image Deformation Using Generalized Barycentric Coordinates

Ákos Tóth
University of Debrecen, Doctoral School of Informatics; Faculty of Informatics, University of Debrecen
4028 Debrecen, Hungary
toth.akos@inf.unideb.hu

Roland Kunkli
Faculty of Informatics, University of Debrecen
4028 Debrecen, Hungary
kunkli.roland@inf.unideb.hu

ABSTRACT

In computer graphics, the generalized barycentric coordinates (GBC) are often used for image deformation. To manipulate an input image using a cage based image deformation method, we usually have to consider a source polygon with a triangulation; but defining the triangulation of the source polygon is not a trivial task in most cases. In this paper, a uniform and a non-uniform triangulation technique—which can be the basis of the cage based image deformation—are introduced and compared. Moreover, different texture filtering methods are tried, producing various deformation results. Experimental results and demonstrative pictures show the behaviors of the triangulation methods.

Keywords
generalized barycentric coordinates, image deformation, triangulation

1 INTRODUCTION

Barycentric coordinates are frequently used to represent a point inside a polygon as the weighted sum of its vertices. In the last years, many generalizations (e.g., harmonic coordinates [Jos07], mean value coordinates [Ju05], local coordinates [Zha14], or blended coordinates [Ani17a]) and techniques [Ani16] have appeared with a different set of properties. However, most of them satisfy the *linear reproduction property*; therefore, the barycentric coordinates are often used for different interpolation tasks, e.g., shading, mesh parameterization, and shape [Cas18] or image deformation [Hor06].

To use the generalized barycentric coordinates [Flo15, Hor17, Nie13] for cage based image deformation, we usually have to create a triangulation of the source polygon, which envelops the input image to be deformed. However, this mentioned triangulation is not always self-evident because it can affect the quality of the deformation [Ani17b]. Nowadays, the cage based image deformation algorithms are usually implemented on the GPU and operate with a uniform triangulation (e.g., Delaunay) [Web09]. Although they can create a smooth deformation with several thousands of interior vertices in most cases, we can decrease the number of the triangles to save computation time and cost by using a non-uniform triangulation (see Figure 1).

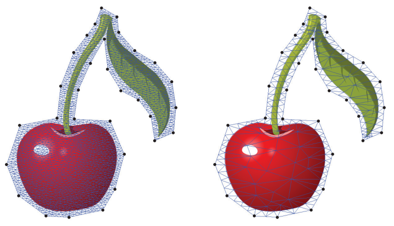


Figure 1: Input image with source polygon using (left) a uniform and (right) a non-uniform triangulation technique. The black dots mark the source polygon defined by the user manually.

Therefore our goal was to examine and compare the two different triangulation techniques in the aspect of the applied cage based image deformation methods. In our comparison, we take into account the input image, the source polygon—which is often determined by the user—, and the used coordinate method. Moreover, we investigate different texture filtering algorithms to improve the quality of the deformation results.

In the next section, we give a short overview of the image deformation methods based on the generalized barycentric coordinates. Then, in Section 3, we discuss the different triangulation techniques. In Section 4, we introduce our non-uniform triangulation algorithm in detail. We show the way how the quality of the deformation results can be improved in Section 5, while in the last two sections, we present our results and future work as well.

2 IMAGE DEFORMATION BASED ON GENERALIZED BARYCENTRIC COORDINATES

As we recalled in Section 1, one of the main application areas of GBC is image deformation.

We can deform an input image I, which is enveloped by a source polygon P with vertices $\mathbf{v}_i$. The source polygon has a triangulation T with vertices $\mathbf{t}_j$. After relocating the source polygon P to P' with vertices $\mathbf{v}'_i$, the new positions $\mathbf{t}'_j$ of the vertices of the triangulation can be computed by the following interpolation function:

$$\mathbf{t}'_j = \sum_{i=1}^{n} b_i(\mathbf{t}_j)\mathbf{v}'_i, \quad (1)$$

where $b_i(\mathbf{t}_j)$ are the barycentric coordinates of the vertex $\mathbf{t}_j$ of the triangulation T respect to the vertex $\mathbf{v}_i$, while n is the number of the vertices of the source polygon P.

The generalized barycentric coordinates $b_i(\mathbf{v})$ of a point $\mathbf{v}$ inside a polygon can be computed by the equations below:

$$\mathbf{v} = \frac{\sum_{i=1}^{n} w_i(\mathbf{v})\mathbf{v}_i}{\sum_{j=1}^{n} w_j(\mathbf{v})}, \quad (2)$$

$$b_i(\mathbf{v}) = \frac{w_i(\mathbf{v})}{\sum_{j=1}^{n} w_j(\mathbf{v})}, \quad (3)$$

where $w_i(\mathbf{v})$ are the homogeneous coordinates. The above mentioned barycentric coordinates are usually computed in a precomputation step before the deformation.

Image deformation based on GBC is widely used in computer graphics because of the straightforward and real-time computation of barycentric coordinates [Ska08]. However, we have to notice that the deformed image depends on the initial and the deformed source polygons, the used coordinate method [Ani19], and the given triangulation as well.

3 GENERATING THE 2D MESH

As we have mentioned in Section 2, defining a triangulation is a crucial task of image deformation based on GBC. After the user marks the desired initial source polygon of the input image (see Figure 1) by defining its vertices manually using a graphical user interface, we have to create a 2D triangular mesh. To create that, we have to partition a given region into simplices which satisfy different criteria. In our case, the triangulation domain of the region is marked by the initial source polygon, and we use shape and size criteria. The shape criterion is an upper bound $\boldsymbol{B}$ on the circumradius-to-shortest edge length ratio, while the size criterion is an upper bound $\boldsymbol{S}$ on the length of the longest edge of triangles.

We used Shewchuk's algorithm [She02]—which is based on the Delaunay refinement method—to produce 2D meshes for the given domain. The algorithm starts with a constrained Delaunay triangulation and inserts new vertices until it satisfies the criteria.

Definition 3.1. A triangulation T is a *Delaunay triangulation* if there exists a circle C for each edge e of T with the following properties:

- the endpoints of e are on the boundary of C, and
- the circle C does not contain another vertex of T in its interior.

Definition 3.2. Let G be a planar straight-line graph (PSLG). A triangulation T of G is a *constrained triangulation* if it contains all edges of G as a part of the triangulation. The edges are called *constrained edges.*

Definition 3.3. A *constrained Delaunay triangulation* (CDT) [Che89] of G is a *constrained triangulation* of the vertices of G, which is as close to the *Delaunay triangulation* as possible.

As we can see in the definitions above, we can construct a uniform triangulated 2D mesh using the Delaunay refinement algorithm if we define G from the edges of the source polygon.

A non-uniform triangulated 2D mesh can be constructed if we define further segments or vertices as constraints in G from the triangulation domain and set a proper criterion.

The modification of shape and size criteria is an excellent way to increase or decrease the number of vertices of the mesh. By default, $\boldsymbol{B} = \sqrt{2}$, which guarantees that the Delaunay refinement algorithm will terminate, but as we increase it, the resolution of the triangulation will be changed. In the same way, the adjustment of $\boldsymbol{S}$ will affect on the constructed mesh (see Figure 2).

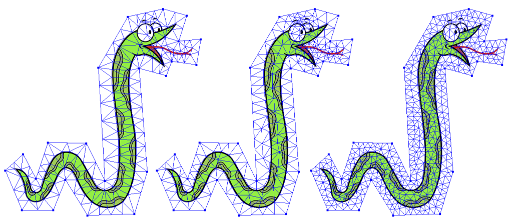


Figure 2: Generated meshes for a given initial source polygon with different parameters. The shape and size criteria are (left) $\boldsymbol{B} = 0.125$; $\boldsymbol{S} = 0.2$, (middle) $\boldsymbol{B} = 0.25$; $\boldsymbol{S} = 0.2$, and (right) $\boldsymbol{B} = 0.125$; $\boldsymbol{S} = 0.05$.

4 OUR NON-UNIFORM TRIANGULATION METHOD FOR CAGE BASED IMAGE DEFORMATION

In the following, we introduce our method, which is able to define an adaptive non-uniform triangulation where the triangles are placed with respect to the input image and the effect of the used coordinate method. Therefore, we can preserve the smoothness of the contour curve of the input image after the deformation, and we can decrease the number of triangles of the triangulation, which results in less computation time. Moreover, the position of the source polygon does not affect the quality of the deformation result.

The inputs of our algorithm are the input shape I and the source polygon P that is often marked by the user manually using a graphical user interface.

4.1 Extracting the contour

After the user defined the source polygon, the input image has to be converted to a binary one, on which a threshold or canny edge detection has to be applied in order to use a contour detection technique, e.g., the method of Suzuki et al. [Suz85]. The input image has to be separable from the background, or it needs to have a coherent black contour, so the contour detection method can work successfully.

In the following part of the paper, we refer to the mentioned contour as $C = \{(x_t, y_t)\}_{t=1}^{N}$, where N is the number of points on the boundary.

4.2 Calculation of curvature

Our algorithm calculates the curvature κ_t of the contour curve of the input shape. The curvature defines the rate of change of the unit tangent vector at a given point, and it can be computed as

$$\kappa(t) = \frac{||\boldsymbol{r}'(t) \times \boldsymbol{r}''(t)||}{||\boldsymbol{r}'(t)^3||}. \tag{4}$$

However, in the discrete world, the contour curve is represented as a chain of segments that are built from a set of points; therefore, we have to associate curvature with vertices. The discrete curvature $\widehat{\kappa}_t$ of the contour curve in vertex $\mathbf{c}_t \in C$ is the change between segments (meeting at $\mathbf{c}_t$) in tangent direction:

$$\widehat{\kappa}_t = \angle\left(\left(\mathbf{c}_t - (\mathbf{c}_{t-1} - \mathbf{c}_t)\right), \mathbf{c}_t, \mathbf{c}_{t+1}\right) = \theta_t, \tag{5}$$

where t is a position in the contour (see Figure 3). The curvature values are higher where the change between segments are high, and it is constant if the chain of segments is almost flat (see Figure 4). Using the curvature values, we can decide in which parts of the curve we have to use a high density of sampled points to follow the shape of the contour curve properly.

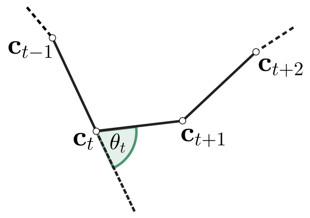


Figure 3: Notations for calculating the discrete curvature.

We notice that our algorithm allows us to use different step sizes for discrete curvature calculation; thus, the noise of the contour curve can be reduced.

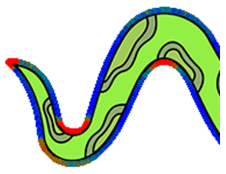


Figure 4: Visualized contour curve curvature. Those parts of the curve where the curvature is high are marked by red.

4.3 Calculation of the effect of the deformation

As we mentioned previously, the deformation results depend on the position of the initial source polygon and the used coordinate method as well (see Figure 5). Therefore, our algorithm calculates the effect $F(\mathbf{c}_t)$ of the nearest vertex of the source polygon P in vertex $\mathbf{c}_t \in C$ by

$$F(\mathbf{c}_t) = \frac{w(\mathbf{c}_t)}{\sum_{k=1}^{n} w_k(\mathbf{c}_t)}. \tag{6}$$

With these values, we can decide which parts of the input image will be deformed better. The effect of deformation is higher in those parts of the input, where the initial source polygon—marked by the user—is close to the image, or the normalized barycentric coordinates values are close to 1.

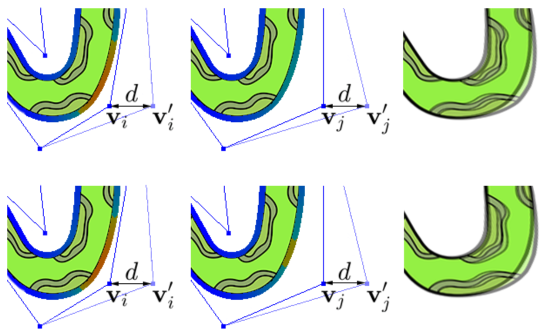


Figure 5: The effect of deformation with different initial source polygons using (top) mean value and (bottom) maximum entropy coordinate methods. The higher the effect of the deformation, the redder the contour. We made deformations with both source polygons, we translated vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ to $\mathbf{v}'_i$ and $\mathbf{v}'_j$ with the same distance d. The comparisons of the deformation results can be seen in the third column. They show us that the deformation depends on the position of the source polygon as well.

4.4 Sampling of the contour curve

An essential step of our algorithm is to define those points of the contour curve, which will be the basis of the non-uniform triangulation. Therefore, we have to sample the contour C with respect to the previously computed $\widehat{\kappa}_t$ and $F(\mathbf{c}_t)$ by

$$r = \begin{cases} r_1 & \text{if } \widehat{\kappa}_t > \Delta\widehat{\kappa} \text{ and } F(\mathbf{c}_t) > \Delta F \\ r_2 & \text{otherwise} \end{cases} \tag{7}$$

$$C' = \{(x_t, y_t) \in C \,|\, t \pmod r = 0\}, \tag{8}$$

where r is the sampling ratio, and $\Delta\widehat{\kappa}$ and ΔF are lower bounds.

Sampling methods for freeform curves [Pag18] are usually used to select sample points from the curve according to some criterion (e.g., curvature, arc length, parameterization, or complexity). Uniform sampling methods for curves are the most popular, but, unfortunately, they are not precise enough in some cases. Other techniques are the adaptive approaches (also called non-uniform sampling), which lead to increased sampled density on those parts of the curve where, e.g., the curvature is high. These techniques give us more efficient approximations.

We use the adaptive technique to determine points to be sampled from the contour curve C with respect to the previously computed curvature and the effect of deformation values (see Figure 6). Our goal is to increase the density of the sampling on those parts of the curve where the curvature and the effect are higher while we want to set the sampling ratio to minimal in other areas. Naturally, other properties can be examined as well.

In our algorithm, we set r_1 to 4, r_2 to 20, ΔF to 0.7, and $\Delta\widehat{\kappa}$ is exactly the average curvature of the contour curve. In that way, we sample every fourth point of the curve if the curvature is greater than the average and the effect of the deformation is greater than 0.7. Otherwise, we sample every twentieth point. We notice that these values highly depend on the resolution of the input image. We worked with full HD images, but if we want to use larger images, we have to increase the values r_1 and r_2.

In addition to all of this, the r_1, r_2, $\Delta\widehat{\kappa}$, and ΔF values in Equation 7 are modifiable freely; thus, the resolution of the approximation can be increased and decreased.

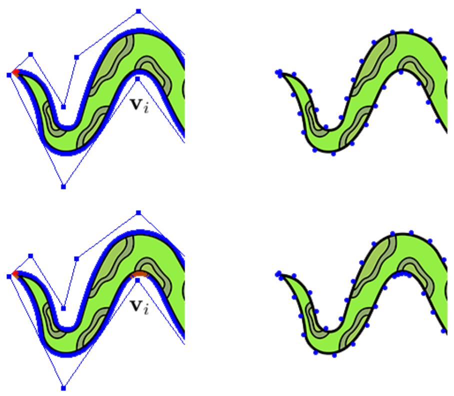


Figure 6: The sampling of the contour curve with the same initial source polygon using (top) mean value and (bottom) metric coordinates. The contour is colored with respect to the curvature and the effect of deformation. We increased the sampled density near to vertex $\mathbf{v}_i$ (bottom) because the metric coordinates has a higher effect on the input image than the mean value.

After the sampling, we have to consider the sampled points as the constraints of a constrained Delaunay triangulation; thus, a non-uniform triangulated 2D mesh can be constructed as we discussed in Section 3. The edges of the generated triangulation follow the contour curve of the input shape. Moreover, in those parts of the triangulation where the curvature and the effect of the source polygon are higher, there are more triangles, while in other areas, our method minimizes the number of triangles.

4.4.1 Holes in the mesh

Another advantage of the non-uniform triangulation over the uniform that it can handle holes in the generated mesh. The user has the opportunity to define holes—which are simple polygons H_i—in the interior of the initial source polygon. In that case, the algorithm uses the edges of H_i as constraint edges in the constrained Delaunay triangulation.

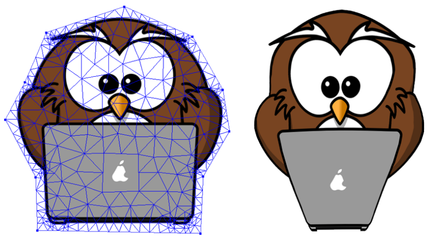


Figure 7: Generated mesh with a hole for an input image and its deformation result.

This solution can be very useful if we want to deform images with logos, texts, or parts which should not be damaged (see Figure 7).

4.5 The step by step process of our non-uniform triangulation algorithm

We can summarize our method in the following steps:

1. Consider the input shape I, the source polygon P, and the hole polygons H_i.
2. Extract the contour $C = \{(x_t, y_t)\}_{t=1}^{N}$ of the input.
3. Calculate the curvature $\widehat{\kappa}_t$ of the contour curve.
4. Calculate the effect $F(\mathbf{c}_t)$ of the nearest vertex of the source polygon P in vertex $\mathbf{c}_t \in C$.
5. Sample the contour curve C respect to $\widehat{\kappa}_t$ and $F(\mathbf{c}_t)$.
6. Generate a 2D mesh obtained from a constraint Delaunay triangulation using C' and H_i.

5 IMPROVING THE QUALITY OF THE DEFORMATION

The image deformation applications based on the barycentric coordinates usually use the OpenGL library to render the deformation. Therefore, as we discussed in Section 2, we have to triangulate the initial source polygon, then we have to apply the input image to it as a texture. In order to draw the texture, OpenGL executes a sampling operation, when the texture coordinates of the vertices are mapped to the texture image. The result of the sampling are texels, which are pixels in the texture image. These texels often contain color information; thus, the final color of the corresponding pixel can be defined. However, it can happen that a computed texture coordinate will not match a texel exactly. In these cases, OpenGL has to figure out which texel corresponds to the texture coordinate. The method, when OpenGL decides the final texel value, is called texture filtering. During the deformation, we usually do many transformations with the vertices of the initial source polygon; therefore, we have to choose well the texture filtering algorithm. The most used filtering methods are the `GL_NEAREST` and `GL_LINEAR`, but these cannot produce good image quality in most cases, e.g., if we stretch the image to be deformed. The `GL_NEAREST` interpolation selects the closest pixel to the texture coordinates; therefore, the resulting images will be blocky. The `GL_LINEAR` method interpolates the closest four pixels, and it creates smoother images than the previous interpolation technique, but staircase effect may appear at the edges of the image.

Using the OpenGL Shading Language (GLSL), we have the opportunity to write shaders on the GPU, where we can use our own filtering methods. In order to increase the quality of the deformation results, we implemented a generalized bicubic interpolation method [Pra13] by the following equation:

$$F(p',q') = \sum_{m=-1}^{2} \sum_{n=-1}^{2} F(p+m, q+n) \cdot R_C\{(m-a)\} R_C\{-(n-b)\}, \tag{9}$$

where the pixel $F(p,q)$ is the nearest to the pixel to be interpolated, a and b are distances between the previously mentioned pixels, while $R_C(x)$ is a bicubic interpolation function such as triangle, cubic B-spline, or Catmull-Rom interpolation.

While the `GL_LINEAR` method uses the nearest four pixels to define the color of the intermediate pixel, the bicubic interpolation uses the nearest sixteen pixels (see Figure 8). Using a texture filtering method based on a bicubic interpolation function, the quality of the deformation can be further improved, as we can see in Figure 9.

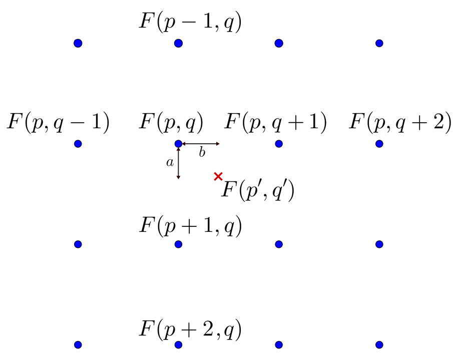


Figure 8: Notations for the bicubic interpolation.

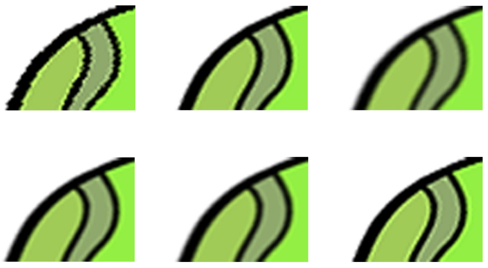


Figure 9: From left to right, top to bottom: deformation result is rendered by `GL_NEAREST`, `GL_LINEAR`, triangle, B-spline, bell, and Catmull-Rom texture filtering methods.

6 RESULTS

6.1 Implementation

We implemented the prototype of the uniform and our non-uniform triangulation algorithms in C++ on an Intel i7 6700K CPU at 4.0 GHz with 16 GB memory. In the case of non-uniform triangulation, to extract the contour of the input, we used the method of Suzuki et al. [Suz85], which is implemented in the OpenCV library [Bra00]. In order to produce a 2D mesh for a source polygon using the extracted contour points, we have to build a constrained Delaunay triangulation then we have to use the Delaunay refinement algorithm on it. In the case of uniform triangulation, the mentioned algorithms can be useful as well. These methods can be found in the CGAL library [The19]. The implementation of the coordinate methods was based on the CGAL library and the Ph.D. thesis of Anisimov [Ani17b].

6.2 Validation and comparison

We tried the methods on many input images, and we compared the deformation results using uniform and non-uniform triangulations (see Figure 10). We used different coordinate methods for the deformations: mean value, maximum entropy, metric, Poisson, and blended coordinates. We can say that the uniform and the non-uniform triangulations works well with every coordinate method.

The introduced non-uniform triangulation method only takes care of the outer contour of the input images. However, in those situations when we want to deform, e.g., cartoon figures—which interiors are not so detailed—, images with big homogeneous regions, it produces smooth deformation results. Furthermore, it can preserve the boundary of the image after the deformation. Moreover, if the graphical resources of the used platform or device are limited, or we want to save computation cost by reducing the number of triangles, the non-uniform method is usable as well.

Although, if we can use a GPU implementation of the cage based image deformation methods, and can increase the number of the triangles up to several thousands, the uniform triangulation can produce smooth enough deformation results as well.

To validate the algorithms, we also created a pixel-based deformation (see the middle column of Figure 10) for the inputs. It can be used as a reference resulting image because we computed the barycentric coordinates of all the pixels of the input image in the precomputation step, then we computed new positions of them in the deformation phase. In the case of large deformations, it can produce holes because there are regions which are not covered by deformed pixels, but using a filtering algorithm, the result will be fully contiguous without holes.

We used the same deformation polygon P' to get three resulting images based on the uniform triangulation, the non-uniform triangulation, and the pixel-based one. For the comparison, we measured the computation times of the triangulations and SSIM (Structural Similarity Index) [Wan04] values. The SSIM value is frequently used to define the similarity of two images $\mathbf{X}$ and $\mathbf{Y}$ with the following formula:

$$\text{SSIM}(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha} \cdot [c(\mathbf{x},\mathbf{y})]^{\beta} \cdot [s(\mathbf{x},\mathbf{y})]^{\gamma}, \quad (10)$$

where $\mathbf{x}$ and $\mathbf{y}$ are square windows of $\mathbf{X}$ and $\mathbf{Y}$ (located at the same spatial position). The $l(\mathbf{x},\mathbf{y})$, $c(\mathbf{x},\mathbf{y})$, and $s(\mathbf{x},\mathbf{y})$ are luminance, contrast, and structure comparison functions, respectively. $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ are parameters which define the relative importance of the three comparison functions. The SSIM value can be between 0 and 1. The value 1 means that the two images are similar, while the value 0 means that they are very different. Table 1 summarizes the computation times of the triangulation on CPU and the SSIM values between the two triangulations on different input images and the number of triangles. In the case of the *Snake* input image, the two triangulations have almost the same number of triangles, and the same computation times, while the non-uniform solution increases the SSIM value. In the

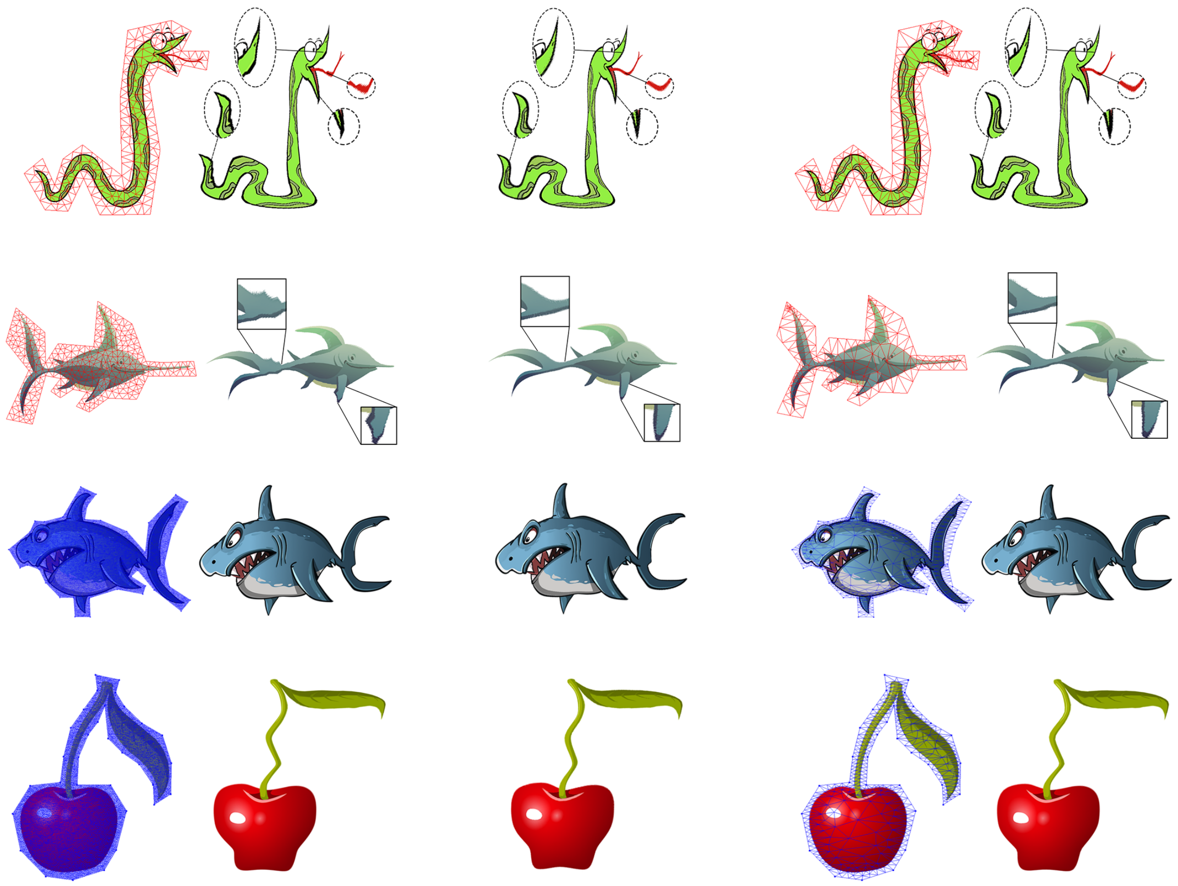


Figure 10: Input images with uniform triangulation on the left column and with non-uniform triangulation on the right column, and their deformation results using mean value coordinates, while the reference resulting images are in the middle.

	Snake		Swordfish		Shark		Cherry	
	u	nu	u	nu	u	nu	u	nu
Triangles	340	335	638	360	29926	1342	16185	715
SSIM	0.956	0.959	0.978	0.979	0.939	0.938	0.984	0.984
Computation time (s) of the triangulation	0.001	0.001	0.003	0.001	1.898	0.008	0.563	0.003

Table 1: The comparison of the uniform (u) and the non-uniform (nu) triangulations.

case of the *Swordfish*, the non-uniform method uses much fewer triangles than the uniform triangulation, so it decreases the computation time and increases the SSIM value. In the case of the *Shark* and the *Cherry*, the uniform method uses ≈30k and ≈16k triangles; thus, the computation times are increased, while the two types of triangulations have almost the same SSIM value. More comparison figures and values can be seen in our supplementary material.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have shown how the different triangulation techniques can be used for cage based image deformation using the generalized barycentric coordinates. Our non-uniform triangulation method was introduced in detail as well. The algorithm takes into consideration the position of the source polygon and the used coordinate method; moreover, it can handle holes in the interior of the polygon. The main difference from the earlier systems that the edges of the non-uniform triangulation follow the contour of the input shape, therefore it can preserve the smoothness of the curves of the input. We compared the uniform and the non-uniform triangulation methods as well. The non-uniform one can decrease the computation time by minimizing the number of triangles of the triangulation on those parts where the effect of the deformation is lower. The uniform triangulation method can produce smooth deformation results by an increased number of triangles. The quality of the deformation can be increased by using different texture filtering methods. Based on our results, we can say that the non-uniform triangulation is more efficient in cases when we work with fewer triangles, and it can produce more accurate deformation results than the methods using the uniform one. Furthermore, we have to notice that the deformation depends on the deformed source polygon as well because the barycentric coordinates are computed with respect to the vertices of the source polygon, which are not changed after the precomputation.

In our non-uniform triangulation, the vertices on the boundary of the hole polygon do not change positions in the deformation phase, and it can lead to deformation errors around the hole in some cases. In order to solve this problem, further investigation is required in the future.

8 ACKNOWLEDGEMENT

This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

9 REFERENCES

[Ani16] Anisimov, D., Deng, C., and Hormann, K. Subdividing Barycentric Coordinates, Comput. Aided Geom. Des. 43 pp.172–185, 2016.

[Ani17a] Anisimov, D., Panozzo, D., and Hormann, K. Blended barycentric coordinates, Comput. Aided Geom. Des. 52–53 pp.205–216, 2017.

[Ani17b] Anisimov, D. Analysis and New Constructions of Generalized Barycentric Coordinates in 2D, Ph.D. thesis, Universitá della Svizzera italiana, 2017.

[Ani19] Anisimov, D., Hormann, K., and Schneider, T. Behaviour of exponential three-point coordinates at the vertices of convex polygons, J. Comput. Appl. Math. 350 pp.114–129, 2019.

[Bra00] Bradski, G. The OpenCV Library, Dr. Dobb's Journal of Software Tools 2000.

[Cas18] Casti, S., et al. CageLab: an Interactive Tool for Cage-Based Deformations. In *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference 2018*, pp.65–74. Eurographics, 2018.

[Che89] Chew, L. P. Constrained Delaunay Triangulations, Algorithmica 4 pp.97–108, 1989.

[Flo15] Floater, M. S. Generalized barycentric coordinates and applications, Acta Numer. 24 pp.161–214, 2015.

[Hor06] Hormann, K., and Floater, M. S. Mean Value Coordinates for Arbitrary Planar Polygons, ACM Trans. Graph. 25 No.4 pp.1424–1441, 2006.

[Hor17] Hormann, K., and Sukumar, N. (eds.). Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics. CRC Press, 2017.

[Jos07] Joshi, P., et al. Harmonic Coordinates for Character Articulation, ACM Trans. Graph. 26 No.3 pp.71:1–71:9, 2007.

[Ju05] Ju, T., Schaefer, S., and Warren, J. Mean Value Coordinates for Closed Triangular Meshes, ACM Trans. Graph. 24 No.3 pp.561–566, 2005.

[Nie13] Nieto, J. R., and Susín, A. Cage Based Deformations: A Survey. In González Hidalgo, M., Mir Torres, A., and Varona Gómez, J. (eds.) *Deformation Models: Tracking, Animation and Applications*, pp.75–99. Springer, Dordrecht, 2013.

[Pag18] Pagani, L., and Scott, P. J. Curvature based sampling of curves and surfaces, Comput. Aided Geom. Des. 59 pp.32–48, 2018.

[Pra13] Pratt, W. K. Introduction to Digital Image Processing, CRC Press, 2013.

[She02] Shewchuk, J. R. Delaunay refinement algorithms for triangular mesh generation, Comput. Geom. 22 No.1-3 pp.21–74, 2002.

[Ska08] Skala, V. Barycentric coordinates computation in homogeneous coordinates, Comput. & Graph. 32 No.1 pp.120–127, 2008.

[Suz85] Suzuki, S., and Abe, K. Topological Structural Analysis of Digitized Binary Images by Border Following, Comput. Vis. Graph. Image Process. 30 No.1 pp.32–46 1985.

[The19] The CGAL Project. CGAL User and Reference Manual. CGAL Editorial Board, 4.14 edition, 2019.

[Wan04] Wang, Z., et al. Image Quality Assessment: From Error Visibility to Structural Similarity, IEEE Trans. Image Process. 13 No.4 pp.600–612, 2004.

[Web09] Weber, O., Ben-Chen, M., and Gotsman, C. Complex Barycentric Coordinates with Applications to Planar Shape Deformation. Comput. Graph. Forum 28 No.2 pp.587–597, 2009.

[Zha14] Zhang, J., et al. Local Barycentric Coordinates, ACM Trans. Graph. 33 No.6 Article 188. 2014.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.28, No.1-2, 2020

Ultrawide Field of View by Curvilinear Projection Methods

Jonah Napieralla
Blekinge Institute of Technology
Valhallav. 1
SE-371 79 Karlskrona, Sweden
jonah.napieralla@gmail.com

Veronica Sundstedt
Blekinge Institute of Technology
Valhallav. 1
SE-371 79 Karlskrona, Sweden
veronica.sundstedt@bth.se

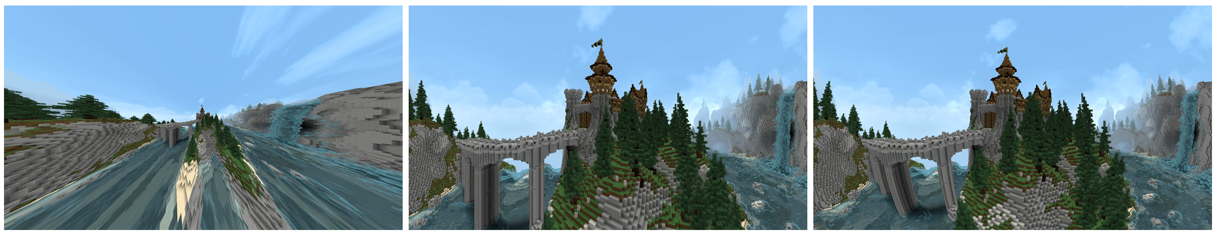

(a) Perspective **(b)** Panini **(c)** Stereographic

Figure 1: One view, rendered at 170 degrees of field of view using different projection methods (a, b, c). Scenery provided by FALL Studios.

ABSTRACT

The rectilinear Perspective projection produces natural-looking results on the condition that the degree of field of view (FoV) is narrow, as raising it causes an exponential increase in visual distortion. Curvilinear perspective projection methods that counter this issue exist in photography, but unlike the rectilinear Perspective projection, these are neither used nor technically documented in computer graphics. This paper contributes by presenting results from a perceptual experiment comparing the rectilinear Perspective projection method to the curvilinear Panini and Stereographic projection methods. These are commonly used with wide-angle lenses in photography and have been digitally recreated for use in computer graphics. The experiment shows a clear preference for the two curvilinear projection methods at high degrees of FoV, as not a single participant prefers the rectilinear Perspective projection at degrees of FoV approaching the human breadth of vision.

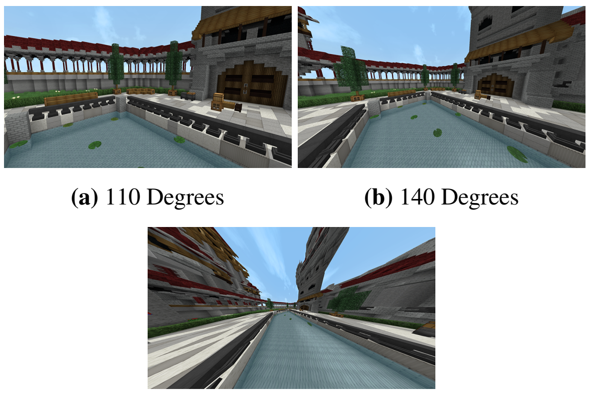
Keywords

Computer graphics, Perception, Human computer interaction (HCI), Graphical projection methods, Field of view, Perspective, Panini, Stereographic, Rectilinear, Curvilinear.

1 INTRODUCTION

The rectilinear Perspective projection method has always been the standard in realistic computer renditions. Of the projection methods made readily available by the common 3D graphics libraries, it is the only one that takes the user's positional perspective into account, a necessity for a natural look. But it does suffer from an unaddressed shortcoming; as the degree of FoV increases, it experiences increasing amounts of distortion that leads to the rendition eventually becoming unrecognizable [Yan08], as demonstrated in Figure 2. The consequences of this have not yet been realized because the current hardware (computer monitors and TV screens) has not incentivized anything but unrealistically low degrees of FoV. But as ultrawide monitors and head-mounted VR-goggles attempt to push the limits of immersive digital experiences, the shortcomings of the Perspective projection method will have to be addressed.

Higher degrees of FoV have been shown to benefit the user in many ways, by, for example, allowing them to perform visual search tasks more effectively [Osm14], which makes them popular in gaming. In one of the most popular competitive 3D games at the time of writing - Overwatch - almost 99% of professional players use the highest degree of FoV available to them [Pro], a trend that spans many competitive games. As computer monitors have become wider, many applications have also started automatically setting the FoV higher for users of widescreen-monitors [Wgf], as monitors that cover a greater portion of the user's view are shown to merit the use of higher degrees of FoV [Ste11]. Assuming that the current trend continues; larger screens that encompass a greater portion of the user's view will become more prevalent in the future, and this increase in size will warrant increasingly higher degrees of FoV. However, that will only be feasible if the picture can be drawn without noticeable distortion. This issue merits the evaluation of alternative projection methods, and of whether they look natural enough to enable the use of high degrees of FoV.


(a) 110 Degrees **(b)** 140 Degrees

(c) 170 Degrees

Figure 2: Difference between 110, 140, and 170 degrees of FoV, using the Perspective projection method. Demonstrating the exponential increase in distortion. Scenery provided by Elysium Fire.

2 OVERVIEW OF PROJECTION METHODS

In the process of converting the view of a virtual 3D world into a 2D rendition; projection methods are used to map points from the 3D world onto points on a 2D plane that can thereafter be drawn on a physical display. What differentiates projection methods is how they distribute these points onto the plane, as the resulting projection is produced. FoV is a parameter of these projection methods. It is defined by a numeric degree that specifies the angle from the left to the right extent of the view; the breadth of vision of the virtual camera.

The key difference between the Perspective projection and the two alternative projection methods evaluated, the Panini and the Stereographic projections, is that the Perspective projection is rectilinear while the Panini and the Stereographic projections are curvilinear, as visualized in Figure 3. What defines a rectilinear projection method is that it keeps the path between every pair of points in 3D space straight in the resulting projection onto the 2D plane, producing no unnatural curvature of straight lines. Curvilinear projection methods do not share this trait, as they bend lines that do not pass through the center of the projection. What differentiates the Panini and the Stereographic projection method is that the Panini projection only bends horizontal lines, whereas the Stereographic projection is fully curvilinear and will bend both horizontal and vertical lines. The Stereographic projection method also has the uncommon characteristic of being conformal, which implies that the angle at which curves intersect will be correctly preserved in the projection, despite other kinds of distortions.

The Panini projection method [Sha10] was included in this study because it had received very high ratings in a recent user study comparing projection methods using still pictures [Kim17]. The Stereographic projection method was included for its characteristics, that contrast strongly with the Perspective projection, making for an interesting comparison, and because it was commended as the best choice for general-purpose use in photography at high degrees of FoV, in a qualitative analysis of various projection methods [Fle95].

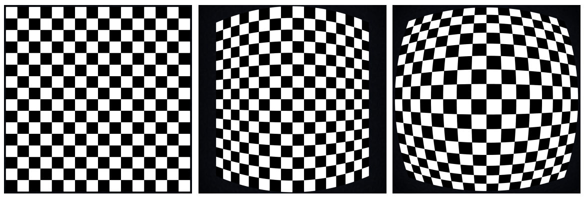

(a) Perspective **(b)** Panini **(c)** Stereographic

Figure 3: Visualization of the point distribution and curvature of the three projection methods [Nap18].

3 METHOD

To assess the naturalness of the curvilinear Panini and Stereographic projections, and how they compare to the Perspective projection at high degrees of FoV, they were each rated in perceptual evaluations, by volunteering participants and then compared through a statistical analysis. The following sections present the data gathering process. For more information see [Nap18].

3.1 Design

The three projection methods were assessed in tests that simulated the most common usage scenarios of 3D software-applications, using a variety of virtual scenery that was intended to assess a wide enough spectrum of potential use. These tests were grouped into the following usage scenarios:

- **Photographic** tests would present regular still pictures. These would allow the projection methods to be assessed more precisely, as it allowed the participants to carefully analyze the aesthetic qualities of each rendition.
- **Cinematographic** tests would consist of the camera moving, in both position and rotation, along a predefined path through a scene. Moving too fast for minute details to be discernible, these tests would instead show how points in space flow across the screen.
- **Interactive** tests would consist of the camera being fixed to a set vantage point, allowing only rotation, by the movement of the computer mouse. These would allow the participants to consider the feeling of actively engaging with an application employing the specific projection methods.

Because the photographic tests were expected to take the least time to rate, and the interactive tests the most time; four different sceneries were used in evaluating the photographic tests, three in the cinematographic, and two in the interactive. Each scenery would be employed in nine tests, as the Perspective, Panini, and Stereographic projection method each were evaluated at 110, 140, and 170 degrees of (horizontal) FoV, at a 16:9 aspect ratio. These degrees were chosen because it was expected that those below 110 would show too little distortion, and those above 170 too much distortion, to have produced valuable results.

3.2 Rendering

The experiment-application was built on top of a modified version of the game Minecraft [Moj], adjusted to fit the demands of the perceptual experiment. This approach was chosen because using a preexisting implementation of the projection methods would facilitate the development of the experiment, and an open-source rendering modification [Git], that had implemented the projection methods to be evaluated in this study, existed only for this application. To produce the curvilinear Panini and Stereographic projections, the rendering modification had to use an uncommon, and computationally less efficient approach, compared to rendering with the rectilinear Perspective projection. This involved first rendering five square views around the position of the camera, using the Perspective projection method; one in the direction that the camera was facing, as well as one 90 degrees to the left, to the right, above, and below that view, as shown in Figure 4. These views would all originate from the position of the camera, but they were rotated to be facing different directions. They were then sent to a fragment shader specific to the desired projection method, where they formed a theoretical cube around the origin that was ray-casted to for each pixel of the screen. Where the ray intersected with one of the view-planes, it sampled a color to output for that pixel. This process faithfully reproduced the Panini and the Stereographic projection from views rendered with the Perspective projection method, as the outcome would be virtually identical to ray-casting directly into the complex environment, but computationally a lot more efficient.

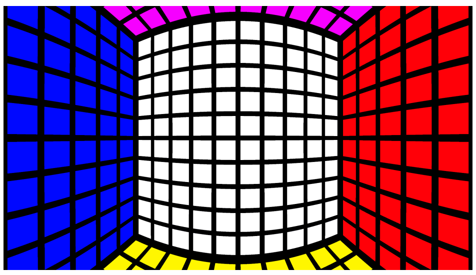


Figure 4: Picture taken from inside the cube that is ray-cast to produce the two curvilinear projection methods, shown using the Panini projection [Nap18].

3.3 Perceptual Evaluation

To assess each projection method at every degree of FoV; a perceptual experiment was conducted in which every participant rated a total of 81 tests, presented in randomized order. The average time to rate all tests took approximately 14.5 minutes. These were displayed on a 23-inch monitor, at an approximate distance of 71 centimeters from the participants' eyes, a distance that each participant was asked to maintain by keeping their head approximately aligned with a given point of reference. One test would be displayed at a time, consisting of a view that filled the entire screen, and it would be rated from within the experiment-application. Participants were asked the same question for each test; *How unnatural (1) or natural (5) does the current projection look to you?*, which was rated using a 5-point Likert scale. A total of 24 people took part, whose ages ranged from 18 to 29 years, and who used 3D software-applications at an average of approximately once a week.

4 RESULTS

Every projection method would generally be rated less favorably when the degree of FoV increased, as that would always increase distortion, but the Perspective projection method was affected more significantly by this increase than the Panini and Stereographic projections were. The drop in ratings can, for the most part, be considered exponential, as there was a lesser difference in ratings between 110 and 140 degrees than between 140 and 170 degrees. Only the Perspective projection would drop close to the lowest possible mean rating, as it fell from the universally highest ratings at 110 degrees to the lowest at 170 degrees, shown in Figure 5. While the Perspective projection method received relatively even mean ratings across the three usage scenarios, ratings of the Panini and Stereographic projections were less consistent. In the photographic and cinematographic tests, the Panini projection was rated only marginally lower than the Perspective projection at 110 degrees, with no significant difference between them ($p > 0.05$) at that degree of FoV, but in the interactive tests, it dropped to unusually low ratings at every degree of FoV. The Stereographic projection received lower ratings than the Panini projection in the photographic tests, and in the cinematographic tests, its ratings fell further as those of the Panini projection rose. Only in the interactive tests would the Panini and Stereographic projections be rated similarly, with no significant difference between the two ($p > 0.05$) at any degree of FoV. There was also no significant difference between the Perspective and Stereographic projections in the cinematographic tests at 140 degrees. Apart from the aforementioned results; there was a significant difference ($p < 0.05$) between ratings of every projection method employed in the same usage scenario and at the same degree of FoV. All statistical significance tests were computed using a Pairwise Wilcoxon signed-rank test.

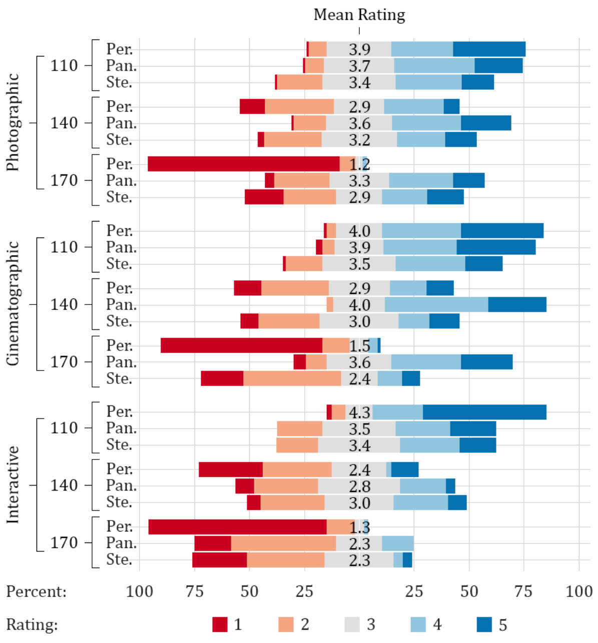


Figure 5: Distribution of ratings across tests visualized by their respective colors, for every projection method and at every degree of FoV, separated by scenario.

The mean change in individual participants' rating of a projection method, when only the scenery changed but not the projection method or FoV, was (0.66, 0.92, 1.20) for the Perspective, Panini, and Stereographic projection methods, respectively. Showing that the favorability of the curvilinear projection methods was more dependent on the scenery displayed. The distribution of individual participants' preferred projection method, across the three usage scenarios, calculated by counting every participant's ratings separately, was (63%, 31%, 6%) at 110 degrees, (19%, 69%, 12%) at 140 degrees, and (0%, 90%, 10%) at 170 degrees, for the Perspective, Panini, and Stereographic projection methods, respectively. These differ from the mean ratings, as, for example, the Stereographic projection received higher overall ratings than the Perspective projection at 140 degrees, but was less often a preferred projection method.

5 DISCUSSION

Every usage scenario would be tested using a number of different sceneries, and ratings of the Panini and the Stereographic projection would vary significantly across these. The proximity of the foremost objects in the scenery appeared to be the pivotal aspect, as sceneries with objects in close proximity to the camera were rated unusually poorly. This is likely due to the curvilinear nature of these two projection methods, which arguably becomes much more noticeable when objects of distinct form take up a large portion of the view. When a distinct shape, such as a straight line, is distorted, the user will easily notice the abnormality as it deviates from their expectations of what something is supposed to look like, as the curved posts in Figure 6 (left). When objects lie close to the camera; shapes are likely to dominate the look of the scenery. Whereas if the foremost objects lie at a greater distance from the camera, then composition will dominate, and a composition of objects tends to be irregular, which renders any distortion less noticeable, as in Figure 6 (right).

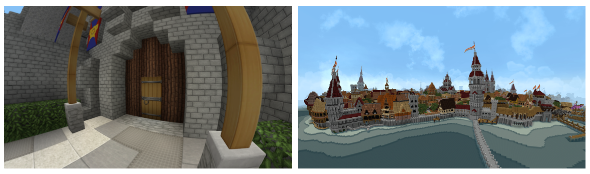


Figure 6: Difference between a focus on objects (left) and a focus on composition (right), demonstrated using the curvilinear Stereographic projection method at 170 degrees of FoV.

Scenery provided by Elysium Fire.

The further a point lies from the center of the screen, the more it is distorted. So when the display takes up only a small portion of the user's view, the distortion will remain very noticeable, as it lies physically close to the assumed center of focus, the center of the screen. However, when the display takes up a large portion of the user's view, then the distortion will lie physically further away from the center of focus, within the user's peripheral vision, where it might be less perceptible. Any unnatural curvature caused by either the Panini or the Stereographic projection method might then bother the user less. This theory would however not apply to the Perspective projection, as it suffers from a further problem; it distorts the proportions of the entire view. When the degree of FoV increases, parts of the view must shrink, as they must take up relatively less space on the screen to fit that which becomes visible with the additional degrees of FoV. However, the Perspective projection does not shrink the earlier view by a proportionally accurate amount. It shrinks it to a disproportionately small size, leaving the outer parts stretched beyond the proportions of the central parts which become inappropriately small, as illustrated in Figure 7.

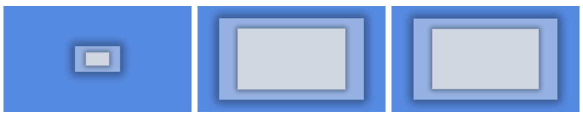

(a) Perspective **(b)** Panini **(c)** Stereographic

Figure 7: Proportional differences between 110, 140, and 170 degrees of FoV, from inner to outer rectangle, illustrating a disparity between projection methods.

6 CONCLUSIONS AND FUTURE WORK

The Perspective, Panini, and Stereographic projection methods were evaluated in simulations of common usage scenarios, using different virtual scenery, by 24 volunteering participants of a perceptual experiment. The Perspective projection was shown to be the optimal choice at narrow degrees of FoV, as it was rated the most favorably in every test at 110 degrees, although often with an insignificantly small margin to the Panini projection. Because the Perspective projection was much more significantly affected by every increase in the degree of FoV, its ratings fell past the two curvilinear projection methods at 140 degrees. The Panini and the Stereographic projection were favored at 140 and 170 degrees, but their ratings depended largely on the usage scenario within which they were employed, as well as on the virtual scenery that was displayed. While the use of curvilinear projection methods has been justified, they will first become practically applicable when they can be computed efficiently enough not to inhibit the effective use of the applications they are employed in. Unless dedicated ray-tracing hardware becomes more prevalent in consumer devices, an investigation of alternative approaches to rendering with curvilinear projection methods, and the computing overhead associated with them, is merited. In addition, a further exploration of more realistic interactive experiences and spatial navigation tasks would be appropriate.

7 REFERENCES

[Fle95] Fleck, M. Perspective Projection: The Wrong Imaging Model. Department of Computer Science, University of Iowa, Technical Report TR 95-01, 1995.

[Kim17] Kim, Y.W. and Lee, C.-R. and Cho, D.-Y. and Kwon, Y.H. and Choi, H.-J. and Yoon, K.-J. Automatic Content-aware Projection for 360 Videos. 2017 IEEE International Conference on Computer Vision (ICCV), pp.4753-4761, 2017.

[Moj] Mojang Synergies AB. Official Minecraft Website: https://minecraft.net. Online, accessed 27-February-2020.

[Osm14] Malla Osman, Z. and Dupire, J. and Topol, A. and Cubaud, P. A Non Intrusive Method for Measuring Visual Attention Designed for the Study and Characterization of Users' Behavior in Serious Games. IARIA, International Journal On Advances in Internet Technology Vol. 7, pp.262-271, 2014.

[Pro] ProSettings.net. Statistics on Video Settings of Professional Overwatch Players: https://www.prosettings.com/overwatch-pro-settings. Online, accessed 27-February-2020.

[Sha10] Sharpless, T. K. and Postle, B. and German, D. M. Pannini: a new projection for rendering wide angle perspective images. Eurographics Association, Proceedings of the Sixth international conference on Computational Aesthetics in Graphics, Visualization and Imaging, pp. 9-16, 2010.

[Ste11] Steinicke, F. and Bruder, G. and Kuhl, S. Realistic Perspective Projections for Virtual Objects and Environments. ACM Trans. Graph., pp.112:1-112:10, 2011.

[Wgf] Widescreen Gaming Forum. Widescreen Gaming Forum article about Hor+ Technique: http://www.wsgf.org/article/screen-change. Online, accessed 27-February-2020.

[Yan08] Yankova, A. and Franke, I. Angle of View vs. Perspective Distortion: A Psychological Evaluation of Perspective Projection for Achieving Perceptual Realism in Computer Graphics. ACM, Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization, pp. 204-204, 2008.

[Git] 18107. Render 360 Mod GitHub Repository: https://github.com/18107/MC-Render360. Online, accessed 27-February-2020.

[Nap18] Napieralla, J. Comparing Graphical Projection Methods at High Degrees of Field of View. Blekinge Institute of Technology, Bachelor Thesis, 2018.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.28, No.1-2, 2020

Improved Lossless Depth Image Compression

Roland Fischer Philipp Dittmann Christoph Schröder Gabriel Zachmann

University of Bremen, Germany
{rfischer, dittmann, schroeder.c, zach}@cs.uni-bremen.de

ABSTRACT

Since RGB-D sensors became massively popular and are used in a wide range of applications, depth data compression became an important research topic. Live-streaming of depth data requires quick compression and decompression. Accurate preservation of information is crucial in order to prevent geometric distortions. Custom algorithms are needed considering the unique characteristics of depth images. We propose a real-time, lossless algorithm which can achieve significantly higher compression ratios than RVL. The core elements are an adaptive span-wise intra-image prediction, and parallelization. Additionally, we extend the algorithm by inter-frame difference computation and evaluate the performance regarding different conditions. Lastly, the compression ratio can be further increased by a second encoder, circumventing the lower limit of four-bit per valid pixel of the original RVL algorithm.

Keywords
Lossless Compression, Depth Image Compression, RGB-D Streaming, Azure Kinect

1 INTRODUCTION

Over the past years, RGB-D sensors became an important topic in many research areas, including computer graphics and computer vision. Their popularity increased enormously when Microsoft released its first Microsoft Kinect, an inexpensive, small, and easy-to-use RGB-D camera, which captures rectangular color images and corresponding depth images. Since then, even smaller RGB-D sensors were developed, which are now integrated into many devices. Prominent examples are the Intel RealSense RGB-D cameras, the subsequently released Kinect V2, and Azure Kinect cameras from Microsoft, whose sensors can also be found in the augmented reality headsets HoloLens 1 and 2, and Lidar scanners. Depth images can also be calculated by using two RGB cameras and semi-global matching (SGM).

Common applications are, for example, telepresence [BKKF13, CK12, SST+18], Virtual/Augmented reality (VR/AR) applications with remote sensors capturing real word scenes [JSM+14], gesture and object recognition and tracking [CJK15], 3D reconstruction [NIH+11, WMS16] and SLAM (simultaneous localization and mapping) [MT17, WKJ+15].

In many cases, the data needs to be streamed over a network beforehand, which runs into the problem of limited network bandwidth, i.e., 100 Mbit/s or 1 Gbit/s for Ethernet, or 300-450 Mbit/s for 802.11n WiFi. The sensors accumulate a large amount of data, and the bandwidth becomes a limiting factor quickly, especially if multiple sensors are combined for better coverage. For instance, a sensor with Full HD color resolution and the depth with a resolution of 512×424 (Kinect V2) produces more than 6.6 MB of raw data per frame. At a typical frame rate of 30 Hz, the network would have to support at least 1.60 Gbit/s. More recent depth sensors often support even higher resolutions. For example, the new Azure Kinect is able to capture color, depending on the mode, up to a resolution of 4K, and has a one-megapixel depth sensor.

Data compression is essential in order to reduce the required bandwidth. This enables lower-bandwidth scenarios, makes room for other payloads, and increases the number of possible cameras. Individual compression of color and depth images is, in general, computationally less expensive than compressing reconstructed 3D data like point clouds or surfaces. Therefore, this approach is often preferred for real-time applications [LBW+15]. The color component of RGB-D sensors can easily be compressed with standard image and video compression algorithms like JPEG [Wal92] or H.264 [WSBL03] as they are optimized precisely for this task.

However, applying the same encoders to the depth data would often result in sub-optimal compression performance or, more crucial, severe artifacts, and geometric distortions [Wil17]. The reason for this is that depth data usually is represented in a different format, 13 or 16-bit single channel, and has inherently different characteristics than natural color images. In general, depth images consist of more homogeneous regions with abrupt discontinuities at object borders. In addition, individual not-captured pixels and regions of invalid pixels are scattered throughout the image, if not filtered beforehand.

In recent years, the research and development of compression algorithms specially designed for depth images became an important topic. Creating efficient and, at the same time, geometry-preserving (lossless) ones is still a very active area of research.

Our work focuses on analyzing and improving the novel RVL algorithm [Wil17], especially regarding achieving higher compression ratios. More detailed, our main contributions are:

- An improved adaptive intra-image prediction step for the RVL algorithm, enhancing its compression ratio.
- The addition of a final entropy-coder stage, further enhancing the compression ratio.
- A multi-threaded implementation of the RVL algorithm and variants speeding it up further.
- An additional inter-frame delta step as well as an examination of its effectiveness regarding the compression ratio over different application scenarios.
- Extensive experiments comparing the effectiveness of various lossless depth compression algorithms for different range cameras.

2 RELATED WORK

Data compression is an important and widely used topic in computer graphics, image processing, and many other research areas. For example, [KÖP13] and [BÖK11] present methods to achieve compact representations of surface-light interactions like subsurface scattering and BRDFs. The output of RGB-D sensors is often visualized in 3D as mesh or point cloud; therefore, a lot of focus was put on compressing these geometric representations. In 2007 MPEG issued a call for proposals on point cloud compression to develop an ISO standard [SPB$^+$19]. Point cloud compression algorithms are mostly based on spatial data structures like an octree. For example, [MBC17] is based on octrees and is also able to exploit temporal redundancies. [TCF16] introduced a time-varying approach that can predict graph-encoded octree structures between consecutive frames. Real-time capable mesh-based compression and streaming techniques like proposed by [MSA$^+$13] and [BGTB12] are in most cases lossy algorithms. As both of these representations, point clouds, and meshes, are three dimensional, it is more challenging to find and encode redundancy in the data efficiently. Specialized data structures are needed and raise the complexity and computation time for high compression ratios.

A different approach is to encode the raw depth images. Many standard lossless image and video codecs like PNG [RK99], JPEG-LS [WSS96, WSS00] or H.264 [WSBL03] can be applied, but the results are rather poor, founded in the inherent differences between natural images and depth images [SMAP16, Wil17, HKR18].
Similarly, general-purpose compression algorithms can be applied on depth images, but intrinsically are not optimized for this kind of data. Therefore they are not ideal solutions.

In recent years some effort was made to adapt common video codes like H.264 and its successor HEVC [SOHW12] to suit depth data better. [PKW11] proposed an encoding scheme to convert the single-channel depth data to the RGB format used by H.264 and VP8, reducing the occurring artifacts. This technique still produces noise at the borders. [LBW$^+$15] developed a hybrid lossless-lossy algorithm, where the first bits are encoded lossless, and the remaining ten bits are compressed using H.264. The HEVC standard [SOHW12] features an extension called 3D-HEVC designed for 3D video coding, which uses the multiview texture videos plus depth maps (MVD) format. This extension addresses the compression of depth images through multiple techniques. The complexity is rather high. Aimed at lowering it and enhancing the compression speed, [ZCH$^+$15] proposed multiple modifications exploiting depth map and texture video correlation. Further speedups can be gained by limiting costly intra-image compression steps to regions, where they are effective according to a decision model based on tensor feature extraction, as proposed by [HE19].

Recently, [HKR18] proposed another technique in which even noisy depth images and videos are encoded through planar segmentation. Each frame gets segmented into a number of planes by using the Graph Cut algorithm over the image defined as a Markov Random Field. While the proposed method achieves a high Rate-Distortion performance, it is rather time-consuming and only effectively applicable to scenes that can be approximated by planes.

Most work of depth image compression is about lossy compression. Lossless solutions are quite rare. [MZC$^+$11] combines run-length encoding and variable bit length coding for lossless encoding of depth images. [Wil17] took a similar but even simpler approach and achieves with the proposed RVL algorithm higher speeds at comparable compression ratios.

3 OUR APPROACH

We created a pipeline of four steps (Fig. 1) to compress depth images losslessly from real sensors with a high compression ratio. If we have a sequence of images, we optionally calculate deltas between the frames. Afterward, we define spans and calculate individual predictors for each of them, before it will be compressed by RVL's [Wil17] alternating run-length and variable-length coding. To decrease the lower bound of RVL, we decided to use Zstandard as a final, optional step. The focus lies on captured 16-bit depth values, which can be handled as a single channel grayscale image. The corresponding color images are not considered in this work.

RVL consists of alternating run-length coding of zero-values, which represents invalid pixels, and variable-length coding for the rest. Like many compression algorithms, RVL compresses not the raw pixel values but the residuals, which remain after calculating the delta to a prediction of the current value. This leads to a decorrelation of the data, which in turn improves the effectiveness of subsequent compression steps. Smaller residuals have a low entropy, and fewer bits are needed to encode them. In the case of RVL, the prediction of the current pixel value is simply the last valid pixel, which is, in most cases, the pixel to the left.

3.1 Adaptive Span-Based Prediction

One crucial aspect was to improve the simple inter-image delta calculation of RVL to generate smaller residuals. As the employed decorrelation heuristic is not ideal, we propose to replace it with an adaptive selector of different predictor functions. In lossless compression, the transformations and functions applied in the compression stage must be reversible in the corresponding decompression stage; hence, the used prediction method must be encoded in the form of bitflags, too, because all the transformations must be reversible. Pixel-wise switching of the predictors leads to too many bitflags. Therefore the image is dynamically partitioned into spans of valid pixels in our approach. A span can be described as a one-dimensional block of a fixed number of consecutive pixel values. Invalid zero-pixels are skipped. Figure 2 shows an example partitioning of pixels into spans of length four. Using spans instead of 2D blocks reduces the computational complexity and leads to faster computation.

Our adaptive prediction then works as follows: For each valid pixel p in the span S, all possible predictor functions $\mathrm{Pred}_i(p)$ are evaluated and the one, which in total leads to the smallest absolute residuals, gets chosen for all pixels in the span to calculate the final residuals r_p:

$$r_p = \mathrm{Pred}_k(p) \tag{1}$$

where

$$k = \underset{i\in[0,3]}{\operatorname{argmin}} \left\{ \sum_{p\in \mathrm{valid}(S)} |\mathrm{Pred}_i(p)| \right\} \tag{2}$$

A high-level overview of our prediction can be seen in Algorithm 1.

Algorithm 1 Span-Based Prediction

```
Require: inputImage
  initialize data structures
  for each non-zero pixel do
      calculate possible predictor values
      increment corresp. accumulated span errors
      if span is full then
          choose best predictor in span
          save predicorID and corresp. pixel deltas
          reset span errors
      end if
  end for
```

We decided to use four different predictor functions, which then can be represented by exactly two bits per span. In principle, the actual predictors, as well as their quantity, are easily exchangeable, focusing either on computationally simpler and, therefore, faster ones or more complex and effective ones. To predict a pixel p, we opted to use RVL's standard predictor, given in Equation 3, as default case, as it is similar to the common "left pixel" approach, but handles the occurrences of intermixed zero-pixels very well by skipping them. In Fig. 3, this process is illustrated.

Additionally, we use predictors based on the delta to the upper pixel (Eq. 4), the average of the left and upper pixel (Eq. 5), and lastly, the result of a combination of the left, upper and upper left pixels (Eq. 6).

$$\mathrm{Pred}_0(p) = p_X - p_A \tag{3}$$

$$\mathrm{Pred}_1(p) = p_X - p_B \tag{4}$$

$$\mathrm{Pred}_2(p) = p_X - \frac{p_A + p_B}{2} \tag{5}$$

$$\mathrm{Pred}_3(p) = p_X - (p_A + p_B - p_C) \tag{6}$$

Equation 6 is used as the default case in the Paeth [RK99, pp. 159–162] as well as the MED predictors [WSS96]. We use only this part of them as our fourth predictor because it is computationally less expensive than using the complete Paeth or MED predictor. Tests

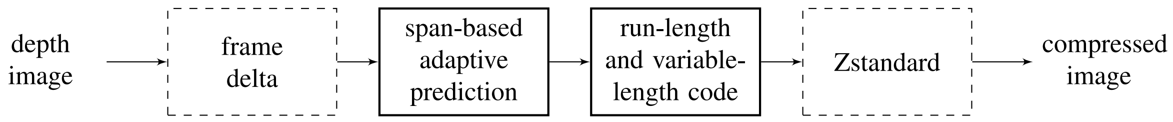

depth image
frame delta
span-based adaptive prediction
run-length and variable-length code
Zstandard
compressed image

Figure 1: The pipeline of our approach. Stages in dashed lines are optional. They allow a better balance between compression ratio and compression speed. First stage: frame delta, second: our span-based adaptive intra-image prediction, third: RVL's run-length and variable-length coding, fourth: an additional Zstandard [Col16] pass.

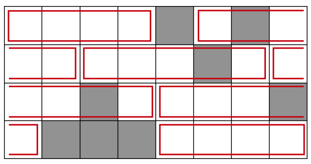


Figure 2: The pixel grid segmented in spans of four valid pixels (red boxes). Invalid pixels (grey) are skipped.

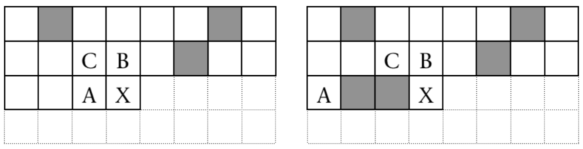


Figure 3: The prediction for pixel p depends on the pixel itself (X) and its neighbors last (A), above (B), and above left (C). Contrary to B and C, A is the last valid pixel and, therefore, not fixed, as the right image depicts.

we conducted showed that using the full MED predictor is much more time consuming, but does not improve the compression ratio significantly.

In this way, the number of additional bits needed for the predictor representation can be significantly reduced, while retaining the ability to adapt to the local image characteristics dynamically. The exact number of predictor bits for the image can be computed as

$$x = \left\lceil \frac{(n-z)}{s} \right\rceil \cdot \lceil \log_2(f) \rceil \tag{7}$$

where n denotes the number of pixels in the image, z the number of zero-pixels, s the span size and f the number of predictor functions.

3.2 Inter-Frame Delta Computation

Another aspect we concentrated on is adding a frame delta component as a first step in the algorithm's pipeline. This is a commonly used technique in video compression, where differentials between subsequent images are encoded instead of individual images one by one. Figure 4 illustrates the process. The effectiveness depends on the application scenario, the content, and how dynamic it is. At least in cases where the change between the images is small, or only some dynamic elements occur, this technique should be beneficial for the compression ratio and speed. In the original RVL paper [Wil17], it is mentioned that such a frame delta

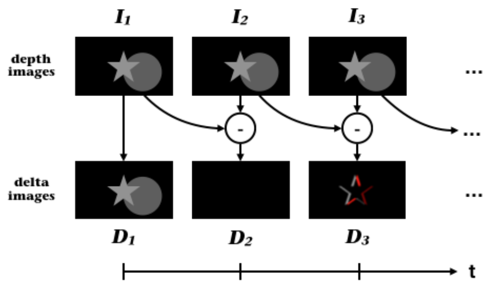


Figure 4: Frame delta computation: Sequence of images I_i and the corresponding differential images D_i between every two consecutive ones.

calculation was experimented with but the compression was even worse. According to the paper, the test scene was a dynamic scene in which the camera constantly moved, which would be the worst-case scenario for this kind of technique. It is not clear if other scenarios were tested. For our pipeline, we designed the frame delta computation as an optional first stage so that it can be skipped in scenarios where it is not effective. Another aspect to consider is the inherent noise of the sensor. Some pixels switch between being valid and invalid, and in addition, the depth readings continuously vary, even for physically static objects. This results in a decreased effectiveness of frame delta computation but can be compensated for by temporal filtering as a preprocessing step (at least to some degree). Depending on how intelligent and vigorous the employed filter is, other artifacts may be introduced, which could be tolerated depending on the application.

3.3 Further bit reduction

The coding of the residuals in RVL is done by variable bit length, where each valid pixel is at least one nibble (four bits) long. For each nibble, the first bit functions as a continuation bit and the other three bits are used to represent (a part of) the actual value of the residual. Therefore, in a single nibble, a residual $r \in [0,7]$ can be represented. In cases where image regions are very homogeneous, and the computed residuals are frequently below this maximum value of seven, the algorithm can lead to comparatively poor compression results. We propose to couple RVL with a second encoder without such a limitation to mitigate this drawback. In our approach, the compressed output of RVL is, therefore, further processed by Zstandard [Col16]. As a combination of a dynamic dictionary-based component with a sliding search window and an ANS-based entropy component, Zstandard can potentially compress symbols smaller than four bits. According to empirical tests, Zstandard performed best as a backend.

3.4 Parallel Execution

To mitigate the inevitably increasing computation time by the more complex prediction, we implemented multi-threaded versions. The principle is the same for all variants: first, for each thread, all necessary data and an output buffer are initialized, then the image, represented as a one-dimensional buffer, is segmented into blocks by the number of threads. Lastly, the compressed parts are stitched together to one continuous block. Especially our improved prediction step in the compression stage should benefit from parallel execution as it is computationally the most expensive part and there are only locally very confined dependencies between the pixel computations.

4 RESULTS

All of our tests were conducted on an Intel Core i7-7800X, 16 GB of system memory, and under Windows 10 x64 Enterprise using the Visual Studio 2017 compiler. Each test was performed multiple times, and the average was taken.

We recorded sequences of depth images of six different test scenes with the Azure Kinect RGB-D camera in both of the two available modes, narrow and wide field of view (NFOV, WFOV). In NFOV, the depth is recorded in 640×576 at 30 Hz and in WFOV at 1024×1024 at 15 Hz. Each depth value is represented as a 16-bit integer (short). Four of those scenes were static, and two dynamic. In the first dynamic scene, the camera is fixed, and a person moves in front of the camera. In the second scene, the camera is handheld and moved around. For the static scenes, 30 frames were recorded, while for the dynamic ones, 120 frames are used. Additionally, we tested three single depth images from the Middlebury dataset [SP07] with very homogeneous depth values, and a dynamic scene of 600 frames from TUM [SEE+12], in which the camera (Kinect 1) is handheld, slightly shaken, and people moving, while seated.

In each test case, we omit the first recorded frame from the evaluation as we do a lot of initialization work for the algorithms here (e.g. reserving memory) which holds for the rest of the test. The measurements of all the other subsequent frames in a test are then averaged.

The Azure Kinect camera has the unique attribute that the output depth-image is rectangular, but depending on the mode, the content is only hexagonal or spherical. The corners are zero-pixels. While this is the standard output of the camera and could be representative for other cases, where significant static areas fail to get captured by an RGB-D sensor, it is a rather uncommon situation. In order to not only test the standard configuration but also more broadly comparable scenarios, we also evaluate depth maps cropped by 25 %. As a result, most of the static invalid regions are dropped.

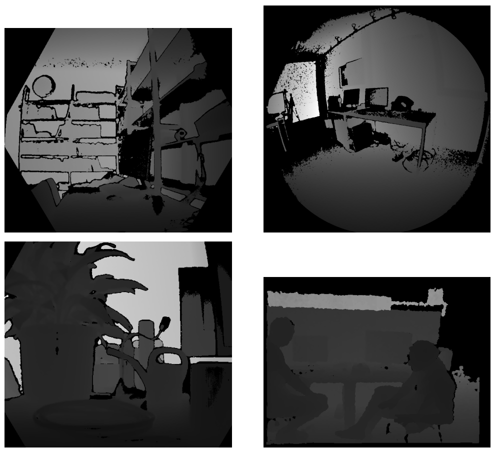


Figure 5: Example depth images of four different scenes. We record the first three with the Azure Kinect in different modes (NFOV, WFOV, NFOV with cropping). The last scene from TUM [SEE+12].

Figure 5 shows a selection of our test scenes. For an overview of all scenes and corresponding metrics, we refer to Figure 11 and Table 2 in the supplementary material.

We compare our novel compression algorithm against the original RVL algorithm [Wil17], PNG [RK99] as a classical lossless image compression algorithm that is widely used, and LZ4 as a fast representative for the LZ77 family of dictionary-based encoders. Additionally, as an example of a modern and efficient entropy coder, we decided to use an ANS implementation by F. Giesen [Gie14]. In empiric tests, this implementation performed best. Lastly, Zstandard as dictionary coder is also compared.
For our algorithm Pred, we chose a span size of 16 valid pixels. For our PredZ variant, which extends Pred by applying Zstandard for further bit reduction, we chose a span size of 16 valid pixels, and two as Zstandard compression parameter. These configurations yielded the highest compression ratios, while still achieving interactive speeds. For algorithms featuring a selectable acceleration (or compression) factor, we tested them in multiple configurations and chose the one, which minimized the difference of the compression ratio in comparison to our algorithm. In the case of PNG, a quality

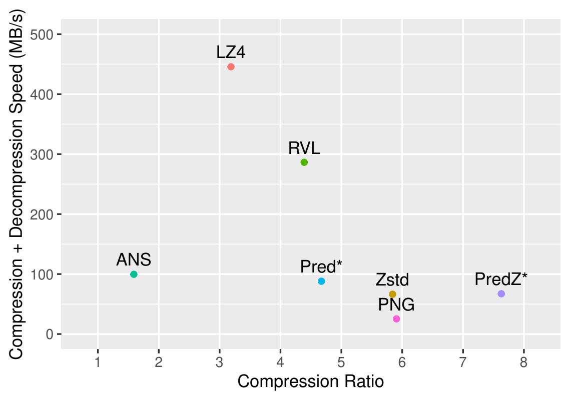


Figure 6: Average combined compression and decompression speed over compression ratio for all algorithms. Algorithms with asterisk mark our algorithms. Our PredZ algorithm achieves the highest absolute compression ratio, while still delivering real-time performance.

level of five was chosen, for LZ4, an acceleration factor of one, and for Zstandard a compression value of six.

The first test we conducted is a general comparison of all the algorithms in terms of their respective compression ratio *cr* and combined compression and decompression speed *sp*. Equations 8 and 9 describe the calculation of the metrics.

$$sp = \frac{2 \times ob}{ct + dt} \tag{8}$$

$$cr = \frac{ob}{cb} \tag{9}$$

ob stands for the original image size in megabytes, *ct* and *dt* for the compression and decompression times in seconds and *cb* for the compressed size in megabytes.

Figure 6 presents the results of this initial comparison, where for each algorithm, the thread configuration was selected, which yielded the best compression rate; therefore, the number of threads was not equal for all algorithms. For this test, the mean over all scenes and modes were taken. The asterisks identify our algorithm variants.

It can be seen that, on average, ANS is not competitive in our test, neither in speed nor compression ratio. LZ4 is the fastest but compresses rather poor. PNG has a high compression ratio of 5.9:1 but at the cost of a very slow combined speed. RVL achieves good performance and a compression ratio of nearly 4.4:1. Our span-based adaptive prediction Pred can boost the compression ratio on average by 6.5 %, but the performance decreases considerably. A regression in performance to a certain degree was to be expected by the inherently more complex delta computation and less efficient memory accesses. A possible explanation for the strength of the drop in performance, despite the rather low complexity predictors, might be the lack of optimization done compared to RVL. Interestingly, the general-purpose Zstandard algorithm can achieve an even higher compression ratio of roughly 5.8:1. However, it is also slower. Lastly, with our most sophisticated variant PredZ, we achieve the best compression ratio of more than 7.6:1, which is significantly higher than the ones of RVL or Zstandard. At the same time, the combined compression and decompression speed of 67.2 MB/s is more than sufficient to maintain interactive frame rates. For more comprehensive data of this test, we refer to Figure 12 in the supplementary material.

Predictor ID	0	1	2	3
Usage %	24.4	26.6	21.2	27.7

Table 1: Distribution of adaptively selected predictors, average over all scenes and modes.

To further analyze the effectiveness of our span-based adaptive prediction component, we counted the frequency how often each predictor is chosen as the best one per image. Table 1 shows the percentages averaged over all scenes and frames.

Each predictor is equally used, which indicates that the adaptive prediction works as intended, and all of the chosen predictors complement each other to effectively reduce the average residual in contrast to a static prediction like it is used, for example, in the original RVL algorithm.

For all subsequent tests, we only consider the most promising algorithms and omit the ones without competitive results, namely ANS, LZ4, and PNG.

In order to review our multi-threading implementation and analyze the behavior of the algorithms with increasing parallel execution, we performed all tests as well with two, four, and eight threads. Measuring initial thread creation overhead is prevented by early thread allocation. The multi-threading performance is shown in Fig. 7, in general, the combined speed rises.

As expected, the performance gains shrink with more threads as a result of diminishing returns. However, we were able to achieve considerable speedups for RVL, although it has low arithmetic complexity and therefore becomes quickly memory-bound. With four threads, the performance increased by factor 1.42. Our more complex RVL-based algorithm benefits highly from the parallel execution, as increasing the threads from one to four leads to a speedup of 2.16. While all algorithms gain performance by multi-threading, at least for up to four threads, the speedup is generally much lower than the theoretical optimum. This may be partly because not all parts of the algorithms are multi-threaded, e.g., merging the results, and that some sections of the algorithms are rather bandwidth- than compute-bound.

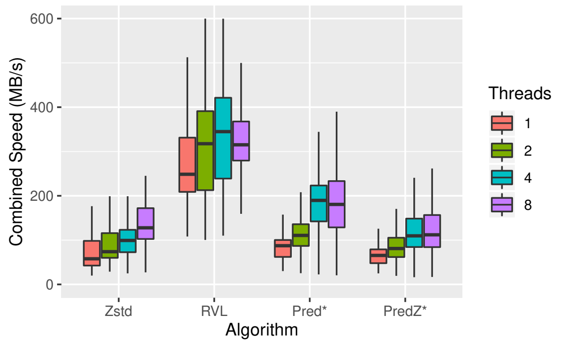


Figure 7: The image illustrates the impact of parallelization on the combined compression and decompression speed for different algorithms. All algorithms including RVL and our RVL-based ones benefit from parallel execution.

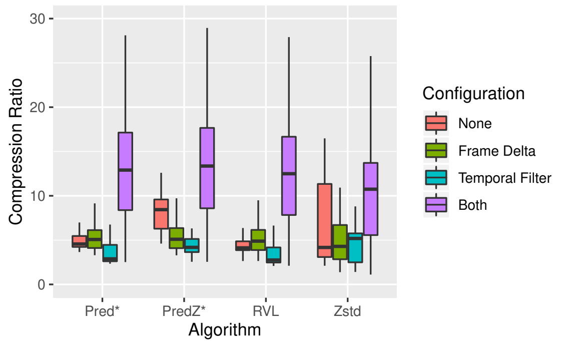


Figure 8: Comparison of the impact of our inter-frame delta computation on the compression ratio, both with and without a preceding temporal filter. The compression ratio increases hugely with the combination of temporal filter and frame delta, but using frame delta computation without the preceding filter also, most often, accomplishes notable gains.

An evaluation of the timings of the individual parts of our Pred algorithm indicates that the more computationally complex prediction stage benefits significantly more from the parallelization than the run- and variable length coding stage. Furthermore, the decompression component does not profit as much as the compression component. The compression ratio did slightly, but not considerably, decrease with more threads, as it was expected due to the separated image blocks. In the case of Zstandard, the compression ratio decreased the most with 2.68 %. RVL, on the other hand, had a decrease of less than 0.1 %. The compression ratio of our adaptive prediction dropped by 0.25 %.

To analyze the effectiveness of frame delta coding in RVL, each test is executed with and without our implementation of the addition as well as with and without a preceding temporal filter (see the last paragraph of section 3.2), which makes four variants. The filter we implemented and use is rather simple and only meant as an example. Nonetheless, we aimed at preserving the legit information (in contrast to the sensor noise) and avoiding motion artifacts. It works as follows: Pixels of an incoming image will get ignored, whose delta to the corresponding pixels of each of the last two images is below 2 %.

The combination of frame delta coding and temporal filtering increases the compression ratio for all algorithms the most, as is shown in Figure 8. While the combination of frame delta coding and temporal filtering, in general, leads to substantial improvements, our algorithm PredZ still achieves the highest compression ratio of up to 13.8:1. It should be noted though, that with temporal filtering, the compression pipeline is not strictly lossless anymore.

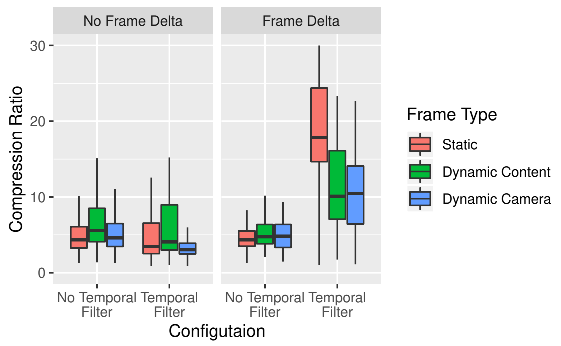


Figure 9: The image illustrates the impact of the scene type (static, dynamic content, dynamic camera) on the performance of the frame delta computation with and without filtering. Static scenes benefit the most by the combination of temporal filtering and frame delta computation.

With only the frame delta extension, the compression ratio still increases for Pred and RVL, which contradicts the statement of [Wil17]. It should be considered that for our tests we took the average of static and also highly dynamic scenes. Nonetheless, RVL's compression rate increased by 18.54 % and with our improved prediction by 11.1 %. Zstandard does not benefit from frame delta, but only from the temporal filter instead, although the confidence interval is very wide. Surprisingly, the data indicates that our PredZ performs best with the raw data compared to the others.

A detailed breakdown w.r.t. different classes of scenes and their influence on the compression ratio, while computing frame delta, can be seen in Fig. 9.

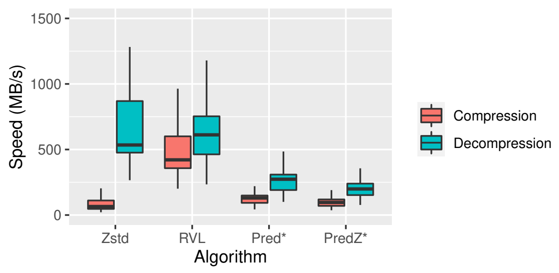


Figure 10: The image displays the individual compression- and decompression speed for different algorithms. The decompression is significantly faster for everyone.

The results indicate that the type of the scene, static or dynamic, or more specifically, the amount and the intensity of movement of the content and the camera itself, do have a significant impact on the compression rate. Without temporal filtering, the average compression rate of all algorithms is about equal. According to Fig. 8, not all algorithms benefit directly from frame delta. If, however, such a filter is used, the influence of the scene type raises strongly, specifically rather static scenes can be compressed extremely well. Interestingly, sometimes the scene which involves camera movement is better compressible using both, the temporal filter and frame delta, instead of the scene, where the camera is static and only parts of the captured environment move.

How the combined speed of the tested algorithms is distributed over compression and decompression can be seen in Fig. 10. The decompression speed generally outperforms the compression speed, especially in the case of Zstandard.

5 CONCLUSION AND FUTURE WORK

We proposed a lossless RVL-based algorithm for real-time depth image compression aimed at high compression ratios. Our experiments show that our algorithm achieves the highest compression ratios compared to competing real-time capable algorithms. With our method, depth images can be compressed up to 73 % smaller than with the original RVL algorithm and 30 % smaller than with the slower Zstandard. Using temporal filtering beforehand, we accomplish even higher compression ratios of 13.8:1, which is still the highest compared to the other evaluated algorithms but the compression pipeline becomes lossy. Thanks to parallel execution, our performance is still sufficient for typical RGBD-sensor frame rates and real-time streaming in bandwidth-limited applications. Finally, we were able to show that the original RVL can also be sped up with multi-threading, and it can, for some use-cases, benefit significantly from frame delta computation. As future work, further improvements could be made by optimizing the span-based prediction stage to lower the computation times, and maybe even SIMD can be applied. Also, it may be worth to investigate zigzag scan and to use a block-based adaptive prediction instead. The latter may exploit a possibly higher spatial coherency, however, increase the computational complexity further. Another promising idea would be to integrate the inter-image- into the intra-image delta computation by also using the neighbor pixel values from the previous frame as possible predictors.

6 REFERENCES

[BGTB12] F. Bannò, P. S. Gasparello, F. Tecchia, and M. Bergamasco. Real-time compression of depth streams through meshification and valence-based encoding. In *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '12, page 263–270, New York, NY, USA, 2012. Association for Computing Machinery.

[BKKF13] S. Beck, A. Kunert, A. Kulik, and B. Froehlich. Immersive group-to-group telepresence. *IEEE transactions on visualization and computer graphics*, 19:616–25, 04 2013.

[BÖK11] A. Bilgili, A. Öztürk, and M. Kurt. A general brdf representation based on tensor decomposition. *Computer Graphics Forum*, 30(8):2427–2439, December 2011.

[CJK15] C. Chen, R. Jafari, and N. Kehtarnavaz. Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 168–172, Sep. 2015.

[CK12] Y. Champawat and S. Kumar. Online point-cloud transmission for tele-immersion. pages 79–82, 12 2012.

[Col16] Y. Collet. Zstandard - a fast real-time compression algorithm. Github repository, 2016. `https://github.com/facebook/zstd`; accessed on 26. Februar 2020.

[Gie14] F. Giesen. Implementation of several rans variants. Github repository, 2014. `https://github.com/rygorous/ryg_rans`; accessed on 26. Februar 2020.

[HE19] H. Hamout and A. Elyousfi. Fast depth map intra coding for 3d video compression based tensor feature extraction and data analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2019.

[HKR18] S. Hemanth Kumar and K. R. Ramakrishnan. Depth compression via planar segmentation. *Multimedia Tools and Applications*, 78, 07 2018.

[JSM$^+$14] B. Jones, R. Sodhi, M. Murdock, R. Mehra, H. Benko, A. Wilson, E. Ofek, B. Macintyre, N. Raghuvanshi, and L. Shapira. Roomalive: Magical experiences enabled by scalable, adaptive projector-camera units. 10 2014.

[KÖP13] M. Kurt, A. Öztürk, and P. Peers. A compact tucker-based factorization model for heterogeneous subsurface scattering. In S. Czanner and W. Tang, editors, *Proceedings of the 11th Theory and Practice of Computer Graphics*, TPCG '13, pages 85–92, Bath, United Kingdom, 2013. Eurographics Association.

[LBW$^+$15] Y. Liu, S. Beck, R. Wang, J. Li, H. Xu, S. Yao, X. Tong, and B. Froehlich. Hybrid lossless-lossy compression for real-time depth-sensor streams in 3d telepresence applications. In *Advances in Multimedia Information*

Processing – PCM 2015, pages 442–452, Cham, 2015. Springer International Publishing.

[MBC17] R. Mekuria, K. Blom, and P. Cesar. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):828–842, April 2017.

[MSA+13] R. Mekuria, M. Sanna, S. Asioli, E. Izquierdo, D. C. A. Bulterman, and P. Cesar. A 3d tele-immersion system based on live captured mesh geometry. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, page 24–35, New York, NY, USA, 2013. Association for Computing Machinery.

[MT17] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[MZC+11] S. Mehrotra, Z. Zhang, Q. Cai, C. Zhang, and P. A. Chou. Low-complexity, near-lossless coding of depth maps from kinect-like depth cameras. In *2011 IEEE 13th International Workshop on Multimedia Signal Processing*, pages 1–6, Oct 2011.

[NIH+11] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, Oct 2011.

[PKW11] F. Pece, J. Kautz, and T. Weyrich. Adapting standard video codecs for depth streaming. pages 59–66, 01 2011.

[RK99] G. Roelofs and R. Koman. *PNG: The Definitive Guide*. O'Reilly & Associates, Inc., USA, 1999.

[SEE+12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[SMAP16] S. Shahriyar, M. Murshed, M. Ali, and M. Paul. Lossless depth map coding using binary tree based decomposition and context-based arithmetic coding. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, July 2016.

[SOHW12] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, Dec 2012.

[SP07] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[SPB+19] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, March 2019.

[SST+18] C. Schröder, M. Sharma, J. Teuber, R. Weller, and G. Zachmann. Dyncam: A reactive multithreaded pipeline library for 3d telepresence in vr. In *Proceedings of the Virtual Reality International Conference - Laval Virtual*, VRIC '18, New York, NY, USA, 2018. Association for Computing Machinery.

[TCF16] D. Thanou, P. A. Chou, and P. Frossard. Graph-based compression of dynamic 3d point cloud sequences. *IEEE Transactions on Image Processing*, 25(4):1765–1778, April 2016.

[Wal92] G. K. Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[Wil17] A. D. Wilson. Fast lossless depth image compression. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, ISS '17, page 100–105, New York, NY, USA, 2017. Association for Computing Machinery.

[WKJ+15] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research*, 34(4-5):598–626, 2015.

[WMS16] O. Wasenmüller, M. Meyer, and D. Stricker. Augmented reality 3d discrepancy check in industrial applications. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 125–134, Sep. 2016.

[WSBL03] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[WSS96] M. J. Weinberger, G. Seroussi, and G. Sapiro. Loco-i: a low complexity, context-based, lossless image compression algorithm. In *Proceedings of Data Compression Conference - DCC '96*, pages 140–149, March 1996.

[WSS00] M. J. Weinberger, G. Seroussi, and G. Sapiro. The loco-i lossless image compression algorithm: principles and standardization into jpeg-ls. *IEEE Transactions on Image Processing*, 9(8):1309–1324, Aug 2000.

[ZCH+15] Q. Zhang, M. Chen, X. Huang, N. Li, and Y. Gan. Low-complexity depth map compression in hevc-based 3d video coding. *EURASIP Journal on Image and Video Processing*, 2015(1):2, 2015.

Visualizing Massive Pristine LIDAR Amplitude Responses

Werner Benger
AirborneHydroMapping GmbH
A-6020 Innsbruck, Austria
w.benger@ahm.co.at

Center for Computation & Technology
Louisiana State University
Baton Rouge, USA

Wolfgang Leimer
AirborneHydroMapping GmbH
A-6020 Innsbruck, Austria
w.leimer@ahm.co.at

Ramona Baran
AirborneHydroMapping GmbH
A-6020 Innsbruck, Austria
r.baran@ahm.co.at

ABSTRACT

Airborne light detection and ranging (LIDAR)- based bathymetry is a highly specialized field within the widely known and used geoscientific surveying technology based on green spectrum lasers. Green light can penetrate shallow water bodies such that river and lake beds can be surveyed. The result of such observations are point clouds, from which geometries are extracted, such as digital elevation maps for lake, river or sea floors. The quality of those maps is crucially dependent on the amount of reliable information that can be extracted from the noisy LIDAR signals. The primary LIDAR data consists of amplitude response curves for each emitted laser signal. A direct visualization of these "raw", pristine data is crucial to verify, assess and optimize subsequent data processing and reduction methods. In this article we present a method for scientific visualization of these amplitude response curves *en mass*, i.e. for millions and tens of millions thereof simultaneously. As part of this direct visualization also preliminary analysis and data reduction operations can be performed interactively. This primary and direct inspection allows studying and evaluating the full potential of acquired data sets such that data processing methods can be fine-tuned to squeeze out all needed information of interest. Ideally such improved data processing avoids subsequent surveying when output from already measured data sets is improved, resulting in reduced economical and environmental costs.

Keywords: airborne LIDAR, visual analysis, data processing, geoscience, bathymetry, scientific visualization, big data, GPU shaders

1 INTRODUCTION

The rendering of large point clouds has been a topic of research for a long time [14] and has even been used as an alternative faster than traditional rendering methods based on triangular meshes. Once the data sets become larger than RAM, out-of-core methods [8] become mandatory and eventually a hierarchical representation is needed. High performance in rendering is tightly related to efficient usage of GPUs. With the advancement and availability of WebGL even browser-based rendering of arbitrarily large point clouds have become possible [10]. However, rendering performance drops significantly once the amount of data to be visualized per frame no longer fits onto GPU RAM. For such situations a purely CPU-based solution may scale better, as presented by [13] using balanced P-k-d Trees. Such a sorting method for points without memory overhead is similar to the space-filling curves [9] used in smoothed particle hydrodynamics for extremely large data [12].

While point clouds are often considered as the primary data source to derive geometries such as triangulate meshes off them, rather little attention is given to the origin of the point clouds itself. In many cases those point clouds are based on LIDAR data acquisitions and the provided point clouds are taken as-is. However, LIDAR technology does not produce point clouds per se, but the instruments measure a time series of amplitude signals from the reflected laser source. The point cloud used for further processing is derived from this primary data, see Fig. 1 and Fig. 2.

These sequences of amplitude modulations constitute the actual "raw" data from the LIDAR sensor, out of which the final point cloud has to be derived. This is usually done by identifying maxima in the amplitude modulation and outputting their geometrical location along the laser shot direction. As these received signal curves may very well contain multiple maxima, this identification will produce many geometrical points per laser shot. Finding only valid maxima within the noisy signal is a matter of signal processing artwork such that the quality of the results depends on the respective algorithm. While extensive work has been undertaken in this area [11], special cases still occur where an automatized algorithm may miss some relevant information. This is particularly of interest in cases of complex geometries such as vegetation, or for underwater LIDAR penetration, Fig. 1. By inspection of the final point cloud after data processing, it is impossible to find such missed features in the raw data, they only give hints of regions where more information would be desirable. Thus, a comprehensive visualization method that allows to investigate, study and explore the entire raw LIDAR data is a must. Only then it is possible to verify and optimize point extraction algorithms, such as to fine tune them and to squeeze out the last little bit of valid information from the already available and recorded data, e.g. in order to complement the representation of relevant object geometries in the resulting point cloud.

This primary data, the amplitude time series per laser shot, is larger by two orders of magnitude than the point cloud derived from them. Due to this huge amount of data amplitude signals have so far only been visualized individually. Here we present an approach for the massive visualization of *all* amplitude time series signals that are acquired from a LIDAR observation, a massive amount of information up to $100\times$ larger than just point clouds. Furthermore, we emphasize the practical implications of improved point extraction based on LIDAR amplitude signal analysis especially for bathymetric (underwater) LIDAR data acquisition.

2 DATA STRUCTURES

Our data model of choice is the Fiber Bundle HDF5 "F5" model , which casts all data into a hierarchy of five (plus two optional) levels [3]. The premise is that nature is best described by a differentiable manifold [4]. This data model is very suitable for I/O and is directly compatible with GPU data structures such as vertex and index buffer objects. In particular, it maps easily onto the underlying Hierarchical Data Format V5 (HDF5) as it allows for random data access and partial file I/O. These features are usually not available via proprietary file formats for the raw data delivered by the LIDAR hardware manufacturers. Reading those native files may last several days, while after conversion into the F5 layout and storing as HDF5, reading them is reduced to mere seconds or milliseconds. The initial step is thus to convert the data from the respective proprietary, hardware-specific file formats into the open, application-independent HDF5 file format [5].

2.1 Review: The HDF5 File Format

Common data formats used in geoscience (e.g. LAS[1], Shapefile[2]) are often restricted to certain data fields and data types. The HDF5 file format provides more flexibility with the data layout. It is hierarchically structured similar to a file system, where folders are equivalent to HDF5 Groups and files equivalent to HDF5 Datasets. HDF5 files are not limited in size and number of objects. Datasets provided additional structures such as dimension and type information. Each of them may be compressed independently with various distinctly tune-able compression methods. The file format allows to construct compound data types like higher dimensional geometric vectors or points from simple, hardware-independent basic types and is therefore self-describing. Furthermore it offers important features such as partial data loading and random access, which is crucial for indexing operations relating data structures to each other even beyond files. Finally it is a free and open standard providing a high-level API with interfaces for C, C++, Java, Python and Fortran. In its most recent version it also offers remote data access via web services, thus enabling inspection of big data via the internet without modifying the main application.

2.2 Review: The F5 Data Model

The F5 Data Model bundles geometric entities, their topological properties and attributes into a combined, complex object that decomposes into the following layers:

1. **Slice** Combines all data related to a certain moment in time, or - in case of a time series - until the next sampling moment in time.

2. **Grid** Combines all data related to a certain geometric entity.

3. **Skeleton** Combines all data of a Grid that are related to a specific topological property of this Grid, for instance its vertices, edges, triangles, agglomerations thereof or hierarchical instances. A Skeleton defines an "index space" where each index refers to one such element of a topological space.

4. **Representation** Combines all data of a Skeleton that are relative to either other Skeletons or to charts that define coordinate systems. Basically, a Representation only makes sense in relation to the object that it refers to, for instance the Representation of triangles relative to vertices, or the Representation of vertices relative to Cartesian coordinates.

[1] `https://www.loc.gov/preservation/digital/formats/fdd/fdd000418.shtml`
[2] `https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`

5. **Field** Combines all numerical values that have a specific semantic meaning, for instance physical fields such as "temperature" or "velocity".
6. **(Fragment)** An optional 6th level to optimize performance for big data by allowing to split a contiguous data set into many smaller fragments.
7. **(Compound)** An optional 7th level to allow storing compound data types as distinct arrays instead of interleaved elements, i.e. offering the choice of using a layout of the kind XXXYYYZZZ instead of XYZXYZXYZ.

This data model is capable to handle geometrical objects such as triangular meshes, point clouds, curvilinear grids, uniformly rasterized images or time series within one universal framework. It also allows to specify relationships between such objects and hierarchical multi-resolution instances. The complexity of the model grows with the complexity of the data, simple cases are modeled with just the minimally required layers filled out.

2.3 Topology of LIDAR Signals

With a LIDAR equipment on board an airplane or drone (UAV), a laser source emits pulsed signals at a certain frequency and a receiver measures the intensity of the reflected light, Fig. 1. Both events are recorded with a time stamp at sub-nanosecond time resolution which allows geometrical reconstruction of the observation events (in one nanosecond light travels ca. 30cm). If

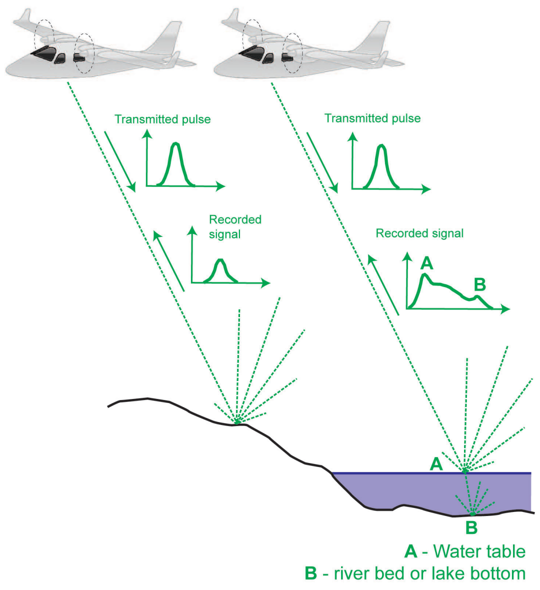


Figure 1: Data acquisition from airborne LIDAR mapping. The laser equipment emits signals at a certain frequency and measures the time when the reflected signal arrives at the airplane again.

the laser beam hits a mirror orthogonal to its view direction, then the received signal is an exact copy of the emitted pulse, yet subject to distortions happening in the detector itself. The location of such a mirror target can then be reconstructed from the time stamps within the precision of the clock. Eventually this location is assigned three-dimensional coordinates, forming one element of a point cloud that describes the entire observed geometry from the particular flight, as in Fig. 1.

Since actual geometries deviate significantly from an idealized perfect mirror, the actually received signal will be a superposition of many reflections that are detected at different times according to their reflections at different locations in space, Fig. 1 and Fig. 2.

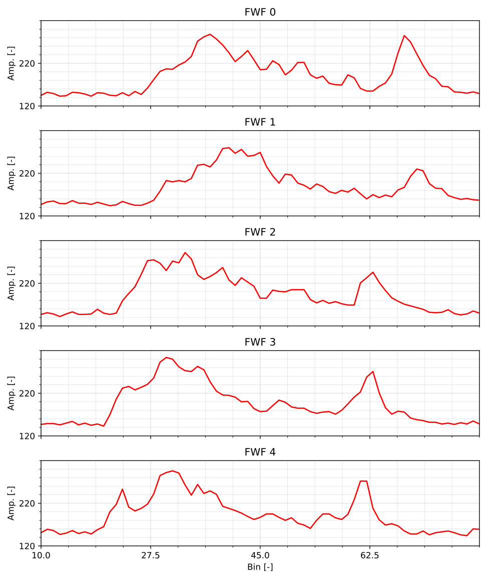


Figure 2: Received amplitude modulation over time for subsequent laser shots.

Within the context of the fiber bundle data model, data are identified via their base and fiber space. For LIDAR point clouds the base space is the geometry, the fiber space consists of all the attributes on them (such as color, intensity, time stamp, ...). Since each point in the final point cloud has been extracted from a specific amplitude modulation those time series can be assigned to each point as additional data, thereby expanding the fiber space without changing the base space. This has the advantage that the raw data can be investigated immediately for each identified point. These points have to reside at a maximum of the signal curve. However, if this point has been extracted from a local maximum, then other points will exist for the same signal curve, requiring duplication of the same curve onto all those points extracted from it.

Alternatively the base space can be considered as constructed from the sequence of “laser shots”. Each laser shot has been emitted at a certain time, forming a one-dimensional sequence with monotonously increasing timestamp. Using the flight path’s trajectory information geometrical 3D coordinates can also be assigned to each individual time stamp. Here, the amplitude modulations are fibers over the base space that is constituted by the flight trajectory. In this approach there is no need for data duplication, but the relationship between extracted geometries and the original raw data is missed. While superior with respect to data efficiency, this approach is thus less suitable to study the computational performance of the algorithm which is the ultimate objective of the visualization approach.

2.4 Massive Variable-Length Multiplicity

Multiplicity is a property of a field that tells its number of components. For instance, a scalar field has multiplicity 1, a coordinate field and a velocity field both have multiplicity 3, even though they have algebraically distinct properties. Within the classical fiber bundle data model as illustrated in Fig. 3, the multiplicity of a field must be constant for each point. This invariance of the multiplicity is essential for performance on I/O, parallelization and GPU processing as it directly fits with the requirements of vertex buffer objects. A field on a Grid's vertex Skeleton is just a vertex array in OpenGL. While this concept applies well to vertices and vertex-

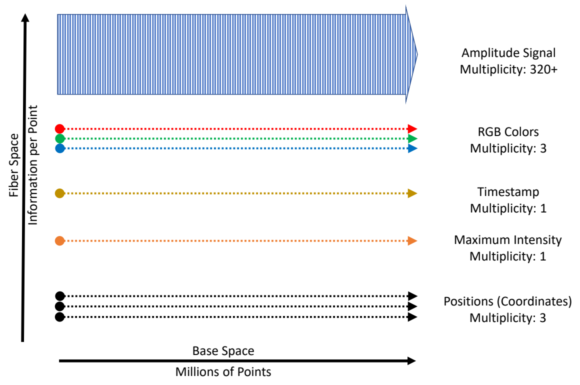


Figure 3: Fiber Bundle data layout for amplitude time series signals on point clouds: The amplitude signal is modeled as a 2D field given over a 1D base space.

related data as well as to homogeneous topologies such as triangular meshes, it fails to cover inhomogeneous cases such as mixed meshes constructed intermittently from quads, triangles, or arbitrary polygons (only after explicit triangulation they become homogeneous). They can be covered by extending the element type by a variable-length data type per point. Such data structures are possible both in C++ and in HDF5, but inefficient, both in terms of I/O and handling on the GPU, so they should be avoided. Staying at the classical fiber bundle model as defined above as close as possible guarantees better performance.

The LIDAR raw signal as received by the instrumental detector is a time series of the light amplitude over time which is different for each laser shot (Fig. 2). The number of sampling points ("bin") for this physically continuous signal varies from a few dozen to a few hundreds. The sampling interval in time was 0.57 nanoseconds for our hardware equipment, which corresponds to about 8.5cm in space considering the light travel time during that period, including double traversal time due to the reflection on the observed point (Fig. 1). Thus, when taking each single laser shot as the elements of a topological space within the framework of the fiber bundle model, then the field of the corresponding amplitude signal will have varying multiplicity (Fig. 2). In order to maintain high performance while not losing any potentially important information each signal is expanded to the maximally found length. While this approach blows up the raw data by adding zero values, it is still preferable over dealing with variable-length data. When storing data to disk, high-speed compression methods such as LZ4 are able to reduce disk storage requirements without performance penalties, and even performance gain [1].

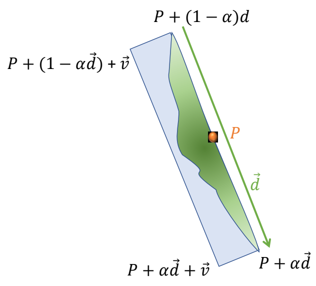
2.5 Data Processing Pipeline

Prior to visualization, the raw data delivered from the scanner has to be processed. In this case the source data consists of the flight trajectory, the point cloud and the amplitude response. To speed up the subsequent processes all source data is imported to the F5 fiber-bundle format first. The second step is to transform the point cloud position from the scanner's internal coordinate system to a geographic one, in this case UTM coordinates. This also includes the direction vector tracing a point to the scanner's position at the time of echo signal detection. To visualize the amplitude response as described in the next section 3 a spacial point is needed to mount the amplitude response onto. One possibility would be to calculate the coordinate location of the start of the amplitude response: $\vec{o}+\vec{d}(c_a*(t_r-t_e)/2)$, where $\vec{o}$ is the origin of the laser pulse, $\vec{d}$ is the direction vector, c_a is the speed of light in air, t_e the time of laser pulse emission, t_r the starting time of received signal record, and division by two because the laser pulse travels back and forth). The advantage is that amplitude responses can be visualized and processed independently from the point cloud. Some scanners are storing the amplitude response apart from the point cloud. In this case the relation "points belonging to an amplitude response" is not available but must be recomputed if it is relevant for data analysis. This is done by calculating the instant of time where the laser pulse is reflected by a detected point (t_e+d*2/c_a, where d is the distance between scanner and point) and search for the amplitude response with an overlapping time interval. This is a significantly time consuming process for millions of points. For better performance these correlations and intermediate values must be precomputed.

3 VISUALIZATION METHODOLOGY

3.1 Basic Functionality

It is natural to display a time series as a curve such as in the technical display of Fig. 2. For LIDAR signals, each amplitude of the time series has a spatial correspondence - namely a point with 3D coordinates - that allows for assigning the amplitude value to locations in space along the line of the respective laser shot. A billboard-like display does the job, where one orientation if the billboard is determined by the direction of the laser shot in 3D and the other one by the plane orthogonal to the view direction, Fig. 4. The coordinates of this laser-shot oriented billboard are easily computed by an OpenGL geometry shader, based on a vertex coordinate P with a vector vertex attribute specifying the direction $\vec{d}$, a scalar vertex attribute α (can be omitted if P refers to always the same position within the amplitude signal) and a uniform variable specifying the view direction (which is, for instance, available via the modelview matrix), as illustrated in Fig. 4.



$P + (1-\alpha)d$
$P + (1-\alpha\vec{d}) + \vec{v}$
P
$\vec{d}$
$P + \alpha\vec{d} + \vec{v}$
$P + \alpha\vec{d}$

Figure 4: “Mounting” an amplitude signal to a point P, given the laser shot direction $\vec{d}$, at positional location $\alpha \in [0,1]$ and a billboard view direction vector $\vec{v} = \vec{d} \times \vec{c}$ with $\vec{c}$ the view direction from the camera.

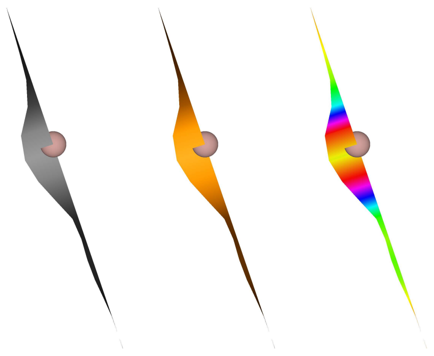


Figure 5: Visualization of a time series along the downwards direction of a laser shot with different color-coding of the amplitude value. The single point is located at the amplitude maximum, which is usually the only source of information constituting final point clouds.

The geometry shader thus “blows up” a single vertex into a screen-orthogonal quad and provides two-dimensional texture coordinates for its four corners. The first component of the texture coordinates correlates to the time index of the amplitude signal, normalized to $[0,1]$ (along $\vec{d}$ in Fig. 4). The second component corresponds to the geometrical distance from the precise line of the laser shot, also normalized to $[0,1]$ (along $\vec{v}$ in Fig. 4). It is then the duty of the OpenGL fragment shader to display the values of the time series based on those texture coordinates, interpolated by the hardware for each pixel, in a visually pleasing and insightful way similar to Fig. 2; but an opaque area - Fig. 5 - is preferable to a mere contour line.

3.2 Base Space Chopping

In order to access the - huge amount - of amplitude signal data per vertex, all these amplitude data need to be provided as a texture buffer to the GPU which extends the capabilities of the commonly used vertex attributes by orders of magnitude: An amplitude signal may encompass hundreds of entries, whereas vertex attributes are limited to the types supported by OpenGL, the largest one being a 4×4 matrix that could store at most 16 values. This one contiguous buffer[3] (OpenGL’s `glTextureBuffer()`, accompanied by GLSL’s `usamplerBuffer`) needs to be accessed as a two-dimensional array then, with one index given by the number of the vertex for which the fragment shader is called, the second index given by the texture coordinate along the direction of the laser shot.

However, while the size of a buffer is merely limited by the amount of total RAM available, the size of a texture buffer constrained to the extent given by `GL_MAX_TEXTURE_BUFFER_SIZE`. This maximally allowed texture buffer size is easily exceeding by the vast amount of amplitude signal information. To handle this constraint, the base space (Fig. 3) needs to be chopped into smaller pieces with the texture referencing a subset of the entire buffer for each such piece via `glTextureBufferRange()` with increasing offset, as illustrated in Fig. 6. This offset is required to be an integer multiple of the hardware-dependent value of `GL_TEXTURE_BUFFER_OFFSET_ALIGNMENT`[4].

Consequently, the base space cannot be chopped at arbitrary vertex indices, but only at index locations that are an integral multiple of both the hardware-dependent texture buffer offset alignment and the multiplicities of the fiber space. The “minimal chop size” is then determined by the *least common multiplier* of the alignment and the multiplicities. For instance, for a common alignment of 16, an amplitude signal length of 300 results in a minimal chop length of 1200, whereas a signal length of 320 results in a minimal chop length of 320 (which already is an integer multiple of 16). Of course, minimizing the number of rendering instances is mandatory, thus the largest integral multiple of this minimal chop length that still fits into the

[3] `https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glTexBuffer.xhtml`

[4] `https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glTexBufferRange.xhtml`

`GL_MAX_TEXTURE_BUFFER_SIZE` will be the one to be used such that all but the last chop will be of the same - maximally possible - size for the rendering instances. This is the effectively used chop length as illustrated in Fig. 6.

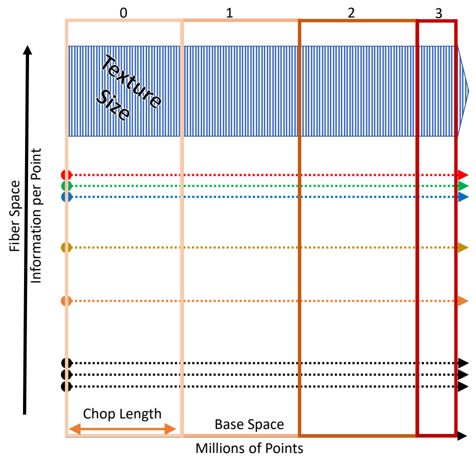


Figure 6: Chopping of the base space (per fragment) such to allow multiple rendering passes to operate on texture sizes that fit into the GPU hardware limits. The last render pass receives a partial texture.

The maximal number of chops per rendering instance is given by the `GL_MAX_TEXTURE_BUFFER_SIZE` divided by the minimal chop length, meaning that many chops can be rendered at once in a single rendering call. Thus, the number of rendering instances actually needed for the entire data set is given by this maximal number of chops divided by the total number of vertices. This will be a fractional number: Its integer part gives the number of "full-length" passes, the remainder part gives the last chop's length. Table 1 gives some exemplary numerical values for a typical data set totaling more than 2GB of raw, uncompressed data corresponding to a single fragment in the F5 data model.

Vertices	3 420 772
Amplitude signal length ..	320
Amplitude signal type ...	16-bit integer
Storage size	2GB $\approx$ 2 189 294 080
compressed disk space ...	200MB $\approx$ 201 920 475
OpenGL max. texture size	128M = 134217728
OpenGL offset alignment	16
Min. chop length (lcm) ..	320
Max. number of chops ...	419430 = 134217728 / 320
Total number of instances	8.1 = 3420772/419430

Table 1: Exemplary data for base space chopping using an NVidia GTX 1650 graphics card for an amplitude signal stored as 16-bit integers on about 3.5 million points.

3.3 Performance

It is evident that with data sizes of 2GB *per fragment* the available memory of smaller graphics cards are quickly reached even with medium-sized point clouds encompassing a few tens of million vertices. As the GPU drivers are able to trade CPU RAM for performance, the rendering rate quickly drops by a factor of 20 or more from 14 frames per second to less than 1 frames per second (data measured for an NVidia GTX 1650 GPU, utilized for laptops). Still, this makes rendering huge data at least possible even on less powerful hardware, while navigation within singular fragments - ideally the "most visible" one [2] - remains interactive. A more powerful graphics card such as a GTX 1080 GPU with 8GB of on-board graphics RAM utilizes 6.3GB for a larger dataset of 14 Mio points and can maintain rendering rates faster than 15fps throughout all navigation events.

IO and GPU transfer The main bottlenecks occur during data transfer at two stages, firstly from the disk to CPU RAM, secondly from CPU RAM to GPU RAM. HDF5 offers a wide choice of compression filters. The high-speed lossless LZ4[5] filter easily achieves data reading rates of more than 500MB/s. For massive data such as the 2GB required per dataset fragment for amplitude signals, this still results in noticeable 4 seconds. Even though this is a one-time effort only, such initial I/O operations significantly stall an interactive application and strongly justify asynchronous data loading in some subthread.

But also the second part of CPU-GPU data transfer done via OpenGL's `glBufferData()` call lasts 1-2 seconds for realistic rates of 1.5GB/s. Such behavior also results in noticeable "stutter" of a rendering application each time a new fragment is loaded. As of now we have not yet implemented asynchronous GPU data transfer [6] but such would be a next step for enhanced user experience.

4 HANDS-ON DATA ANALYSIS

4.1 Visual Enhancement Techniques

The fragment shader provides various opportunities of enhancements for the most insightful visualization of LIDAR amplitude series among millions of combined renderings each of them containing complex information on their own. Some of these opportunities are envisioned first in (Fig. 7(a)) for a single LIDAR amplitude series before pointing out their meaning for the visualization of multiple LIDAR amplitude series as presented in Fig. 8.

Beyond the mere factual plain data display as in Fig. 7(a), contours (Fig. 7(b)) are easily added by overlaying the amplitude signal value of the pixel from the

[5] `https://lz4.github.io/lz4/`

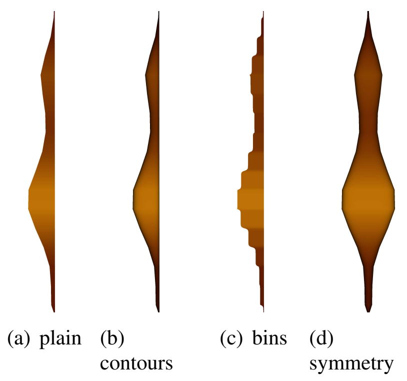


Figure 7: Visualization options for LIDAR amplitude series.

laser shot axis with the color value. Once this technique is applied to massive data Fig. 8(a), the singular amplitude signals are exposed with much better contrast Fig. 8(b). The interpolation of the signal bins, i.e. the texel access, is a potential tuning parameter as well such to change linear interpolation to nearest neighbor Fig. 7(c) as means to more accurately display the exact data values instead of a smooth visualization. In some situations displaying the amplitude signal symmetrically as in Fig. 7(d) is more appropriate since the choice to limit the visualization to the "left side" is entirely arbitrary.

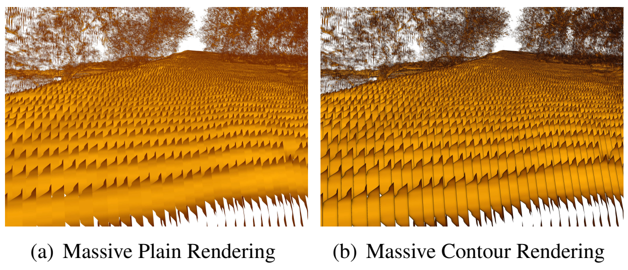


Figure 8: Visualization of a series of amplitude signals representing terrain surface and vegetation (trees) in the background using differing shading enhancements as described in the text. Colorization by amplitude replicates the geometric shape. 8(a) plain rendering, 8(b) contour rendering.

4.2 Feature Detection

The ultimately outcome of analysing a cloud of amplitude signals is the production of a cloud of points that correspond to physical objects. Such data reduction functionalities can be built into the fragment shader to provide immediate, real-time interactive assessment capabilities to the raw data as shown fully in Fig. 8. Since the fragment shader has full access to the texture describing the amplitude at each point, it can also compute the gradient of the signal. Thus, a color value can be limited to pixels where the sign changes for the texel left and right, i.e. indicating a local maximum at each of the pixels in Fig. 9 (same data as Fig. 8). In practice, the recorded amplitude series come with noise

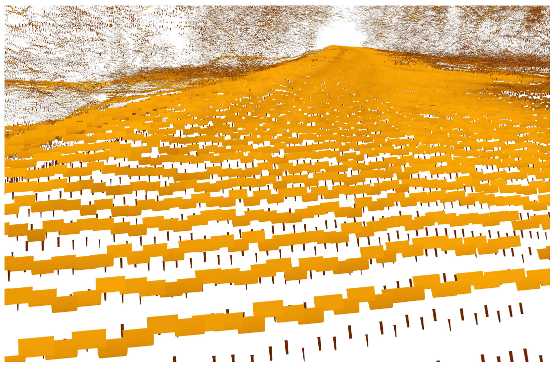


Figure 9: Visualization of same series of amplitude signals as in Fig. 8, but display of pixels restrained to bins containing a local maxima.

such that displaying any local maxima as shown in Fig. 9 results in undesirable visual clutter (Fig. 10(a)). A simple thresholding by user-specified noise level already improves the situation significantly, as shown in Fig. 10(c) and Fig. 10(d), where the river bed becomes distinguishable from the waterbody in the left and right part of the cross-section. Of course, more complex data

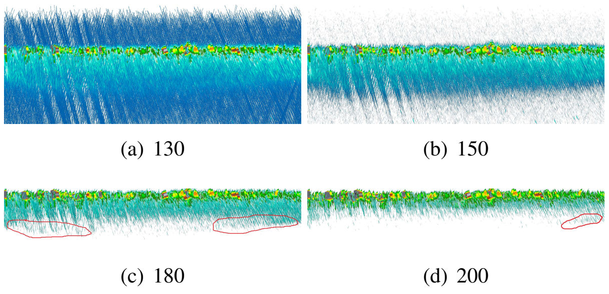
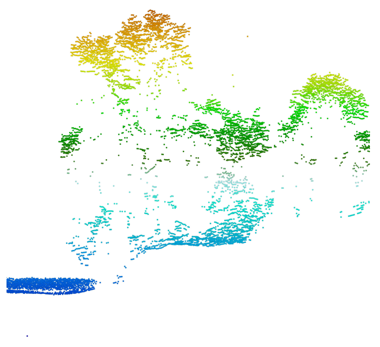
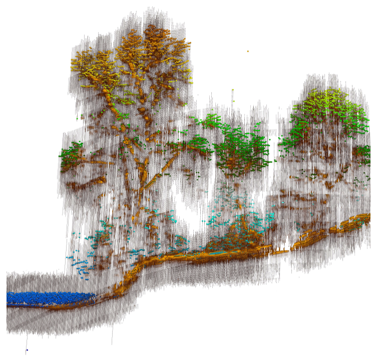


Figure 10: Cross-sectional view of a series of amplitude signals through a waterbody demonstrating the effect of realtime noise reduction on bins containing local maxima only, using increasing threshold values for data cutoff.

analysis and feature extraction methods can be incorporated into the fragment shading step, but come at the cost of reduced rendering speed. Simple but fast analysis parameters may already prove useful for subsequent data analysis routines that are performed on the GPU. For instance, Fig. 9 depicts the global maxima (band of broad stripes) followed by local maxima (band of thinner stripes). Such local maxima are not necessarily physical targets, but may be the result of the signal detector's response curve exhibiting overshooting and ringing behaviors. The mathematically correct approach to remove these artifacts is via deconvolution [7] with the perfect signal response of a delta function. However, they drastically increase the noise level numerically . Performing a full deconvolution at each frame rendering is unreasonable, but can be done as a one-time preprocessing step with the presented visualization method as verification and data assessment tool.

The immediate visualization of raw amplitude signal series from LIDAR datasets alone already allows for a direct and detailed exploitation of their potential. This is specifically emphasized in Fig. 11, where different amplitude visualization and point extraction methods are combined and compared in order to improve feature detection results in vegetation and terrain mapping (Fig. 11(a) compared to Fig. 11(c)).


(a) Maxima Points, colored by altitude


(b) Points + Full Amplitude

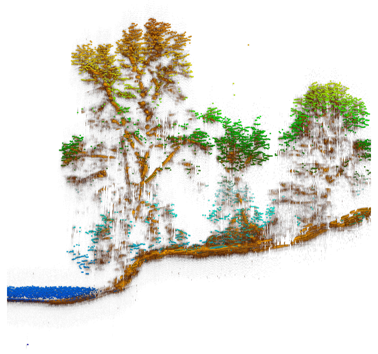

(c) Points + Noise cutoff at 150

Figure 11: Improvements of feature detection for vegetation and terrain mapping shown for a cross-section through a densely vegetated river foreland area. The combined visualization of originally digitized 3D points at data acquisition Fig. 11(a) and signal amplitude maxima Fig. 11(c) indicates that feature recognition can be substantially improved by further amplitude signal processing.

4.3 Water Clarity

Beyond point cloud extraction and identification of physical objects, the full amplitude signal provides more information that is not even of geometrical nature. Particularly, when observing lakes or rivers with a green laser source capable of penetrating the water body, the attenuation coefficient becomes measurable. The attenuation coefficient can be viewed as a proxy for determining the water clarity respectively water turbidity as an areal measurement. Such measures are important specifically for quantifying active fluvial sediment transport processes, or for habitat modeling in rivers and lakes and assessing their ecologic state.

The Beer-Lambert law is valid in the linear regime where the material is not highly scattering and states that the optical attenuation $I(s)/I_0$ is an exponential function of the path length s through the medium: $I(s) = I_0 e^{-\kappa s}$ with κ as the attenuation coefficient. This attenuation coefficient is constant for a fluid with homogeneous scattering properties. This exponential decay of the reflected light intensity along the path of the ray is immediately suitable for analysis from the amplitude signal.

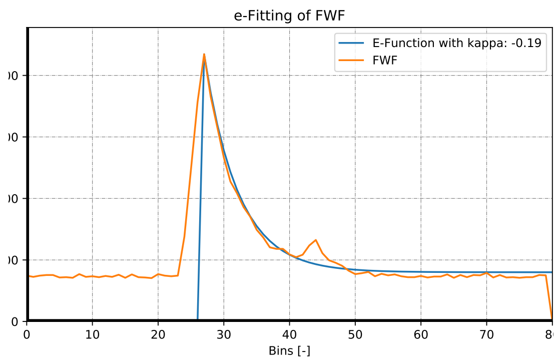
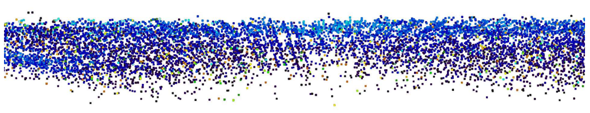
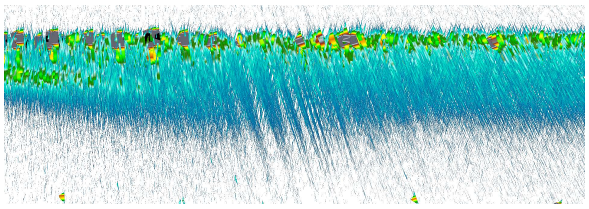


Figure 12: e-Fitting

A point cloud with reflectivity information contains this information as well, but provides only non-equidistant, lower spatial resolution along the ray of penetration (Fig. 13(a)). The information of subsequent connected data values is instead available from an extracted point cloud, whereas it is primarily available in the full signal (Fig. 13(b)). Actual amplitude signals are noisy, so fitting an exponential curve to each individual curve is needed to determine the attenuation coefficient for each laser shot (Fig. 12). Such analysis is too time-consuming to be performed at the rendering stage - which is aiming toward running at multiple frames per second, but it makes sense as a pre computation step. In future these pre-computed attenuation coefficients may be used at the visualization stage to subtract an exponential bias curve in order to enhance peaks above the exponential decay, instead of simple noise thresholding as in Fig. 10.



(a) Point Cloud of a water body



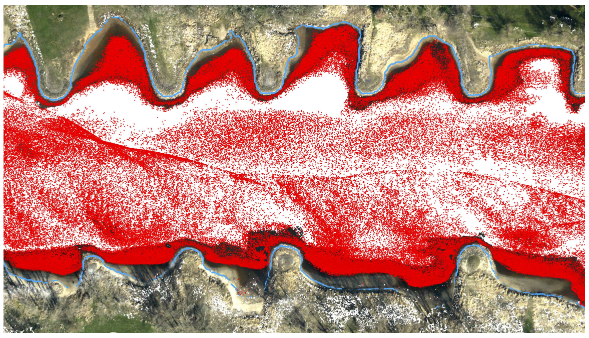
(b) Myriads of Amplitude Signals

Figure 13: Cross-section through a waterbody. (a) originally digitized 3D points at data acquisition colored by reflectivity decreasing from water surface (light blue indicating higher reflectivity) to depth (dark blue equaly lower reflectivity). (b) Associated amplitude signal series showing signal fade out with water depth with largest maxima at water surface and water ground (green to red color).

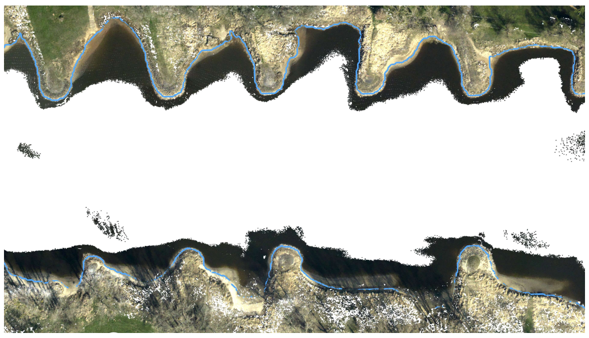
5 RESULTS

The presented visualization enhancement approaches of raw LIDAR amplitude signals are of great advantage when analyzing complex topobathymetric datasets, because they allow for direct and fast data access without time-consuming data post-processing efforts. Especially, the full amplitude visualization exposes river bed areas where the point cloud shows no coverage Fig. 15(a). Displaying local maxima also shows those signals on the water surface that triggered the original point cloud (compare Fig. 15(c) with Fig. 15(a)). Moreover, full amplitude visualization can also reveal terrain coverage below dense vegetation where the original point cloud indicates no coverage (Fig. 11). In practice, these two findings are of particular relevance to improve terrain extraction from LiDAR point clouds above and below water, and thus to significantly improve the quality level of survey data, which are already of high quality when considering only the original point clouds Fig. 14.

Further insights on water conditions in terms of clarity respectively turbidity derived from signal analysis can be of relevance for determining sediment transport processes in rivers related to floods or during snow melt. We are currently evaluating the potential of signal analysis for automated substrate mapping in water areas in terms of grain-size distribution along the river bed and surface roughness, which are verified by ground truth data.



(a) Enhanced Coverage



(b) Original Coverage

Figure 14: Comparison of a river bed based on amplitude signal analysis versus immediate LIDAR point cloud data.

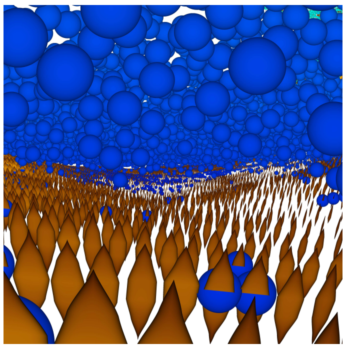
6 CONCLUSION

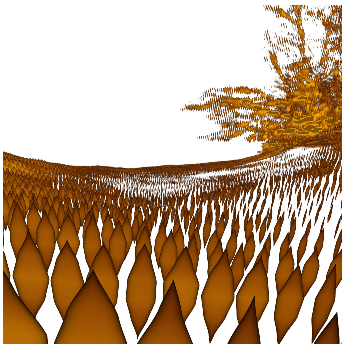
In this article we presented a rendering method for the direct, immediate visualization of massive raw amplitude response curves from LIDAR data acquisitions. The method provides intuitive insight into the potential of the data sets to allow for optimizing further data processing steps and potentially reducing the actual number of observation flights needed for airborne LIDAR bathymetry. Simple data processing steps can be performed interactively in realtime. As per our knowledge this is the first time that a direct, simultaneous visualization of dozens of millions of LIDAR amplitude response curves has been done. Its formulation within the fiber bundle model ensures a rigid mathematical basis yielding high performance independent of the application domain.

Our visualization technique has potential beyond LIDAR and can be applied to e.g. multispectral imaging with hundreds of spectral channels as well. Furthermore, the technique straightforwardly extends to other topological properties than vertices, e.g. edges or triangles, and can thus be applied also to high dimensional data given on more complex geometries.

REFERENCES

[1] F. Alted. Why modern cpus are starving and what can be done about it. *Computing in Science and Engg.*, 12(2):68–71, Mar. 2010.


(a) Point Cloud + Amplitude Signals


(b) Amplitude Signals

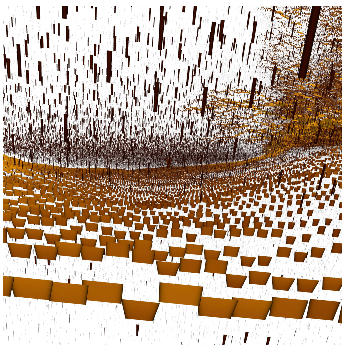

(c) Signal Maxima

Figure 15: Detailed view of a topobathymetric LIDAR dataset looking from a position at the river bed towards the water surface. (a) combined visualization of originally digitized 3D points at data acquisition (blue) and associated full amplitude signal in symmetric display (compare to Fig. 7(d)). (b) Visualization of amplitude signal series without points. (c) Visualization of amplitude signal maxima (compare to Fig. 9).

[2] W. Benger, D. Hildenbrand, and W. Dobler. Optimizing refined geometric primitive‘s leaflet visibility for interactive 3d visualization via geometric algebra. In *Proceedings of Computer Graphics International 2018*, CGI 2018, pages 267–272, New York, NY, USA, 2018. Association for Computing Machinery.

[3] W. Benger, M. Ritter, S. Acharya, S. Roy, and F. Jijao. Fiberbundle-based visualization of a stir tank fluid. In 17th *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 117–124, 2009.

[4] D. M. Butler and M. H. Pendley. A visualization model based on the mathematics of fiber bundles. *Computers in Physics*, 3(5):45–51, sep/oct 1989.

[5] T. Habermann, A. Collette, S. Vincena, J. J. Billings, M. Gerring, K. Hinsen, W. Benger, F. R. Maia, S. Byna, and P. de Buyl. The hierarchical data format (hdf): A foundation for sustainable data and software. *2014 AGU Fall Meeting*, 2014.

[6] L. Hrabcak and A. Masserann. Asynchronous buffer transfers. In P. Cozzi and C. Riccio, editors, *OpenGL Insights*, chapter 28, pages 391–414. CRC Press, July 2012.

[7] J. Pfleiderer. Methods of deconvolution. In W. C. Seitter, H. W. Duerbeck, and M. Tacke, editors, *Large-Scale Structures in the Universe Observational and Analytical Methods*, pages 298–305, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[8] R. Richter, S. Discher, and J. Döllner. *Out-of-Core Visualization of Classified 3D Point Clouds*, pages 227–242. Springer International Publishing, Cham, 2015.

[9] H. Sagan. *Space-Filling Curves*. Universitext. Springer-Verlag New York, 1994.

[10] M. Schütz. Potree: Rendering large point clouds in web browsers. Master’s thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/E193-02, A-1040 Vienna, Austria, Sept. 2016.

[11] R. Schwarz, G. Mandlburger, M. Pfennigbauer, and N. Pfeifer. Design and evaluation of a full-wave surface and bottom-detection algorithm for lidar bathymetry of very shallow waters. *ISPRS Journal of Photogrammetry and Remote Sensing*, 150:1 – 10, 2019.

[12] V. Springel. The cosmological simulation code gadget-2. *Monthly Notices of the Royal Astronomical Society*, 364(4):1105, 2005.

[13] I. Wald, A. Knoll, G. P. Johnson, W. Usher, V. Pascucci, and M. E. Papka. Cpu ray tracing large particle data with balanced p-k-d trees. In *Proceedings of the 2015 IEEE Scientific Visualization Conference (SciVis)*, SCIVIS 15, pages 57–64, USA, 2015. IEEE Computer Society.

[14] M. Wimmer and C. Scheiblauer. Instant points: Fast rendering of unprocessed point clouds. In *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG 06, pages 129–137, Goslar, DEU, 2006. Eurographics Association.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.28, No.1-2, 2020

Intermediate Representations for Vectorization of Stylized Images

D.-Amadeus J. Glöckner*
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam, Germany
daniel-amadeus.gloeckner@hpi.de

Lisa Ihde*
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam, Germany
lisa.ihde@student.hpi.de

Jürgen Döllner
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam, Germany
juergen.doellner@hpi.de

Matthias Trapp
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam, Germany
matthias.trapp@hpi.de

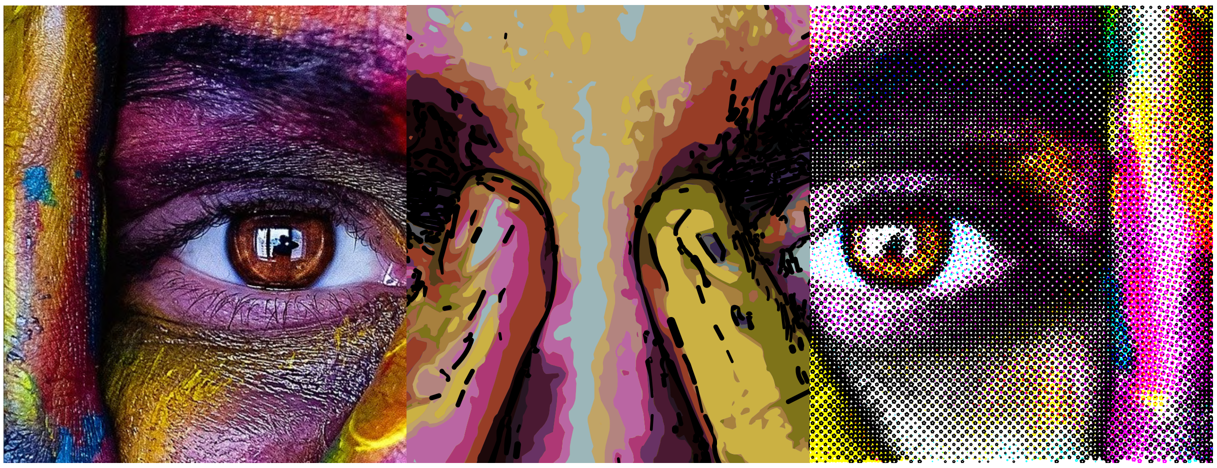


Figure 1: Exemplary results of the presented approach: input image (left), vectorized result of toon stylization (middle), and vectorized result of half-tone stylization (right).

ABSTRACT

This paper presents a new approach for the vectorization of stylized images using intermediate data representations to interface image stylization and vectorization techniques. It enables the combination of efficient GPU-based implementations of interactive image stylization techniques and the advantages of vectorized image representations. We demonstrate the capabilities of our approach using half-toning and toon stylization techniques.

Keywords: Image stylization, image processing, vectorization, half-toning, toon, interaction

1 INTRODUCTION

1.1 Motivation

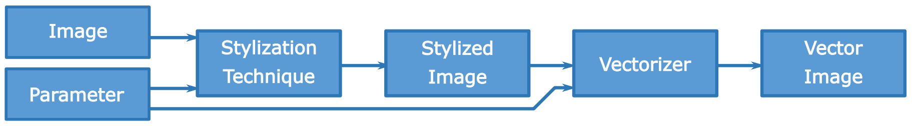
Image stylization techniques (e.g., half-toning [3] or toon [14]), and their applications offer users the opportunity to express their creativity and increasingly attracted attention during recent years. There are numerous stylization techniques that operate on raster images. Often, these can be implemented efficiently using Graphics Processing Units (GPUs). However, with respect to viewing or reproduction (e.g., for printing purposes), the resulting raster images are limited due to their limited spatial resolution.

In contrast thereto, vector images or graphics can be easily rasterized to required resolutions. However, their creation, editing and manipulation is often a cumbersome task. This work aims at combining efficient GPU-based implementation for image stylization and the advantages of vector-based presentations of their results (Figure 1).

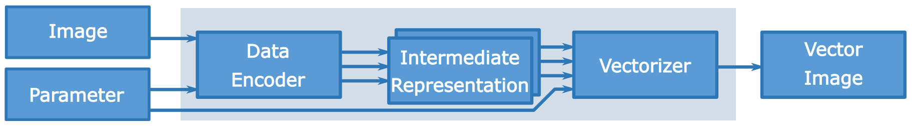
*These two authors contributed equally

1.2 Problem Statement

Vectorization techniques use raster images as input and generate a respective output vector image based on user defined settings such as number of colors and respective thresholds. Thus, their output often differs with respect to the number of colors being conveyed and the number of polygon required to represent image features such as colored areas or edges. However, they do not take information about the respective stylization technique into account (e.g., regions of unified color, edges, etc.). This often impacts the visual and data quality of results (e.g., high number of polygons or no explicit edge representations) and can lead to imprecise presentations of the stylization to be achieved (Figure 3).



(a) Common vectorization approach



(b) Vectorization with intermediate representation

Figure 2: Comparison between a common vectorization approach (a) and the concept of stylization-specific intermediate representations (b).

To counterbalance this, vectorization would require high-resolution input data to convey fine features. Subsequent editing or animation of vectorization results, e.g., changing half-tone elements or its size and orientation, is hard or would require to perform stylization and vectorization repeatedly, which can be a tedious process. Thus, to support fast image stylization and high-quality vectorization results, it seems promising to research intermediate representations that encode specifics of the respective stylization approach and facilitate representation and manipulation of the vectorized results.

1.3 Approach and Contributions

Using the naive method, the raster image is read in with the custom parameters for a stylization technique, which creates a stylized image. The stylization output is again represented as a raster image and serves as the subsequent input image for vectorization that generates the final vector image. However, this vectorization is performed without any prior information describing the nature of the stylization. Instead of directly applying the stylization to an input image, our approach encodes a stylization-specific raster-based intermediate representation that is used to generate vector images, subsequently. Such representations acting as "data maps" are suitable for fast GPU-based processing and facilitate interactive applications.

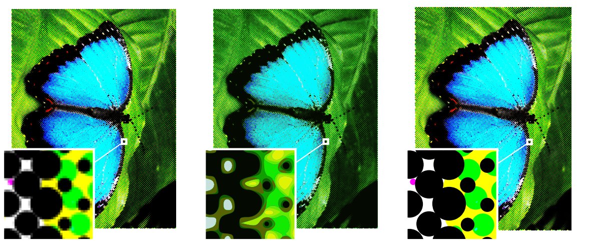
To summarize, this paper makes the following contributions:

(i) Concept for generating intermediate representation in order to create a vector image of a stylization effect.

(ii) Minimal adaptation of existing effect pipelines and their rendering passes with shader programs to use the raster-based outputs as basis for vectorization.

(iii) Provision of parameters for interactive modification of vectorization, which we demonstrate by stylization effects such as toon and half-tone filter.

The remainder of this paper is structured as follows. Section 2 reviews related work with respect to vectorization approaches in general and specific to stylized images. Section 3 presents the basic concept of our approach as well as implementation details specific to two exemplary image stylization techniques. Section 4 discusses the presented approach and Section 5 concludes this paper.



(a) Input raster image (1440 × 1920 pixels) (b) Inkscape vectorization (12 colors) (c) Proposed approach (114 120 elements)

Figure 3: Comparison of vectorization results of a half-toned input image (a). Using the method of (b), the input image was vectorized with Inkscape yielding half-tone elements that can hardly be recognized. With the method of (c), the element's position, size, and color is defined by an intermediate representation that is used for its instancing during vectorization.

2 BACKGROUND

Antoniou *et al.* [1] has pointed out the inflexibility of raster image formats and describes a conversion of such into XML-based structure like Scalable Vector Graphics (SVG) files. While raster images are based on pixels, in an XML-based environment elements such as points, lines or polygons can be reused. However, this approach only describes how to convert each pixel directly into a rectangle and thus does not use other shapes.

Further, Olsen and Gooch introduced an edge-based image reconstruction method that extracts vector edges using tracing, which can be applied on photographs [8].

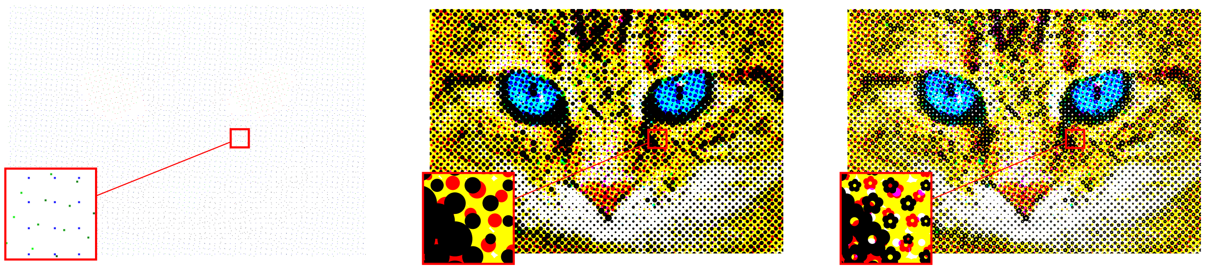

(a) Intermediate representation (adjusted) (b) Vector image using a dot element (c) Vector image using a flower element

Figure 4: Comparison of the intermediate representation (a) and two vector images (b) and (c). Each pixel of (a) encodes element color, size, and position.

The result is a vector image in grayscale, which is qualitatively well done, but this simplification does not include the color details from the original image. With respect to this, they showed that the vector-based image representation can be more memory efficient than raster images.

Noris *et al.* [7] present a vectorization method for raster images of line drawings with a focus on junction configurations. It produces a vectorization representation of the lines, but these have all the same width. Additionally, small details are not be captured by the topology extraction and the method needs solid color areas in the input image. Our approach allows the extraction of line widths from the input image and thus lines of variable width.

Simo-Serra *et al.* [13] focused on simplify sketch drawings and trained a convolutional neural network with a new data set of pairs of rough and simplified sketch drawings. The network provides a clean vector result out of a raster image with a rough pencil sketch. Their approach establishes the state of the art in sketch simplification. Additionally, the computation time using the GPU-based model achieves times under a second. Unfortunately, the approach depends on the quality and quantity of the training data. Furthermore, only line drawings are recognized, but no color areas or other compositions of an image. Therefore, this method is not suitable for vectorizing complex image stylization effects.

Xie *et al.* [15] treats vector images in an artistic way and promote the user to guide the vectorization interactively. For this purpose, the possible edges are calculated at first and the user then decides which ones should be used. In addition, the user can also draw edges or modify edges. The colors have to be applied manually, so there is no recognition of colored areas.

Favreau *et al.* [6] present a vectorization algorithm for photographs into cliparts based on stacking colored polygons. These layers are either opaque or semi-transparent and are faded to give the final result. But a segmented image is used as input, which must be created manually to generate a more stylized clipart.

Faraj *et al.* [5] explores the geometrical structure of an image to perform local operations like replacing or rotation. For it, a topographic map is used, which has a hierarchical order of color shapes in stacked layers [2]. Thus, this work can be considered as an image-abstraction method, which does not store the recognized mathematical forms in a vector image but in a tree. So this work is limited to the few operations that are offered. Our approach, however, saves all recognized components as SVG and can therefore be manipulated with common editing tools.

Several papers focus on image vectorization by detecting edges and shapes. Most of these works are limited to the color variety or only line vectorization. Both aspects are considered in our approach and are especially important for e.g., the toon effect. Moreover, our approach deals with the composition of stylization effects and examines the structure of effects in order to create the best possible generation of a vector image variant with the effect.

The previous works would try to generate a vector image from an input image with applied stylization effect. Thus the recognized areas and lines are vectorized independently of the applied style and the vector image is not sustainable. Possible later editing of this image

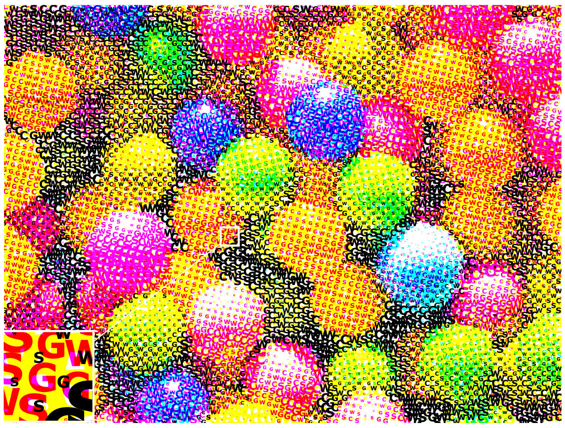
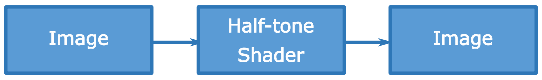
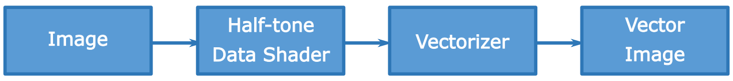


Figure 5: Half-tone image with multiple different elements ('W','S','C',G') at once.



(a) Raster image approach



(b) Our vectorized approach

Figure 6: Comparison between a raster image half-toning pipeline (a) and our slightly modified vectorized approach (b).

e.g., replacing of elements by half-toning would be very complex or not possible (see Figure 3b).

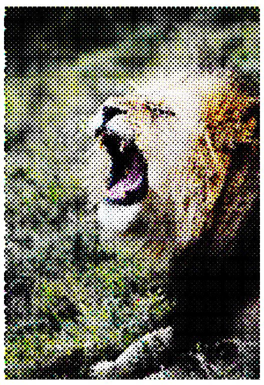
3 VECTORIZATION CONCEPT

3.1 GPU-based Image Stylization

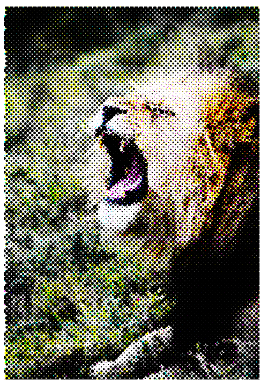
In order to access different intermediate steps of a stylization effect to be used for vectorization, a system is required where a stylization effect is divided into individual sub-steps, which can be combined as desired. Our approach is based on the work of Semmo *et al.* [12], which provides a GPU-based framework for complex image stylization effects and was also used for ProsumerFX [4]. It consists of three components, which are explained below:

Effect pipeline: An *effect pipeline* defines the individual steps that are applied to an input image to produce an output image at the end.

Effect: A single step of the effect pipeline is an *effect* (e.g., difference-of-Gaussians filter [14]). This effect can have parameters that can be adjusted by the user.



(a) Half-tone vector image without sub pixel offset



(b) Half-tone vector image with sub pixel offset

Figure 7: Comparison of the half-tone stylization with and without sub pixel offset. Without it the rounding to whole pixel positions can create error patterns which are visible in the output (a). The use of the offset textures increase the positional resolution enough to eliminate the error patterns (b).

Rendering pass: For the execution of the effects the *rendering passes* are needed, which are available as shader programs or C++ code.

3.2 General Approach

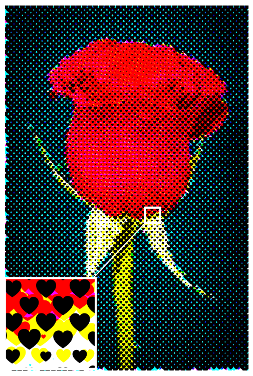
Figure 2 shows a comparison between a common vectorization approach (a) and the proposed one (b). While common vectorization of stylized images is performed by applying a stylization technique to an input image, resulting in a stylized raster image that is converted to a vector image using tracing [11]. In contrast thereto, our approach comprises the following main components:

Data Encoder: The encoder basically implements the image stylization technique and generates the intermediate data specific to this technique. Therefore, it reads in the image data and user parameters and executes the GPU-based stylization.

Intermediate Representation: The intermediate data represents the major components of the stylization Gestalt (e.g., colored areas edges, half-toning elements) and is represented using textures. These different textures can have different channel counts, do not necessarily have to have the same resolution as the input image, and can be obtained using render-to-texture in combination with multiple render-targets.

Vectorizer: A stylization-specific vectorizer receives the intermediate representation with user parameters and generates a final vector image. This makes it possible to separate the vectorization into substeps: thus, different components of the effect can be vectorized separately and then merged for the result. This approach enables to generate an specifically structured vector image for the respective stylization. To allow easy reuse of data encoders and vectorizers we integrated these as image processing passes into a modular system. These components can be combined with different image preprocessing steps to create new image stylizations.

In the following, we describe how these three components can be adapted to different stylization techniques.


(a) Half-tone vector image with no rotation applied to the elements

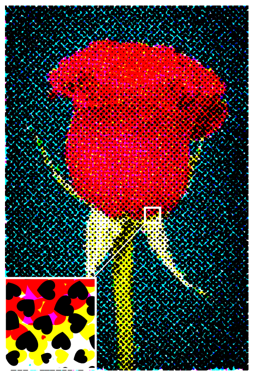

(b) Half-tone vector image with random rotation applied to the elements

Figure 8: Comparison of aligned and randomly rotated halftone elements. Aligned half-tone elements with an recognizable orientation introduce noticeable patterns (a). This can be avoided by randomly rotating the elements (b).

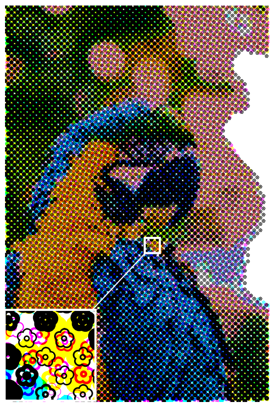
3.3 Half-toning Vectorization

Half-toning is a common reproduction technique for photography in printing, where the continuous tones of the images are represented by full-tone dots of varying size, shape, and density [3]. For the printing medium, usually the Cyan-Magenta-Yellow-Key (CMYK) colors are used for the point-data, which are mixed subtractively. This image style can be recreated perfectly with SVGs since it requires only single colored shapes, which can easily be represented in the SVG format.

Effect Pipeline. The original pipeline consists only out of one step. This is a GLSL shader getting a raster image and some parameters as input and outputting a raster image with the half-tone effect applied (Figure 6a). In our pipeline the shader is slightly modified and becomes the data encoder and its output is the intermediate representation. Furthermore, as a new step, a vectorizer is added, that reads the intermediate representation and generates the SVG output (Figure 6b). Vectorization for half-toning stylization can be achieved by implementing these components as follows.

Data Encoder. We use a variation of a single-pass GLSL shader-based half-tone stylization. Here, the individual elements constituting a half-tone stylization are not rendered directly into a raster image, but the determined element positions, colors, and sizes are encoded in three textures.

Usually the half-tone shader would determine for each pixel the position of the closest half-tone element center and the size of this element. This information is than used to determine if the examined pixel is inside the element radius and colored accordingly. This


(a) Line thickness modulated multi channel image

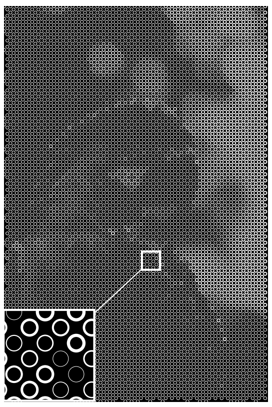

(b) Line thickness modulated single channel image[1]

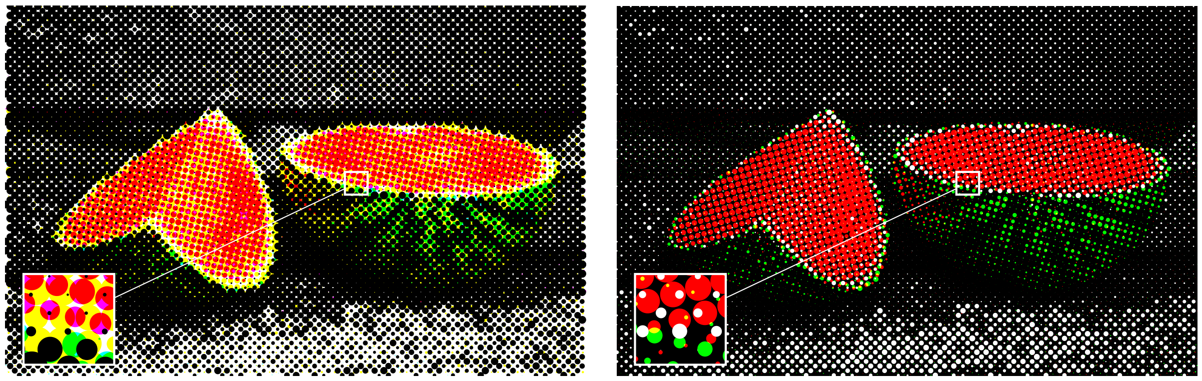
Figure 9: Color intensity modulated using line thickness instead of shape size.

is done for each of the color layers (usually cyan, magenta, yellow and black). In our modification we only set the value if the pixel is less then half a pixel away from the center, therefore only modifying the pixel at the center of the element. Also instead of choosing the color dependent on the half-tone color that is processed, this just determines which texture channel is written to, as described in the next paragraph.

Intermediate Representation. This is a set of three textures with pixels of different colors (Figure 4a). The color of the half-tone element is determined by the color channel where the red, green, blue and alpha channels represent the cyan, yellow, magenta and black channels respectively. The half-tone element size is determined by the intensity of each pixel in the corresponding channel in the first texture. Pixels with a value of zero do not represent an element. The position of the half-tone element is determined by the position of the pixel. To improve the positional accuracy and therefore prevent artifacts introduced by rounding to a full pixel position, we use the other two textures. These encode a sub-pixel offset in the x- and y-direction respectively (Figure 7). This increase in resolution allows us to decrease the actual resolution of the three intermediate textures. The resolution of the three textures is not dependent on the input image but on the half-tone frequency. The resolution must be big enough that two elements of the same color layer do not correspond to the same pixel position.

Vectorizer. This outputs an SVG for further editing. First, half-tone elements are generated using a template (cf. Line 5 in Listing 1). Following to that, a group element (`<g>`) is created for each of the CMYK color channels. Inside these, instances of the template are created for each of the non zero pixels in the main tex-

[1] Depending on the used PDF viewer it might be necessary to zoom this figure to full screen size.



(a) Half-tone vector image with the color space CMYK for the half-tone elements. (b) Half-tone vector image with the color space RGB for the half-tone elements.

Figure 10: Comparison of the CMYK and RGB color spaces for the halftone elements. Images with a high black content would have to generate a lot of black halftone elements (a). It is easier to change the color space, since black is the basis for the RGB color space (b).

ture (`<use>`). The position and intensity of the pixel is used for the transformation, which consists of a scaling and a shift. The x- and y-offset textures are used for fine adjusting the position. To achieve the characteristic subtractive color mixing, the SVG blending mode is set to multiplication (Line 2) and applied to the respective color group. For wider support of display and editing tools the blend mode is specified using the `comp-op` attribute and Cascading Style Sheet (CSS) styling.

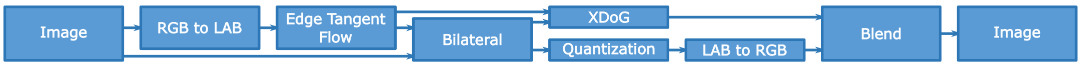
Figure 3b shows the vectorization result of a stylized half-tone image (Figure 3a) in a common way using Inkscape. For comparison, Figure 3c shows that the appearance of half-tone elements can be conveyed, yielding high-quality visual output. The common vectorization achieves only colored areas that do not represent the half-tone elements sufficiently. Using the blend mode for creating the mixed color areas instead of creating separate shapes for the overlapping areas, we can save calculation time and decrease the file size. This also allows users to modify the resulting SVG more intuitively because the shapes of the half-tone elements are still intact and can be manipulated as a whole.

```
1  <svg width="1280" height="484" ...>
2  <style>.blend{mix-blend-mode: multiply;}</style>
3  <g>
4    <g opacity="0">
5      <g id="element"> <!-- Graphical element to re-use -->
6        <path transform="..." d="m 0.0,-0.5 ..."/>
7      </g>
8    </g>
9    <!-- Cyan color group -->
10   <g fill="#0ff" comp-op="multiply" class="blend">
11     <use xlink:href="#element" transform="..."></use>
12     <use xlink:href="#element" transform="..."></use>
13     ...
14   </g>
15   <g fill="#f0f" comp-op="multiply" class="blend">...</g>
16   <g fill="#ff0" comp-op="multiply" class="blend">...</g>
17   <g fill="#000" comp-op="multiply" class="blend">...</g>
18 </g>
19 </svg>
```

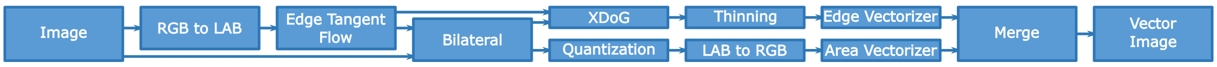
Listing 1: Exemplary structure of a SVG re-using elements to represent the results of an half-tone stylization.

The usage of an instanced template decreases the overall file size for half-tone elements more complex than a circle. This also allows the user to modify the shape of all the elements at once by just editing the template. Actually instancing the group containing the shape even allows swapping the shape for an entirely new one easily. Also, the reference for all the elements does not always have to refer to the same template and thus displaying different elements at once is possible. Figure 5 shows how four templates are displayed at once, which have the four letters "W", "S", "C" and "G" defined as path. Not only the reference can be changed, also the transformation matrix. Thus each element can be rotated as desired. This has the benefit that the grid structure is not immediately visible. Figure 8a shows that the half-tone element of a heart reveals the direction of the grid through its tip, but by random rotation the grid is no longer recognizable. Furthermore, the scaling in the transformation matrix can also be adjusted. Thus, the size can be easily changed for all elements and allows the user to define the size according to his own perception.

It is also possible to make all elements the same size and allow intensity modulation through the outlines of the elements. Therefore, the fill is changed to transparent and the stroke width is scaled differently. Figure 9a shows how even a complex half-tone element can express the intensity through the line thickness and Figure 9b shows this with a circle as half-tone element. Currently, the color space CMYK is used with a subtractive blend mode. This can also be changed to additive and the color space Red-Green-Blue (RGB). For this purpose, the SVG blend mode must be changed from *multiply* to *lighten* and the shader of the rendering pass must use the color space of RGB instead of CMYK, accordingly. Also only certain color channels can be displayed. For this purpose, the grayscale rep-



(a) Raster image approach as abstract overview of the 29 rendering passes based on Semmo *et al.* [12]



(b) Our vectorized approach

Figure 11: Comparison between a raster image toon pipeline (a) and our slightly modified vectorized approach (b).

resentation of the image is used for the intensity value. There are many ways to generate a grayscale representation, e.g., using the brightness value per pixel. Figure 10 shows how the respective color components can also be displayed individually. The frequency can be adjusted by the user for the level of detail of the display. A higher value means that the grid for the color channels has a higher resolution and therefore more elements are displayed.

In addition to the many interaction possibilities, the SVG structure allows easy access to a representative element. Thus, further modifications, e.g., using JavaScript, are possible to animate the elements.

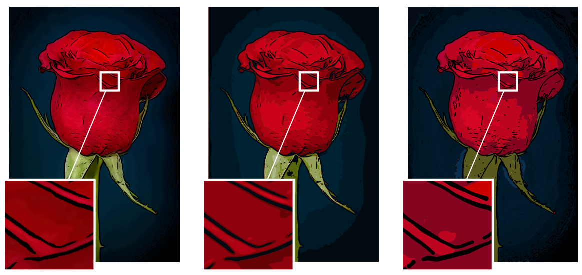
3.4 Toon Vectorization

Toon stylization [14] is a typical representative for image abstraction techniques, e.g., for expressing caricature drawings. This image style is a good candidate for vector image representation because it is composed of mostly uniform colored shapes as well as lines, which both are geometric primitives of SVG.

Effect Pipeline. The original pipeline already consists of several steps. The underlying GPU-based effect pipeline based on Semmo *et al.* [12] for toon stylization calculates two parts from an input image. On the one hand an edge detection is performed with the extended difference-of-Gaussians and on the other hand a color quantization based on local luminance in the LAB color space. These two partial results are merged again for the output (Figure 11a). In our pipeline the result of the edge detection is passed to a thinning step. The thinned edges are then used for the edge vectorization. Additionally, the quantized color data is used for an area vectorization. Instead of blending the intermediate results pixel based, the two vectorization results are merged to create the final vector image (Figure 11b).

Data Encoder. We use the output of the multi-pass stylization technique to generate on the one hand a stylized color quantized raster image and on the other hand an edge raster image from the input. Usually these two images would be blended on a per-pixel basis to generate the stylized output image. The edge image is additionally passed through a thinning pass to create lines of one pixel thickness.

Intermediate Representation. Two textures are used to represent the stylization outputs for subsequent vectorization: (1) a RGB texture stores quantized colored areas and (2) a luminance texture stores edge data. The edge data contains a Boolean (Figure 13b), whether an edge exists at the pixel and another value, how thick the edge would be at the location. To calculate these values the rendering pass is used, which creates the edge image for the toon stylization. This is used as the input image to perform a thinning [9, 10]. Thereby the edge widths in the image are always reduced by one pixel from out-

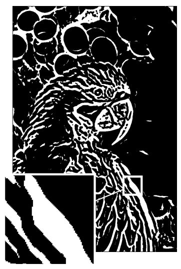


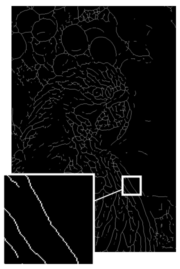
(a) Input raster image (1280 × 1920 pixels) (b) Inkscape vectorization (11 colors) (c) Proposed approach

Figure 12: Comparison of the vectorized results for a toon effect (a). In method (b), the stylized image was vectorized with Inkscape. In method (c) the intermediate representation was used to generate a vectorized result, which has a typical toon color area and the edges are paths with individual line width.

```
<defs>
 <inkscape:path-effect
   end_linecap_type="round"
   scale_width="1"
   miter_limit="4"
   linejoin_type="round"
   start_linecap_type="round"
   interpolator_beta="0.2"
   interpolator_type="CentripetalCatmullRom"
   sort_points="true"
   offset_points="0,9.5| 1,2.5 | 2,9.5"
   lpeversion="1"
   is_visible="true"
   id="path-effect998"
   effect="powerstroke" />
</defs>
<path
  id="CPUVectorizeEdgePassProcessor998"
  style="fill-rule:evenodd"
  inkscape:path-effect="#path-effect998"
  fill="rgb(0,0,0)"
  stroke="none"
  stroke-linecap="round"
  inkscape:original-d="M 79 9.5 Q 74.1 1.8 68.5 1 L 67 4.5"
  d="m 87.014,4.399 c 0,0 0.411,0.350 0,0 C 85.193,2.849
       81.295,0.691 77.096,-0.392 73.863,-1.227 70.964,-1.173
       68.853,-1.474 a 2.5,2.5 15.554 0 0 -2.651,1.490 c
       -0.5,1.166 -6.611,0.618 -7.934,0.742 -0.180,0.016 0,0 0,0 A
        9.5,9.5 90 0 0 75.731,8.242 c 0,0 0.112,0.142 0,0 C
       74.909,7.199 70.297,3.151 70.797,1.984 l -2.651,1.490 c
       1.338,0.191 2.260,1.885 2.607,3.967 0.428,2.568 0.093,5.734
        0.231,7.157 0.033,0.349 0,0 0,0 A 9.5,9.5 90 0 0
       87.014,4.399 Z" />
```

Listing 2: Definition of offset points in Inkscape to apply this effect to a path via the Id.


(a) Thresholded edge-pass texture (720 × 1080 pixels)


(b) Edge texture after thinning

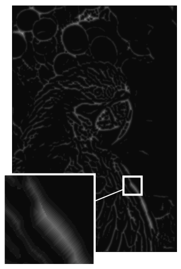
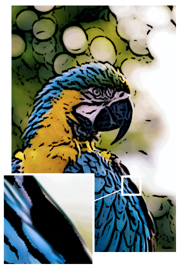
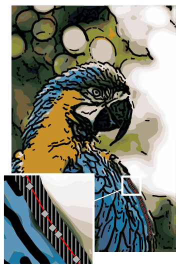

(c) Distance data used for local line thickness

Figure 13: Edge preprocessing for the toon stylization. The edge pass from the toon effect is thresholded to create a binary image (a). This is than thinned iteratively to yield the edge skeletons (b) necessary for the vectorization. During the thinning the shortest distance from outside the to the middle is determined (c).

side to inside during an iteration until only one-pixel thick skeleton results (Figure 13b). Thus, the number of times a pixel has been deleted is saved as distance data in the resulted skeleton and this value can be used later to reconstruct the actual width of the line (Figure 13c).
Vectorizer. This consists of three components: a area vectorizer for colors, a edge vectorizer, and a merger. The area vectorizer generates colored polygons from separate quantized color-areas using a standard vectorization approach[2]. The edge vectorizer generates paths based on the edge image. This is performed by first creating paths with one node per pixel and than simplifying the path using line and curve segments where possible. The resulting nodes read the actual width from the luminance texture. This can then be used for the varying line thickness of a path. The SVG standard does not support varying line widths, but there is a proposal for this[3]. However, the most common vector graphics editors e.g., Inkscape and Adobe Illustrator support the display of this. In our approach, we have exported the results for display in Inkscape. For it, we set the path effect *powerstroke*. This requires offset-points (Listing 2), where the first value is the position on the path (e.g., 0 for the first node, 1 for the second node, etc.) and the second value is the distance from the node, the value we got during the thinning process.

Finally, the merger combines the two partial results to an SVG (Figure 12c). Figure 14b and Figure 14c show the difference and added value achieved by varying path thicknesses. This option has advantages over previous


(a) Combined color and edge layer of the raster sylization (720 × 1080 pixels)


(b) Vector output with average line width per path

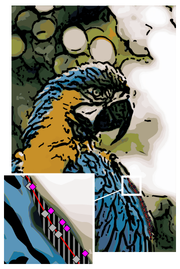

(c) Vector output with dynamic line width along paths

Figure 14: Different line width representations. In the raster based stylization the line thickness can vary along the line (a). The SVG standard (1.1) does not include a way for varying line thickness along a path. Therefore the width information can only be used to set the thickness to e.g., the average (b). Vector graphic editors like Inkscape support dynamic stroke widths with there own extinctions and therefore allow for a higher degree of detail (c).

work, where only paths of constant thickness could be created.
Figure 12a shows an exemplary raster-image output of the toon stylization technique. By using the common method, both color areas and edges are represented as colorized polygons in the final vector image (Figure 12b). In contrast thereto, our approach handles colored areas and edges separately by creating line paths on top of polygons (Figure 12c), which can be edited individually. This technique is closely related to how vector artists work, namely creating different layers for different elements of the image like the line and color layer. This also allows for easy manipulation of all the lines or shapes at once (e.g., changing the thickness or color). Also by including the outlines as paths in the SVG they can be edited as such e.g., path patterns or stylizations can be applied.

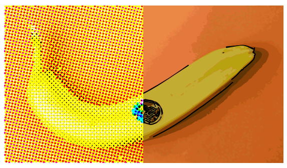
4 DISCUSSION

4.1 Performance Evaluation

We tested the run-time performance of prototypical applications of our approach to two stylization techniques with respect to the following resolutions: 1280 × 720 (HD), 1920 × 1080 (FHD), 3840 × 2160 (4K), and 7680 × 4320 (8K-UHD) pixels. The performance tests were conducted using a NVIDIA GeForce GTX1080Ti GPU with 12 GB VRAM on a Intel i7 CPU with 3.7 GHz and 32 GB RAM. We used two different input images with different level of detail in the form of existing edges and number of colors (Figure 15). Measurements are averaged over 100 runs per input image (Table 1).

[2] `https://github.com/jankovicsandras/imagetracerandroid/blob/master/process_overview.md`
[3] `https://www.w3.org/Graphics/SVG/WG/wiki/Proposals/Variable_width_stroke`

(a) Low detail raster image used for performance test.


(b) Split view of toon and half-tone vector outputs of the low detail image.

(c) High detail raster image used for performance test.

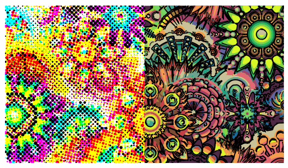

(d) Split view of toon and half-tone vector outputs of the high detail image.

Figure 15: Images used for the performance test. Image (a) has very little detail. Image (c) on the other hand has a lot of details.

One can observe an increase of run-time with respect to the (1) input image resolution, (2) the complexity of the stylization techniques, (3) the number of vectorization passes, and (4) level of detail. For simple single-pass stylization techniques such as half-toning, which yield only a single vectorization pass, interactive frames rates could be achieved. Additionally, this image stylization technique is independent from the level of detail in the input image.

However, for more complex stylization techniques that yield multiple vectorization passes, the resulting performance is below interactive constraints for input resolutions greater than High Definition (HD). Furthermore, the level of detail affects the running time of the complex toon effect. An input image with more edges and colored areas leads to slower performance especially for the tracing step. Thus, this effect takes almost double the time with a more detailed image as with a less complex one. This is mainly due to the higher number of steps for thinning and also a higher quantity of path effects that have to be generated for the individual line thicknesses.

4.2 Image Stylization Representation

Our approach to use intermediate representations to create stylized vector images was successfully demonstrated using two stylization effects. Here we use the output of the existing rendering passes and extend them only with calculations like thinning. Furthermore, we enable interaction with parameters to change the vectorization. We could show that our result of the stylized vector image is more suitable for further use than a vector image that is only based on a stylized image. This is due to our representation in the SVG structure, which facilitates editing. For example, the use of cloning in the half-tone effect allows the shape of the half-tone element to be easily replaced in the final SVG. Only the one template element needs to be adjusted. Besides the exchange of elements, these can also be animated by accessing the certain SVG elements and their attributes, e.g., line thickness, displacement or rotation.

4.3 Limitations

Besides the many advantages of an SVG, the presented implementation has a few limitations.

Currently, the variable line thickness of the toon effect can only be displayed in an appropriate editor (e.g., Inkscape) that supports variable line thickness. For each effect we have considered individually how this effect can be logically broken down into vectorization steps. Now vectorization steps exist to trace edges and surfaces. There are also other pipelines for the placing of clones and the thinning step. All these vectorization steps can be reused for future effects.

Another limit is e.g., if the user edits the shape of the template object in a half-tone SVG, then all clones would be edited as well. This leads to high update times in vector image editors like Inkscape. But this is rather due to the software than the vector file.

4.4 Future Work

For future work, we plan to test our approach with additional image stylization techniques and explore possibilities of further intermediate representation variants. With respect to run-time performance improvements, a more compact intermediate representation could speed-up the vectorizer by avoiding scan-lining during tracing.

5 CONCLUSIONS

This paper shows how the extension of image vectorization by means of intermediate data can be used to increase the output quality of vector images for certain types of image stylization techniques. Our approach, which is to perform stylization as part of vectorization rather than as a pre-process, is actually close to how vector artists work, since they commonly create complex artworks as multiple layers (e.g., one layer for color regions and another layer for outlines), and fill-in regions with vector patterns. We enable the combination of fast GPU-based image stylization techniques and high-quality visual output by means of vector images. However, this requires minimal changes for encoding the output of each technique to yield the intermediate representation that drives the particular vectorization. We implemented and tested the presented

Table 1: Run-time performance comparison in total (*Total*), for the stylization (*Style*), and tracing stages (*Trace*) (in milliseconds) for input images of different spatial input resolutions (in pixels).

	Half-tone (H) **[3]** Single GPU-pass 1 Vectorization pass						**Toon** (T) **[14]** 29 GPU-passes 2 Vectorization passes					
	Low Detail Image			High Detail Image			Low Detail Image			High Detail Image		
Input Resolution	H_{Style}	H_{Trace}	H_{Total}	H_{Style}	H_{Trace}	H_{Total}	T_{Style}	T_{Trace}	T_{Total}	T_{Style}	T_{Trace}	T_{Total}
1280×720	26.5	35.0	61.5	24.7	37.7	62.4	24.8	792.2	817.0	20.0	1389.4	1409.4
1920×1080	23.6	57.2	80.8	20.6	57.3	77.9	26.5	1590.7	1617.2	30.4	2822.5	2852.9
3840×2160	33.1	154.9	188.0	34.7	147.3	182.0	37.2	6037.0	6074.2	35.3	10655.1	10690.4
7680×4320	52.3	545.9	598.2	47.6	544.1	591.7	91.0	28167.6	28258.6	102.3	53362.9	53465.2

concept using GPU-based half-tone and toon stylization techniques.

ACKNOWLEDGMENTS

We thank the reviewers for their insightful comments. The project was supported by the Federal Ministry of Education and Research (BMBF), Germany (mdViPro, 01IS18092).

REFERENCES

[1] Vyron Antoniou and Lysandros Tsoulos. Converting raster images to xml and svg. In *SVG Open*, 01 2004.

[2] Vicent Caselles, Bartomeu Coll, and Jean-Michel Morel. Topographic maps and local contrast changes in natural images. *International Journal of Computer Vision*, 33:5–27, 1999.

[3] Oliver Deussen and Tobias Isenberg. Halftoning and stippling. In Paul Rosin and John Collomosse, editors, *Image and Video-Based Artistic Stylisation*, pages 45–61. Springer London, London, 2013.

[4] Tobias Dürschmid, Maximilian Söchting, Amir Semmo, Matthias Trapp, and Jürgen Döllner. Prosumerfx: Mobile design of image stylization components. In *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*, SA '17, New York, NY, USA, 2017. Association for Computing Machinery.

[5] Noura Faraj, Gui-Song Xia, Julie Delon, and Yann Gousseau. A generic framework for the structured abstraction of images. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, NPAR '17, New York, NY, USA, 2017. Association for Computing Machinery.

[6] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. Photo2clipart: Image abstraction and vectorization using layered linear gradients. *ACM Trans. Graph.*, 36(6):180:1–180:11, November 2017.

[7] Gioacchino Noris, Alexander Hornung, Robert W. Sumner, Maryann Simmons, and Markus Gross. Topology-driven vectorization of clean line drawings. *ACM Trans. Graph.*, 32(1):4:1–4:11, February 2013.

[8] Sven Olsen and Bruce Gooch. Image simplification and vectorization. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, NPAR '11, pages 65–74, New York, NY, USA, 2011. ACM.

[9] T. Pavlidis and S. L. Horowitz. Segmentation of plane curves. *IEEE Transactions on Computers*, C-23(8):860–870, Aug 1974.

[10] Theo Pavlidis. A thinning algorithm for discrete binary images. *Computer Graphics and Image Processing*, 13(2):142 – 157, 1980.

[11] Philip J. Schneider. An algorithm for automatically fitting digitized curves. In Andrew S. Glassner, editor, *Graphics Gems*, pages 612–626. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[12] Amir Semmo, Tobias Dürschmid, Matthias Trapp, Mandy Klingbeil, Jürgen Döllner, and Sebastian Pasewaldt. Interactive image filtering with multiple levels-of-control on mobile devices. In *SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications*, SA '16, New York, NY, USA, 2016. Association for Computing Machinery.

[13] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: Fully convolutional networks for rough sketch cleanup. *ACM Trans. Graph.*, 35(4), July 2016.

[14] Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. *ACM Trans. Graph.*, 25(3):1221–1226, July 2006.

[15] Jun Xie, Holger Winnemöller, Wilmot Li, and Stephen Schiller. Interactive vectorization. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 6695–6705, New York, NY, USA, 2017. ACM.

OPEDD: Off-Road Pedestrian Detection Dataset

Peter Neigel[1,2], Mina Ameli[1], Jigyasa Katrolia[1], Hartmut Feld[1], Oliver Wasenmüller[1], Didier Stricker[1,2]

[1]German Research Center for Artificial Intelligence
Trippstadter Str. 122
67663 Kaiserslautern
Germany

[2]Technische Universität Kaiserslautern
Gottlieb-Daimler-Str., Gebäude 42
67663 Kaiserslautern
Germany

{firstname.lastname}@dfki.de

ABSTRACT

The detection of pedestrians plays an essential part in the development of automated driver assistance systems. Many of the currently available datasets for pedestrian detection focus on urban environments. State-of-the-art neural networks trained on these datasets struggle in generalizing their predictions from one environment to a visually dissimilar one, limiting the use case to urban scenes. Commercial working machines like tractors or excavators make up a substantial share of the total number of motorized vehicles and are often situated in fundamentally different surroundings, e.g. forests, meadows, construction sites or farmland. In this paper, we present a dataset for pedestrian detection which consists of 1018 stereo-images showing varying numbers of persons in differing non-urban environments and comes with manually annotated pixel-level segmentation masks and bounding boxes.

Keywords
Pedestrian Detection, Instance Segmentation, Non-Urban Environment, Off-Road, ADAS, Commercial Vehicles

1 INTRODUCTION

The detection of pedestrians is a major problem for Advanced Driver Assistance Systems (ADAS) and substantial effort has been made in the past decade to advance the performance of detection methods. Current state-of-the-art approaches to pedestrian detection in monocular RGB-images rely on convolutional neural networks (CNN) that output either bounding boxes or pixel-level segmentation masks for every depicted person [1]–[6]. To train these networks for pedestrian detection in a supervised manner large image datasets are needed that come with ground truth annotations for person bounding boxes or segmentation masks. The most commonly used datasets for this task portray scenes in *urban environments*, motivated by the need for and the recent progress in autonomous driving systems for private passenger cars and commercial cargo trucks.
In contrast, most industrial vehicles operate in completely different environments. Although used in dozens of industries, from coarse earthwork operations to tactful harvesters, and making up a substantial share of the total number of motorized vehicles, they are currently neglected by published datasets available for pedestrian detection. In many cases, neural networks trained on urban images fail to generalize from the context of the city to a visually different one, resulting in reduced detective capabilities for off-road environments and posing a problem for ADAS in the context of mobile working vehicles, e.g. automated emergency brakes for tractors, excavators or harvesters. Additionally, urban environments constrain the variety of poses that pedestrians are portrayed in: Most are seen walking or standing upright on the pavement. Industrial or agricultural vehicles in off-road environments can find people in unusual poses, e.g. crouching or lying down while picking crops or doing construction work. These points pose an obstacle to the safety of humans around autonomously operating vehicles in non-urban contexts.

Since there is evidence suggesting that data may be more important than algorithms for performance [7], [8], in this paper we aim to contribute a stereo image dataset of pedestrians in 5 different off-road environments: *Meadows, woods, construction sites, farmland and paddocks*. The persons shown in the

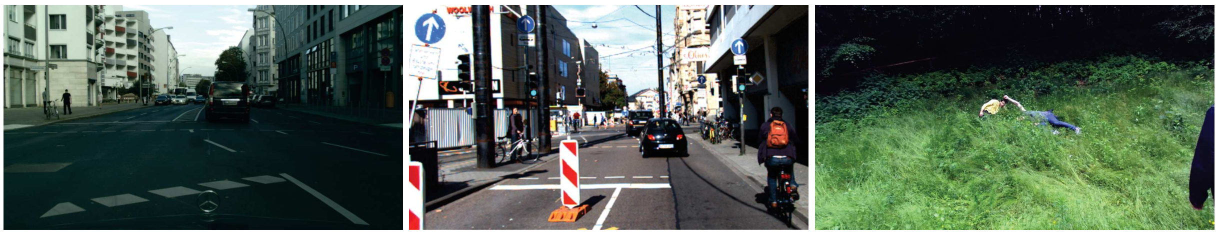


Figure 1: Example images from different datasets. Left: Cityscapes dataset [9]. Center: KITTI Stereo 2015 Dataset [10]. Right: OPEDD. From colour spectrum, over gradient orientations to pedestrian's poses: The visual makeup of urban scenes is substantially different from off-road environments, impeding generalization in detections by neural networks.

dataset are portrayed in varying poses, with some being highly unusual in the ADAS context, e.g. extended limbs, handstands, crouching or lying down. Additionally, our dataset shows significant occlusion of persons from vegetation, crops, objects or other pedestrians. The dataset itself consists of 1018 stereo images, where the left image comes with manually created ground truth pixel-level segmentation masks and individual IDs for every portrayed pedestrian, allowing the data to be used for tasks like object detection (bounding boxes), semantic segmentation (pixel masks) or instance segmentation (pixel masks and IDs). In addition to the dataset itself, we provide depth maps generated from the stereo images and the stereo video sequences from which the images of the dataset were selected.

2 RELATED WORK

Since the popularization of neural networks for object detection, many datasets with annotated pedestrians have been published, either explicitly for the task of pedestrian detection or as part of complete scene segmentation.
Cityscapes [9] is a widely used dataset for urban street scenes. Captured with a stereo camera setup on a car in 50 different cities, it consists of 25000 images, out of which 5000 are provided with fine-grained semantic labels and 20000 are coarsely labelled. The depicted classes include persons, cyclists, cars and other motorized vehicles, while pixel-level semantic labels and instance IDs enable the evaluation of object detection, semantic- , instance- and panoptic-segmentation tasks.
KITTI [10] is a popular driving dataset similar to Cityscapes in terms of portrayed environments, offering benchmarks for different object detection and segmentation tasks. The capture setup consists of a stereo camera in addition to a 360° laser scanner and GPS, providing video sequences and ground truth for the evaluation of tasks like detection, segmentation, sceneflow, depth estimation, odometry, tracking and drivable road detection.
The Caltech Pedestrian Detection Benchmark [11] consists of about 250,000 images frames of regular traffic in an urban environment. A total of 350,000 bounding boxes label circa 2300 unique pedestrians, but no annotations in terms of pixel-level segmentation masks are included.
These datasets are a subsample of publications that focus solely on urban environments and roads, making them unsuitable for the magnitude of industrial and agricultural commercial vehicles.
In contrast, the NREC Agricultural Person-Detection dataset [12] provides a large number of images for off-road pedestrian detection in apple and orange orchard rows, taken from a tractor and a pickup truck. Pedestrian poses include non-standard stances found typically in the orchard environment, only pedestrian bounding boxes are included however, making them incompatible for semantic- and instance segmentation tasks.
In total, the currently published datasets do not allow for comprehensive benchmarking on different pedestrian recognition tasks in off-road-environments. Our presented work offers manually generated ground truth segmentation masks besides bounding boxes, displays a larger and more varying number of environments and includes poses typical for the corresponding off-road environment as well as stances that are completely arbitrary and unusual, filling a gap in published datasets not yet covered.

Dataset	Number of Images	Depth	Segm. Masks	Environment	Poses
KITTI	14,999	LIDAR	✓	Urban	Std. Urban
Cityscapes	25,000	Stereo	✓	Urban	Std. Urban
Caltech	39,702	-	-	Urban	Std. Urban
NREC	23,950	Stereo	-	Agricultural	Std. Agricultural
OPEDD	**1,018**	**Stereo**	✓	**Multiple Off-Road**	**Wide Range, Unusual**

Table 1: Comparison of contents of datasets for pedestrian detection.

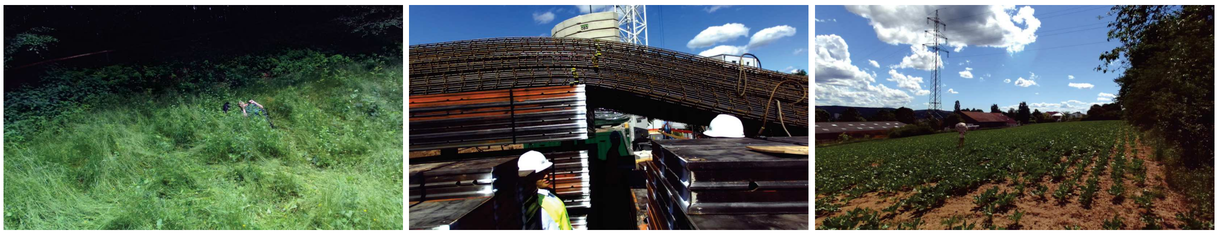


Figure 2: Our dataset shows different types of occlusion in varying environments, including naturally occurring obstacles (left: vegetation, center: construction materials) and unusual objects (right: umbrella).

3 CHARACTERISTICS OF OFF-ROAD ENVIRONMENTS

Off-road, agricultural or rural environments show several characteristics that differentiate them from urban surroundings in a number of ways:

Visuals The largest differences are recognizable in the visual domain. In urban images, the background is mostly characterized by buildings and paved roads, yielding a colour spectrum dominated by greys. In contrast, off-road environments can depict a multitude of backgrounds. Agricultural and wooded surroundings usually show ample vegetation with a colour spectrum controlled by greens and browns, while construction sites display a mix of urban and non-urban components. In terms of texture, backgrounds dominated by vegetation show heavy textural repetition. Moreover, Tabor et al. [13] have shown that the gradient orientation alignment is very distinct between the different types of environments.

Composition In urban settings, pedestrians are one visually distinct object class out of many, including cars, cyclists, trucks and many more. In off-road environments, pedestrians tend to appear as much more strongly separated objects.

Occlusion In surroundings dominated by vegetation partial occlusion of persons by leaves, grass or branches is very common. Examples are people harvesting fruit in orchards or a person standing in field crops, having parts of the lower body obstructed. Additionally, the boundary of occlusions is often much fuzzier than in the case of occlusions by e.g. cars in the urban setting.

Poses Due to the nature of city scenes, datasets for pedestrian detection in urban environments show persons predominantly standing or walking upright. Additionally, because the data is usually captured from a vehicle driving on the road, most pedestrians are located on the lateral edges of the image, with persons only directly in front of the camera if the data-capturing-vehicle is positioned in front of a cross- or sidewalk. Contrary to that, many agricultural or industrial scenes show persons in unusual and more challenging poses: Often the person is seen working in a crouching or bent position and limbs extended in differing ways are common. Due to the hazardous environment on construction sites, the vehicle could encounter people lying on the ground. In general, off-road scenes display a much larger variety of poses than the average urban scene. Many of these difficulties are addressed in our dataset, described in the following chapter.

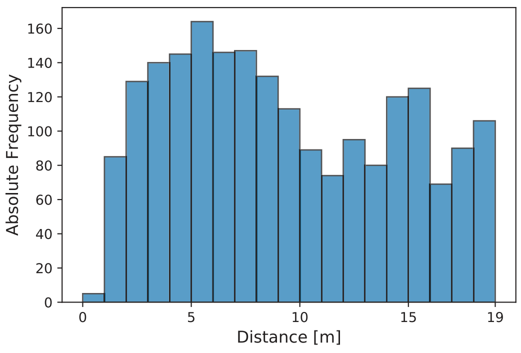


Figure 4: Histogram of distances of the portrayed pedestrians to the camera.

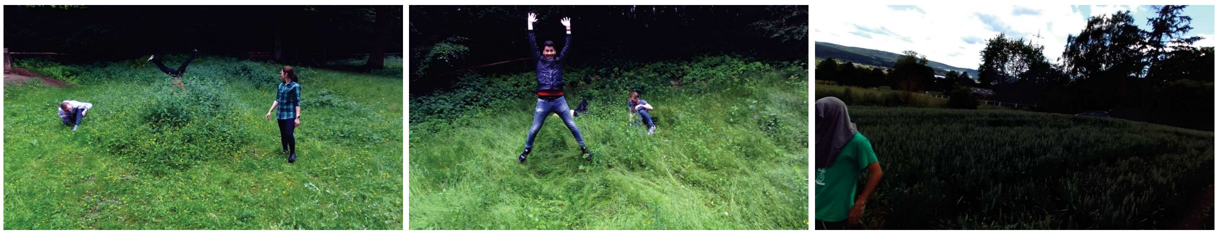


Figure 3: Special attention was paid to capture a wide range of poses not usually encountered in urban driving datasets. Left: Handstand. Center: Jumping with extended limbs. Right: Head covered with clothes.

Task	Implementation	Trained On	AP50	AP75	AP
Instance Segmentation	Mask R-CNN	COCO	0.6831	0.4355	0.3935
Instance Segmentation	Mask R-CNN	COCO + Ours	0.80031	0.4880	0.4500
Object Detection	YOLOv3	COCO	0.5666	0.3966	0.3437

Table 2: Test-set results on object detection and instance segmentation tasks with Mask R-CNN and YOLOv3.

4 DATASET

4.1 Data Capturing

We record all sequences of our dataset using a Stereolabs ZED Camera [14]. The stereo camera has a baseline of 120mm and is able to capture video sequences with a side-by-side output resolution of 4416x1242 pixels at 15 frames per second. In order to prevent compression artifacts, which can impair detection performance [15], the video sequences are captured with lossless compression.

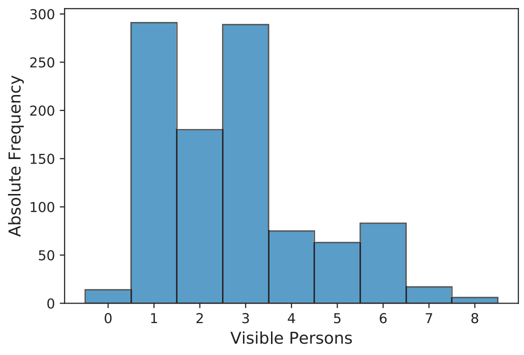


Figure 6: Histogram depicting how many pedestrians are visible in the images.

Besides rectifying the images, the ZED camera also outputs depth maps from stereo for every frame. Data was captured in short video sequences of 1 to 50 seconds with a framerate of 15 Hz.

Environments We capture data in different locations to cover a broad range of possible ADAS application scenarios: *Meadows, woods, construction sites, farmland and paddocks*.

While capturing, emphasis was laid on covering many scenarios that complicate pedestrian detection in off-road environments.

Occlusions In all of our environments, occlusion happens with locally characteristical obstacles like grass, leaves, field crops, construction materials or fences, as well as more unusual barriers like stone walls, garbage bins or objects held by persons (umbrellas, paper files). Examples can be found in Fig. 2. Moreover, we took care to include many instances of person-to-person occlusion, oftentimes by a pedestrian standing close to the camera.

Poses Our dataset shows a variety of uncommon and challenging poses including people doing handstands, lying on the ground or on objects, lying on the back or on the side, sitting, crouching or bent over, limbs extended as well as running and jumping.

Composition Special attention was paid to have multiple positions in the image covered by pedestrians, to avoid the urban situation where persons are located mainly at the sides. Additionally we vary the number of persons, see Fig. 6, and the distances they appear to the camera (Fig. 4). Most images are taken from eye-level up to 1m above, facing forward, to simulate taller vehicles like tractors or excavators, with images showing a more downward facing angle.

Image Resolution	2208 x 1242
Stereo Camera Baseline	120mm
Number of video sequences	1004
Video Framerate	15 Hz
Compression	Lossless
Number of Images	1018
Total pedestrian instances	2801

Table 3: Capturing and dataset statistics.

Lighting The light conditions vary naturally as well as intentionally, with some images being taken against direct sunlight or with people being hidden in the shadows of walls or trees.

Miscellaneous Further variations include clothing, helmets or gimmicks like clothes being thrown around or people deliberately hiding.

Image Selection From the video sequences 1018 image pairs are selected. Since the images are often

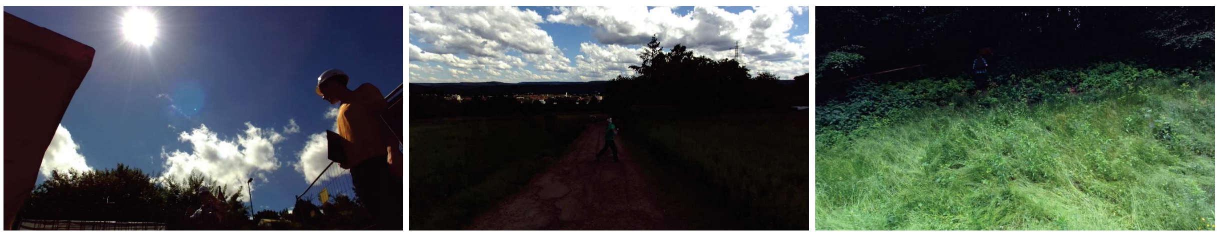


Figure 5: Samples of images with difficult lighting conditions.

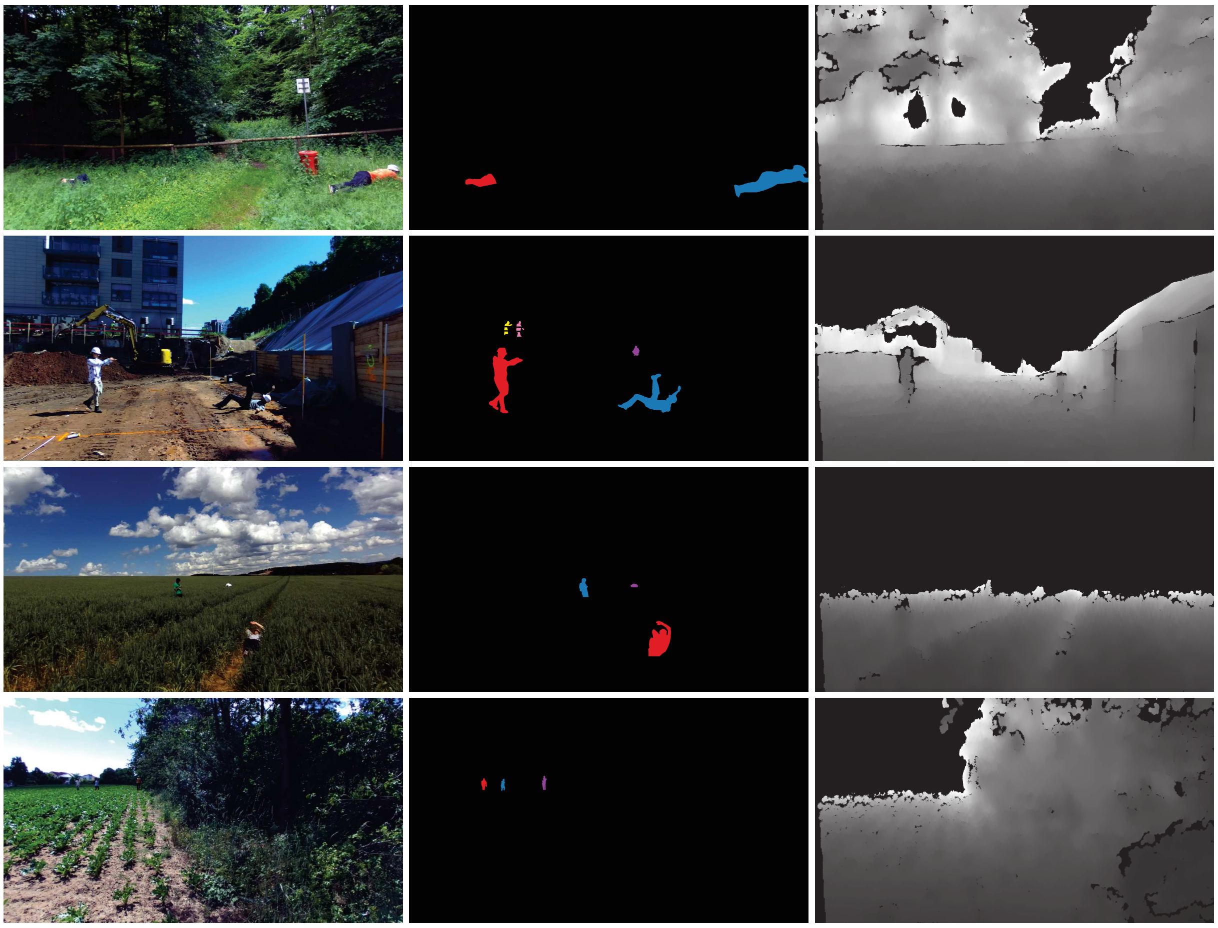


Figure 7: Selection of environments contained in the presented dataset. Left column: Left frame of stereo image. Center column: Corresponding segmentation masks. Right Column: Corresponding depth maps. Our dataset shows a variety of environments and poses.

almost identical from one frame to the next, we make sure to choose the next frame in such a way that sufficient alteration is visible, often by clear repositioning of persons or after a pan of the camera. The images that make it to the final dataset are selected by hand.

Annotation The ground truth annotations of the images were created using the VGG Image Annotation Tool (VIA) [16], [17]. All visible persons were annotated with a segmentation mask and given an (imagewise-) unique ID. Since drawn masks sometimes overlap, the IDs are assigned in increasing order with increasing depth of the person in the image. All labelling information is stored in a json file that can also be imported as a VIA project, allowing users to easily modify and expand on the annotations. The project files are available on the GitHub Repository provided at the bottom. Additionally, we supply scripts to extract segmentation masks and bounding boxes.

4.2 Related Detection Algorithms

Object Detection Object detection describes the task of extracting bounding box coordinates and dimensions of target objects in the image. We use YOLOv3 [2] trained on the COCO dataset [18] to make a first rough evaluation on our data. The CNN first predicts bounding box coordinates from anchors, regresses an objectness score and classifies the image patch.

Instance Segmentation In contrast to plain object detection, instance segmentation algorithms also output pixel-level segmentation masks for every detected bounding box. We apply Mask R-CNN [3] to our dataset, first as-supplied ([19]) trained on COCO, then fine-tuned on our training set. A region proposal network first predicts possible objects and their bounding boxes. Further branches then classify the object and output corresponding pixel masks. The results of our first evaluations can be taken from table 2. We compute the average precision (AP) similar to [20], where the number specifies the minimum intersection over union (IoU) for a predicted bounding box or segmentation mask to be assigned to a ground truth instance, e.g. AP50 meaning a minimum of 50% IoU. In addition, we average AP over multiple IoU thresholds from 0.5 to 0.95, simply denoted as AP, to avoid bias towards a specific value [9], [18].

5 CONCLUSION AND FUTURE WORK

In this paper, we have introduced a new Off-Road Pedestrian Detection Dataset (OPEDD). It consists of 1018 manually annotated images and displays persons in scenarios broadly characterized as meadows, woods, construction sites, farmland and paddocks. The people portrayed show a wide variety of poses usually not encountered in urban environments and corresponding datasets. In all settings, it supplies a variety of types of occlusions, compositions and lighting conditions. Ground truth annotations are available as pixel-level segmentation masks, with each person in an image having an individual ID, making it possible to use the dataset for object detection, semantic- and instance segmentation tasks. For future work, we aim to add additional annotations to our currently unlabelled sequences. Furthermore, instead of labelling unique images in the captured data, complete sequences could be labelled for tasks like multiple object tracking.

6 ACKNOWLEDGMENTS

We thank Maximilian Palm for aiding us in the capturing of sequences and selection of images.

GitHub https://github.com/PNeigel/OPEDD

REFERENCES

[1] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single Shot Multi-Box Detector,” *ECCV*, pp. 21–37, 2016.

[2] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *ArXiv*, vol. abs/1804.02767, 2018.

[3] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask R-CNN,” *ICCV*, pp. 2980–2988, 2017.

[4] W. Liu, S. Liao, W. Hu, X. Liang, and X. Chen, “Learning Efficient Single-Stage Pedestrian Detectors by Asymptotic Localization Fitting,” *ECCV, Proceedings*, pp. 618–634, 2018.

[5] Y. Pang, J. Xie, M. H. Khan, R. M. Anwer, F. S. Khan, and L. Shao, “Mask-Guided Attention Network for Occluded Pedestrian Detection,” *ICCV*, pp. 4967–4975, 2019.

[6] J. Zhang, L. Lin, Y.-C. Chen, Y. Hu, S. C. H. Hoi, and J. Zhu, “CSID: Center, Scale, Identity and Density-Aware Pedestrian Detection in a Crowd,” *CoRR*, 2019.

[7] A. Halevy, P. Norvig, and F. Pereira, “The unreasonable effectiveness of data,” *Intelligent Systems. IEEE*, 2009.

[8] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan, “Do We Need More Training Data?” *IJCV*, pp. 1–17, 2015.

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” *CVPR, Proceedings*, pp. 3213–3223, 2016.

[10] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite,” *CVPR, Proceedings*, pp. 3354–3361, 2012.

[11] P. Dollár, C. Wojek, B. Schiele, and P. Perona, “Pedestrian Detection: An Evaluation of the State of the Art,” *PAMI*, vol. 34, 2012.

[12] Z. Pezzementi, T. Tabor, P. Hu, J. K. Chang, D. Ramanan, C. Wellington, B. P. Wisely Babu, and H. Herman, “Comparing apples and oranges: Off-road pedestrian detection on the National Robotics Engineering Center agricultural person-detection dataset,” *Journal of Field Robotics*, vol. 35, no. 4, pp. 545–563, 2018.

[13] T. Tabor, Z. Pezzementi, C. Vallespi, and C. Wellington, “People in the weeds: Pedestrian detection goes off-road,” in *2015 IEEE SSRR*, 2015, pp. 1–7.

[14] *Https://www.stereolabs.com/zed/*, accessed May 4, 2020.

[15] M. Dejean-Servières, K. Desnos, K. Abdelouahab, W. Hamidouche, and L. Morin, “Study of the Impact of Standard Image Compression Techniques on Performance of Image Classification with a Convolutional Neural Network,” *INSA Rennes; Univ Rennes;IETR; Institut Pascal*, 2017.

[16] A. Dutta, A. Gupta, and A. Zissermann, *VGG Image Annotator (VIA)*, http://www.robots.ox.ac.uk/˜vgg/software/via/, 2016, accessed May 4, 2020.

[17] A. Dutta and A. Zisserman, “The VIA Annotation Software for Images, Audio and Video,” in *27th ACM Multimedia, Proceedings*, ser. MM ’19, New York, NY, USA: ACM, 2019.

[18] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” in *ECCV*, 2014.

[19] W. Abdulla, *Matterport Mask R-CNN*, https://github.com/matterport/Mask_RCNN, 2017, accessed May 4, 2020.

[20] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, “Simultaneous detection and segmentation,” in *ECCV*, 2014.