ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

CSRN 2801

(Eds.)

- **Vaclav Skala**
  University of West Bohemia, Czech Republic

Computer Science Research Notes

26. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision
WSCG 2018
Plzen, Czech Republic
May 28 – June 1, 2018

Proceedings

WSCG 2018

Full Papers Proceedings

ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

CSRN 2801

(Eds.)

- **Vaclav Skala**
  University of West Bohemia, Czech Republic

Computer Science Research Notes

26. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision
WSCG 2018
Plzen, Czech Republic
May 28 – June 1, 2018

Proceedings

WSCG 2018

Full Papers Proceedings

Vaclav Skala - UNION Agency

ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

Computer Science Research Notes
CSRN 2801

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia
Univerzitni 8
CZ 306 14 Plzen
Czech Republic
skala@kiv.zcu.cz http://www.VaclavSkala.eu

Managing Editor: Vaclav Skala

Publisher & Author Service Department & Distribution:
Vaclav Skala - UNION Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic
Reg.No. (ICO) 416 82 459

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

ISSN 2464-4617 (Print) ***ISSN 2464-4625*** (CD/DVD)
ISBN 978-80-86943-40-4 (CD/-ROM)

WSCG 2018

International Program Committee

Benes,B. (United States)
Benger,W. (United States)
Bouatouch,K. (France)
Bourke,P. (Australia)
Daniel,M. (France)
de Geus,K. (Brazil)
Dingliana,J. (Ireland)
Durikovic,R. (Slovakia)
Feito,F. (Spain)
Feng,J. (China)
Ferguson,S. (United Kingdom)
Galo,M. (Brazil)
Garcia Hernandez,R. (Germany)
Gavrilova,M. (Canada)
Giannini,F. (Italy)
Gudukbay,U. (Turkey)
Juan,M. (Spain)
Klosowski,J. (United States)
Lobachev,O. (Germany)
Molla,R. (Spain)
Montrucchio,B. (Italy)
Muller,H. (Germany)
Patow,G. (Spain)
Pedrini,H. (Brazil)
Renaud,C. (France)
Richardson,J. (United States)
Rojas-Sola,J. (Spain)
Sanna,A. (Italy)
Santos,L. (Portugal)
Segura,R. (Spain)
Skala,V. (Czech Republic)
Sousa,A. (Portugal)
Szecsi,L. (Hungary)
Teschner,M. (Germany)
Thalmann,D. (Switzerland)
Trapp,M. (Germany)
Wuensche,B. (New Zealand)
Wuethrich,C. (Germany)
Xu,K. (China)
Yin,Y. (United States)

WSCG 2018

Board of Reviewers

Aburumman,N. (France)
Assarsson,U. (Sweden)
Ayala,D. (Spain)
Azari,B. (Germany)
Benes,B. (United States)
Benger,W. (United States)
Bouatouch,K. (France)
Bourke,P. (Australia)
Carmo,M. (Portugal)
Carvalho,M. (Brazil)
Daniel,M. (France)
de Geus,K. (Brazil)
De Martino,J. (Brazil)
de Souza Paiva,J. (Brazil)
Dingliana,J. (Ireland)
Durikovic,R. (Slovakia)
Feito,F. (Spain)
Feng,J. (China)
Ferguson,S. (United Kingdom)
Galo,M. (Brazil)
Galo,M. (Brazil)
Garcia Hernandez,R. (Germany)
Garcia-Alonso,A. (Spain)
Gavrilova,M. (Canada)
Gdawiec,K. (Poland)
Giannini,F. (Italy)
Goncalves,A. (Portugal)
Gudukbay,U. (Turkey)
Hernandez,B. (United States)
Horain,P. (France)
Charalambous,P. (Cyprus)
Juan,M. (Spain)
Kanai,T. (Japan)
Klosowski,J. (United States)
Kurt,M. (Turkey)
Lee,J. (United States)
Lisowska,A. (Poland)
Lobachev,O. (Germany)
Luo,S. (Ireland)
Marques,R. (Spain)
MASTMEYER,A. (Germany)
Metodiev,N. (United States)
Molla,R. (Spain)
Montrucchio,B. (Italy)
Muller,H. (Germany)
Oliveira,J. (Portugal)
Oyarzun Laura,C. (Germany)
Papaioannou,G. (Greece)
Patow,G. (Spain)
Pedrini,H. (Brazil)
Peytavie,A. (France)
Puig,A. (Spain)
Ramires Fernandes,A. (Portugal)
Renaud,c. (France)
Ribeiro,R. (Portugal)
Richardson,J. (United States)

Rodrigues,J. (Portugal)
Rojas-Sola,J. (Spain)
Sanna,A. (Italy)
Santos,L. (Portugal)
Segura,R. (Spain)
Skala,V. (Czech Republic)
Sousa,A. (Portugal)
Subsol,G. (France)
Szecsi,L. (Hungary)
Tavares,J. (Portugal)
Teschner,M. (Germany)
Thalmann,D. (Switzerland)
Todt,E. (Brazil)
Tokuta,A. (United States)
Trapp,M. (Germany)
Vanderhaeghe,D. (France)
Vidal,V. (France)
Vierjahn,T. (Germany)
Wuensche,B. (New Zealand)
Wuethrich,C. (Germany)
Xu,K. (China)
Yin,Y. (United States)
Yoshizawa,S. (Japan)
Zwettler,G. (Austria)

WSCG 2018

Full Papers Proceedings

CSRN 2801

Contents

Keynote - Abstract

Faramarz Samavati: From Geometric Modeling to Digital Earth
University of Calgary, Canada

Domaradzki,J., Martyn,T.: Procedural Fracture of Shell Objects 1

Getto,R., Fina,K., Jarms,L., Kuijper,A., Fellner,D.: 3D Object Classification and Parameter Estimation based on Parametric Procedural Models 10

Molchanov,V., Linsen,L.: Generation of Implicit Flow Representations for Interactive Visual Exploration of Flow Fields 20

Chakib,R., Merillou,N., Vincent,P.-J., Merillou,S.: Calibrating Low-cost Structured-light 3D Sensors 30

Azari,B., Bertel,S., Wüthrich,C.A.: Assessing Objective Image Quality Metrics for Bidirectional Texture Functions 39

Morlet,L., Neveu,M., Lanquentin,S., Gentil,Ch.: Barycentric Combinations Based Subdivision Shaders 49

Vlasanek,P.: Fuzzy image inpainting aimed to medical images 59

Tripathi,G., Etemad,K., Samavati,F.: Single image summary of time-varying Earth-features 68

Krämer,M.S., Kuhnert,L.,Kuhnert,K.-D.: Realtime Visual Off-Road Path Detection 78

Papachristou,A., Zioulis,N., Zarpalas,D., Daras,P.: Markerless Structure-based Multi-sensor Calibration for Free Viewpoint Video Capture 88

From Geometric Modeling to Digital Earth

Faramarz Samavati
http://pages.cpsc.ucalgary.ca/~samavati/
Computer Science Dept.
University of Calgary
Canada

Abstract
The Earth is immense, and abundant with interesting information. Recent advancements in geospatial sensors have resulted in the development of technologies capable of collecting large and dynamic geospatial data. As a result, we are experiencing an explosion in the volume and variability of geospatial data, and yet many remain unaware of or unable to access this wealth of data to make informed decisions regarding our planet. One viable solution to this challenge is to use a digital model of the Earth (a Digital Earth) as a place holder for integrating all sorts of geospatial datasets. The Digital Earth is a vision for Earth-based applications in which geospatial data may be assigned and retrieved using the 3D Earth as a reference model, rather than a 2D map. In this talk, an overview is provided of research projects and recent achievements from my group, related to Digital Earth. In these projects, we explore problems of creating and managing grid systems, large geospatial data processing and streaming, as well as creative visualization and interaction in Digital Earth.

Short Bio
Faramarz F. Samavati is a Professor of Computer Science at the University of Calgary. He is currently one of the Associate Heads (Graduate Director) of the Department of Computer Science. Faramarz's research interests include Computer Graphics, Visualization, 3D Imaging and Geometric Modeling. Dr. Samavati has published more than hundred and twenty peer reviewed papers, one book, and two patents. In the past seven years, he has received seven best paper awards, Digital Alberta Award, Great Supervisor Award, and University of Calgary Peak Award, which honors his contribution to the development of new technologies and innovations. Faramarz was one of the ASTech 2017 Nominees & Finalists for the award on Outstanding Leadership in Alberta Technology.

ISSN 2464-4617(print)
ISSN 2464-4625(CD)
Computer Science Research Notes
CSRN 2801
Full Papers Proceedings
http://www.WSCG.eu

Procedural Fracture of Shell Objects

Jakub Domaradzki
Institute of Computer Science
Warsaw University of Technology
ul. Nowowiejska 15/19
00-665 Warsaw, Poland
J.Domaradzki@stud.elka.pw.edu.pl

Tomasz Martyn
Institute of Computer Science
Warsaw University of Technology
ul. Nowowiejska 15/19
00-665 Warsaw, Poland
martyn@ii.pw.edu.pl

ABSTRACT

We propose a novel algorithm to fracture brittle objects that are characterized by an empty interior and thick surface (which we denote as *shell objects*), such as: vases, pots, pitchers, antique ceramic, etc. Our method augments the previous ones based on fracture patterns and utilizes sparse voxel octrees (SVOs) as a highly efficient and detailed object representation. In our method, the fracture pattern relies on Voronoi diagrams and is calculated on-the-fly. The outcomes of applying the fracture pattern differ from the ones obtained with the previous methods in that it solves the problem of planar faces of the newly generated pieces of geometry, allowing them to have concave shapes. Without any precomputation, we are able to achieve various and interesting fractures that are unique to each destructed object. Finally, our approach is intuitive, adaptable and fast, which makes it a good candidate for applications in computer game industry.

Keywords

sparse voxel octree, Voronoi decomposition, pattern fracturing, procedural surface generation

1 INTRODUCTION

Nowadays various effects of object fracturing are widely present in computer games. Wandering through the game worlds we can usually interact with lots of environment items and in order to increase the realism of "being-in-the-world", we are more often allowed to destruct them. Nevertheless, the players' requirements and expectations pertaining the nowadays gameplays in general and in particular freedom in the player's interactions with the virtual world are continuously rising. Needless to say that the objects populating the game level are usually required not only to possess a capability of being fractured, splintered, crushed, or destroyed in some other way, but also the process of breaking them into pieces should be *unique* for each instance of a destructible geometrical asset and for the same instance—with every restart of the game.

Therefore the approaches based on a predefined destruction of geometrical assets and successfully utilized in games of the previous generations are now slowly being replaced with more and more sophisticated techniques, in which some of the calculations are performed on-the-fly during the gameplay itself. Due to the constant increase of computer power, especially graphics cards, we are now very often able to perform all calculations underlying the object destruction in real-time. On the other hand, the amount of details of the geometry that constitutes the scenes in modern games is also still increased, which makes the task of performing such a unique destruction of objects directly during gameplay even more challenging.

Thanks to the development of new voxel-based techniques in a few recent years, it is now possible to utilize voxels as a representation of solid objects in real-time systems. Memory usage, one of the main problems associated with voxels, has been drastically reduced mainly due to taking advantage of the sparse voxel octree structure (SVO) [Cra11]. This way a genuine, highly detailed geometry with many levels of detail can be easily accessed.

In this paper, we show how to enrich with details and make unique the process of destruction when applied to shell objects, which are featured by an empty interior and a thick surface like vases, pots, pitchers, antique ceramic, etc. Our approach augments the previous ones based on fracture patterns and benefits from the SVO representation. The outcomes of our method stand out from those obtained with the related solutions in that the pieces the destroyed object is falling apart to can be concave in shape and, moreover, the geometry of the pieces' surface is not limited to planar facets. And what is most important, all that we achieve in time sufficient for computer game applications.

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.1

2 RELATED WORK

There is a wide selection of literature on fracturing brittle objects. Over the years, many methods have been developed. The early works in the animation of fracture used connected point-masses and the fracturing process was performed by the removal of some of the links. This method was introduced by Terzopoulos et al. [TF88] and Norton et al.[NTB*91], who pioneered this area of study in computer graphics. Afterwards, the approach was improved by O'Brien and Hodgins, who followed them and focused on simulating brittle [OH99] and ductile fracturing [OBH02]. They used a fully dynamic finite element method (FEM), in order to compute the internal stress in the object being destructed. However, this approach requires very small time steps and costly cutting mesh operations [WRK*10]. To avoid this issues, the fracture simulation tends to be formulated as a quasi-static stress analysis [GMD13][ZBG15].

On the other hand, the real-time systems, especially computer games, utilize a more practical approach which is based on a predefined partition of the destructible object into pieces, which replace the original object while it is destroyed. While this method is fast and gives the designer a full control over the shape and location of fractures, its results are often a far cry from realism. In order to improve the visual quality of this approach and, at the same time, to stay within the limits imposed by real-time applications, the geometry-based methods were introduced [SSF09][BCC*11][MCK13]. Although the methods also utilize a predefined fracture pattern, the pattern is separated from the actual geometry and is applied, oriented and scaled on demand at the impact location. That way the observer gets the impression that the fracturing is unique for each destruction. The common technique to generate the fracture patterns consists in constructing a 3D space decomposition based on the Voronoi diagram or by means of a simulation [IO09]. The further development in this direction resulted in generating the fracture pattern on-the-fly while a destructive impulse occurs [SO14][DM16].

Although the current pattern-based approaches usually produce good results, there is still room for improvement. From our standpoint, there are two essential problems, which we regard as challenges and would like to address, namely: the non-planar surfaces of the fractured pieces and their concave shape. One should note that both problems have already been referenced in [SO14], however, the solution presented there requires a pre-computation step and the calculations are performed on a rather fine tetrahedral mesh. The method we present in this paper follows Domaradzki's work et al. [DM16] and takes a different approach to tackle the mentioned issues. Looking for a predecessor of the general conception our method is founded on, one can point out a paper by Chen [CYFW14], in which visual details were added to coarse simulation results in a post-process step.

What is more, we strongly believe that the solution to the addressed problems can be found, quite naturally, in the voxel-based representation. Currently, the popular data structure for voxels to represent boundaries of solid objects is the sparse voxel octree (SVO) [Cra11]). Thanks to the computational power offered by today's GPUs, along with new GPU-specialized programming techniques newly developed algorithms designed for sparse voxel octrees are ready for real-time applications.

Although the SVO representation is not yet well established in the commercial real-time graphics software, such as professional game engine being still dominated by triangle meshes, it has already proven its usefulness in a number of fields. To begin with, Cyril Crassin was able to visualize global illumination in real time using voxel cone tracing [CNS*11]. Following this work, Laine developed an efficient SVO ray-tracing algorithm [LK10], proving that way that ray-casting the SVO can be done faster than when using triangle meshes. Furthermore, the search for even more compact scene representations led to the development of sparse voxel directed acyclic graphs (SVDAG [KSA13]), which have been then improved to Symmetry-Aware SVDAGs [VMG16], that reduce memory footprint even further. There are also other ways to represent the SVO-based geometry effectively, to mention only a method of unlimited object instancing presented by Jabłoński et al. [JM17], that can be combined with a continuous and symmetrical LOD transition [JM16].

Apart from visualization itself, there has also been a development in other graphics areas including object animation [Bau11], deformation [Wil13], and fracturing in real-time [DM16]. Last, but not least, octrees can be built very fast using some recent techniques presented in [ZGHG11][GPM11][Kar12] and even by means of the out-of-core approaches described in [BLD14][PK15], which utilize only a fraction of memory required to store a model.

3 SVO FRACTURING

In this section, we outline our algorithm for fracturing SVO objects. We especially target shell objects that are empty inside and are formed by thick surfaces, such as vases, pitchers, pots, antique ceramics, etc. Our goal is to cut them into pieces with a slightly and locally disturbed fracture pattern that is calculated on-the-fly. The algorithm consists of two separate parts that combined together produce a final result. The first part, we call *Basic Fracture Algorithm* (or BFA—Sec. 4), is based

 ISBN 978-80-86943-40-4

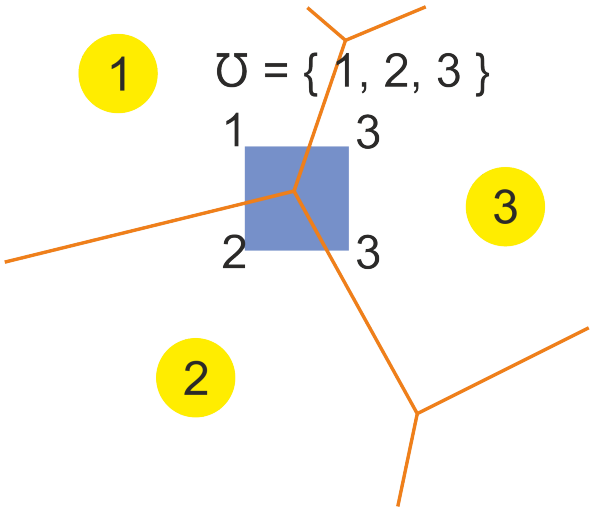


Figure 1: A voxel-pattern intersection test with three Voronoi seeds.

on the previous work of Domaradzki et al. [DM16]. Its underlying idea is to divide the SVO object into convex pieces by means of a fracture pattern consisting of planar faces, which determine the slicing areas used to partition the object. The outcome is then processed in the second part, we call *Enhanced Fracture Algorithm* (or EFA—Sec. 5), in which the cuts are additionally deformed by means of a local procedural surface creation. As a consequence, we enhance the previous Domaradzki's algorithm in that the final cuts are less regular and more intricate geometrically and, thus, they render a more natural fracture.

4 BASIC FRACTURE ALGORITHM

The goal of BFA is to determine the subsets of SVO voxels that represent the surfaces of the particular fractured pieces at the accuracy of the SVO highest level. To this end, the SVO is traversed from the root to the leaves, and at each SVO level, the intersections of voxels with the pattern faces are tested. Next, if a face-voxel intersection is detected, the voxel is either expanded to its eight children or, in the case the voxel belongs to the object's interior (i.e. it is not an element of the surface), the children are dynamically created. The resulting subsets are composed of the voxels intersected by the fracture pattern faces at the highest SVO level and, of course, the original leaf voxels of the SVO object surface. A voxel is assigned to a given subset on the basis of the closest Voronoi seed to the voxel's location (Sec. 4.1). Then, for each subset, we build a SVO founded on the subset's voxels treated as the SVO leaves. A more detailed description of the algorithm can be found in [DM16].

4.1 Fracture Pattern

The fracture pattern used in this algorithm is represented by a finite set of 3D points, which represent the seeds of a Voronoi diagram. In order to achieve a more realistic fracture, we want it to concentrate around the impact location. This can be obtained by aligning the center of an existing fracture pattern with the impact location as presented in [SSF09] and [MCK13]. However, following work in [DM16], we also create the fracture pattern on-the-fly. In this goal, we generate the Voronoi seeds at random on a set of spheres of growing radii, which are centered at the point of impact. Having this set of seeds it is not required to determine the faces of the Voronoi diagram for the voxel-pattern intersection test, which can be performed directly based on the set as follows:

Let $S = \{1,\dots,n\}$ be the set of the indices of the Voronoi seeds $\{s_i\}_{i\in S}$. Define a function $\gamma : \mathbb{R}^3 \to 2^S$ such that

$$\gamma(x) = \{k \in S : \|s_k - x\| = \min_{i\in S} \|s_i - x\|\}. \qquad (1)$$

Given a point $x \in \mathbb{R}^3$, the function γ returns the set of the indices of the Voronoi cells that include x. (Note that the resulting set is not a singleton, if x is located on a face, an edge, or a vertex of the Voronoi diagram).

The voxel-pattern intersection test can be done by means of the following function:

$$\Ʊ(V) = \bigcup_{v\in V} \gamma(v). \qquad (2)$$

which, given a voxel specified by the set $V = \{v_i\}_{i=1,\dots,8}$ of its vertices, maps the voxel into the set of the indices of the Voronoi cells the vertices are situated in.

It is easy to see that the voxel is intersected by a face of the fracture pattern if and only if the set of indices given by $\Ʊ(V)$ is not a singleton (fig. 1).

5 ENHANCED FRACTURE ALGORITHM

In the second step, we improve the result obtained by BFA to give the fracture a more realistic appearance. In this purpose, we must face the two main deficiencies of the previous algorithm: the lack of concave fractured pieces and their totally planar surfaces.

5.1 Distance Metric Approach

To begin with, there is a solution that might be used to produce outcomes satisfying our needs. However, it is expensive in memory and computation. The approach is based on influencing the distance metric that is used in the calculations in the voxel-pattern intersection test (sec. 4.1). The distance metric can be altered in two ways.

ISBN 978-80-86943-40-4

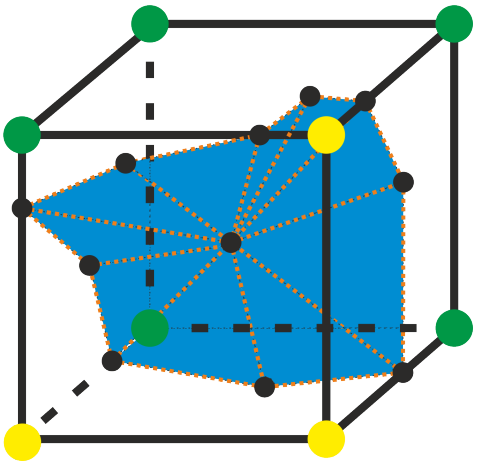


Figure 2: An example of a crack surface represented by a set of triangles, created within a voxel on the SVO transition level.

First, one can deform the fracture pattern as in [M05]. Following that, we would have to use a 3D noise texture and sample it along the ray from a voxel corner to a Voronoi seed, accumulating the result and treat it as a distance. Such an approach would take a lot of memory to store the texture and load operations to sample it with the 3D interpolation to receive a smooth outcome.

Another method, presented in [SO14], first applies a deformation to the object and then the fracturing process operates on the deformed object. Finally, the reverse deformation is applied to the result, producing deformed pieces of fractured geometry. Although this method is faster than the previous one, it encounters difficulties with the hierarchical representation of the destructed object. What is more, it would have to be performed on-the-fly, as building the new deformed SVO object comes with the loss of information (as presented in [Wil13]).

5.2 Procedural Crack Surface Creation

Taking into consideration challenges presented in the previous section, we propose another solution. Following BFA (sec. 4), we traverse the SVO from the root to the leaves. On each SVO level we get a set of voxels containing the information whether a voxel is intersected by a face of the fracture pattern and, if so, which Voronoi seed is the nearest to the voxel's vertices. As a result, if we stop BFA on any intermediate SVO level, then the outcome can be viewed as a partition of the SVO object with an approximation of the Voronoi diagram. Depending on the level we stop, the size of voxels differs and so the volumes of the pieces the voxels make up. The volumes may be treated as subspaces, within which *crack surfaces* can be procedurally generated. The crack surfaces will be used in the final step of the algorithm to enrich the original planar and convex fracture geometry provided by BFA. The SVO level at which we stop the algorithm and construct the crack surfaces we call the *transition level*.

There are two main problems that we need to address. First, how to generate the geometry of these crack surfaces with the voxels delivered by BFA. Secondly, we aim at a GPU-based parallel implementation that constructs the crack surfaces concurrently in the voxels. Therefore the algorithm is to operate locally within a voxel, and this implies the problem of connectivity of the surfaces between neighboring voxels.

We tackle the second problem by founding the crack surface construction on an information shared by vertices of neighboring voxels. As the aftermath of BFA, each vertex possesses the information about its nearest Voronoi seed, which we now additionally enrich with a procedurally generated value, for example, obtained from a 3D noise function.

The surface we want to create should separate the voxel's vertices that have assigned different Voronoi seeds and thereby belong to different fracture pieces of the objects. To this end, for each voxel's edge that connects vertices having different Voronoi seeds, the *edge separation point* is created. The position of this point is determined by adding to one of the edge's endpoints an offset computed with the following equation:

$$eo_n = f(a,b) * v_s, \tag{3}$$

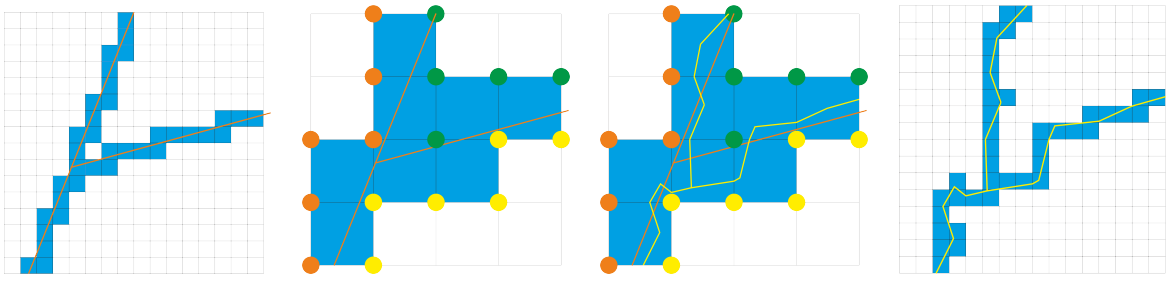
where: eo_n is the edge offset on $n \in \{X,Y,Z\}$ axis; $a,b \in [0,1]$—values assigned to the voxel's vertices; f is a user function returning a value in range (0, 1) based on the values a and b; and v_s is the size of the voxel edge. The user function can be implemented, for example, as a balance point based on the given weights or a deviation from the intersection point with the Voronoi pattern steered by the given weights.

Subsequently, for each voxel's face, the *face separation point* is calculated as the average of the *edge separation points* belonging to this face, and then the point is displaced by a slight offset within the face's area. Next, the *voxel separation point* is determined from *face separation points* in an analogous way. Finally, for each tuple of three points: *edge separation point*, *face separation point* and *voxel separation point* a triangle is created.

The set of the triangles forms a portion of the crack surface that will then be approximated by descendant voxels of the processed transition voxel (fig. 2).

The presented construction of continuous crack surfaces within voxels is only exemplary and one can develop different methods for this purpose. For example, one can base the construction of the surface on mathematical functions.

ISBN 978-80-86943-40-4



(a) (b) (c) (d)

Figure 3: (a) A result of BFA on the SVO highest level. (b) A result of BFA on the SVO transition level. (c) A new crack surface constructed by EFA relative to the BFA result. (d) The final voxelized crack obtained with EFA.

5.3 SVO Traversal

Putting all together, we can describe the overall method for fracturing a SVO object as follows:

We begin with BFA, which traverses the SVO tree from the root to the leaves: voxels on each SVO level are being tested for intersections with faces of the fracture pattern (sec. 4), and the ones that pass the test, are expanded to their children (which are generated on-the-fly in the case of the intersected voxels located inside of the object). This way we proceed over a chosen number of the SVO levels to the desired transition level. At this level, we assign the nearest Voronoi seeds to the voxels' vertices, and EFA begins.

From now on, the final surface of fractured pieces is being created using the crack surfaces. We generate the crack surfaces within the voxels at the transition level in the way described in the section 5.2. Then, for every voxel that gave birth to a crack surface, each child of the voxel is tested against an intersection with the surface by means of the method described in [AA05]. That way we proceed to the SVO leaves, where each voxel is assigned to an appropriate group of voxels which defines a fractured piece, using the same rules as in BFA (fig. 3). The final voxels properties comes from a volumetric texture for color and normal vector taken from the triangles the voxel intersects.

In order to fully define a fractured piece, apart from the voxels intersected by the crack surfaces, we also need to identify the appropriate voxels that come from the original surface of the destructed object. In order to assign these voxels to the proper Voronoi seed (and thus the group specifying a fractured piece), while executing EFA we utilize the following test:

While checking a voxel for intersections with the triangles of a crack surface, we also check on which side of each triangle the voxel is located and accumulate this information. By the construction of the crack surface (Sec. 5.2), each triangle has a vertex located on an edge of a transition level voxel, such that the edge ends with the voxel's vertices assigned to two different Voronoi seeds. On that basis, we assign the appropriate Voronoi seed to each side of the triangle. Using this information, we can assign a surface voxel (not intersected by a crack surface) to the appropriate Voronoi seed on the basis of the voxel's location relative to the sides of the relevant crack surface's triangles—we choose the seed that dominates in the sides of the triangles, as shown in fig. 4.

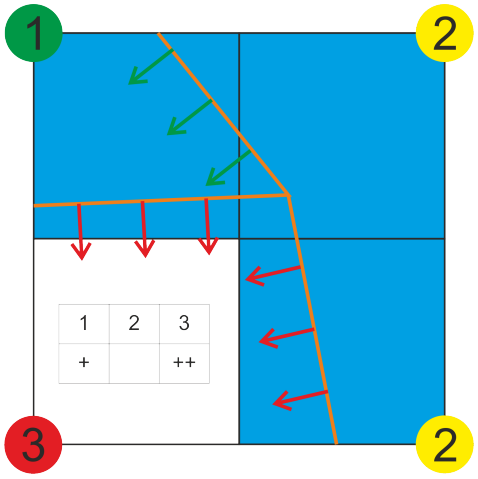


Figure 4: The assignment test for a voxel (marked with white) that doesn't intersect a crack surface.

The method presented above features a lossless fracturing process. It means that the surfaces of cracks match perfectly each other. In the real world, however, there are also materials bearing different properties, which result in less stable cracks. During fracturing process, apart from a separation of an object into a number of pieces, a part of the object's volume is converted into separate tiny dust particles of an individual size much smaller than the volume of the SVO leaf voxel. The formation of these dust particles during fracturing process results in irregular empty volumes between crack surfaces (fig. 5). We can easily incorporate this effect into our method by changing the crack surface creation algorithm presented in Sec. 5.2 to a more suitable one. For this purpose, we utilize the surface creation technique used in the Marching Cubes algorithm [LC87]. Specifically, using the information of the assignment of the transition voxels' vertices to Voronoi seeds, we match the voxel to one of the cube configurations used

ISBN 978-80-86943-40-4

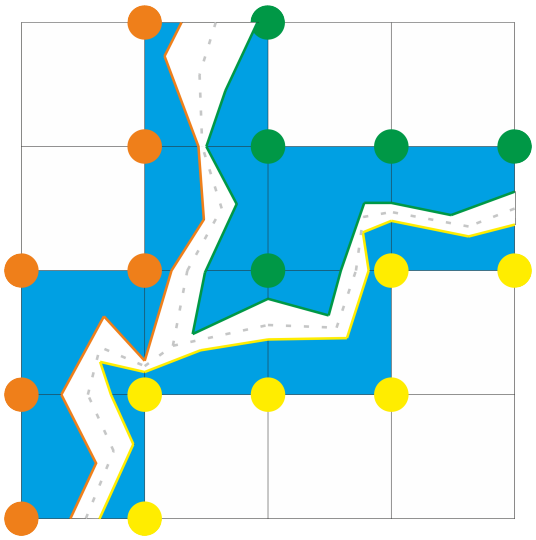


Figure 5: An empty volume between cracks' surfaces.

in the Marching Cubes algorithm. Moreover, we need to split the *edge separation points* and shift them in order to separate new surfaces.

6 RESULTS

In this section, we discuss the performance and quality of the presented solution. All depicted timings were obtained on Intel Core i7 960 CPU with Nvidia GeForce GTX Titan Black GPU. All algorithms were implemented using CUDA framework to fully exploit the parallelism delivered by GPU.

The presented results show that our new fracturing technique greatly improves on the visual quality of fractured objects relative to the relevant methods that utilize fracture patterns. A direct comparison with work in [DM16] is shown in fig. 6. One can notice that thanks to the distortions in the planar fracture pattern faces, the result looks more natural. What is more, the outcomes of our technique are always unique as they are created on-the-fly while a destruction occurs.

Furthermore, our method gives the user a simple way to influence the fracturing process outcome by controlling the level at which the transition between BFA and EFA takes place. Studying the pictures in fig. 7 reveals the relationship between this parameter value and the final result. The closer the transition level to the root, the more the basic fracture pattern (build from the Voronoi seeds) is distorted. The change in the transition level, when regarded from the point of view of the procedural creation of crack surfaces, also has an influence on the smoothness of the cracks.

We also presented a way to incorporate in the fracturing process a decline in the original object's volume within the area of cracks—fig. 9 exemplifies the feature. The variety of the cracks' possible shapes and widths we can obtain with our method greatly enhances the visual quality of results and allows one to produce countless of unique outcomes. In addition, the cracks are not only more detailed but also more noticeable.

We also succeeded in the integration of the results of our fracturing method with a physics simulation technique presented in the paper [DM16]. The performance of the approach presented there depends strongly on the volume of the objects subjected to simulation. Even though the fractured pieces generated with our method are usually relatively thin, the results of the physics simulation are acceptable (fig. 8).

Last but not least, the time performance of our method, as measured per voxel, is slower than in the one featuring the method in [DM16]. First, it is mainly due to that our technique produces more voxels in the final outcome and, secondly, the test for voxel intersection with a procedurally generated crack surface is more computationally demanding. It should be also noted that the performance of all the algorithms used in the fracturing process (fracture boundary set extraction, islands detection, internal nodes detection, etc.—details in [DM16])) depends directly on the total number of voxels, which also influence the timing of the whole process. However, in our tests, fracturing the SVOs with the number of levels up to 9 (included), resulted in the timings not exceeding 50 ms.

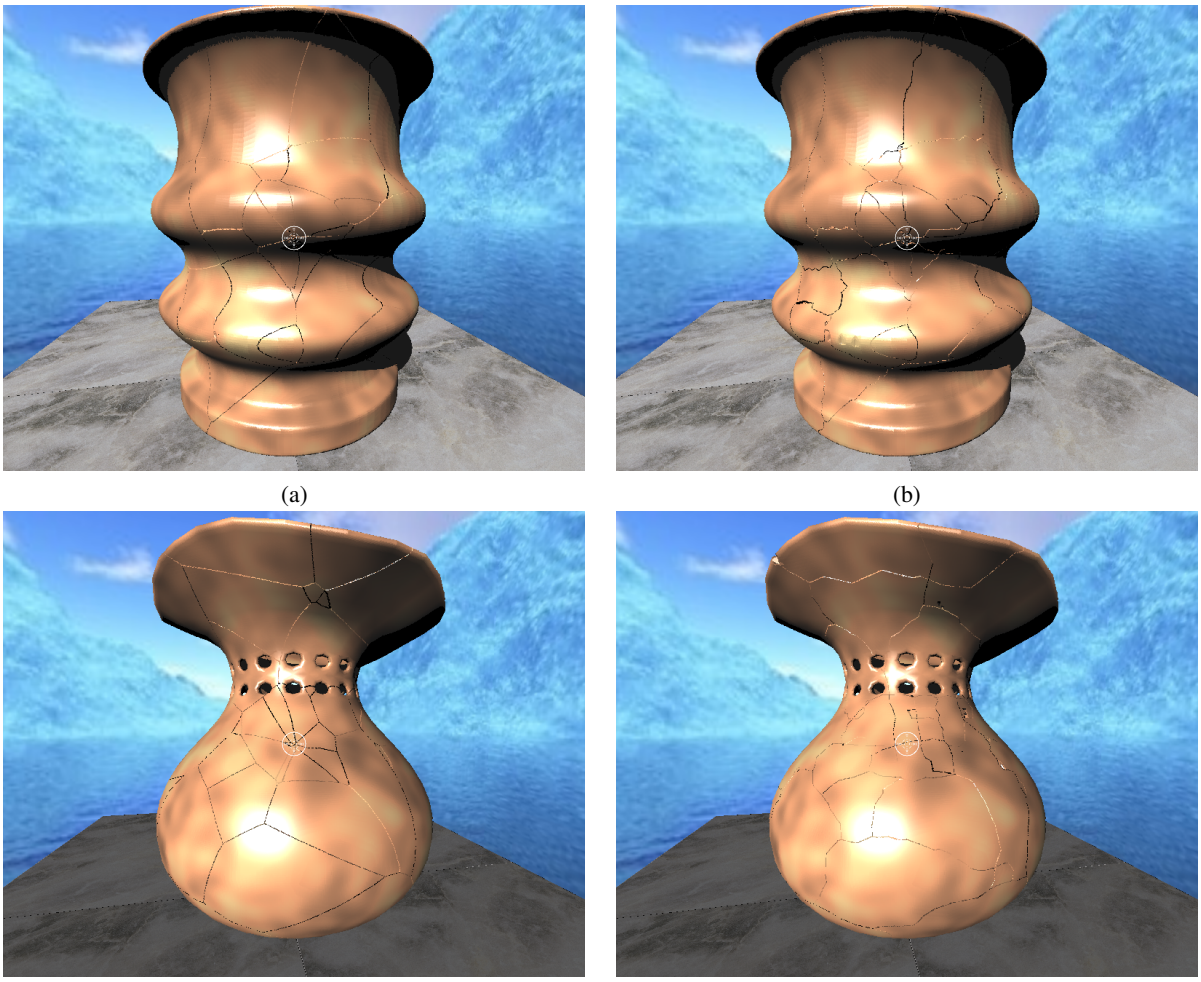
7 CONCLUSION AND FUTURE WORK

We have presented a novel method for fracturing shell objects represented with sparse voxel octrees. Our method allows for creating detailed surfaces of fractured pieces. To this end, it applies a fracture pattern to the object at the impact location and cuts the object with the pattern into pieces, and then enhances this partial result by creating the final surfaces of the pieces procedurally. As a consequence, new detailed pieces of geometry are created and represented as individual SVOs, which can then be subjected to a rigid body simulation using the approach presented in [DM16].

Although in this paper, we target only the objects that are empty inside but possesses thick surfaces, we strongly believe that our method could also be applied to objects with noticeable inner volumes. In that case, it would require either the change of the method used to procedurally create new surfaces or the application of more suitable weights for the current method for voxels' vertices on the transition level, so that the method would be aware of the voxels' boundaries in a local neighborhood. It is due to the fact that the current method generates cracks without any global structure, which is desirable for objects' surfaces but not necessarily for objects' interior.

Finally, the presented fracturing method operates on voxels, however, with some changes, it could also be

ISBN 978-80-86943-40-4



(a) (b)

(c) (d)

Figure 6: A comparison of results obtained by our fracturing technique (fig. b and d) and the method presented in [DM16] (fig. a and c).

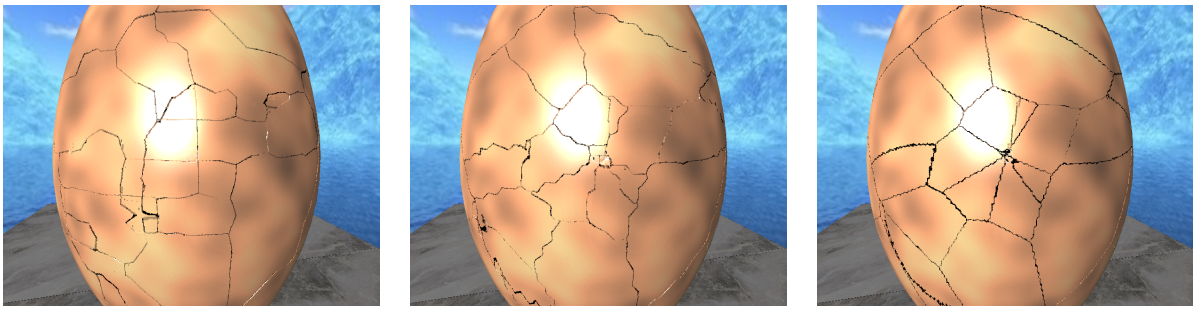


Figure 7: Different transition levels (from the left accordingly: 4, 6, 8) and the same fracture pattern.

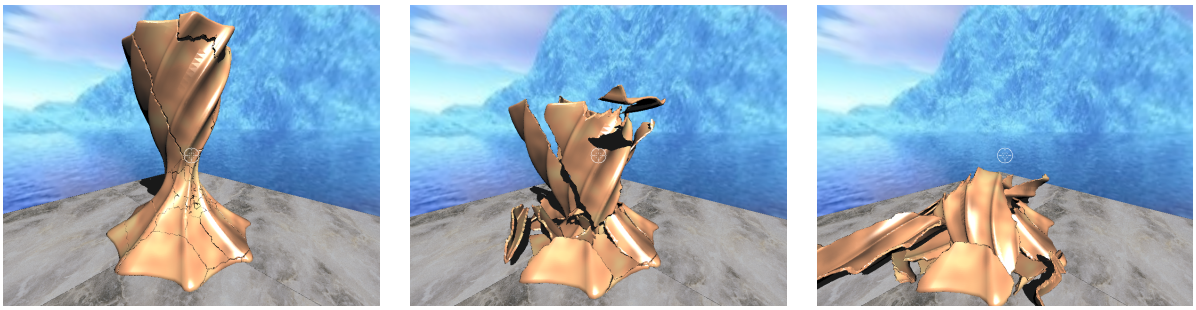


Figure 8: Fracturing and physics simulation of a vase.

ISBN 978-80-86943-40-4

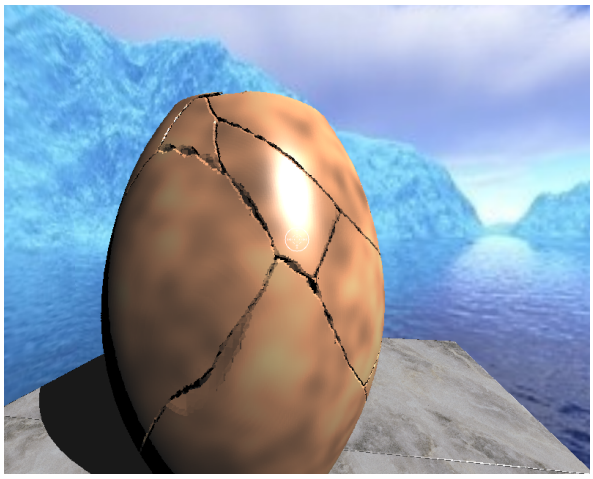


Figure 9: Cracks with irregular space between them.

adapted for objects represented with meshes. For this purpose, the first part of the algorithm could be performed on a tetrahedral mesh and within its cells, new surfaces would be created and cut against the destructed object.

8 REFERENCES

[AA05] Akenine-Möller, T., Aila, T.: Conservative and tiled rasterization using a modified triangle set-up. In Journal of Graphics Tools 10 (2005), 3, pp. 1-8.

[Bau11] Bautembach D.: Animated sparse voxel octrees. Bachelor's Thesis (feb 2011).

[BCC*11] Baker M., Carlson M., Coumans E., Criswell B., Harada T., Knight P., Zafar N. B.: Destruction and dynamic artist tools for film and game production. In ACM SIGGRAPH 2011 course notes (2011).

[BLD14] Baert J., Lagae A., Dutra' P.: Out-of-core construction of sparse voxel octrees. Computer Graphics Forum 33, 6 (2014), pp. 220-227.

[CNS*11] Crassin C., Neyret F., Sainz M., Green S., Eeisemann E.: Interactive indirect illumination using voxel cone tracing. Computer Graphics Forum (Proceedings of Pacific Graphics 2011) 30, 7 (sep 2011).

[Cra11] Crassin C.:. PhD thesis, Grenoble University, 2011.

[CYFW14] Chen Z., Yao M., Feng R., Wang H.: Physics-inspired adaptive fracture refinement. ACM Trans. Graph. 33, 4 (July 2014), 113:1-113:7.

[DM16] Domaradzki J., Martyn T.: Fracturing Sparse-Voxel-Octree objects using dynamical Voronoi patterns. In Computer Graphics, Visualization and Computer Vision WSCG 2016. Full Papers Proceedings, Computer Science Research Notes, vol. 2601, 2016, pp. 37-46.

[FBAF08] Faure F., Barbier S., Allard J., Falipou F.: Image-based Collision Detection and Response between Arbitrary Volume Objects. In Eurographics/SIGGRAPH Symposium on Computer Animation (2008).

[GPM11] Garanzha K., Pantaleoni J., Mcallister D.: Simpler and faster HLBVH with work queues. In Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, HPG '11, ACM, pp. 59-64.

[GMD13] Glondu, L., Marchal, M., Dumont, G.: Realtime simulation of brittle fracture using modal analysis. IEEE TVCG 19, 2013, pp. 201-209.

[IO09] Iben H. N.,O'Brien J. F.: Generating surface crack patterns. Graph. Models 71, 6 (Nov. 2009), 198-208.

[JM16] Jabłoński S., Martyn T.: Real-Time Rendering of Continuous Levels of Detail for Sparse Voxel Octrees. In Computer Graphics, Visualization and Computer Vision WSCG 2016. Short Papers Proceedings, Computer Science Research Notes, vol. 2602, 2016, pp. 79-88.

[JM17] Jabłoński S., Martyn T.: Unlimited Object Instancing in real-time. In Computer Graphics, Visualization and Computer Vision WSCG 2017. Short Papers Proceedings, Computer Science Research Notes, vol. 2702, 2017.

[KSA13] Kämpe, V., Sintorn, E., Assarsson, U.: High resolution sparse voxel DAGs. In ACM Trans. Graph. 32, 4, 2013, pp. 101:1-101:13.

[Kar12] Karras T.: Maximizing parallelism in the construction of BVHs, Octrees, and k-d Trees. In High Performance Graphics (2012), Eurographics Association, pp. 33-37.

[LC87] Lorensen W., Cline H.: Marching Cubes: A high resolution 3D surface construction algorithm. In Proceedings of the 14th annual conference on Computer graphics and interactive techniques (1987), vol. 21, pp. 163-169.

[LK10] Laine S., Karras T.: Efficient sparse voxel octrees. In Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, I3D '10, ACM, pp. 55-63.

[M05] Mould, D.: Image-guided fracture. In Proceedings of Graphics Interface 2005. Canadian Human-Computer Communications Society, 2005. p. 219-226.

[MCK13] Müller M., Chentanez N., Kim T.-Y.: Real time dynamic fracture with volumetric approximate convex decompositions. ACM Trans. Graph. 32, 4 (July 2013), 115:1-115:10.

ISBN 978-80-86943-40-4

[NTB*91] Norton A., Turk G., Bacon B., Gerth J., Sweeney P.: Animation of fracture by physical modeling. The Visual Computer 7, 4 (1991), 210-219.

[OBH02] O'Brien J. F., Bargteil A. W., Hodgins J. K.: Graphical modeling and animation of ductile fracture. In Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '02, ACM, pp. 291-294.

[OH99] O'Brien J. F., Hodgins J. K.: Graphical modeling and animation of brittle fracture. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, pp. 137-146.

[PK15] Pätzold M., Kolb A.: Grid-free Out-of-core Voxelization to Sparse Voxel Octrees on GPU. Proceedings of the 7th Conference on High-Performance Graphics, HPG '15, ACM, pp. 95-103.

[SO14] Schvartzman S. C., Otaduy M. A.: Fracture Animation Based on High-dimensional Voronoi Diagrams. Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, I3D '14, ACM, pp. 15-22.

[SSF09] Su J., Schroeder C., Fedkiw R.: Energy stability and fracture for frame rate rigid body simulations. In Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2009), SCA '09, ACM, pp. 155-164.

[TF88] Terzopoulos D., Fleischer K.: Modeling inelastic deformation: Viscolelasticity, plasticity, fracture. SIGGRAPH Comput. Graph. 22, 4 (June 1988), pp. 269-278.

[VMG16] Vllanueva, A. J.; Marton, F.; Gobbetti, E.: SSVDAGs: Symmetry-aware sparse voxel DAGs. In Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM, 2016. pp. 7-14.

[Wil13] Willcocks C. G.: Sparse volumetric deformation. PhD Thesis (apr 2013).

[WRK*10] Wicke M., Ritchie D., Klingner B. M., Burke S., Shewchuk J. R., O'Brien J. F.: Dynamic local remeshing for elastoplastic simulation. ACM Transactions on Graphics 29, 4 (July 2010), 49:1-11. Proceedings of ACM SIGGRAPH 2010, Los Angles, CA.

[ZBG15] Zhu Y., Bridson R., Greif C.: Simulating rigid body fracture with surface meshes. ACM Transactions on Graphics (2015).

[ZGHG11] Zhou K., Gong M., Huang X., Guo B.: Data-parallel octrees for surface reconstruction. IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS (2011).

 ISBN 978-80-86943-40-4

ISSN 2464-4617(print)
ISSN 2464-4625(CD)
Computer Science Research Notes
CSRN 2801
Full Papers Proceedings
http://www.WSCG.eu

3D Object Classification and Parameter Estimation based on Parametric Procedural Models

Roman Getto Kenten Fina Lennart Jarms
Technische Universität Darmstadt
Fraunhoferstr. 5 64283 Darmstadt, Germany
{firstname.lastname}@gris.tu-darmstadt.de

Arjan Kuijper Dieter W. Fellner
Technische Universität Darmstadt & Fraunhofer IGD
Fraunhoferstr. 5, 64283 Darmstadt, Germany
arjan.kuijper@mavc.tu-darmstadt.de
dieter.fellner@gris.tu-darmstadt.de

ABSTRACT

Classifying and gathering additional information about an unknown 3D objects is dependent on having a large amount of learning data. We propose to use procedural models as data foundation for this task. In our method we (semi-)automatically define parameters for a procedural model constructed with a modeling tool. Then we use the procedural models to classify an object and also automatically estimate the best parameters. We use a standard convolutional neural network and three different object similarity measures to estimate the best parameters at each degree of detail. We evaluate all steps of our approach using several procedural models and show that we can achieve high classification accuracy and meaningful parameters for unknown objects.

Keywords

Procedural model, parametric model, parameterization, 3D object classification, deep learning.

1 INTRODUCTION

The most widely accepted approach for 3D Object Classification is the database-approach. A class is learned by having all types of example objects within a database. However, this approach is not applicable to all domains. A large database with all examples for the desired classes is not always available. In research environment there are several big databases that provide enough data to learn different classes and then evaluate the performance of a classification algorithm. In real applications we have actual classes of objects in mind which do not fit to the classes offered in the test databases. The amount of 3D data is often not available and the cost and time effort to produce such a database is tremendous. For 2D (image) classifications the data-approach is more affordable since images are available for literally everything. Many approaches tried to make 3D data more available and delivered environments to easily create new 3D objects, even for non-expert users. However, in direct comparison to image data, 3D data is still near non-existent. When insufficient data is available, one approach is to represent a class directly by a procedural model. The procedural model is a more abstract description of an object by representing the object implicitly by a parameterizable object construction algorithm. Therefore, the procedural model corresponds to a blueprint of an object class. Creating a single procedural model includes some effort but can then be used as a complete data foundation for a desired class. In many cases it is more affordable to create a blueprint instead of gathering a vast amount of example objects.

Our contribution is a complete processing pipeline to achieve classification and parameter estimation using procedural models as basis. Also, the pipeline contains three separate contributions: The algorithm to (semi-)automatically generate parameters for a procedural model, a scheme to use procedural models for deep learning, and the parameter estimation technique using three different similarity measures.

In the following Section we review related work. Section 3 presents our methodology including the procedural model definition, the classification with deep learning, and the parameter estimation. In Section 4 we evaluate and discuss each step of the pipeline individually. Finally, we conclude and outline future work.

2 RELATED WORK

Procedural models are often referred to as grammars [Tal11] or L-systems [Šťt'10]. In general a 3D procedural model is a description of building scheme for a class of 3D objects, which allows to easily generate many different variations. Therefore, procedural models excel in content generation. Instead of an implicit grammar representation, a procedural model can also be represented by a concatenation of parameterized procedures. The

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.2

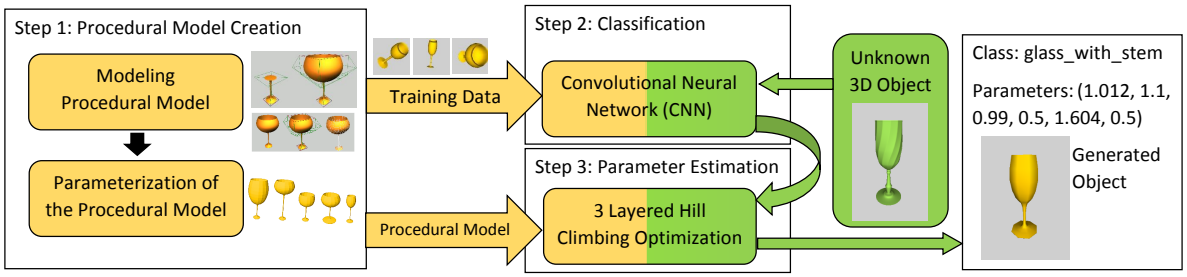


Figure 1: The full pipeline of our system: procedural models are created to represents blueprints of an objects class. These are used to classify and estimate the parameters of an unknown database object.

sequence describes the building process and the parameterization allows the variation of the building process.

Bokeloh et al. [Bok12] propose a procedural modeling algorithm which works on regular structured polygon meshes. A procedural model is automatically generated, so that parameters change the shape of the object while preserving the regular patterns optimally. This approach shows that procedural models are generally very powerful in terms of flexibility. Still, this approach is only suited for cases with regular structured objects.

Other approaches tackle the problem of variation generation by recombining several objects. Jain et al. [Jai12] create variations by part-based recombinations. Yumer et al. [Yum15] define variations with terms like 'luxurious' 'sporty' or 'expensive'. Wang et al. [Wan11] use a symmetry hierarchy to vary objects. Other approaches use box templates [Ave14] to represent a blueprint of a class. Generally, all these approaches are limited to the already available 3D objects. In terms of flexibility procedural models are vastly superior.

In a previous work [Get17] we proposed a definition of procedural models as a concatenation of procedures by using modeling operations. We use this framework for our work to define our initial procedural models.

Ullrich et al. [Ull11] presented a work with a concept similar to ours. They define a 3D object procedurally and compare the procedural model to a query object to estimate the parameters. However, in their approach, the procedural model is designed and parameterized manually and the similarity is only based on a surface difference measure. Our approach includes semi-automatization of the creation of the procedural model, deep learning of the class and a more reliable layered parameter estimation.

To measure the difference between two 3D Objects many so-called descriptors have been proposed. These are focused on different aspects, using histograms [Osa02], topology graphs [Mar07] or image properties of rendered images [Vra05]. For our initial parameter estimation we use the panorama descriptor [Pap10] which is considered to be one of the best geometrical descriptors [Li15] .

For the 3D object classification the descriptors have also been used to directly learn single classes of 3D objects [Wes08, Wan15]. However, deep learning mostly outclassed previous approaches. Maturana et al. [Mat15] and Wu et al. [Wu15] proposed convolutional neural network (CNN) approaches directly learning on voxel representations. Su et al. [Su15] developed a multi-view CNN learning on rendered 2D Images of 3D objects. With this approach they achieve higher accuracy than any comparable approach. The authors reason that currently the relative efficiency using 2D data is higher than using 3D representations. For this reason we also use a CNN approach learning on 2D rendered images.

3 METHODOLOGY

We propose a system based on procedural models. We train a Convolution Neural Network (CNN) with the procedural model and propose a technique to estimate the values of all parameters of the procedural model. We present the concept of our pipeline in Figure 1.

The procedural model itself consists of a concatenation of parameterized procedures. In contrast to an explicit surface representation like a polygon mesh, the procedural model is an implicit object representation. The procedures describe a construction algorithm. When the procedures are executed subsequently an instance of the procedural model is generated. The instance of the procedural model is a 3D polygon mesh itself. When the parameters of the procedures are changed, the resulting mesh changes. Therefore, the procedural model offers the possibility to generate infinite variations by varying the parameters.

For the initial creation of a procedural model we use the tool and the algorithm of Getto et al. [Get17]. The concept of the tool is that a procedural model is automatically generated during the modeling of a single object. The modeling operations are transformed to procedures

ISBN 978-80-86943-40-4

of the procedural model obeying several rules, e.g. the rule of locality, so that parameter changes only have local effects. The boundary representation is a sparse control mesh of a subdivision surface. The edges can be marked as smooth or sharp. It offers basic operations, allowing to insert, remove, drag and connect vertices, edges and faces. Additional a path of face extrusions can be performed by sketching a line from a face.

3.1 Semi-Automatic Parameter Insertion

While the procedural model is modeled manually, we propose a semi-automatic parameter insertion technique to enhance the process of creating the fully parameterized procedural model. Our goal is to compute several possible variations and show them to the user, so that the user can decide which variations make sense. Therefore, the user can define all parameters with a few clicks. As the user cannot inspect every possible parameter, we order the possible variations by 'importance' and furthermore automatically group related parameters together. Table 1 shows the complete overview of the relevant operations. Operations only including ids (e.g. connect faces) are not relevant for the parameterization.

Procedure Parameters: An extrusion and a drag is described with cylindrical coordinates ρ, ϕ and z. A rotation-extrude is defined by the width w and length l. The insert vertex operation defines the the position of the new vertex on an edge as barycentric coordinate λ_1 ($\lambda_2 = 1 - \lambda_1$). A scale has a relative size σ and a rotate has a rotation angle α.

Automatic Variations: For all these operations, we define parameter variations to evaluate the importance. These are shown in Table 1. The automatic variations mostly include doubling, halving or inverting the parameters as suitable.

Importance Evaluation Measures: To measure the importance of an operation we follow a simple rule: The bigger a change the higher the importance. We generate a single mesh for every variation and compare the varied mesh to the original mesh taking into account 5 measures. The overview Table 1 shows the composition of the evaluation measure for each parameter variation. For all measures the base mesh is normalized, so that the centroid is at the origin and the mesh is within a radius of 1. The volume is computed with the method of Zhang et al. [Zha01]. The surface area is the sum of all polygon areas. The bounding sphere has its center at the origin and its radius is the distance to the furthest vertex of the mesh. The coordinate plane projection difference is computed by projecting the surface of the mesh on to the three coordinate plane. We conduct this projection by creating an image of 64x64 pixels for each plane. A pixel is set to true if any part of the object is projected onto this pixel. The average distance is the average Euclidean distance of a vertex of the original mesh and the respective vertex of the varied mesh. The final value of the difference of the two meshes is calculated by the following equations:

$$\delta_p = \frac{p(v) - p(b)}{p(b)} \tag{1}$$

$$\delta_{projection} = \frac{\sum_{i=0}^{m} xor(v_{pixel\_i} - b_{pixel\_i})}{m} \tag{2}$$

$$\delta_{vertexdistance} = \frac{\sum_{i=0}^{n} |v_{vertex\_i} - b_{vertex\_i}|}{n} \tag{3}$$

$$v = variation\ mesh, b = base\ mesh,$$
$$p \in \{volume, surface, BSvolume\},$$
$$m = number\ of\ pixels\ in\ projection\ planes,$$
$$n = number\ of\ vertices\ in\ the\ mesh$$

Parameter Grouping: Groups of parameters are new parameters themselves. When the group parameter is changed all underlying operations are changed respectively. Groups of parameters are formed by finding related operations with related parameters. This is generally the case if two operations are similar. Operations are similar if their values are similar. Therefore, we first define the similarity of two values x and y and two angles α and β:

$$similarity(x,y) = 1 - \frac{|x-y|}{max(1,|x|,|y|)} \tag{4}$$

$$similarity(\alpha,\beta) = 1 - \frac{|\alpha-\beta|}{c} \quad c \in \{45,90\} \tag{5}$$

For the similarity of angles we need to cover additional special cases since two angles of related operations should be considered similar if the one angle is the mirrored version of the other. The angles are defined in a local plane in u-v-space. We consider 4 different angles: the original, mirrored on the u-axis, mirrored on the v-axis and mirrored on both. Furthermore, we check 4 additional angles: the original angle rotated by 90 degrees and all 3 mirrored version of this angle. For this 4 angles the criteria for the similarity are more tight: we half the range of similarity, which is achieved by exchanging the 90 by a 45 within the equation.

To calculate the similarity of two operations we multiply all similarities of their parameters. Finally, we set a threshold for the similarity of each operation as some operations are much more likely to have a higher similarity (e.g. insert vertex) than others.

After finding all similar operations we need to further process them to identify actual groups. We can deduct the relation of operations by their relative position in the sequence of operations. We identified that related instructions are either present in the pattern AA or with the pattern ABAB. This means they are not only similar but also subsequent, as in pattern AA. Or a combination of operations AB is subsequent, forming the pattern ABAB. Finally, we build groups of similar operations which are present in one of these two patterns within the sequence of operations of the procedural model.

User based parameter choice: A parameter is considered to be important if the importance value is bigger or

 ISBN 978-80-86943-40-4

Operation	Procedure Parameters	Automatic Variation	Importance Evaluation Measure	New Inserted Parameter x	Initial Range of x	Similarity Threshold
Extrude (Sketching)	(ρ, ϕ, z)	$(2\rho, \phi, 2z)$	$0.1\delta_{volume} + 0.25\delta_{surface} + 0.1\delta_{BSvolume} + 0.2\delta_{projection}$	$(x\rho, \phi, xz)$	[0.125,8]	0.3
		$if\ \rho > 0.15z\ and\ \phi \in [90,270): (\rho, \phi + 180(\sqrt{0.5} - 1), z)$	$0.75\delta_{projection} + 0.1\delta_{vertexdistance}$	$(\rho, \phi + 180(x-1), z)$	[0,2.0]	
		$if\ \rho > 0.15z\ and\ \phi \in [270,90): (\rho, \phi + 180(\sqrt{1.5} - 1), z)$				
Rotation-Extrude (Sketching)	(w, l)	$(w, 2l)$	$0.15\delta_{volume} + 0.3\delta_{surface} + 0.1\delta_{BSvolume} + 0.2\delta_{projection}$	(w, xl)	[0.125,8]	0.45
		$(0.5w, l)$	$0.7\delta_{volume} + 0.15\delta_{surface} + 0.05\delta_{projection}$	(xw, l)	$[-2,1]$	
Insert Vertex	(λ_1) $\lambda_2 = 1 - \lambda_1$	$if\ \lambda_1 \geq 0.5 : (0.5 + 0.5\lambda_1)$	$0.1\delta_{surface} + 0.3\delta_{projection} + 0.15\delta_{vertexdistance}$	$(0.5 + x\lambda_1)$	[1,1.9]	0.95
		$if\ \lambda_1 < 0.5 : (0.5\lambda_1)$		$(x\lambda_1)$	[0.1,1.0]	
Drag	(ρ, ϕ, z)	$(2\rho, \phi, 2z)$	$0.1\delta_{surface} + 0.5\delta_{projection} + 0.15\delta_{vertexdistance}$	$(x\rho, \phi, xz)$	$[-8,8]$	0.85
Scale	(σ)	$if\ \sigma > 1 : (2\sigma)$	$0.5\delta_{volume} + 0.25\delta_{surface} + 0.1\delta_{BSvolume} + 0.15\delta_{projection}$	$(x\sigma)$	$[-3,3]$	0.9
		$if\ \sigma < 1 : (0.5\sigma)$				
Rotate	(α)	$(-\alpha)$	$0.7\delta_{projection} + 0.1\delta_{vertexdistance}$	$(x\alpha)$	[0.125,5]	0.7

ρ = radial distance, ϕ = angular coordinate, z = height , w = width, l = length, λ = baryzentric coordinate, σ = relative scale, α = rotation angle
δ_{volume} = volume difference, $\delta_{surface}$ = surface area difference, $\delta_{BSvolume}$ = bounding sphere volume difference,
$\delta_{projection}$ = coordinate plane projection difference, $\delta_{vertexdistance}$ = average vertex distance difference

Table 1: Overview of all relevant operations used to automatically parameterize the procedural model.

equal than 1. Additionally, we include the user in this step and offer a simple interface to inspect all parameters and choose all important parameters. This interface is shown in Figure 2. The parameters are ordered by their importance. The user can check or uncheck any individual parameter or parameter group. He can create new groups and rename parameters.

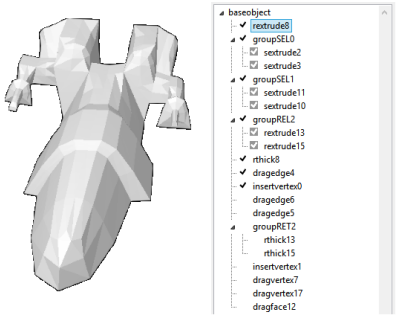


Figure 2: The user interface of the parameter insertion.

Range Estimation: For the random generation of variations we need to additionally define a valid range for the parameter x. The overview Table 1 shows the initial range estimations for every parameter type. For each parameter we generate a mesh with the maximal and minimal value for the specific parameter and measure the difference to the base mesh using the panorama distance [Pap10]. If the panorama distance is bigger than a threshold $C \cdot t$ the range is diminished and reevaluated. t is obtained by measuring the panorama distance to all generated variations of all inserted parameters.

$$Threshold = C \cdot t \tag{6}$$

$$C = constant\ multiplier (default\ C = 1)$$
$$t = max \in \{panorama\ distance\ to\ all\ generated\ meshes\}$$

3.2 Classification with Deep Learning

We propose to use the very deep Convolutional Neural Network (CNN) Inception [Sze15] to directly learn the 3D object with rendered images of the object. The last fully connected layer of the inception network can be retrained with a relative small amount of 3D data. Also the retraining is tremendously faster than training a network from scratch. We retrain the last fully connected layer with a randomized set of images of rendered views of the 3D object. Also, we additionally generate random variations of the 3D object within these images.

Each procedural model represents an object class. For each class we generate 1000 variations (3D mesh). We vary each parameter of the procedural model randomly within the parameter range. For each of the 1000 variations we generate 10 images. In sum, we use 10 000 images per class to train the network.

Figure 3: Examples of generated learning images

Image Generation: We use rendered images of the generated 3D objects with random perspectives. Like [Su15] we noticed that different illumination setups did not make significant differences in the results. In Figure 3 we show example images of our setup. Before generating the images the 3D object is first normalized. The center of mass of the mesh is translated to the origin. The object is scaled, so that all coordinate values are within -1 and 1. We include a random rotation and scaling of the object for each image. The scaling is limited to the range $[0.8, 2]$.

Object Classification: With the retrained network we can classify images. The output of the network for a single image is a class probability for each trained class. To classify a new 3D object we generate 10 images and average the classification values for each class. The 3D object is then put in the class with the highest value.

ISBN 978-80-86943-40-4

3.3 Parameter Estimation

The procedural model has several parameters to generate variations. We estimate those parameters for a new 3D object having the same class as the procedural model. The parameters can either be labeled by the user, e.g. 'wing length', or the influence of the parameters can be shown visually to the user by generating exemplary objects for different values. In both cases estimating the parameters for an unknown object gives valuable information to the user.

The parameter estimation is based on geometrical similarities of the unknown 3D object and the objects generated by the procedural model. It is important to note that the procedural models cannot reproduce any object perfectly in full detail. We use 3 different measures with different degrees of detail. The panorama distance [Pap10], the surface distance and a z-buffer distance.

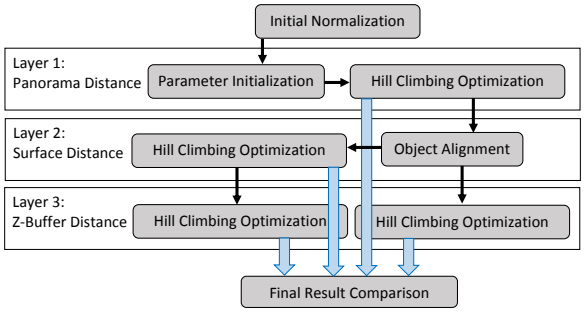


Figure 4: The parameter estimation consists of 3 layers using 3 different levels of distance measures. The final result is the best result of 4 different optimizations.

Our algorithm includes 3 layers for these 3 measures. We show an overview of the parameter estimation in Figure 4. We use all measures subsequently in a hill climbing optimization to refine the estimated optimal parameters step by step. Additionally we set thresholds for the measures, so that the result of the preceding layer is taken if the object cannot be represented precisely on a layer. As a result we do not only estimate the best parameters but actually are able to tell how well the parameters of the procedural model can represent the unknown 3D object.

Panorama Distance: The panorama distance is defined by the panorama descriptor [Pap10], which is a hybrid descriptor based on geometrical features and image features of panoramic views of the object.

Surface Distance: The surface distance, also known as point-to-surface-distance is based on the distance between the actual surface polygons of both meshes. To compute the surface distance between an instance of the procedural model and the unknown object we generate a set of points for both meshes. For the unknown object we use the Poisson disk sampling [Cor12] with 2000 points. For the mesh generated by the procedural model we take all vertices of the mesh after 2 iterations of the subdivision. For each set of points the surface distance of a single point is the distance to the nearest point of the other set. We average this distance over all points.

Z-Buffer Distance: We compare the z (depth) information of both objects pixel wise. For this distance we generate a total of 14 views with 256x256 pixels. We use an orthogonal projection with $[-2,2]$ for all boundary planes. The 14 views are the 6 views directly from the positive and negative coordinate axes and the 8 views from the corners of a cube around the origin. We Present the distance calculation in Algorithm 1. Note that we penalize an undersizing of the generated object more than an oversizing. This reinforces the initial growing into all regions.

Algorithm 1 Z-Buffer Distance

```
1:  procedure (Original Obj. O, Generated Obj. G)
2:      Generate 14 views V_O for O and V_G for G
3:      Distance d ← 0
4:      for all Views v_O ∈ V_O do
5:          for all Pixels p_O ∈ v_O do
6:              if p_O = background ∧
                   p_G != background then
7:                  d = d + 1
8:              else if p_O != background ∧
                   p_G = background then
9:                  d = d + 2
10:             else if p_O != background ∧
                   p_G != background then
11:                 d = d + |z(p_O) − z(p_G)|
12:             end if
13:         end for
14:     end for
15:     d is the final distance
16: end procedure
```

Initial normalization: At the start we bring both objects into a shared coordinate system. The average position of the vertices of the mesh (center of mass) is translated to the origin. The object is scaled, so that all coordinate values are within -1 and 1.

Hill climbing algorithm: Each layer includes a hill climbing search with one of the distance measures. We show the hill climbing search for a distance measure D in Algorithm 2. All parameters are optimized with diminishing step sizes. Note that to measure the distance with D and parameters P we generate a new object using the procedural model with the parameters P.

For the surface distance (layer 2) and the z-buffer distance (layer 3) an additional intermediate step has to be inserted: When parameters of the procedural model change and the object shape changes, the position of the object is shifted respectively. Therefore after each change of a parameter value the objects have to be realigned before the distance measure is computed. Just

ISBN 978-80-86943-40-4

Algorithm 2 Hill Climbing Optimization

```
1:  procedure (Distance Measure D, Parameters P)
2:      Distance d ← D(P)
3:      do
4:          Step size s ← 1.0
5:          do
6:              for all p ∈ P do
7:                  for all ⊕ ∈ {+, −} do
8:                      p_new = p ⊕ s
9:                      clamp p_new to [min, max] of p
10:                     Distance d_new ← D(p_new)
11:                     if d_new < d then
12:                         d = d_new
13:                         p = p_new
14:                     end if
15:                 end for
16:             end for
17:         while no parameter has been changed
18:         s = next s ∈ {1.0, 0.5, 0.25, 0.1, 0.01, 0.0}
19:     while s! = 0.0
20: end procedure
```

like the parameter adjustment, we perform a greedy search for the best translation, rotation and scaling. The step size is fixed for this realignment: 0.01 for the translation and scaling, and $0.01 \cdot 180^\circ$ for the rotation. The scaling is limited to a minimum of 0.5 and a maximum of 1.5 of the original scale.

Layer 1 - panorama distance: Before the first layer the initial normalization (scaling and translation) is performed. In the first layer the parameter initialization and hill climbing optimization with the panorama distance is performed (See Figure 4). The initialization of the parameters of the procedural model is of major importance since the following greedy hill climbing algorithms can get stuck in a local extremum. The panorama distance generally measures the distance between two 3D objects and is optimally suited to fulfill this task. We generate a total of 10 000 objects from the procedural model with random parameterization, covering a large range of possible initialization values. We compute the panorama distance of each generated object to the unknown object. The parameters of the generated object with the smallest panorama distance are taken as our starting point. Then we perform a hill climbing optimization of all parameters using the panorama distance.

Layer 2 - surface distance: In the second layer the object alignment and hill climbing optimization with the surface distance is performed (See Figure 4). For the surface distance measure and the following z-buffer distance measure we need to optimize the alignment of both objects. We optimize the translation, rotation and scaling in a greedy search like introduced in the hill climbing description. However, this greedy search only remedies small misalignments. Since the initial orientation can be majorly flawed we additional consider 24 possible coordinate system rotations. The 24 rotations include all main rotation possibilities: the x-axis can be rotated to match one of the 6 possible positive or negative coordinate system axis and can be rotated around itself by $0, 90, 180$ or 270 degrees. Giving a total of $6 \cdot 4 = 24$ possibilities. For each of the 24 possibilities we perform the alignment optimization and evaluate the case with surface distance. The case with the lowest surface distance is taken as the initial alignment. Finally, we perform the hill climbing optimization of all parameters using the surface distance.

Layer 3 - z-buffer distance: In the final layer two hill climbing optimizations with the z-buffer distance are performed (See Figure 4). In the first case we use the output of layer 2 as input and in the second case we use the output of layer 1 (after the alignment in layer 2) as input. Hence, we compute optimal parameters for 2 different starting point. Then we compare the z-buffer distance of the two final optimization results and take the better solution as final result.

At the end of our parameter estimation we decide which layer result is the most adequate representation. The procedural model might not be able to represent every object to a pixel wise degree, hence we set thresholds for the final results to decide to which extent the procedural model represents the unknown object. We analyzed several objects, results and distance measures values and identified shared thresholds for the distance measures. A z-buffer distance of lower than 0.7 and a surface distance of lower than 0.04 represents an adequate match. The final parameters correspond to the result of layer 3 if the z-buffer distance is below 0.7. Else it corresponds to the result of layer 2 if the surface distance is below 0.04. If both are not the case than the result of layer 1 gives the final parameters.

4 EVALUATION AND DISCUSSION

In this section we evaluate our approach, including our 3 steps: the parameter insertion, classification with deep learning and the parameter estimation. For each step we separately show results and discuss the results.

4.1 Parameter Insertion

To evaluate the correctness of the proposed semi-automatic parameter insertion technique we evaluate the importance evaluation measure and the parameter grouping. We created several example models (See Figure 5) with the modeling tool and manually annotated important parameters and appropriately grouped related parameters. Then we retrieved the proposed important parameters and groups automatically detected by the default threshold of 1. We present our

 ISBN 978-80-86943-40-4

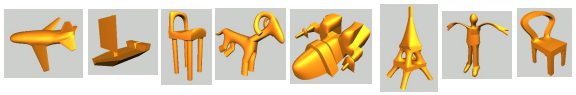


Figure 5: Models of the parameter insertion evaluation.

	Importance					Grouping			
	CP	FP	FN	CN	Accuracy	CG	FG	MG	Accuracy
Airplane	9	1	1	11	90.91%	7	0	3	70.00%
Ship	7	3	0	8	83.33%	3	1	0	66.67%
Stool	6	3	0	12	85.71%	8	1	0	87.50%
Animal	11	12	1	18	69.05%	7	0	0	100.00%
Spaceship	6	1	0	6	92.31%	3	0	0	100.00%
Tower	11	4	1	7	78.26%	4	0	1	80.00%
Humanoid	10	8	1	12	70.97%	9	0	1	90.00%
Chair	6	6	0	8	70.00%	7	1	0	85.71%
Average	9.42	5.43	0.57	11.71	80.07%	6.00	0.38	0.63	84.98%
CP = Correct Positive FP = False Positive, FN = False Negative CN = Correct Negative						CG = Correct Group FG = False Group MG = Missed Group			

Table 2: The accuracy of the importance evaluation measure and the accuracy of the parameter grouping

results in Table 2. The most relevant parameters are automatically categorized as important in 80% of the cases. 84% percent of the groups are correctly identified by the algorithm.

Discussion: Table 2 shows that false negatives are seldom. Our algorithm rather finds too many important parameters, so that false positives occur. In several cases parameters are found to be important even though the change of the parameter does not lead to a semantically consistent outcome. For this reason, we include a user phase where the user can refine parameters. Therefore, the user is able to correct semantic inconsistencies.

The parameter grouping also has erroneous cases. The majority of the missing groups and false groups are caused by the insert vertex operation. The insert vertex only includes a single parameter and the values are mostly within $[0.25, 0.75]$. Even though we have set a tight threshold with 0.95 for this operation, the similarity computation is less reliable for this operation. We highlight the insert vertex groups for the user, so that he can decide in these cases.

4.2 Classification with Deep Learning

To evaluate our classification approach with deep learning on rendered images, we constructed 10 procedural models. Figure 6 shows the models. The 10 procedural models represent 10 different classes. To evaluate our approach we use the NIST database [Fan08] and the Princeton shape benchmark (PSB) [Shi04] together. Table 3 shows the number of objects of each class in the databases and also show properties of the procedural models.

We took the preset classes within the given databases (since the NIST database only has single a class with spiders and insects our spider class includes both). We trained our network with a total of 100 000 images (10

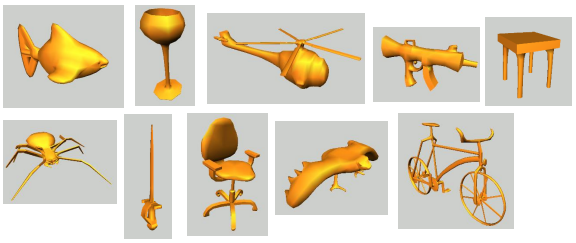


Figure 6: All models that were used to evaluate the classification and parameter estimation step.

	Database			Procedural Model			
	NIST	PSB	Total	po	to-ops	par-ops	par
fish	18	17	35	1348	150	68	13
glass_with_stem	18	9	27	332	44	11	6
helicopter	18	35	53	1560	294	184	12
gun	36	39	75	888	161	54	15
table	36	63	99	614	87	51	6
spider	18	16	34	2976	197	137	9
sword	18	31	49	568	64	30	7
office_chair	18	15	33	1724	193	86	8
bird	18	21	39	2040	241	134	13
bicycle	18	7	25	4966	510	189	10
others	504	1562	2066				
total within classes	216	253	469				
total	720	1815	2535				
po = number of polygons to-ops = total number of operations par-ops = parameterizable operations par = number of parameters							

Table 3: The number of objects for each class in the PSB and the NIST database, and the properties of the used procedural models

classes,1000 variations with 10 images). The retraining took about 10 hours on a casual PC. We used our algorithm for 2 different scenarios: the classification of database objects and a 3D object retrieval scenario.

Classification: We classified every object of the databases that belong into one of the 10 learned classes and measured the overall classification accuracy. Figure 7 (right) shows the accuracy for all 10 classes resulting in the average accuracy of 86.14% (404 of 469 objects are classified correctly).

3D object retrieval: The output of a 3D object retrieval query is a list of retrieved objects, ordered from the most similar to the least similar. We directly use the class probabilities given by the neural network to sort the list of retrieved objects. A class label is the query itself and the first object of the retrieval list is the 3D object with the highest class probability for this class. Here we include all 2535 objects of all databases.

Figure 7 shows the precision recall curve for the 3D object retrieval. We compare this result with the panorama distance by using the default instance of the procedural model as query for the database. The panorama distance from this object to all objects in the database is calculated and the retrieval list is sorted respectively. We also show the result of our approach without vari-

ISBN 978-80-86943-40-4

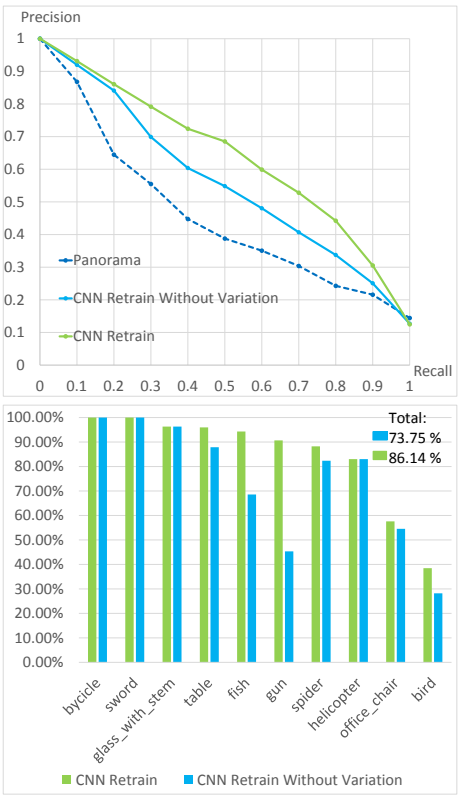


Figure 7: The 3D object retrieval precision-recall curve and the classification accuracy.

ations: we generate all images without changing any parameters and retrain the network with these images.

Discussion: The classification and the 3D object retrieval illustrate several properties of the approach. An important insight is that including variations into the learning process leads to improvements. This is not as trivial as it might seem at first glance. We tested several other possibilities of image generation, including random translations and higher variations of scaling and found out that it is easier for the neural network to learn the class when the images are more consistent. At the same time a good amount of variability is needed in the images to prevent overfitting and promote generalizability. However, our results clearly show that object variations enhance the results in all cases.

The average classification accuracy is 86%. This is comparable to state-of-the-art approaches like [Su15] achieving 83-90% accuracy on the classification task. Only the office chair and bird class achieved a lower accuracy. The database objects are not sufficiently similar to the initial procedural model. In Figure 8 we show falsely classified objects. The parameters did not compensate very exceptional variations of the objects.

In the precision-recall curve (Figure 7) our approach also outperforms the panorama distance, even though the panorama distance is among the best geometrical distance measures. In sum, our deep learning retraining

Figure 8: Images of falsely classified objects.

approach with rendered images of variations is fast and works with less data than a full network learning and still generates comparable results.

4.3 Parameter Estimation

For the final step we took all correctly classified examples of our 10 classes and estimated the parameters of the procedural model for every unknown database object. In total 66.09% of the final parameter estimations origin from layer 3. 10.15% from layer 2 and 23.76% from layer 1. Figure 9 shows the distribution of the surface distance and z-buffer distance for the 4 different results in the 3 layers. Figure 10 presents several exemplary parameter estimations for all classes.

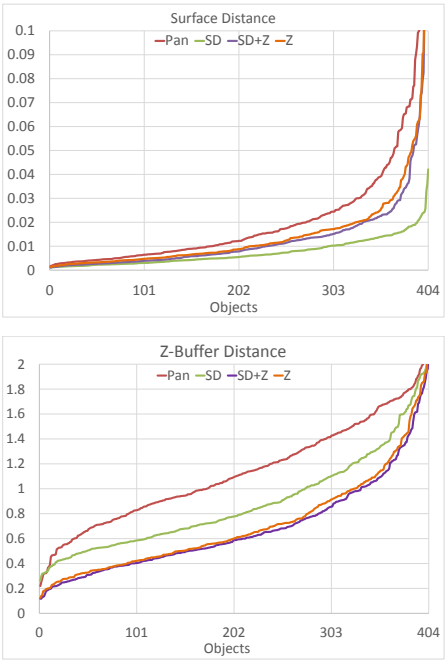


Figure 9: The distribution of the surface distance and the z-buffer distance for all objects for the 4 different results from the 3 layers.

Discussion: In Figure 9 we can detect a general advantage of the layered optimization system. In the plots we present the two different results from layer 3 separately: using the surface distance with z-Buffer distance(SD+Z) and only the z-buffer distance(Z). Here, we can see that the distribution of the z-buffer distance in the final layer is better on average when the output of the 2nd layer is taken as input (SD+Z). The hill climbing algorithm naturally profits from a good initialization. We can see that not only the initial setup improves the results, but also the intermediate optimization of layer 2 improves the results of the final layer 3.

ISBN 978-80-86943-40-4

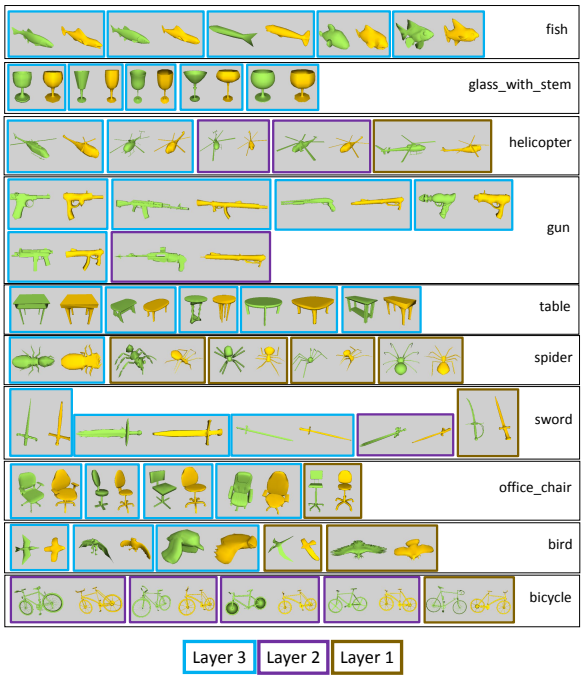


Figure 10: Exemplary results for all classes. The colored borders show from which layer the result origins.

Figure 11: Ten different glasses of the database sorted by the ratio of stem length to the bowl length.

The examples presented in Figure 10 show that the parameter estimations lead to generated objects with similar overall appearance compared to the unknown database objects. Most objects could be estimated on layer 3 (z-buffer). However, the bicycle, spider and helicopter class did not have enough flexibility to represent most of the objects on layer 3. Especially the rotors of the helicopter, the legs of the spiders and the thin spokes and connection bars of the bicycle could not be matched pixel-wise. The user can improve the estimations for the classes by adding additional parameters to increase the flexibility. Nonetheless, our system is able to provide meaningful results from layer 2 (surface distance) and layer 1 (panorama distance) for the cases where the procedural model is not suitable enough for the objects.

Figure 11 presents an object characteristic derived by the parameters. Here we order the objects by the ratio of the stem length to the bowl length. Important to note in this context is that ratios and differences between parameters are more meaningful than the comparison of values of a single parameter. This is the case because the database objects have to be normalized and the instances of the procedural models have to be scaled and aligned accordingly. Therefore, the actual values itself are less comparable when the coordinate systems of different objects do not match. In the use case of having scanned objects as input, no normalization is needed since the values are related to real millimeter values. In this case the actual values of single parameters are also completely comparable.

Figure 12 shows two types of errors that we found in the results. The bird is mostly symmetrical, so that the instance of the procedural model happens to be misaligned. The head and the tail are facing in the wrong direction. These cases happened at some symmetrical objects of the bird, fish and gun class. In the future we will have to integrate an additional symmetry detection to handle these cases explicitly.

The second error type is represented by the glasses with stem in Figure 12. The database object does not have a real stem. The bowl is directly connected to the base. The procedural model does not include the case of a stem having 0 length. Even though this result comes from layer 3, the final parameters are distorted by the falsely estimated stem length.

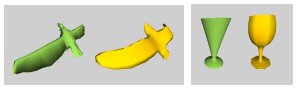


Figure 12: The bird is falsely aligned. The glass has an estimation of the stem length even though the glass of the database has no stem.

5 CONCLUSION & FUTURE WORK

We proposed a new approach including a system to model and parameterize complete procedural models, train a convolutional neural network solely with the procedural models and finally classify an unknown object from a database and additionally estimate all parameters of the procedural model for the unknown object. Hence, our system does not only classify unknown objects but also retrieve additional information.

The proposed system has a very high potential when suitable procedural models can be created. Therefore, the currently biggest drawback is the need to model the initial model with the modeling tool. We will further investigate the possibilities of automatizing this step. Creating a method that can automatically construct a procedural model from a single object in mesh representation would highly enhance the ease and usability of our system.

Our learning method shows a clear enhancement of the results by using the variations of the objects. A further investigation of the exact mechanisms leading to this effect should be performed. This would enable advanced possibilities of enforcing this mechanisms.

The accuracy of the final parameter estimation step is directly dependent on the provided procedural models. Therefore, the final estimation will improve by further enhancing the creation of the procedural model itself.

ISBN 978-80-86943-40-4

6 REFERENCES

[Ave14] Averkiou M., Kim V. G., Zheng Y., Mitra N. J. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. In Computer Graphics Forum, vol. 33, Wiley Online Library, pp. 125–134, 2014.

[Bok12] Bokeloh M., Wand M., Seidel H.-P., Koltun V. An algebraic model for parameterized shape editing. ACM Transactions on Graphics 31, No. 4, pp. 1–10, 2012.

[Cor12] Corsini M., Cignoni P., Scopigno R. Efficient and flexible sampling with blue noise properties of triangular meshes. IEEE Transactions on Visualization and Computer Graphics 18, No. 6, pp. 914–924, 2012.

[Fan08] Fang R., Godil A., Li X., Wagan A. A new shape benchmark for 3d object retrieval. Advances in Visual Computing, pp. 381–392, 2008.

[Get17] Getto R., Merz J., Kuijper A., Fellner D. W. 3d meta model generation with application in 3d object retrieval. In Proceedings of the Computer Graphics International Conference, ACM, p. 6, 2017.

[Jai12] Jain A., Thormählen T., Ritschel T., Seidel H.-P. Exploring Shape Variations by 3d-Model Decomposition and Part-based Recombination. In Computer Graphics Forum, vol. 31, Wiley Online Library, pp. 631–640, 2012.

[Li15] Li B., Lu Y., Li C., Godil A., Schreck T., Aono M., Burtscher M., Chen Q., Chowdhury N. K., Fang B., et al. A comparison of 3d shape retrieval methods based on a large-scale benchmark supporting multimodal queries. Computer Vision and Image Understanding 131, pp. 1–27, 2015.

[Mar07] Marini S., Spagnuolo M., Falcidieno B. Structural shape prototypes for the automatic classification of 3d objects. IEEE Computer Graphics and Applications, No. 4, pp. 28–37, 2007.

[Mat15] Maturana D., Scherer S. Voxnet: A 3d convolutional neural network for real-time object recognition. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE, pp. 922–928, 2015.

[Osa02] Osada R., Funkhouser T., Chazelle B., Dobkin D. Shape distributions. ACM Transactions on Graphics (TOG) 21, No. 4, pp. 807–832, 2002.

[Pap10] Papadakis P., Pratikakis I., Theoharis T., Perantonis S. Panorama: A 3d shape descriptor based on panoramic views for unsupervised 3d object retrieval. International Journal of Computer Vision 89, No. 2, pp. 177–192, 2010.

[Shi04] Shilane P., Min P., Kazhdan M., Funkhouser T. The princeton shape benchmark. In Shape modeling applications, 2004. Proceedings, IEEE, pp. 167–178, 2004.

[Št'10] Št'ava O., Beneš B., Měch R., Aliaga D. G., Krištof P. Inverse procedural modeling by automatic generation of l-systems. In Computer Graphics Forum, vol. 29, Wiley Online Library, pp. 665–674, 2010.

[Su15] Su H., Maji S., Kalogerakis E., Learned-Miller E. Multi-view convolutional neural networks for 3d shape recognition. In Proceedings of the IEEE international conference on computer vision, pp. 945–953, 2015.

[Sze15] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9, 2015.

[Tal11] Talton J. O., Lou Y., Lesser S., Duke J., Měch R., Koltun V. Metropolis procedural modeling. ACM Transactions on Graphics (TOG) 30, No. 2, p. 11, 2011.

[Ull11] Ullrich, Torsten, Fellner, Dieter W. Generative Object Definition and Semantic Recognition. 2011.

[Vra05] Vranic D. V. Desire: a composite 3d-shape descriptor. In Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on, IEEE, pp. 4–pp, 2005.

[Wan11] Wang Y., Xu K., Li J., Zhang H., Shamir A., Liu L., Cheng Z., Xiong Y. Symmetry Hierarchy of Man-Made Objects. In Computer graphics forum, vol. 30, Wiley Online Library, pp. 287–296, 2011.

[Wan15] Wang Y., Liu Z., Pang F., Li H. Boosting 3d model retrieval with class vocabularies and distance vector revision. In TENCON 2015-2015 IEEE Region 10 Conference, IEEE, pp. 1–5, 2015.

[Wes08] Wessel R., Baranowski R., Klein R. Learning distinctive local object characteristics for 3d shape retrieval. In VMV, pp. 169–178, 2008.

[Wu15] Wu Z., Song S., Khosla A., Yu F., Zhang L., Tang X., Xiao J. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1912–1920, 2015.

[Yum15] Yumer M. E., Chaudhuri S., Hodgins J. K., Kara L. B. Semantic shape editing using deformation handles. ACM Transactions on Graphics 34, No. 4, pp. 86:1–86:12, 2015.

[Zha01] Zhang C., Chen T. Efficient feature extraction for 2d/3d objects in mesh representation. In Image Processing, 2001. Proceedings. 2001 International Conference on, vol. 3, IEEE, pp. 935–938, 2001.

 ISBN 978-80-86943-40-4

ISSN 2464-4617(print)
ISSN 2464-4625(CD)
Computer Science Research Notes
CSRN 2801
Full Papers Proceedings
http://www.WSCG.eu

Generation of Implicit Flow Representations for Interactive Visual Exploration of Flow Fields

Molchanov Vladimir
The University of Münster
Schlossplatz 2
48149 Münster, Germany
molchano@uni-muenster.de

Lars Linsen
The University of Münster
Schlossplatz 2
48149 Münster, Germany
linsen@uni-muenster.de

ABSTRACT

A stream function is an implicit flow representation in form of a function, whose values are constant along streamlines of the underlying velocity field. To generate a stream function, a common approach is to use a streamline tracking technique after assigning scalar function values on the inflow/outflow domain boundary (pre-processing step). However, non-trivial flows generally have streamlines that do not start or end at the domain boundary. We propose an automatic approach that defines a stream function along such streamlines. To do so, we construct optimal termination surfaces inside the domain and assign scalar values to all streamlines crossing these surfaces. Furthermore, we propose a proper functional to characterize the quality of the approximated stream function. Using a variational approach, we derive a partial differential equation for the minimization of the derived functional. This minimization procedure is an effective tool to improve the stream function. It can also be used to significantly improve the pre-computation times by creating a high-quality high-resolution stream function from a low-resolution estimate. Once the implicit flow representation is established and improved, we can efficiently extract flow geometry such as stream ribbons, stream tubes, stream surfaces, etc. by applying fast marching algorithms. Tracking time recorded during the pre-processing step can be coupled with the stream function or used directly to extract time surfaces. Thus, the entire flow field can be explored interactively. There is no need for time-consuming particle tracking and mesh refinement during the visual exploration process.

Keywords

Flow visualization, streamlines and -surfaces, implicit representation, stream function.

1 INTRODUCTION

Modern flow visualization systems are required to handle large volumetric datasets of high complexity, to extract and transform requested information fast and accurately, and to meet users' intuition and expectation when rendering. The enormous demand on such systems caused an intensive research on this topic over the last decades. As a result there have appeared various visualization algorithms combining ideas from numerical methods, fluid dynamics, geometry, and other fields.

Most of the existing approaches can be classified into four large groups: direct, geometric, texture-based, and feature-based methods [LHD+03]. All these approaches have their own application areas and differ in efficiency, generality, and expressiveness. Direct methods are intuitive but only allow for local comprehension of the flow and are of limited use when considering volume data. Texture-based techniques produce dense flow representations by applying filters to three-dimensional textures. Occlusion becomes an issue. Scalar characteristics are in the focus of feature-based methods, which often require more experience from the user. Geometric approaches are considered to be quite intuitive and expressive.

Our paper is devoted to three-dimensional geometric flow visualization using an implicit flow representation. The core of most geometric approaches is an integration of the flow field, which can be extremely time consuming when postulating high accuracy. To allow for an interactive visualization that involves many geometric objects, the integration needs to be executed in a pre-processing phase. Our algorithm takes advantage of an implicit representation of flow, thus, effectively converting the problem to a scalar field visualization task. Given the implicit flow representation in the form of a collection of stream functions, an extraction and rendering of geometric stream elements is performed efficiently using the available pre-integrated information.

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.3

Implicit flow representation is a collection of stream functions together with advection times and lengths recorded for each node. A (generalized) stream function is a non-trivial function, whose values are constant along streamlines of the underlying velocity field. To generate a stream function for a given velocity field, a common approach is to use a streamline tracking technique after assigning scalar function values (parametrization) on the inflow/outflow domain boundary, see Section 3. However, non-trivial flows generally have streamlines that do not start or end at the domain boundary. We propose an automatic approach that defines a stream function along such streamlines. To do so, we construct optimal termination surfaces inside the domain and assign scalar values to all streamlines crossing these surfaces, see Section 4. We also support the interactive modification of position and parametrization of the termination surfaces by the user based on the information obtained by the automatic procedure. After having computed one or several distinct stream functions for gridded data, marching algorithms can be applied to the grid to visualize implicit stream elements, such as streamlines, stream tubes, stream ribbons, stream surfaces, etc, see Section 6. Moreover, tracking time can be recorded during the pre-processing step, which allows for the extraction of time surfaces or for enhancing other stream elements with time information.

Another aspect of our work is concerned with the quality of the stream functions. To our knowledge, there exists no tool to measure and improve the quality of the pre-integrated data. Our efforts were concentrated on developing such an approach that improves a stream function with respect to the underlying velocity field. Using a variational approach, we derive a partial differential equation to optimize the derived quality measure, see Section 5. The procedure can be useful in many regards, including the following:

- *Improvement*: A stream function constructed by tracking of samples may contain noise, exhibit sampling artifacts, or have high local errors due to a non-uniform behavior of the velocity field. Our minimization procedure improves the quality of the stream function and can eliminate these artifacts.
- *Refinement*: Computing a stream function over a large domain can be rather expensive when tracking all nodes. Using our approach, we can downsample the data, compute a coarse approximation of the stream function, use interpolation for upsampling to the original resolution, and correct the interpolated stream function values via the proposed minimization procedure.

The main contributions of the paper can be summarized as follows: (1) Automatic generation of implicit flow representation for the entire flow domain; (2) Termination surfaces to generate a parametrization for streamlines not crossing the domain's boundary; (3) Proper functional to control the stream function quality; (4) Variationally derived procedure for stream functions improvement; (5) Effective algorithms for extraction of various stream elements with the possibility to represent advection-time information in form of color (transparency) encoding or extraction of time surfaces.

2 RELATED WORK

Nontrivial real-world and modeled flows have variations in velocity and curl magnitude, an inhomogeneous distribution of helicity and divergence, and a non-degenerated determinant of the gradient tensor. All these scalar fields associated with a flow are features that play an important role in flow analysis. An approach to highlight regions of a non-uniform flow behavior is to use a multi-dimensional transfer functions [PBL+04, PBL+05], or glyphs [GRT17].

Flow direction – one of the simplest flow characteristics – is hardly described by a scalar quantity. To depict this information the Line Integral Convolution method was proposed by Cabral and Leedom [CL93]. The idea is to blur textures along a given vector field over the domain producing intuitive patterns, especially in two spatial dimensions. In the case of a volumetric flow, the method can be combined with other approaches. For instance, Schafhitzel et al. [STWE07] computed and rendered stream surfaces and path surfaces of a three-dimensional flow with a texture-based surface flow structure.

Rendering of flow-related geometrical objects is an extremely helpful visualization method. Colored points, curves, and surfaces may be used to define the topological skeleton of a vector field, i.e., critical points, periodic orbits, separatrices, etc. Existing approaches focus on topological segmentation of two-dimensional [SHJK00] and three-dimensional steady vector fields [MBS+04], an analysis of time-dependent vector field topology [SRP09], and extraction of two-dimensional separatrices of three-dimensional saddles and saddle type periodic orbits [PS09].

The basic underlying principle of topology-based and geometric methods is the tracking of imaginary particles introduced into the flow. The idea was adopted from real-world experiments on injection of an extraneous, clearly visible fluid material into a stream. Propagation of the material displays the stream- or pathline structure of the flow. A proper optical model for smoke advection in an unsteady flow was proposed [vFWTS08]. Li et al. [LTH08] developed a dye propagation scheme overcoming non-physical artifacts of integration. Cuntz et al. [CKSW08] advected a dye in an unsteady three-dimensional flow using a hybrid particle-mesh formalization. A dye released into the

 ISBN 978-80-86943-40-4

flow at fixed positions at different times results in *streak lines* [WT10].

An integration along particle paths is commonly done by a fourth-order Runge-Kutta method [PYH+06]. The paths describe streamlines or pathlines and can be extended to stream ribbons or stream tubes [RLN+17]. An improvement of stream ribbon triangulation in divergent or shearing flows was studied in the seminal paper by Hultquist [Hul92].

The construction of a flow topology skeleton is mainly done without any user interaction. Although it also requires significant computational efforts, extraction of stream surfaces and lines is more user-oriented, since the seeding points can be defined arbitrarily. Typically, the number of simultaneously extracted stream elements needs to be limited to allow interactive frame rates. One step towards an interactive visualization application that allows the simultaneous extraction of many stream elements can be taken by moving all time-consuming integration to a pre-processing phase and encoding the flow implicitly in a scalar stream function.

An implicit surface representation is the key idea of a wide class of level-set methods, e.g., [CKSW08, WJE00, WJE01]. Early attempts in implicit representations of stream surfaces go back to van Wijk [vW93]. All grid nodes were tracked in the direction opposite to the flow until they reach the domain boundary. The velocity field was evaluated via a trilinear interpolation from the grid. Values of a smooth scalar function defined at the boundary are then assigned to the nodes based on the assumption that they remain constant along each streamline. Alternatively, a convection equation is solved on a regular grid. Isosurfaces of the resulting gridded volumetric function are then proven to be stream surfaces of the underlying flow. Xue et al. [XZC04] adapted the approach by van Wijk to render implicit volumes. Instead of assigning scalar values on the inflow region, the user is asked to paint a two-dimensional texture on the boundary (termination surface). Properly constructed boolean fields which remain unchanged along streamlines allow for effective flow topology exploration as shown in [SS07]. In this paper, we present an approach that computes stream functions fully automatically. Moreover, we define a quality measure and present an approach for improving stream functions.

A streamline can be found as an intersection of two stream surfaces called dual. A cell-wise trilinear approximation of dual stream functions (f and g) was used by Kenwright et al. to render streamlines [KM92]. A concept of an fg-diagram was then generalized to an irregular tetrahedral mesh [KM96].

3 STREAM FUNCTIONS

A stream function $\psi(\mathbf{x})$ of a two-dimensional potential flow $\mathbf{w}(\mathbf{x}) = (w_1(\mathbf{x}), w_2(\mathbf{x}))$, $\mathbf{x} = (x_1, x_2)$, is known to satisfy the Poisson equation $\triangle\psi(\mathbf{x}) = \dfrac{\partial w_2(\mathbf{x})}{\partial x_1} - \dfrac{\partial w_1(\mathbf{x})}{\partial x_2}$, where $\triangle = \frac{\partial}{\partial x_1} + \frac{\partial}{\partial x_2}$ stands for the Laplace operator. The right-hand side of the equation has the meaning of vorticity with a negative sign. The stream function $\psi(\mathbf{x})$ remains constant along streamlines and the magnitude of its gradient is proportional to the flux. This property holds exceptionally for potential flow, i.e., for velocity field $\mathbf{w}(\mathbf{x})$ with $\nabla \times \mathbf{w}(\mathbf{x}) = 0$. However, a generalized notion of a stream function is still applicable for non-potential flows in spatial dimensions higher than two.

A (nontrivial) scalar function $f(\mathbf{x})$ is said to be a (generalized) *stream function* of a given vector field $\mathbf{u}(\mathbf{x})$ (interpreted as velocity), if $\nabla f(\mathbf{x}) \perp \mathbf{u}(\mathbf{x})$ everywhere in a domain $D \in \mathbb{R}^d$, $d \geq 2$. It implies that $f(\mathbf{x})$ is constant along any streamline of the flow $\mathbf{u}(\mathbf{x})$, i.e., an implicit relation $f(\mathbf{x}) = f_{\text{iso}}$ with some constant f_{iso} defines a streamline or a stream surface for $d = 2$ or $d = 3$, correspondingly.

We assume that the underlying vector field is sufficiently smooth, i.e., its components have continuous first derivatives. Since the stream function definition above is invariant under arbitrary scaling of the velocity $\mathbf{u}(\mathbf{x})$, it is convenient to normalize the flow introducing a new field $\mathbf{v}(\mathbf{x}) = \mathbf{u}(\mathbf{x})/\|\mathbf{u}(\mathbf{x})\|$. The boundary ∂D of the flow domain D can be split into two parts, the *inflow boundary region* ∂D_{in} and the *outflow boundary region* ∂D_{out}, i.e., $\partial D = \partial D_{\text{in}} \bigcup \partial D_{\text{out}}$. By definition, $\mathbf{y} \in \partial D_{\text{in}}$ iff $\mathbf{y} \in \partial D$ and $\mathbf{v}(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) \leq 0$, where $\mathbf{n}$ is a normal to the boundary ∂D pointing outwards and "$\cdot$" denotes the inner product of vectors in $\mathbb{R}^d$.

There exist two main approaches to construct a stream function $f(\mathbf{x})$. A first approach solves the partial differential transport equation with boundary condition

$$\frac{\partial f(\mathbf{x},t)}{\partial t} + \mathbf{u} \cdot \nabla f(\mathbf{x},t) = 0; \quad f(\mathbf{y},t) = f_0(\mathbf{y}), \quad \mathbf{y} \in \partial D_{\text{in}},$$

to track boundary values throughout the domain along streamlines. Alternatively, all grid nodes $\mathbf{g}_i$ can be tracked backwards in the flow (so called, *anti-particles*). Here, the ordinary differential equation

$$\frac{\mathrm{d}\mathbf{g}_i(t)}{\mathrm{d}t} = -\mathbf{u}, \qquad \mathbf{g}_i(0) = \mathbf{g}_i, \tag{1}$$

is to be solved until each tracked particle reaches ∂D_{in} at some time t_i. After that, the inflow boundary region is parametrized, i.e., some scalar values are prescribed to all inflow boundary points. Then, all grid nodes $\mathbf{g}_i$ are assigned with the same scalar values as their footprints $\mathbf{g}_i(t_i)$. The established volumetric scalar field is

 ISBN 978-80-86943-40-4

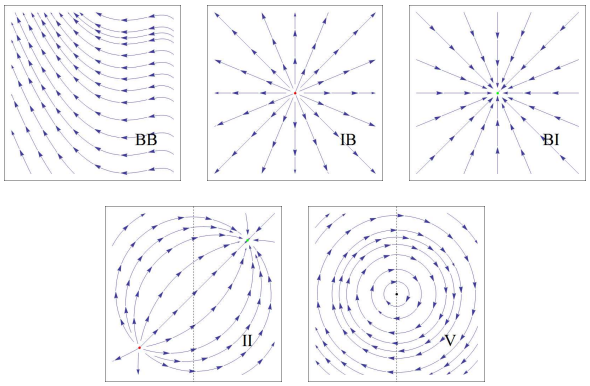


Figure 1: Classification of streamlines with respect to their start and end points lying on the boundary (B) or in the interior (I) of the domain. Case V describes closed streamlines around vortex. By proper splitting (dashed lines), II and V regions can be reduced to two subregions of BB, BI, or IB.

the stream function $f(\mathbf{x})$. The collection of the gridded values t_i determines another scalar field $t(\mathbf{x})$ called the *advection-time function*. In both approaches t stands for an artificial time.

Usually, both the coordinates of the footprints $\mathbf{g}_i(t_i)$ and the advection times t_i are recorded after the backwards tracking step. The user chooses a proper parametrization of ∂D_{in} and specifies an iso-value f_{iso} to extract the implicit stream surface $f(\mathbf{x}) = f_{\text{iso}}$. The advection times t_i can be used either to extract time surfaces $t(\mathbf{x}) = t_{\text{iso}}$ or to color extracted stream surfaces.

A dual technique is to track particles forward in the flow until they reach ∂D_{out} and to record their tracking time. Since we use both of the methods simultaneously in our approach, we denote by t_{in} and t_{out} the advection times by inverse and original flow, respectively.

In most cases, finding a parametrization that results in a globally smooth stream function is not easy for two reasons: First, the domain D is usually chosen to be a rectangular box, which, obviously, has a non-smooth boundary. Second, many flows have streamlines, which do not start on the boundary. We reproduce the flow diagrams from [vW93] in Figure 1. Based on whether a streamline starts/ends on the boundary (B) or in the interior (I), or it forms a loop around a vortex (V), one can classify them in five types: BB, BI, IB, II, and V.

The methods described above require that all streamlines of the flow $\mathbf{u}(\mathbf{x})$ start and/or end at the domain boundary ∂D. However, the presence of sources, sinks, or vortices may lead to stream curves belonging to the domain interior (cases II and V) which remain non-parametrized. These cases can be solved by a proper splitting of domain D into subdomains, see Figure 1, and/or by surrounding singularities with *termination surfaces*.

4 TERMINATION SURFACES

Parametrization of streamlines of type II and V was stated as an open problem by van Wijk [vW93]. Splitting of the flow domain as in Figure 1 (lower row) becomes impractical for three-dimensional fields, since critical points (sinks, sources, vortex cores) can build complicated geometry, e.g. vortex filaments. To handle the II-case with isolated sinks/sources, Xue et al. [XZC04] constructed termination surfaces surrounding the critical points. The streamlines approaching one of these points intersect the corresponding termination surface and pick up a value from its parametrization. However, Xue et al. did not present a methodology on how the radius of the spherical surface should be chosen and left the placement of termination surfaces to the user. Moreover, the streamline density on small spheres is extremely high, which makes the parametrization process unstable with respect to unavoidable tracking errors. Our approach automatically creates termination surfaces inside II or V regions optimally placed with respect to the locations of critical points, which is based on pre-processed information.

In the pre-processing step, we track each grid node forward and backward in the flow to define its type: The type of a node is the type of the streamline the node belongs to. For the nodes of types BB, BI, and IB we record the footprint point(s) and the two advection times. For the nodes of type II we record the grid voxels being visited, the tracking times t_{in} and t_{out} and the advection lengths l_{in} and l_{out}. For the nodes of type V we just record the voxels being visited. As such, we classify all grid nodes. Setting value 1 to all nodes of one class and value 0 to nodes of the other classes, we can extract *separating surfaces* as isosurfaces with respect to the isovalue 0.5. These are stream surfaces that provide important information about the flow structure. However, their quality is low, since they are extracted from a boolean field.

Flow regions that have been categorized as being connected to the domain boundary (types BB, BI, and IB) are then parametrized according to scalar field(s) that are assigned to the domain boundary ∂D. The next step is to create a smooth scalar field for regions of type II and V by constructing proper termination surface(s). For that purpose we first look for a *seeding voxel* S (discussed below). Let $\mathbf{c}$ be the center of S and $\mathbf{m} = \mathbf{v}(\mathbf{c})$ the velocity at $\mathbf{c}$. Starting from the seeding voxel, we grow the termination surface by marking neighboring voxels if they (a) have a non-empty intersection with the plane $\beta : \mathbf{x} \cdot \mathbf{m} = \mathbf{c} \cdot \mathbf{m}$, (b) have unparametrized streamlines crossing them, and (c) their velocity $\mathbf{v}$ has the same orientation as the velocity at $\mathbf{c}$, i.e., $\mathbf{v} \cdot \mathbf{m} > 0$. The procedure results in that part of plane β that is connected with voxel S and has unparametrized streamlines crossing it. Let $\{\mathbf{k}, \mathbf{l}\}$ be an orthonormal basis in plane β.

ISBN 978-80-86943-40-4

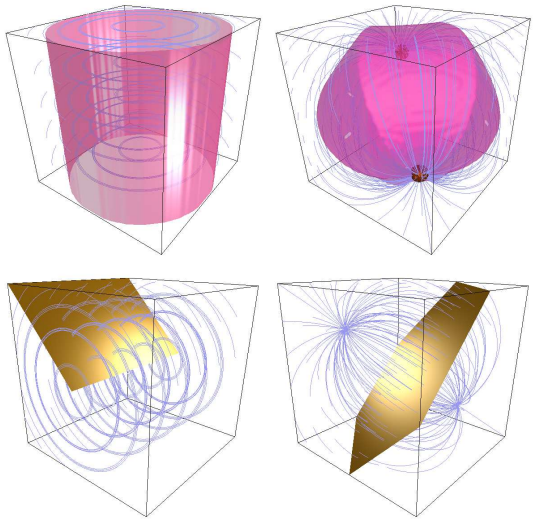


Figure 2: Automatic parametrization of V- and II-regions (left and right column, correspondingly). Upper row: Stream surfaces that separate the V- and II-region from the surrounding regions. Lower row: Termination surfaces provide scalar values for streamlines intersecting them, which allows for the extraction of stream elements.

A streamline crossing the termination surface at point $\mathbf{g}$ gets assigned a scalar according to $f_1 = (\mathbf{g}-\mathbf{c})\cdot\mathbf{k}$ or $f_2 = (\mathbf{g}-\mathbf{c})\cdot\mathbf{l}$. Both scalars are needed for extracting stream tubes and ribbons as discussed below.

For the selection of seeding voxel S, our aims are (1) to provide scalar values for a maximal number of streamlines at once, and (2) possibly avoid an overly dense local concentration of streamlines on the surface. In other words, we want to parametrize the largest part of the domain and make our parametrization less sensitive to computational errors. Several approaches to choose the seeding voxel were tested in our experiments. We came to the conclusion that for II-region with single source and sink the seeding voxel S should lie half way between the sink and the source on the shortest connecting streamline. Thus, S should contain the grid node with minimal total tracking length $(l_{\text{in}}+l_{\text{out}})$ and minimal tracking length difference $|l_{\text{in}}-l_{\text{out}}|$. In a V-region, on the other hand, tracking time for the streamlines has no meaning , since the streamlines are closed. Thus, the choice of S is arbitrary. Generally, one can find the largest termination surface with the maximal number of streamline crossing it by a brute force algorithm testing all possible seed points. Results are shown in Figure 2.

For each streamline we record a label of the termination surface from which it received the scalar values. If not all streamlines were parametrized, we iteratively build further termination surfaces until all streamlines are parametrized.

5 STREAM FUNCTION CONTROL AND IMPROVEMENT

Streamline tracking introduces numerical errors due to imprecise velocity interpolation and integration. The longer a streamline, the larger the error. To our knowledge, there exists no effective procedure to improve a constructed stream function $f(\mathbf{x})$ other than to reconstruct it again using a smaller integration step size, which is an extremely time-consuming process. Our goal is to develop a method to control and improve the quality of a stream function.

5.1 Functional for measuring stream function quality

We start with a construction of a functional measuring the quality of a stream function $f(\mathbf{x})$ with respect to the underlying normalized vector field $\mathbf{v}(\mathbf{x})$. The fundamental characteristic of a stream function is that its isolines (surfaces) are tangent to the flow direction. Therefore, we define

$$E_1(f) = \frac{1}{2}\int_{D'} \left| \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}\cdot\mathbf{v}(\mathbf{x})\right|^2 \mathrm{d}\mathbf{x}. \qquad (2)$$

Here and in the following D' denotes a subregion in D covered by streamlines of the same type. Streamlines within D' either have a common termination surface or start or end at the boundary $\partial D'$. Obviously, the functional takes values from interval $[0,1]$ and vanishes for a perfect stream function. Our goal is to obtain a method, which allows us to minimize E_1 for a given approximation of $f(\mathbf{x})$.

5.2 Minimization algorithm

A standard technique to minimize a functional of the form $E(\phi) = \int L(\mathbf{x},\phi,\nabla\phi)\,\mathrm{d}\mathbf{x}$ is to construct its Euler-Lagrange equation

$$\frac{\partial L}{\partial \phi} - \mathrm{div}_{\mathbf{x}}\left[\frac{\partial L}{\partial \nabla\phi}\right] = 0. \qquad (3)$$

Equation (3) expresses the necessary condition for a stationary point ϕ_0 of the functional and can be derived by usual differentiation of $E(\phi) = E(\phi_0+\varepsilon\psi)$ with respect to ε.

To simplify the resulting equation, we omit the normalization of the gradient field in Equation (2). Our tests show that this modification reduces the computational costs and still serves the goal of minimization of $E_1(f)$. The simplified functional depends only on the gradient of the function $f(\mathbf{x})$, thus the associated Euler-Lagrange equation reduces to the form

$$-\mathrm{div}_{\mathbf{x}}\left[\mathbf{v}(\mathbf{x})\left(\nabla f(\mathbf{x})\cdot\mathbf{v}(\mathbf{x})\right)\right] = 0. \qquad (4)$$

ISBN 978-80-86943-40-4

We introduce an artificial time τ and set the partial temporal derivative of $f(\mathbf{x},\tau)$ to the left-hand side of Equation (4) taken with a negative sign. The resulting evolutional equation

$$\frac{\partial f(\mathbf{x},\tau)}{\partial \tau} = \mathrm{div}_{\mathbf{x}}\left[\mathbf{v}(\mathbf{x})\left(\nabla f(\mathbf{x},\tau)\cdot\mathbf{v}(\mathbf{x})\right)\right], \quad (5)$$

$$f(\mathbf{x},0) = f_0(\mathbf{x}), \quad (6)$$

describes a transformation of an initial approximated stream function $f_0(\mathbf{x})$ towards a local minimum of functional E_1. The algorithm is similar to the steepest descent method for root search, where the divergence term stands for the opposite gradient direction. The governing equation has the form of diffusion in the direction of $\mathbf{v}(\mathbf{x})$ with the diffusion rate $\nabla f(\mathbf{x},\tau)\cdot\mathbf{v}(\mathbf{x})$. Clearly, the diffusion rate vanishes for the perfect stream function. Thus, the perfect stream function is a stationary point of the evolution process.

We discretize Equation (5) in space and time to derive a numerical scheme. In our tests we use central differencing for spatial and forward differencing for temporal discretization resulting in an explicit scheme with second-order accuracy in space. The discretized partial differential equation has the form

$$\frac{f^{n+1}_{i,j,k} - f^{n}_{i,j,k}}{\delta\tau} = \mathrm{div}_{i,j,k}\left[\mathbf{v}^{n}_{i,j,k}\left(\nabla_{i,j,k} f^{n}\cdot\mathbf{v}^{n}_{i,j,k}\right)\right], \quad (7)$$

where gradient $\nabla_{i,j,k}$ and divergence operator $\mathrm{div}_{i,j,k}$ are discretized using central differences, $\delta\tau$ is the time step, the upper indices denote the time, and the lower indices indicate the position in space.

Equation (5) is a parabolic partial differential equation. Thus, both initial and boundary conditions are required for the well-posedness of the problem. The initial condition is given by Equation (6). Imposing a proper boundary condition is not a trivial task, since numerical instabilities can develop close to the boundary $\partial D'$ of the considered region.

The simplest and the safest approach is to fix the values of the stream function at $\partial D'$ for all τ by imposing the Dirichlet boundary condition: $f(\mathbf{y},\tau) = f_0(\mathbf{y})$ for all $\mathbf{y}\in\partial D'$ and all $\tau \geq 0$. The numerical scheme becomes simple and the functional decreases over the first iterations. Moreover, the initial parametrization of the boundary is not affected.

5.3 Application

The minimization procedure described above can be applied to an already generated stream function to make its level sets be better aligned to the given vector field. Since the governing Equation (5) models a diffusion process, the procedure also has a smoothing effect. Van Wijk [vW93] applied an isotropic smoothing filter to the generated stream function to enhance its rendering quality at the cost of losing detailed information. In the proposed method, the smoothing is performed in accordance with the underlying flow field decreasing the error defined in Equation (2).

Another main application of the minimization algorithm can be the reduction of computation time in the pre-processing stage. Accurate advection of all grid nodes can take hours for large data sets. Even if the preprocessing has to be performed only once, the computational efforts are an issue. We propose to construct a rough approximation to the stream function which subsequently can be improved by applying our minimization procedure. The steps of the algorithm are the following:

1 We perform an advection of three subsets of nodes: (a) the boundary nodes $\mathbf{g}_i \in \partial D'$, (b) nodes having vorticity or absolute divergence values larger than specified thresholds, and (c) an evenly distributed sparse subset of nodes in D'.

2 The advected nodes are parametrized according to their footprints at the boundary.

3 The scalar field sampled at the parametrized nodes is interpolated linearly to the nodes which were not tracked producing a rough approximation to a globally defined stream function.

4 The approximate stream function is improved according to Equation (7). The values at the advected nodes (from Step 1) remain unchanged during this optimization.

5 The minimization process is stopped as soon as the error (2) reaches its minimum.

The vorticity used in Step 1 are given by norm of $\nabla\times\mathbf{v}(x,y,z) = \left(\frac{\partial \mathbf{v}_y}{\partial z} - \frac{\partial \mathbf{v}_z}{\partial y}, \frac{\partial \mathbf{v}_x}{\partial z} - \frac{\partial \mathbf{v}_z}{\partial x}, \frac{\partial \mathbf{v}_y}{\partial x} - \frac{\partial \mathbf{v}_x}{\partial y}\right)$, where derivatives are computed by central differencing. High vorticity values indicate that locally the stream function is highly curved. In the neighborhood of large absolute values of divergence, the norm of the gradient of the stream function can grow quickly. To avoid possible instabilities when evolving $f(\mathbf{x})$ according to Equation (7), we explicitly advect and parametrize grid nodes in regions of high vorticity and high absolute divergence values. Analogously, we can parametrize the streamlines crossing a termination surface.

6 EXTRACTION OF STREAM ELEMENTS

Stream elements are the instruments of geometric flow visualization methods. The most commonly used elements are stream surfaces, streamlines, stream tubes, and stream ribbons. Different stream elements serve for an adequate exploration of different flow characteristics and structures. Since the flow through any stream

 ISBN 978-80-86943-40-4

surface vanishes, i.e., $\mathbf{v} \cdot \mathbf{m} = 0$ with surface normal $\mathbf{m}$, these surfaces can be widely used to identify and separate different regions of flow. However, displaying several stream surfaces usually leads to occlusion. One can easily show a direction and magnitude of the local flow using colored stream elements. Stream tubes and ribbons are the proper tools to reflect divergence and torsion of the field, correspondingly. Combination of these basic elements can be applied to provide a versatile picture of the flow. Given the derived implicit flow representation, stream elements can be directly extracted from these scalar fields.

All points satisfying the relation $f(\mathbf{x}) = f_{\text{iso}}$ for arbitrary $f_{\text{iso}} \in \mathbb{R}$ define a stream surface of the flow $\mathbf{v}(\mathbf{x})$. We use standard marching technique to derive a triangulated representation of stream surfaces.

It is well-known that an intersection line of two non-parallel stream surfaces is a streamline [KM92]. Different parametrizations of the boundary (or termination surface) lead to different stream functions for the same flow. Given two stream functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ with the property $\nabla f_1 \cdot \nabla f_2 \neq 0$ in D, a set of streamlines can be obtained by intersection of isosurfaces $f_1(\mathbf{x}) = c_1$ and $f_2(\mathbf{x}) = c_2$ for various constants c_1 and c_2. Therefore, each streamline is uniquely defined by two stream coordinates $\mathbf{c} = (c_1, c_2)$. However, an explicit integration of a single streamline is much easier. This observation changes as soon as one is interested in extracting certain sets of streamlines.

Usually, a stream tube is generated as a collection of streamlines with seeding points lying on an ellipse. An alternative construction of a stream tube can be obtained by generating a proper stream function. Let $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ be two stream functions. It is easy to show that any smooth function $h(f_1(\mathbf{x}), f_2(\mathbf{x}))$ is also a stream function: $\nabla h(\mathbf{x}) \cdot \mathbf{v}(\mathbf{x}) = 0$ [vW93]. Let us assume that isosurfaces $f_1(\mathbf{x}) = c_1$ and $f_2(\mathbf{x}) = c_2$ are orthogonal in a neighborhood of the termination surface: $\nabla f_1(\mathbf{x}) \cdot \nabla f_2(\mathbf{x}) = 0$. Then, the stream surface $h(\mathbf{x}) = 1$ for the function

$$h(\mathbf{x}) = \frac{(f_1(\mathbf{x}) - c_1)^2}{a^2} + \frac{(f_2(\mathbf{x}) - c_2)^2}{b^2}$$

is the desired stream tube with radii a and b.

Similar to the stream tube construction, there are also two methods for extracting stream ribbons. One can seed a set of streamlines along a line segment of interest or one can extract a part of the stream surface $f_1(\mathbf{x}) = f_{\text{iso}}$ satisfying the condition $a \leq f_2(\mathbf{x}) \leq b$. In the latter case, we construct the stream surface with respect to the field $f_1(\mathbf{x})$ by means of a marching algorithm. For each triangle from the derived surface representation we compute values of $f_2(\mathbf{x})$ on its vertices. If all three values are in the range $[a, b]$, the triangle will be accepted; if none of the values belong to the interval,

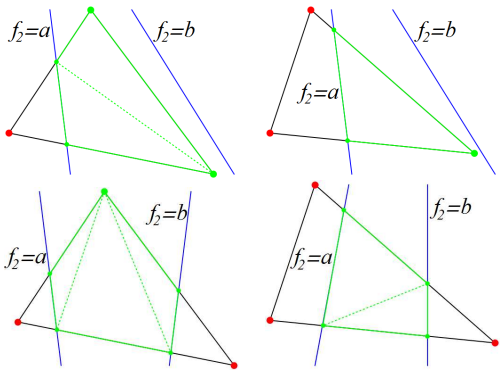


Figure 3: Extraction of stream ribbon $f_1(\mathbf{x}) = f_{\text{iso}}$, $a \leq f_2(\mathbf{x}) \leq b$. A marching algorithm produces a triangulation of the stream surface $f(\mathbf{x}) = f_{\text{iso}}$. These triangles are then rejected, accepted, or accepted with modification based on the values of function f_2 at their vertices. If a triangle intersects the ribbon boundary, it is trimmed producing up to 3 new triangles.

we reject the triangle; if some of the values are in the range, the triangle is trimmed producing up to 3 new triangles. All possible trimming scenarios are shown in Figure 3.

The advection-time field $t(\mathbf{x})$ is also available after the pre-processing step. Its isosurfaces — time surfaces — can be extracted in the same manner as stream surfaces. The advection time information can be encoded on the surface of stream element using color or transparency. Besides that, stream functions and advection-time field can be combined to extract flow volumes. A flow volume is a part of flow domain bounded by surface $S(f_1(\mathbf{x}), f_2(\mathbf{x}), t(\mathbf{x})) = 1$. In practice, we use flow tube and flow cube given by expressions

$$S_{\text{tube}} = \max\left\{ \frac{(f_1(\mathbf{x}) - c_1)^2}{a^2} + \frac{(f_2(\mathbf{x}) - c_2)^2}{b^2}, \frac{|t(\mathbf{x}) - t_0|}{r_t} \right\}$$

$$S_{\text{cube}} = \max\left\{ \frac{|f_1(\mathbf{x}) - c_1|}{r_1}, \frac{|f_2(\mathbf{x}) - c_2|}{r_2}, \frac{|t(\mathbf{x}) - t_0|}{r_t} \right\}.$$

Flow volumes have a meaningful interpretation for steady flows: They define the fluid portion that crosses the boundary at the specified location during the given time interval. For example, the fluid inside flow tube S_{tube} will flow through the elliptical part of the boundary withing time from $t_0 - r_t$ to $t_0 + r_t$, i.e., it will traverse the tube from one end to the other.

7 NUMERICAL EXPERIMENTS

All numerical tests presented in this and the following section were performed on a PC with an Intel Xeon 3.20GHz processor. For surface extraction, a marching cubes algorithm was used. Extraction of any stream element for any examples presented here took only a fraction of a second. Thus, the user experiences a highly interactive system for extracting stream elements. In all

ISBN 978-80-86943-40-4

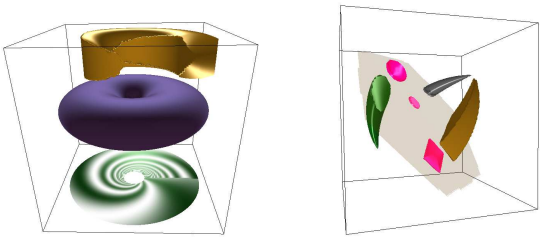


Figure 4: Extraction of flow volumes. *Left:* Type V region. Restricting a stream tube (purple) to a finite time-advection interval results in a flow volume (gold). To obtain an information about velocity magnitude, the sinus of the advection time is mapped to transparency of a stream surface (green). *Right:* Type II region. Three flow volumes together with their footprints (red) on the termination surface are shown.

our tests, we used trilinear interpolation of the velocity field and a fourth-order Runge-Kutta method for integration.

First, we looked into simple synthetic data sets. The first example is that of flow around a vertex line, which we sampled at 100^3 regularly distributed grid nodes. The flow is divided in two subdomains of type BB and V. To parametrize the latter, we construct a termination surface as shown in Figure 2 (left). Several extracted stream elements are shown in Figure 5 (left). Information about velocity magnitude can be obtained by analyzing the shape of flow volumes or by rendering of advection-time values on stream elements. In Figure 4 (left) transparency of the lower stream surface shows the sine of the advection time. Curved patterns show that the magnitude of velocity increases superlinearly with the distance to the vortex line. The same conclusion can be drawn when looking to the shape of the flow tube shown in gold.

A second example is that of flow from a single source to a simple sink. This flow field includes a subdomain of type II. It is parametrized as shown in Figure 2 (right). Extracted stream elements are shown in Figure 5 (center). Three flow volumes and their footprints on the termination surface are shown in Figure 4 (right).

Next, we demonstrate the speed-up of the preprocessing step when applying the minimization procedure presented in Section 5. The tornado dataset [CM93] was sampled on a uniform grid of resolution 50^3 and 128^3. After computing vorticity at all nodes, we set its threshold to 0.15. Then, we track those grid nodes, which belong to the domain boundary, have vorticity values larger than the threshold, or have an even grid index. The tracked nodes get scalar values equal to the z-coordinate of their footprint at the boundary. The resulting sparse scalar field is linearly interpolated to the rest of the nodes. Finally, we perform several iterations to minimize the stream function error.

The time spent at each step of the algorithm for both datasets is summarized in Table 1. When compared to tracking all nodes, we observe that our algorithm requires only 22% and 16% of the tracking time for the data sets with 50^3 and 128^3 nodes, correspondingly. The evolution of the average error during the minimization step is presented in Figure 7. Only few iterations with the artificial time step $\delta\tau = 2.0$ were enough to reduce the error to values that are even below the error one obtains when tracking all nodes. A result for extracted stream elements from this data set can be seen in Figure 5 (right). Areas of high vorticity are shown in Figure 6(left), while Figure 6(right) shows the error on a stream surface.

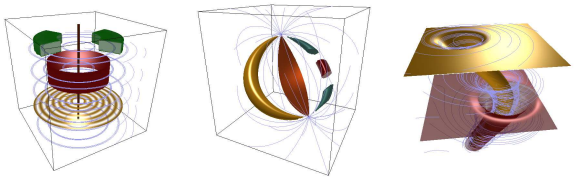


Figure 5: Flow representation of several data sets: flow around a vortex line (*left*), flow between a sink and a source (*center*), and tornado data set (*right*). A set of streamlines together with various stream elements are shown for each data set. The geometric features are extracted interactively from an implicit flow representation.

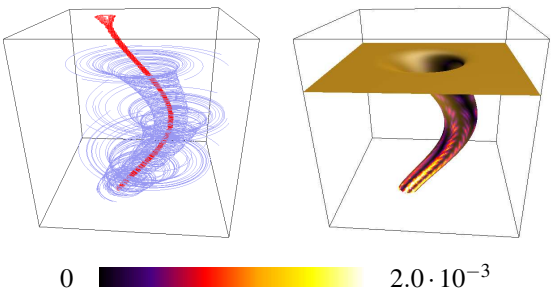


Figure 6: Left: Streamlines computed for tornado dataset. Red are the grid nodes which have vorticity values larger than threshold. A stream surface close to these nodes has high curvature that makes the error minimization procedure unstable in this region. Right: Error visualization on a stream surface. The error increases when the surface approaches the tornado center (red-yellow-white spots) and vanishes at larger distances (black-blue spots). Areas with negligible error remain gold.

Finally, we construct an implicit representation for a simulation of the flow of five jets (dataset courtesy of Kwan-Liu Ma, University of California, Davis). Figure 8 shows a set of streamlines, a constructed termination surface and some extracted stream elements.

ISBN 978-80-86943-40-4

	50^3 nodes	128^3 nodes
tracking time	177.53 s	4631 s
interpolation time	0.15 s	2.07 s
error minimization time	9.19 s	114.62 s
number of iterations	70	50
total time	186.87 s	4797.69 s
final error	$6.278 \cdot 10^{-4}$	$3.343 \cdot 10^{-4}$
time (tracking all nodes)	861 s	30625 s
error (tracking all nodes)	$7.475 \cdot 10^{-4}$	$6.456 \cdot 10^{-4}$

Table 1: Time consumption for different stages of our algorithm in Section 5 for the construction of an implicit flow representation when applied to the tornado dataset sampled at 50^3 and 128^3 nodes. The results show significant reduction of time when compared to the approach of tracking all nodes. Moreover, although we are tracking significantly less nodes, the average error decreases with our approach.

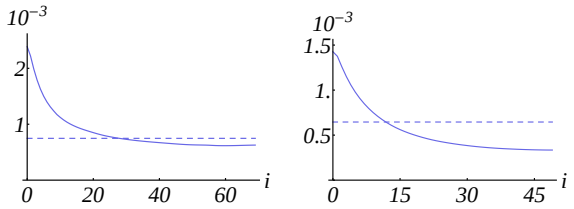


Figure 7: Evolution of average error during the minimization procedure applied to the interpolated data (solid lines). Dashed lines show the error values after tracking all grid nodes. Tornado dataset with 50^3 nodes (left) and 128^3 nodes (right) was tested.

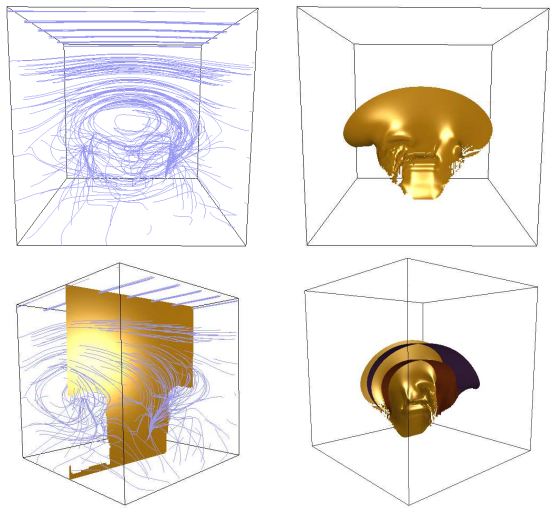


Figure 8: Five jets dataset. Streamlines and a termination surface are shown in the upper row. Extracted stream surface in combination with two stream ribbons is shown in the lower row.

8 CONCLUSION

We presented a method for automatic generation of implicit representation for volumetric flow. The method is based on the classification of streamlines in five types: BB, BI, IB, II, and V depending on whether they start/end on the boundary or in the domain interia, or form a closed trajectory. For the first three cases known techniques as in [vW93, XZC04] are applicable. We focused our efforts on effectively defining a stream function for regions of type II and V. To handle those, a termination surface is created starting with a proper seeding voxel. Two strategies for choosing seeding voxels are proposed: Maximization of number of unparametrized streamlines passing through the voxel (suitable for type V), and comparing advection-time values recorded in the pre-processing step (suitable for type II). Thus, some open problems concerning the construction of a stream function in the entire flow region have been solved. We have avoided any artificial splitting of the domain. Instead, the established subregions reflect the flow topology; they are separated from each other by special stream surfaces. We have also avoided termination surfaces isolating critical points, since it could lead to inaccurate parametrization due to the high density of the streamlines on such surfaces.

We also proposed a tool for improving of stream functions. It is based on variational minimization of a functional describing the function quality with respect to the underlying vector field. The governing diffusion equation is derived.

Various geometrical stream elements (e.g., stream surfaces) can be extracted and displayed interactively. In particular, we proposed novel algorithms for the extraction of stream tubes and ribbons. We also combined the stream function visualization with a visualization of the advection-time field. Both tracking the nodes in the pre-processing step and extraction of stream elements in run time allow for an efficient parallel computing.

Acknowledgments This work was supported in part by DFG grants LI 1530/6-2 and MO 3050/2-1.

9 REFERENCES

[CKSW08] Nicolas Cuntz, Andreas Kolb, Robert Strzodka, and Daniel Weiskopf. Particle level set advection for the interactive visualization of unsteady 3D flow. *Computer Graphics Forum*, 27(3):719–726, May 2008.

[CL93] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 263–270, New York, NY, USA, 1993. ACM.

[CM93] Roger Crawfis and Nelson Max. Texture splats for 3D vector and scalar field visualization. In *Proceedings Visualization '93*, pages 261–266, Los Alamitos, Oct 1993. IEEE Computer Society.

ISBN 978-80-86943-40-4

[GRT17] T. Gerrits, C. Rössl, and H. Theisel. Glyphs for space-time jacobians of time-dependent vector fields. *Journal of WSCG*, 25(1):31–38, 2017.

[Hul92] J. P. M. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *VIS '92: Proceedings of the 3rd conference on Visualization '92*, pages 171–178, Los Alamitos, CA, USA, 1992. IEEE Computer Society.

[KM92] David N. Kenwright and Gordon D. Mallinson. A 3-D streamline tracking algorithm using dual stream functions. In *VIS '92: Proceedings of the 3rd conference on Visualization '92*, pages 62–68, Los Alamitos, CA, USA, 1992. IEEE Computer Society.

[KM96] David Knight and Gordon Mallinson. Visualizing unstructured flow data using dual stream functions. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):355–363, 1996.

[LHD+03] Robert S. Laramee, Helwig Hauser, Helmut Doleisch, Benjamin Vrolijk, Frits H. Post, and Daniel Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum*, 23(2):203–221, 2003.

[LTH08] Guo-Shi Li, Xavier Tricoche, and Charles D. Hansen. Physically-based dye advection for flow visualization. *Comput. Graph. Forum*, 27(3):727–734, 2008.

[MBS+04] Karim Mahrous, Janine Bennett, Gerik Scheuermann, Bernd Hamann, and Kenneth I. Joy. Topological segmentation in three-dimensional vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 10:198–205, 2004.

[PBL+04] Sung W. Park, Brian Budge, Lars Linsen, Bernd Hamann, and Kenneth I. Joy. Multi-dimensional transfer functions for interactive 3d flow visualization. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 177–185, Washington, DC, USA, 2004. IEEE Computer Society.

[PBL+05] Sung W. Park, Brian Budge, Lars Linsen, Bernd Hamann, and Kenneth I. Joy. Dense geometric flow visualization. In *EUROGRAPHICS - IEEE VGTC Symposium on Visualization*, pages 21–28, 2005.

[PS09] Ronald Peikert and Filip Sadlo. Topologically Relevant Stream Surfaces for Flow Visualization. In H. Hauser, editor, *Proc. Spring Conference on Computer Graphics*, pages 43–50, April 2009.

[PYH+06] Sung W. Park, Hongfeng Yu, Ingrid Hotz, Oliver Kreylos, Lars Linsen, and Bernd Hamann. Structure-accentuating dense flow visualization. In Beatriz Sousa Santos, Thomas Ertl, and Kenneth I. Joy, editors, *EuroVis06: Joint Eurographics - IEEE VGTC Symposium on Visualization, Lisbon, Portugal, 8-10 May 2006*, pages 163–170. Eurographics Association, 2006.

[RLN+17] Dylan Rees, Robert S. Laramee, Duong Nguyen, Lei Zhang, Guoning Chen, Harry Yeh, and Eugene Zhang. A Stream Ribbon Seeding Strategy. In *EuroVis 2017 - Short Papers*. The Eurographics Association, 2017.

[SHJK00] Gerik Scheuermann, Bernd Hamann, Kenneth I. Joy, and Wolfgang Kollmann. Visualizing local vector field topology. *SPIE Journal of Electronic Imaging*, 9:367, 2000.

[SRP09] Filip Sadlo, Alessandro Rigazzi, and Ronald Peikert. Time-Dependent Visualization of Lagrangian Coherent Structures by Grid Advection. In *Proceedings of TopoInVis 2009*. Springer, 2009.

[SS07] Tobias Salzbrunn and Gerik Scheuermann. Streamline predicates as flow topology generalization. In Helwig Hauser, Hans Hagen, and Holger Theisel, editors, *Topology-Based Methods in Visualization (Mathematics and Visualization)*, pages 65–78. Springer, July 2007.

[STWE07] Tobias Schafhitzel, Eduardo Tejada, Daniel Weiskopf, and Thomas Ertl. Point-based stream surfaces and path surfaces. In *Graphics Interface*, pages 289–296, 2007.

[vFWTS08] Wolfram von Funck, Tino Weinkauf, Holger Theisel, and Hans-Peter Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization)*, 14(6):1396–1403, Nov 2008.

[vW93] Jarke J. van Wijk. Implicit stream surfaces. In *VIS '93: Proceedings of the 4th conference on Visualization '93*, pages 245–252, Washington, DC, USA, 1993. IEEE Computer Society.

[WJE00] Rüdiger Westermann, Christopher Johnson, and Thomas Ertl. A level-set method for flow visualization. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 147–154, Los Alamitos, CA, USA, 2000. IEEE Computer Society.

[WJE01] Rüdiger Westermann, Christopher Johnson, and Thomas Ertl. Topology-preserving smoothing of vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):222–229, 2001.

[WT10] Tino Weinkauf and Holger Theisel. Streak lines as tangent curves of a derived vector field. *IEEE Transactions on Visualization and Computer Graphics*, 16:1225–1234, 2010.

[XZC04] Daqing Xue, Caixia Zhang, and Roger Crawfis. Rendering implicit flow volumes. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 99–106, Washington, DC, USA, 2004. IEEE Computer Society.

 ISBN 978-80-86943-40-4

ISSN 2464-4617(print)
ISSN 2464-4625(CD)
Computer Science Research Notes
CSRN 2801
Full Papers Proceedings
http://www.WSCG.eu

Calibrating Low-cost Structured-light 3D Sensors

R. Chakib
Université de Limoges
XLIM / ASALI
123 Av. Albert Thomas
87000 Limoges, France

CORUO
reda.chakib@etu.unilim.fr

N. Mérillou
Université de Limoges
XLIM / ASALI
123 Av. Albert Thomas
87000 Limoges, France
nicolas.merillou@unilim.fr

P.-J. Vincent
CORUO
46 Av. des Bénédictins
87000 Limoges, France
pierre-jean.vincent@coruo.com

S. Mérillou
Université de Limoges
XLIM / ASALI
123 Av. Albert Thomas
87000 Limoges, France
stephane.merillou@unilim.fr

ABSTRACT

Consumer-grade RGB-D cameras are widely accessible, but they suffer from a lack of accuracy when compared to professional-grade 3D scanning solutions. In this paper, we propose a new method for calibrating an Intel RealSense SR300 camera, adaptable to other structured light sensors. The method uses classical checkerboard calibration and a coordinate-measuring machine (CMM) based setup with a calibrating plane. It delivers better results than the manufacturers settings.

Keywords

Camera calibration, RGB-D camera, coordinate-measuring machine, pinhole model, intrinsic calibration.

1 INTRODUCTION

Despite being widely accessible and user-friendly, low-cost structured light cameras suffer from a major problem related to their accuracy. The manufacturers generally use proprietary calibration methods with their devices, which leads to semi-closed technologies. Therefore, experienced end-users cannot benefit from the full potential of their sensors. A proper calibration may lead to a better precision when compared with the factory default settings.

The introduction of the Microsoft Kinect was the beginning of the era of consumer grade RGB-D cameras. Then the Intel RealSense sensors line introduced efficient, compact and easily embeddable devices. We chose to work with the Intel RealSense SR300, which covers short-range areas. This camera contains a color sensor, an IR sensor and an IR projector for depth measurement. The onboard imaging chip processes the depth computation [Int16].

In use, the RealSense SR300 presents some inaccuracies, for example when capturing a flat wall, the point cloud is warped at the corners, see Fig. 1. The IR sensor also suffers from distortion at the edges of the IR frames as shown in Fig. 2.

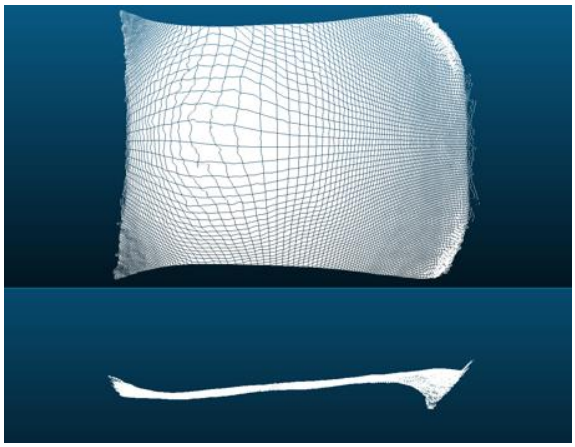


Figure 1. Point cloud of a flat surface captured using the SR300 with default settings.

This paper describes a new calibration method for the Intel RealSense SR300 with a twofold achievement:

- Improving the accuracy over the manufacturer's calibration;
- Providing a general-purpose calibration method that can be applied to similar devices;

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.4

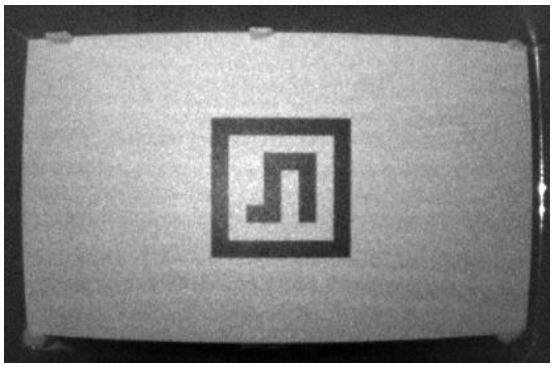


Figure 2. Distortion in the SR300 IR. The panel with the pattern is rectangular.

Our algorithm consists in two main steps:

- A classic checkerboard calibration or 2D calibration to correct the camera rays (IR camera).
- A depth correction performed using a Coordinate-Measuring Machine (CMM) for high precision measurement.

The output is a calibration data file with the camera parameters and a 3D grid of correction coefficients covering the calibration domain in the view frustum of the depth camera.

This paper is organized as follows. Section 2 gives a brief introduction of the camera's intrinsic parameters, then it presents related works about RGB-D cameras calibration. Section 3 presents our method and provides all the details on the hardware setup. Section 4 contains some experimental results along with a validation approach for our method. Finally, Section 5 is a discussion/conclusion on our work.

2 BACKGROUND AND RELATED WORK

Camera calibration is the process of mathematically describing how 3D spatial points project into the camera image sensor. That is, a mathematical model of the camera is required for calibration. We use the pinhole camera model for the camera's parameters description.

2.1 Camera's Intrinsic Parameters

The pinhole camera model describes the projection of 3D world points into the camera's (2D) image plane.

Let us consider a point $M_c = [x_c, y_c, z_c]^T$ in the camera frame. We want to express the projection of M_c using image coordinates which we denote $P_c = [u_c, v_c]^T$ using the pinhole model.

First, we begin by normalizing the point M_n:

$$M_n = [x_n, y_n]^T = [x_c/z_c\,, y_c/z_c]^T.$$

In the pinhole camera model, the rays are considered to pass linearly through the optical center, which in the case of real cameras is not true. In fact, the use of lenses alters the linearity of the light rays which causes non-linear distortion on the final images.

Using the normalized point, the distortion is performed in two steps [HKH12]:

$$M_g = \begin{bmatrix} 2k_3 x_n y_n + k_4(r^2 + 2x_n^2) \\ k_3(r^2 + 2y_n^2) + 2k_4 x_n y_n \end{bmatrix}$$

$$M_k = (1 + k_1 r^2 + k_2 r^4 + k_5 r^6) M_n + M_g$$

where $r^2 = x_n^2 + y_n^2$ and $k_c = [k_1, \dots, k_5]$ is the vector of the distortion coefficients.

The point Pc that we are looking for is:

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} f_{cx} & 0 \\ 0 & f_{cy} \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} u_{0c} \\ u_{0c} \end{bmatrix}$$

The parameters $\{f_{cx}, f_{cy}, p_{0c}, k_1, k_2, k_3, k_4, k_5\}$ are called the *intrinsic parameters of the camera* where $\{f_{cx}, f_{cy}\}$ are the *focal lengths* and $p_{0c} = [u_{0c}, v_{0c}]$ is the camera *principal point*. Intrinsic calibration consists in finding these parameters. To do so, we should establish the correspondence between a set of 3D points and their projected 2D image points [Sem16].

Zhang [Zha04a] made the following classification for calibration techniques, based on the dimensionality of the calibration object:

1) 3D reference object-based calibration: the typical 3D calibration object is composed of two or three orthogonal planes [Hei00]. The geometry of the object should be known with high precision.

2) 2D plane-based calibration: consists in using a planar object such as a checkerboard or circular patterns printed on a panel captured from different point of views. Many resources are available on the subject [Zha00], [SM99].

3) 1D line-based calibration: first proposed by Zhang [Zha04b], it consists in observing a set of collinear points moving around a fixed point.

4) Self-calibration: or 0D calibration as referred to by Zhang [Zha04a] because no calibration object is required. The method consists in calibrating the camera form a sequence of images of a static scene, without any prior knowledge of the camera's motion [HZ05].

2.2 Depth Cameras Calibration

Although built around the Kinect v1 sensor, most of the methods that we cite are supposed to be compatible with a wide range of low cost structured light cameras according to their respective authors. When the calibration object is known (shape, color, size), the calibration method is said to be *supervised*. Otherwise, the method is called *unsupervised*.

 ISBN 978-80-86943-40-4

Smisek *et al*. [SJP11] proposed a geometrical model for the Kinect v1 and estimated the intrinsic parameters of both the IR and RGB cameras as well as their relative pose. They also estimated internal parameters of the depth camera. They used a checkerboard as the calibration target of both the RGB and IR cameras of the Kinect.

Herrera *et al*. [HKH12] have used a high-resolution color camera rigidly attached to the Kinect to compensate for the Kinect lower resolution color sensor. The calibration target is a planar board where a checkerboard is printed or stuck. In addition to the intrinsic parameters and the relative pose, the authors estimated the depth camera intrinsics.

Jin et al. [JLG14] have performed an intrinsic calibration of a Kinect unit, using a set of well-manufactured cuboids as their calibration target. Their objective function is a linear combination of the distance and angle errors from the cuboid. They re-wrote the objective function in terms of the intrinsic parameters of the camera prior to the optimization step.

Staranowicz et al. [SBMM15] have used a video of a spherical object moving in front the camera as input to their method. After a robust feature-extraction process, their algorithm infers an initial estimation of the depth, as well as the other calibration parameters, and then it performs a refinement estimate of the different parameters.

3 CALIBRATION METHOD

Our technique works as follow. First, the camera's intrinsic parameters are computed via a classical checkerboard approach, to correct the *x* and *y* coordinates. Then, the sensor is mounted on a CMM in front of the measure plane. Successive captures of the plane are acquired while moving towards it by using the corrected model from the first step. Then, we compute a 3D grid of correction coefficients that we infer from the collected data (plane's captures).

We could have dropped the checkerboard step, and instead rotated the plane by 45 degrees at each of its axis, but the errors in each direction would mix up. An alternative would be also to drop the checkerboard calibration, and to capture a calibrating sphere at different positions, then compute the errors, but we would be using inaccurate captures as we rely on the manufacturer's calibration.

3.1 2D Calibration

As previously said, to get more accurate camera's intrinsic parameters (i.e. in order to remove the distortion shown in Fig. 2) we use a classical checkerboard calibration. We photograph a checkerboard from different viewpoints using the camera, and simply use OpenCV calibration module to compute the camera's parameters, in our case we are interested in the intrinsic values of the IR sensor.

Practically, we use the intrinsic values to compute the point's coordinates. The relationship between a 3D point *(x, y, z)* in space and its correspondent *(u, v)* in the depth image is as follows:

$$x = \frac{(u - p_x)z}{f_x}$$

$$y = \frac{(v - p_y)z}{f_y}$$

Where: (f_x, f_y) is the focal distance and (p_x, p_y) the optical center coordinates. The coordinate *z* is the depth that the sensor returns for the depth image pixel *(u, v)*.

Finally, we apply on *x* and *y* a similar iterative distortion compensation scheme to the one used in OpenCV. The correction over the X and Y axes is equivalent to correcting the camera's ray directions.

Now, we need to adjust the position of each acquired point all along its corresponding camera ray.

3.2 Depth Calibration

At this step, we compute a regular 3D grid of correction coefficients over the view frustum of the sensor (a truncated pyramid) or a part of it. A set of captures of a calibrating plane is used to "feed" the grid's nodes in terms of point correction.

The process consists in two main steps:

- *Data acquisition*: "Real" points spread over the calibration domain and their correction.
- *Grid definition and nodes filling*: "Virtual" points embedding the local correction information and regularly spread over the calibration domain.

For a given sensor, these steps are performed only once to define its proper correction grid.

3.2.1 Data Acquisition

The input data is a set of points, captured by the sensor that we want to calibrate spread over the calibration domain which is the subspace defined by the correction grid. Every point should have a correction coefficient.

To this end, we used a matte white plane with a marker printed on its center. We place our plane against the inner panel of the CMM as shown in Fig. 3. We adjust the sensor's orientation so that it sits parallel to the calibrating plane (more details about the plane and the sensor adjustments are given in section 3.4). Successive captures of the plane are acquired by starting from the farthest distance in the calibrating domain and moving the sensor towards the plane with a fix step until the whole domain is covered.

 ISBN 978-80-86943-40-4

Figure 3. The calibrating plane and its setup on the CMM.

The 2D calibration process corrected the X and Y coordinates, that is the camera rays. Therefore, for every acquired point (of the plane), the correction coefficient we are looking for should slide the point back or forth along the camera ray so that the point's depth matches the real depth. In other words, we are looking for the real distance between the plane and the sensor to compute the correction coefficient.

To compute the real distance between the plane and the sensor, we use image processing to detect the marker printed on the calibrating plane and we apply the similar triangles principle using the focal distance that we already computed with the checkerboard method. Once the first distance computed, we use the CCM in order to infer the next distances for the successive calibrating plane captures.

The correction coefficient of a given point *P* is equal to the real distance of the plane *tD(P)*, which is the true depth, divided by the depth returned from the sensor *sD(P)* as shown in Fig. 4. Therefore, the correction coefficient *c(P)* is:

$$c(P) = tD(P)/sD(P)$$

3.2.2 Grid Definition and Node Filling

3.2.2.1 Grid Definition

The 3D grid is a regular truncated pyramid shaped set of nodes over the calibration domain. Every node is a 4D vector such that the first three components are the nodes coordinates and the fourth component is the correction scalar corresponding to the node. The nodes are not actually sensor's acquired points, but rather "virtual" points embedding the correction information of their neighborhood.

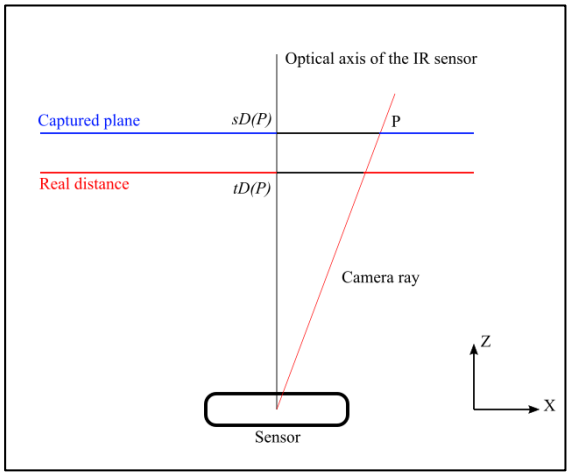


Figure 4. Correction coefficient for a given point *P*: the real depth of the calibrating plane *tD(P)* divided by the z coordinate of *P* returned by the sensor *sD(P)*.

The grid shape was chosen in order to guarantee a fair distribution of the points contributing to the correction computation in each node, regardless of the distance from the sensor.

We divide the Z-axis according to a fix step. We use the same step for capturing the calibrating plane with the couple sensor/CMM.

For the X-axis and Y-axis, we also use fixed steps. In addition, we take into account the maximum resolution of the depth sensor that we should not exceed.

Finally, it is important to consider the approximate number of points that will contribute to the correction of a node via interpolation.

3.2.2.2 Nodes Filling

The nodes positions are defined by the grid construction. Still, we need to compute the error correction in each node. To do so, we begin by defining the neighborhood of a node as all the cells that it belongs to. Using the points from the calibrating plane's captures, we interpolate every subset of points belonging to a neighborhood in order to compute its corresponding node's correction. In fact, each node embeds the correction information of the subspace defined by its neighborhood.

To interpolate over the defined neighborhoods, we used the inverse distance weighting interpolation method. It is defined as follows:

ISBN 978-80-86943-40-4

Let P the point to be corrected (the node), $\{P_i, i=1..N\}$ the vertices of its neighborhood, $d(P, P_i)$ the distance between the node P and the neighbor P_i, c_i the coefficient correction of the neighbor P_i, p a smoothing parameter and $c(P)$ the coefficient correction that we are looking for:

$$c(P) = \begin{cases} \sum_{i=1}^{N} \omega_i(P)\, c_i / \sum_{i=1}^{N} \omega_i(P), & if\ d(P,P_i) \neq 0\ \forall i \\ c_i, & if\ \ d(P,P_i) = 0\ for\ some\ i \end{cases}$$

Where:

$$\omega_i(P) = 1/d(P,P_i)^p$$

The smoothing parameter p controls the influence of far points on the interpolation. We took $p = 3$.

Once filled, the grid can be used to correct any point cloud captured within the subspace defined by it.

3.3 Applying the correction

In order to qualify for correction, a captured point cloud must belong partially or totally to the domain defined by the correction grid. That is, any point outside the calibration area cannot be rectified.

Let *PC* a point cloud captured with a calibrated sensor and *G* its correction grid. For every point *P* in *PC*, we start by finding the point's bounding cell *BC* in the grid *G*. Therefore, the inverse weighting interpolation can be applied across the nodes of *BC* to compute the correction for the point *P*. Finally, we multiply *P* by the computed coefficient to get a rectified point.

To determine the bounding cell of a given point, we define a 3D grid (a truncated pyramid) in which cells are numbered following *IJK* (K direction follows each ray from camera center over our domain). The coordinates *(i,j,k)* refer to the cell with the top-left-front vertex (from the point of view of the sensor. See Fig. 5.

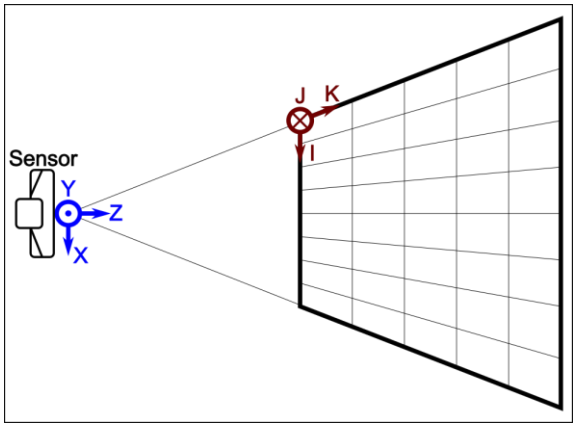


Figure 5. Top view of the newly defined 3D space, *IJK* (top view).

Therefore, beside the *(x,y,z)* coordinates of a given point *M(x,y,z)*, we just defined new coordinates *(i,j,k)* in the *IJK grid* which indicates the bounding cell of the point as follows:

1 - We start by finding *K*-coordinate. In fact, for a given *k*, all the nodes corresponding to the "level" *k* share the same depth. Thus, for every level, we can compare the current point's depth to the first node of each level starting from the farthest level to the sensor. The first level for which the first node's depth is less than the point's depth defines the *K* component. Thus, the bounding cell that we are looking for lays on that level.

2 - To find the *J*-coordinate, we restrict our search to the k^{th} level obtained from the previous step. We compute a signed angle between OM_{YZ} and Z-axis, where $OM_{YZ}(0,y,z)$ is the orthogonal projection of *M* on the plane *YZ*. We compare this angle against the signed angles computed between the projections on the plane *YZ* of the first node of each row from the level *k*, and the *Z*-axis.

3 - For the *I*-coordinate, we restrict our search to the k^{th} level obtained from the first step, and the j^{th} row obtained from the second step. We compute a signed angle between OM_{XZ} and *Z*-axis, where $OM_{XZ}(x,0,z)$ is the orthogonal projection of *M* on the plane *XZ*. We compare this angle against the signed angles computed between the projections on the plane *XZ* of each node of the j^{th} row from the k^{th} level, and the *Z*-axis.

3.4 Hardware Setup

We secure the calibrating plane against the inner panel of the CMM using modeling clay. In fact, it allows adjusting the plane, so it lays orthogonal to the *Y*-axis of the CMM. We attach a mechanical touch probe to the CMM and we "draw" a rectangle near the border of calibrating plane. The probe should touch the calibrating plane in the entire trajectory. If the test fails in some area of the plane, we compensate for the displacement of the calibrating plane using modeling clay. Fig. 6 shows our setup.

Once the calibrating plane is properly set, we detach the mechanical touching probe from the CMM and we attach the couple geared head/sensor instead. Then, we track the marker on the calibrating plane, and use the geared head to fine tune the sensor's orientation. To this end, we perform the detection on the IR camera stream and we highlight the marker's corners when they align over the X-axis or the Y-axis of the sensor in our software. We align the corners couple wise, for example top-left with top-right then top-left with bottom-left. That is, we perform the alignment one direction at a time (fig. 7).

ISBN 978-80-86943-40-4

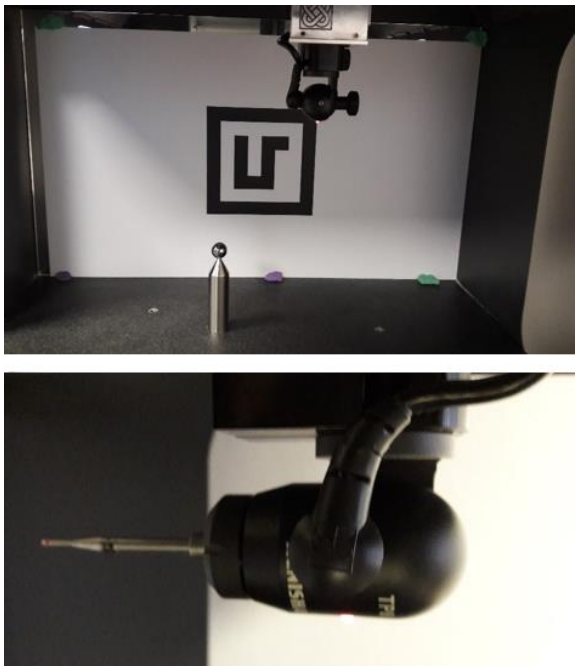


Figure 6. Top: the calibrating plane laying on the "inner panel" of the CMM. Bottom: the mechanical probe used to check the orthogonality of the plane with CMM Y-axis.

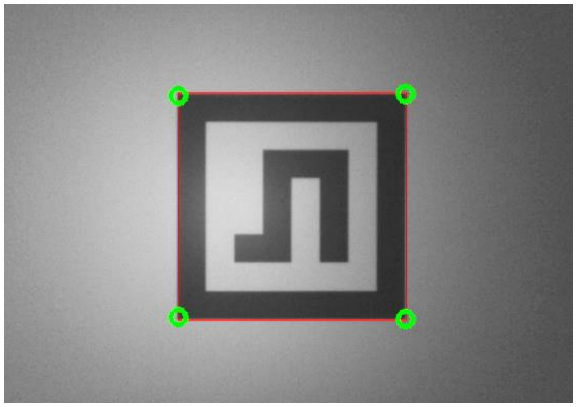


Figure 7. A real successful alignment; we used the green circles to highlight the aligned corners.

When the four corners of the marker align, meaning the sensor is parallel to the calibrating plane, we use the CMM joystick to move the sensor over the X-Y axes of the CMM so that the center of the marker matches the optical center of the sensor in the IR image. We recall that the optical center was computed during the checkerboard calibration. Therefore, we can apply the similar triangles principle to compute the ground truth distance.

In order to enhance the marker's detection, we turn off sensor's IR projector and use an external IR light source to illuminate the plane for a continuous IR illumination as the projector projects changing patterns.

Once the distance is measured, we spray a white matte powder to hide the marker in order to avoid the black color of the marker to distort theses points in the captured point cloud.

4 RESULTS AND VALIDATION

4.1 Calibration domain

According to the inner dimensions of the working space of the CMM, and for the calibrating plane to be fully covering the "frame" for each point cloud captured, we defined our calibration domain as the subspace of the depth view frustum located between 10 cm and 27 cm approximately from the IR camera center. The correction grid is of 64x48x50 size.

4.2 Checkerboard Calibration

We performed a checkerboard calibration on the IR sensor giving the results on table 1. We took 48 pictures of a checkerboard using a 640x480 resolution. The checkerboard has 10x8 square tiles of 3 cm edges.

Fig. 8 shows a picture of the checkerboard before and after the correction via the computed distortion values. See Table I for the numerical results.

Parameter	Our values	Intel SDK's extracted values
Focal distances (pixels)	(473.448, 473.073)	(474.263, 474.263)
Principal point (pixels)	(308.148, 242.341)	(304.816, 245.449)
Radial distortion	(-0.117456, -0.0642003, 0.0390934)	(-0.120845, -0.0660312, 0.0516015)
Tangential Distortion	(-0.00148510, 0.000892128)	(-0.00265185, -0.00182552)
Average re-projection error	0.64	4.79

Table 1. The checkerboard calibration values vs sdk's

To compare the intrinsic parameters that we obtain against those of Intel's SDK, we use the re-projection error. Meaning, we re-project back feature points using the SDK's camera matrix and compare against the checkerboard reference positions, then we repeat the process using our camera matrix. In the end, we compute the average errors. See Table I for all the numerical values. Our computed parameters give a lower re-projection error than Intel's parameters.

ISBN 978-80-86943-40-4

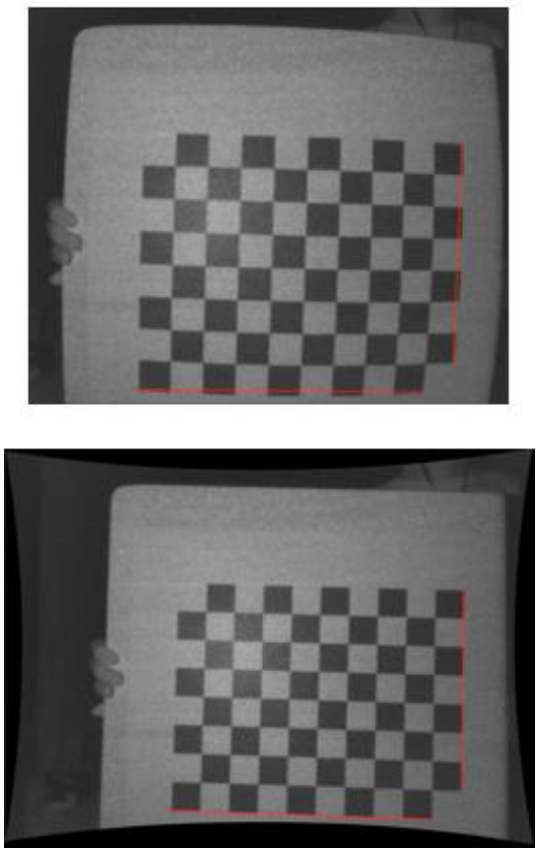


Fig. 8. On the top, a checkerboard picture without correction. On the bottom, the same picture after correcting the distortion. Straight red lines shows the distortion effect.

4.3 Depth Calibration

Before introducing our validation approach, we refer the reader to the in-depth RealSense SR300 assessment from a metrological point of view by Carfagni *et al*. [CFG+17]. Authors give an overview of the RealSense SR300 sensor capabilities and limits as a 3D scanning device.

Fig. 9. The calibration sphere used in our validation process: diameter 50.80 mm (2 inches).

Keeping the same hardware setup that we used for depth calibration, we replace the plane by a calibration sphere with a precisely known diameter Fig. 9. The goal is to capture the sphere at different positions of the calibration domain, then, estimate all the sphere centers using a best-fit approach to form a trajectory with the centers as nodes. For each capture or trajectory node, we acquire two point-clouds, one using the SDK's calibration values and the other using our calibrating values (checkerboard inferred intrinsic parameters). To the set of clouds captured using our values, we additionally apply depth correction.

We compute two errors per trajectory, a global error and a local error.

4.3.1 Global Error

For this estimator, no reference sphere is chosen, hence the term global. We denote the global error *E*.

We compute the distance of each sphere center to the next sphere center, in the order of their captures as no specific order is required. We will refer to the first set of distances as point cloud distances and we will denote it D_{PC}. Equivalently, we compute the distances between the successive CMM positions of the captures that we will call CMM distances and we will denote D_{CMM}. We define the global error as the following:

$$E = \sum_{\substack{d_{PC} \in D_{pc} \\ d_{CMM} \in D_{CMM}}} |d_{PC} - d_{CMM}| \,/ (num\_spheres - 1)$$

Where: d_{CMM} is the correspondent of d_{PC} in D_{CMM}.

4.3.2 Local Error

A local error can be computed at each sphere center that we captured. For a sphere *S*, we compute the local error *e(S)* by taking the distances to all the other sphere centers and comparing them against the respective CMM inferred distances in a similar way of the global error. The local error at the sphere's center is:

$$e(S) = \sum_{\substack{d_{PC} \in D_{pc}(S) \\ d_{CMM} \in D_{CMM}(S)}} |d_{PC} - d_{CMM}| \,/ (n\_spheres - 1)$$

Where $D_{PC}(S)$ is the set of distances computed from the point clouds and $D_{CMM}(S)$ is the set of distances computed from the respective CMM positions. d_{CMM} is the correspondent of d_{PC} in $D_{CMM}(S)$.

4.3.3 Results

We captured the calibrating sphere on twenty-seven different positions as shown in Fig. 10.

We recorded sets of three calibrations using our method under the same conditions. The plots in fig. 11 depict the global and local errors that we obtained. Although there are some positions where the RealSense SDK calibration performed better than our calibration, our average global error is lower in all the experiments, see Table II for the average global error of each experiment. Concerning local error, we can see that our

 ISBN 978-80-86943-40-4

calibration is much better than the SDK's in all the experiments.

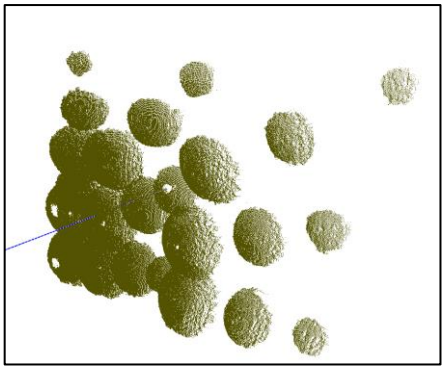


Fig. 10. The calibrating sphere captures over the calibration domain. The blue line corresponds to the Z-axis of the sensor.

Fig. 11. Shows a point cloud before and after the calibration. We chose a flat surface point cloud in order to see the actual difference. In fact, it is near the corners of a flat surface covering the whole "frame" that the distortion is mostly visible.

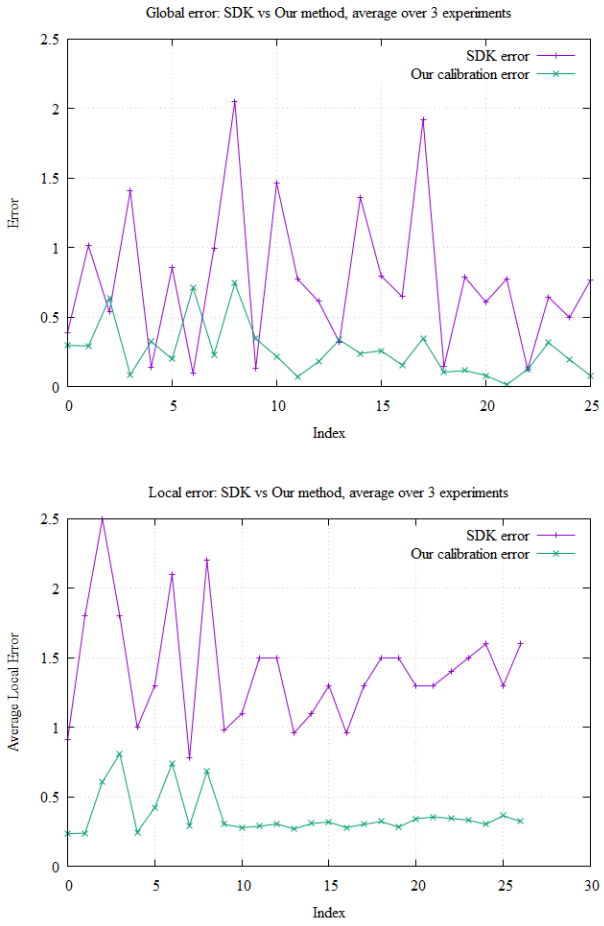


Fig. 11. Compared global error and local error plots SDK versus our method. We averaged over 3 experiments.

	Our calibration average error (mm)	SDK average error (mm)
1st experiment	0.18	0.76
2nd experiment	0.27	0.76
3rd experiment	0.32	0.76

Table 2. The global error evaluation

Fig. 12 Left: front and top view of a point cloud (flat surface) before correction. Right, the same plane after correction using our method.

4.3.4 *Notes on the method's precision*

The accuracy of our method essentially depends on two factors:

- The average re-projection error of the checkerboard calibration (see Table I). In our test, the error is 0.64 pixels.
- The precision of the ground-truth distance computed through image processing.

We will try to evaluate the second factor that is the accuracy of the ground-truth distance. It heavily relies on the average re-projection error as the corrected and undistorted IR frames are used in the image-processing step.

Using the similar triangles principle, the ground truth distance d is computed as follows:

$$d = \frac{L f}{l}$$

Where, L is the marker half-width (in millimeters), l is the marker half-width detected in the IR frame (in pixels) and f is the computed focal distance (in pixels) from the checkerboard calibration.

Now, suppose that we make a mistake of n pixels in our detection, and that the computed distance is d'. Then, the error corresponding to this detection is approximatively:

ISBN 978-80-86943-40-4

$$E(n) \approx \mathrm{d} - \mathrm{d}' = \frac{Lf}{l} - \frac{Lf}{l+n}$$

Thus,

$$E(n) \approx \frac{nLf}{l(l+n)}$$

The first thing to notice is that the bigger the value l, the smaller the error. To increase l, the IR camera should be set to its maximum resolution, that is 640x480 for the RealSense SR300, and the sensor should be very close to the camera in such a way that the marker cover most of the frame while still entirely enclosed in for detection sake.

To get an idea about the precision we achieved in our setup, we could get as close for a value of 225 pixels for l.

Knowing that $L = 79.5\,mm$ and $f = 473.448\,pixel$, the error is:

$$E(0.64\,pixels) \approx 0.47\,mm$$

Thus, we have approximatively a half millimeter precision in our ground truth distance.

5 CONCLUSION

In this paper, we have proposed a supervised intrinsic calibration method for the Intel RealSense SR300 that relies on the use of a CMM for robust ground truth. It has proven to give superior accuracy over the manufacturer's default calibration, as shown in the "Results and Validation" Section. In addition, we can apply it to other structured-light sensors, as we do not use any special or exclusive calibration parameter to the Intel RealSense SR300 sensor.

On the limitations side, when computing the X and Y coordinates, the method involves the use of a non-corrected yet depth coordinate (see equations page 3). Still, our approach performs better than the default manufacturer calibration, but as a future improvement, we will estimate the gap and if needed perform iterative calibration steps. On another side, we plan to make our method fully automated.

6 REFERENCES

[Int16]https://software.intel.com/sites/default/files/managed/0c/ec/realsense-sr300-product-datasheet-rev-1-0.pdf

[HZ05] Hartley, R., & Zisserman, A. (2005). Multiple view geometry in computer vision. Robotica, 23(2), 271-271.

[MVGV09] Moons, T., Van Gool, L., & Vergauwen, M. (2009). 3D reconstruction from multiple images, Part 1: Principles. Now Publishers Inc.

[HKH12] Herrera, D., Kannala, J., & Heikkilä, J. (2012). Joint depth and color camera calibration with distortion correction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(10), 2058-2064.

[Sem16] Semeniuta, O. (2016). Analysis of Camera Calibration with Respect to Measurement Accuracy. Procedia CIRP, 41, 765-770.

[Zha04a] Z. Zhang, "Camera Calibration", Chapter 2, pages 4-43, in G. Medioni and S.B. Kang, eds., Emerging Topics in Computer Vision, Prentice Hall Professional Technical Reference, 2004.

[Hei00] Heikkila, J. (2000). Geometric camera calibration using circular control points. IEEE Transactions on pattern analysis and machine intelligence, 22(10), 1066-1077.

[Zha00] Zhang, Z. (2000). A flexible new technique for camera calibration. IEEE Transactions on pattern analysis and machine intelligence, 22(11), 1330-1334.

[SM99] Sturm, P. F., & Maybank, S. J. (1999). On plane-based camera calibration: A general algorithm, singularities, applications. In Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on. (Vol. 1). IEEE.

[Zha04b] Zhang, Z. (2004). Camera calibration with one-dimensional objects. IEEE transactions on pattern analysis and machine intelligence, 26(7), 892-899.

[SJP11] J. Smisek, M. Jancosek, T. Pajdla, 3D with Kinect, in: IEEE Workshop on Consumer Depth Cameras for Computer Vision, 2011.

[JLG14] Jin, B., Lei, H., & Geng, W. (2014, September). Accurate intrinsic calibration of depth camera with cuboids. In European Conference on Computer Vision (pp. 788-803). Springer International Publishing.

[SBMM15] Staranowicz, A. N., Brown, G. R., Morbidi, F., & Mariottini, G. L. (2015). Practical and accurate calibration of RGB-D cameras using spheres. Computer Vision and Image Understanding, 137, 102-114.

[CFG+17] Carfagni, M., Furferi, R., Governi, L., Servi, M., Uccheddu, F., & Volpe, Y. (2017). On the performance of the Intel SR300 depth camera: metrological and critical characterization. *IEEE Sensors Journal*, *17*(14), 4508-4519.

 ISBN 978-80-86943-40-4

ISSN 2464-4617(print)
ISSN 2464-4625(CD)
Computer Science Research Notes
CSRN 2801
Full Papers Proceedings
http://www.WSCG.eu

Assessing Objective Image Quality Metrics for Bidirectional Texture Functions

Banafsheh Azari
CogVis/MMC,
Faculty of Media,
Bauhaus-University Weimar
Bauhausstrasse 11
99423 Weimar, Germany
banafsheh.azari@uni-weimar.de

Sven Bertel
Usability,
Flensburg University of
Applied Sciences,
Kanzleistrasse 91-93,
24943 Flensburg, Germany
sven.bertel@hs-flensburg.de

Charles A. Wüthrich
CogVis/MMC,
Faculty of Media,
Bauhaus-University Weimar
Bauhausstrasse 11
99423 Weimar, Germany
charles.wuethrich@uni-weimar.de

ABSTRACT

Bidirectional Texture Functions (BTFs) are view- and illumination-dependent textures used in rendering for accurate simulation of the complex reflectance behavior of fabrics. One major issue in BTF rendering is the large number and size of images which requires lots of storage. "Visually lossless" compression offers the potential to use higher compression levels without noticeable artifacts, but requires a rate-control strategy that adapts to image content and loss visibility.

In this contribution, we investigate the applicability of objective image quality metrics to predict levels of perception degradation for compressed BTF textures. We apply traditional error-sensitivity and structural similarity based approaches to predict levels of perceptibility for compressed BTF textures to achieve visually lossless compression. To confirm the validity of the present study, the results of an experimental study on how decreasing the BTF texture resolution influences the perceived quality of the rendered images with the results of the applied image quality metrics are compared.

In order to compare two representatives from each group were selected. The Visible Differences Predictor (VDP) and Visual Discrimination Model (VDM) are typical examples of an image quality metric based on error sensitivity, whereas the Structural SIMilarity index (SSIM) and Complex Wavelet Domain Structural Similarity Index (CWSSIM) are specific examples of a structural similarity quality measure.

Keywords

Perceptual experiment, Realistic rendering, Visual quality metric.

1 INTRODUCTION

To have a photo realistic display of fabrics, a real illumination and view dependent surface texture representation, called the Bidirectional Texture Function (BTF), was introduced in [1].

$$S_{\mathbf{BTF}} = \int_{p \in \mathbf{P}} (\theta_{\mathrm{i}}, \phi_{\mathrm{i}}, x_{\mathrm{p}}, y_{\mathrm{p}}, \theta_{\mathrm{o}}, \phi_{\mathrm{o}})\ \delta p, \qquad (1)$$

BTF is a six-dimensional function representing the appearance of a material sample surface for variable illumination (θ_i, ϕ_i) and view (θ_o, ϕ_o) directions, where θ and ϕ are elevation and azimuthal angles, respectively, and (x, y) is the planar position over a material surface.

The three-dimensional textured models rendered through BTF rendering method are subject to various types of distortion during acquisition, synthesis, compression and processing. An appropriate image quality assessment scheme is a useful tool for evaluating image processing algorithms, specially algorithms designed to leave the image visually unchanged (e.g. compression algorithms) [2].

While the quality assessing task is simple for human observers, it actually involves very complex psychophysical mechanisms. Due to the high complexity of the human visual system (HVS), understanding it with current psychophysical knowledge is nearly impossible.

Currently, the only reliable way is to compare the overall visual similarity of two textures by independent observers in a psychophysical experiment [3–6]. However, this method is expensive, and it is usually too slow to be useful in real-world applications. As an alternative solution, BTF data modeling quality can be verified using objective image quality metrics (IQMs).

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.5

This paper makes an attempt to validate these models with regard to predicting the visible quality differences in images rendered by compressed and non compressed BTFs.

For comparison of the traditional error-sensitivity and structural similarity based approaches, two representatives from each group were selected: The Visible Differences Predictor (VDP; [7]), Visual Discrimination Model (VDM; [8]), the Structural SIMilarity index (SSIM; [9]) and Complex Wavelet Domain Structural Similarity Index (CWSSIM; [10]).

The metrics were implemented and the results obtained from the predictions of the models were compared with each other and with the outcomes of a subjective quality measure experiment, which involved quality comparison tasks with pairs of textured objects of varying BTF quality levels [11].

In the next section, we will introduce the fundamentals of objective image quality assessment and review relevant full-reference objective quality metrics. Next some instances of the predictions of the models are presented and their performance is characterized accordingly. After discussion on the models, the detection results of metrics are compared with each other and with the outcomes of the user study, which is then followed by a conclusion and an outlook.

2 OBJECTIVE IMAGE QUALITY METRICS

The goal of Objective Quality Metrics is to design mathematical models that are able to predict the quality of an image accurately and automatically. An ideal method should be able to mimic the quality predictions of an average human observer.

Pixel-Based Metrics such as Root Mean Square (RMS) error or Peak Signal to Noise Ratios (PSNR) fail to assess the perceived degree of realism since they neglect important properties of the human visual system and poorly predict the differences between the images.

The philosophy used in constructing an objective image quality metrics is one of the major criterion employed for their classification. While traditional perceptual approaches to image quality assessment (bottom-up) are directly connected with the characteristics of HVS and try to simulate all the relevant components and psychophysical features as basic building blocks, and then combine them together, the ultimate goal of the structural similarity based approaches (top-down) is to make hypotheses about the overall functionality of the entire HVS and treat it as a black box, where only its input-output relationship is of concern. This section gives a overview of the general philosophy of both metrics and introduces the most popular and widely used metrics in each category.

2.1 Error Sensitivity Based Image Quality Measurement

A great variety of objective image quality assessment methods follow an error sensitivity based paradigm that attempts to analyze and quantify the error signal in a way simulating the characteristics of human visual error perception. In this part we will outline the perceptually driven image quality metrics that are used in this study that we will describe in the following, namely, VDP and VDM.

2.1.1 Visible Differences Predictor

The Visible Differences Predictor (VDP; [7]) is one of the well-known image distortion metrics, which consists of three main components: calibration of the input images, a HVS model and a method for displaying the visible differences.

The algorithm receives a pair of images (original and compressed images), and parameters for viewing conditions as input. After the calibration of the input images, in the next stage the HVS is modeled i.e. the lower-order processing of the visual system, such as the optics, retina, lateral geniculate nucleus, and striate cortex. The HVS model uses processes to limit the visual sensitivity.

Firstly, the original pixel intensities are compressed by the amplitude non-linearity based on the local luminance adaptation. Afterwards, the contrast sensitivity function (CSF) is processed to model the variations as a function of spatial frequency and so as to take into account the global state of luminance adaptation, orientation, image size and eccentricity from the fovea region. The sensitivity S as a function of ρ radial spatial frequency in c/deg is modeled by the following equation ([7]):

$$S(\rho,\theta,l,i^2,d,e) = \\ P \cdot min[S_1(\frac{\rho}{r_a \cdot r_e \cdot r_\theta}, l, i^2), S_1(\rho, l, i^2)], \quad (2)$$

where θ is the orientation in degrees, l is the light adaptation level in cm/m^2, i^2 is the image size in visual degrees, d is lens accommodation due to distance in meter, and e is eccentricity in degrees. The parameters r_a, r_e and r_θ model the changes in resolution due to the accommodation level, eccentricity and orientation and P is the absolute peak sensitivity of the CSF.

The resulting images are decomposed into the spatial frequency and orientation channels using the cortex transform introduced by [12]. Cortex transform is a multi-resolution pyramid that simulates the spatial-frequency and orientation tuning of simple cells in the primary visual cortex. For every channel and every pixel, the global contrast and elevation of the detection threshold based on masking is calculated. This

 ISBN 978-80-86943-40-4

detecting threshold is then used to normalize the contrast differences between target and mask images. The normalized differences are input into the psychometric function which estimates the probability of detection of differences for a given channel. This estimated probability value is summed across all channels for every pixel, and visualization of visible differences between the target and mask images is performed.

While this metric is designed for low dynamic range (LDR) images, [13] proposed an high dynamic range (HDR) extension of VDP, that can handle the full luminance range visible to the human eye. The modifications improve the prediction of perceivable differences in the full visible range of luminance and under the adaptation conditions corresponding to real scene observation.

2.1.2 Visual Discrimination Model

Another frequently used image discrimination measuring method is the Sarnoff Visual Discrimination Model (VDM; [8]). The Visual Discrimination Model acts in the spatial domain by firstly using an approximation of the point spread function of eye's optics, according to which the input data are convoluted. Next, the signals are re-sampled to be able to reproduce the sampling of photo-receptor in the retina. To break down the images into seven different resolutions, VDM uses a Laplacian pyramid [14]. At this stage each resolution must be one-half of the immediate higher image. Band-limited contrast computations are then performed.

In the next step, the selectivity of orientations in four different orientations is applied. To do this through steerable filters of Freeman and Adelson [15], a group of orientation filters were implemented. CSF was modelled through normalization of the output of every frequency-selective channel by the base-sensitivity for that channel. To implement masking, a nonlinear sigmoid is used. This is performed after convolving the errors at each level with disk-shaped kernels. Eventually, JND (Just Noticeable Differences) map or a distance measure is calculated as the Lp-norm of the responses of the masks. In the visual field of an observer, the eccentricity of images is an important factor. VDM is one of the few models that appropriately takes this into account. For color video, VDM was modified to the Sarnoff JND metric [8],

$$J = \frac{1}{ln2}\int_{V_{max}}^{0}\sqrt{\frac{M(V)}{M_t(V)}}\frac{dV}{V}, \qquad (3)$$

where V_{max} is the maximum spatial frequency displayed, $M(V)$ is the modulation transfer function of display and $M_t(V)$ is the threshold modulation transfer function of the human visual system.

2.2 Structural Distortion Based Image Quality Measurement

The fundamental principle of the structural approach is that the human visual system is highly adapted to extract structural information (the structure of objects) from the visual scene, and therefore a measurement of structural similarity (or distortion) should provide a good approximation of perceptual image quality.

In this section, we will mainly focus on two very recent and exceptionally successful general-purpose image quality assessment approaches, the Spatial Domain Structural Similarity Index (SSIM; [9]) approach and the Complex Wavelet Domain Structural Similarity Index (CWSSIM; [10]) approach. These approaches are based on high-level top-down hypotheses regarding the overall functionality of HVS (see [16]).

2.2.1 Spatial Domain Structural Similarity Index

Under the assumption that human visual perception is not built for detecting absolute, exact intensities, instead it is adapted to help us navigate the three-dimensional space we live in and, consequently, is highly adapted for extracting structural information from a scene, [9] introduced the Structural SIMilarity Index (SSIM).

In particular the SSIM index is a framework for quality assessment based on the degradation of structural information and is mostly sensitive to distortions that break down natural spatial correlation of an image such as blur, blocking, ringing, and noise.

The SSIM separates the task of measurement into three functions: Luminance $l(x,y)$, contrast $c(x,y)$, and structure $s(x,y)$. Given two images (or image patches) of x and y for comparison, the three similarity functions are then combined to yield the general form of the SSIM index structural similarity:

$$SSIM(x,y) = l(x.y)^{\alpha}\cdot c(x.y)^{\beta}\cdot s(x.y)^{\gamma}, \qquad (4)$$

where α,β,γ are positive constants used to weight each comparison function.

SSIM is a window-based algorithm that uses a square window, moving pixel-by-pixel over the image to measure loss of correlation, luminance distortion and contrast distortion locally [9]. To evaluate the overall image quality, a mean SSIM (MSSIM) index is calculated as follows:

$$MSSIM(X,Y) = \frac{1}{M}\sum_{i=1}^{M} SSIM(x_i,y_i), \qquad (5)$$

where M is the number of samples in the quality map, x_i and y_i are the image contents at the i-th local window, and X, Y are the input images.

 ISBN 978-80-86943-40-4

The structural similarity metric yields a result in a range of 0.0 to 1.0, where zero corresponds to a loss of all structural similarities and one corresponds to being an exact copy of the original image. Images with lighting-related distortions alone yield a high SSIM value while other distortions result in low similarities, corresponding well with the intuitive perception of quality.

2.2.2 Complex Wavelet Domain SSIM

A major drawback of the spatial domain SSIM algorithm is that it is highly sensitive to translation, scaling and rotation of images while perceptual metrics can successfully account for contrast and luminance masking, they are quite sensitive to spatial shifts, intensity shifts, contrast changes, and scale changes.

[10] suggested to implement a structural similarity metric in the complex wavelet domain and make it insensitive to these "non-structured" image distortions that are typically caused by the movement of image acquisition devices, rather than the changes in the structure of objects in the visual scene [10]. In addition, if an application requires an image quality metric that is unresponsive to spatial translation, this extension of SSIM can be adopted.

Given complex wavelet coefficients c_x and c_y that correspond to compared image patches x and y, the complex wavelet structural similarity (CWSSIM) is yielded by:

$$CWSSIM(c_x, c_y) = \frac{2 \mid \sum_{i=1}^{N} c_{x,i}, c_{y,i}^{*} \mid +K}{\sum_{i=1}^{N} \mid c_{x,i} \mid^2 + \mid c_{y,i} \mid^2 + K}, \quad (6)$$

where c^* denotes the complex conjugate of c and K is a small positive constant.

The proposed algorithm shows some interesting connections with several computational models that have been successfully used to account for a variety of biological vision behaviors such as those pointed out by [17–19]. However, the algorithm does not provide any information on correspondences between the pixels of the two compared images and the method works only when the level of translation, scaling, and rotation is small (compared to the wavelet filter size).

3 EXPERIMENT

The goal of the experiment is to investigate the validation of error sensitivity and structural distortion based image quality metrics to predict the visible differences between compressed and non compressed texture resolutions. To achieve this, we analyze and compare the results of these models against each other and with the outcomes of the user study such as, performance of subjects (i.e., the subjects' ability to judge image quality differences) and the gaze data (locations and frequencies of fixations).

In the experimental study three datasets have been used. The first one was corduroy, available in the BTF database of the University Bonn[1], which we will refer to as *Cord-256*, whereby its texture pictures are 256x256 pixels. We generated two additional datasets by downscaling the *Cord-256* set through bilinear interpolation to respective resolutions of 128x128 pixels (*Cord-128*) and 64x64 pixels (*Cord-64*). For each of the three texture data sets, a three dimensional textured model of a sofa was rendered through the standard BTF rendering method at a screen resolution of 1920x1080 pixels.

The sofa model was oriented for presentation to the viewer so as to present textured parts across a large range of picture depth. We chose a sofa to have an object with a structured surface and composition (e.g., individual buttons, cushions, etc.). This is important in order to ensure that a large set of fitting BTF pictures will be selected as basis for the object's texture, with widely varying illumination and viewing angles (see Figure 1).

Pairs of images were displayed on a full screen in native resolution mode. Each pair consist of a sequentially presented rendering with the use of two of the three texture resolutions (256x265, 128x128 and 64x64). After the presentation of each pair, subjects were asked to make a decision about the comparative image quality within the pair: was the first or second image of better visual quality? Or were the two images of the same visual quality? A SMI RED250 remote eye tracking system was used in binocularmode with 250 Hz fixation detection, in order to record subjects' fixation behavior. A total number of 20 subjects, 12 males and 8 females, participated in the experiment, and they were not informed about the purpose of the experiment prior to conducting it.

The same sofa object model in the experimental study stimuli as well as one additional spherical object, which contains various angles and depth combinations, were utilized for making performance and detection results comparable with the outcomes of the experimental study. For the texture, two cases were considered including the Cord already known from the experimental study and Pulli, which is also available in the BTF database of the University Bonn.

Both objects are rendered in three levels of resolutions namely: 256x256, 128x128 and 64x64 pixels, which are referred to as *Cord-256* / *Pulli-256*, *Cord-128* / *Pulli-128* and *Cord-64* / *Pulli-64*, respectively.

The images were presented on a 24-inch monitor with a resolution of 1920x1080 pixels at a distance of 70 cm from the viewer. The screen measured 22.35x15.80

[1] http://btf.cs.uni-bonn.de/.

 ISBN 978-80-86943-40-4

	VDP Fixation location	VDM Fixation location	SSIM Fixation location	CWSSIM Fixation location
Cord-256 _ Cord- 64	-0.808	-0.1772	-0.498	-0.643
Cord-128 _ Cord- 64	-0.753	-0.1728	-0.320	-0.582
Cord-256 _ Cord-128	-0.015	-0.473	-0.155	-0.0617

Table 1: Correlations between IQMs results >75% of fixation location independently of presentation order. ($p > 0.0001$)

	VDP-Depth of the pixel	VDM-Depth of the pixel	SSIM-Depth of the pixel	CWSSIM-Depth of the pixel	Fixation-Depth of the pixel
Cord-256 _ Cord- 64	-0,1636	-0,6961	-0,0173	-0,2872	-0.2705
Cord-128 _ Cord- 64	-0,1124	-0,6929	-0,0092	-0,4498	-0.2305
Cord-256 _ Cord-128	-0,0061	-0,6413	-0,0055	-0,5105	-0.2405

Table 2: Correlations between IQMs results, number of fixation and depth of the pixel independently of presentation order. ($p > 0.0001$)

	#equal	#correct	*VDM*	*SSIM*	*CWSSIM*
Cord-256 _ Cord- 64	49	382	0.93624	0.963	0.822
Cord-128 _ Cord- 64	44	383	0.89378	0.971	0.838
Cord-256 _ Cord-128	423	63	0.39004	0.994	0.949

Table 3: Frequencies of correct answers, incorrect equal-quality answers (accumulated over all 20 subjects; sum of answers per pair: 480); and dprime value from VDM, SSIM and CWSSIM

inches and subtended approximately 33 degrees of visual angle. Due to the texture pattern, the minimal texture detail (i.e., for the parts of the sofa at the greatest depth in the image) had a cycle of 4 pixels, which means a subtended angle for a viewer of about 6 cycles per minute of a degree of arc. We employed the same condition for all the metrics.

3.1 Detection Results and Performances

In this section both, the output detection images of the image quality metrics, and the outcome of the user study are compared. The implemented metrics received pairs of images as input. The output detection images of the metrics were then compared and discussed.

For all the models, the following approach was employed [20]: the numerical value of the difference between images is the percentage of pixels for which the probability of difference detection is greater than 0.75. It is assumed, that the difference can be perceived for a given pixel when the probability value is greater than 0.75 (75%), which is the standard threshold value for discrimination tasks, [21]. This output value therefore ranges between 0 and 100 , where 0 means the best result (no pixel with probability of difference detection greater than 0.75), while 100 means that all the pixel differences are above the difference detection threshold (the worst result).

However, since we also need a single overall quality measure, we use a mean index in the case of SSIM and CWSSIM models and JND for VDM. The index values fall within a range of 0 to 1, where 1 in JND value of VDM means the worst quality, and 0 denotes an indistinguishable difference between the input images, which is in case of SSIM and CWSSIM mean index conversely.

Figures [1,2–4] present the output images of the metrics. To have a better comparison between metrics the results of two famous pixel-based metrics, the MSE and PSNR, for each image pair are also presented.

Next gaze fixation distributions of subjects across the sofa images were analyzed in order to assess whether differences exist for different image pair comparisons. Fixation counts for cells in an overlaid 16x16 grid are shown in Figure 1 (upper part) for three conditions.

Correlations between VDP/ VDM results (above 75%) and respective fixation location patterns can be observed in Table 1. We observed strong correlations between locations of predicted visually perceivable differences by VDP and observed fixation patterns only for Cord-256 and Cord-64 as well as Cord-128 and Cord-64, while significant, albeit a very poor correlation exists for VDM and fixation patterns for all image pairs. The results show a poor correlation for SSIM and CWSSIM.

In the next step, the responses of objective quality metrics to pixel depth for each image pair and the percentage of fixation in each depth were controlled.

Table 2 illustrates the correlation between IQMs responses and the depth of pixels as well as the correlation between fixation position and the depth of these pixels. The results show a poor correlation between

 ISBN 978-80-86943-40-4

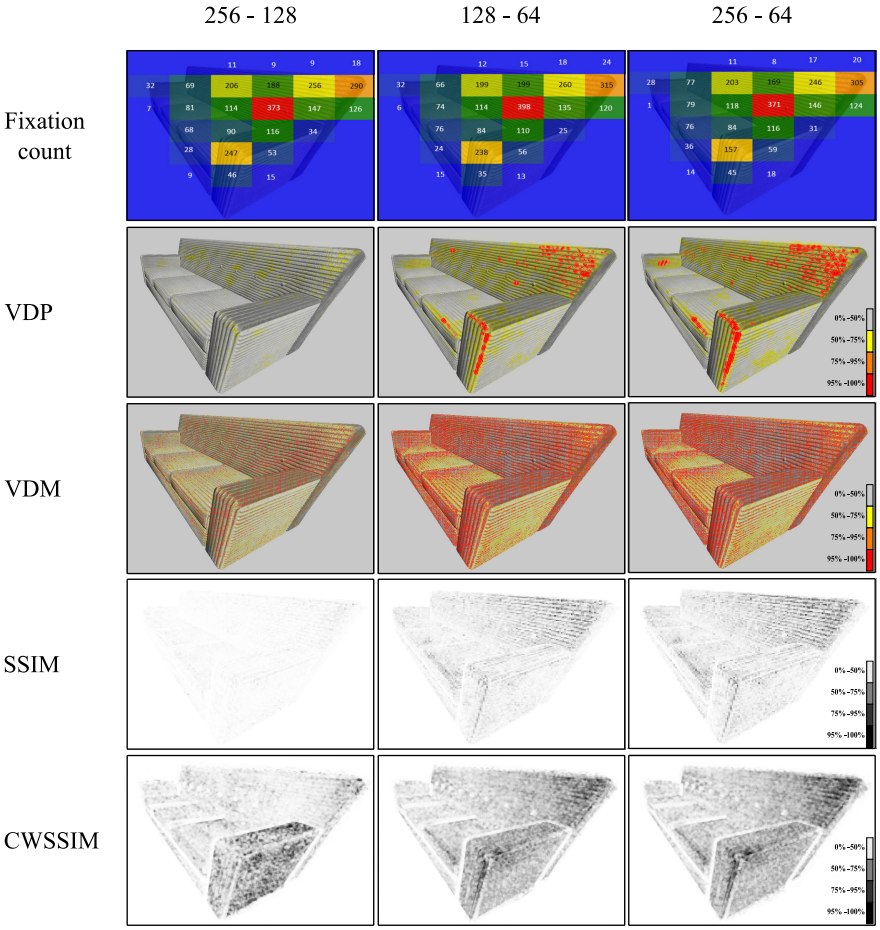


Figure 1: The output images of four IQMs by sofa with different *'Cord'* texture resolution pairs. The color-scales on the right side indicate probability values of metrics in each pixel. The last row presents Just Noticeable Difference (JND) values of VDM, SSIM and CWSSIM. Additionally the MSE and PSNR, for each image pairs are also presented.

VDP, VDM and fixation and a significant correlation between SSIM, CWSSIM and pixel depth.

As shown by Figures 7, all curves react similarly to depth from quality perspective, but VDM is less sensitive than other metrics.

The first two columns of Table 3 illustrate the number of correct and equal answers yielded for each of image pairs, and the remaining columns present the result of VDM, SSIM and CWSSIM. The results show a significant correlation between subjects' ability to perceive differences between images and IQMs predictions.

In order to control the correlation between the saliency map (SM), Regions of Interest (ROI) and the responses of IQMs, we followed [22] and computed a ROI map from the subjects' fixations.

The ROI map is a probability distribution of the gaze direction, therefore its integral is normalized to 1. Figure 5 (left-down) shows the ROI map obtained from individual fixations.

To define the saliency map the algorithm proposed by [23] was employed, with a new definition of the visual features (intensity, colour and orientation), which is the most popular in computer science, and has led to more convincing oculometric validations (see Figure 5 (right-up)). The Saliency Toolbox for Matlab, which is available online, was utilized in the present study [24].

ISBN 978-80-86943-40-4

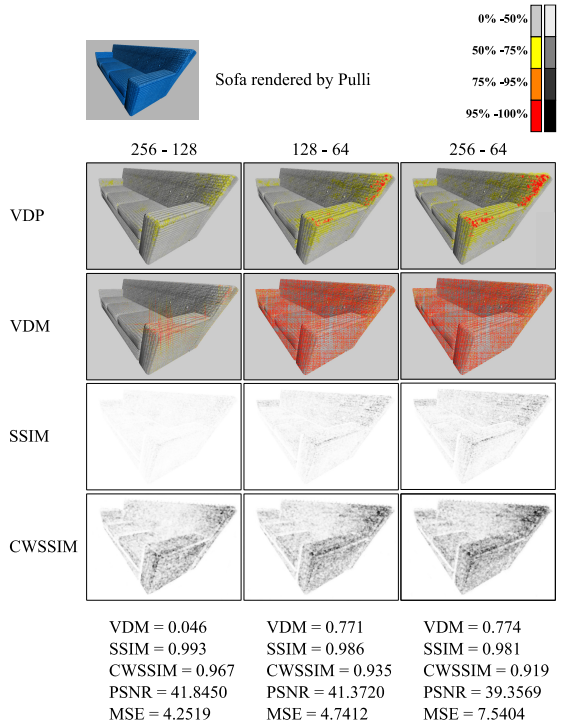


Figure 2: The output images of four IQMs by sofa with different *'Pulli'* texture resolution.

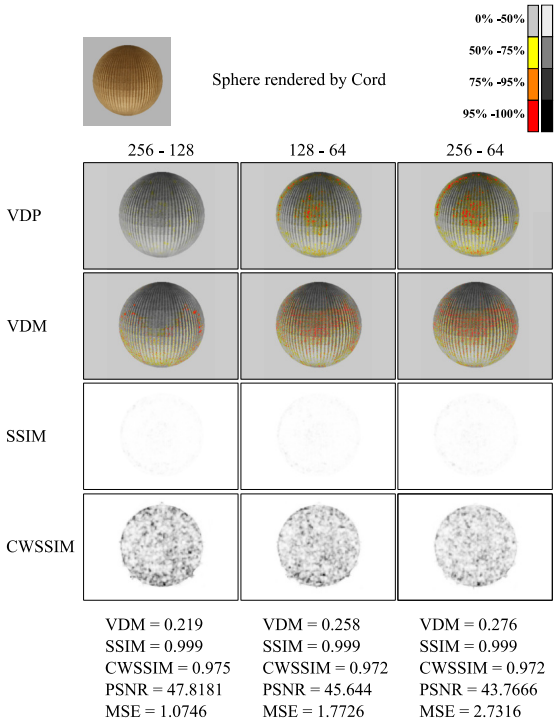


Figure 3: The output images of four IQMs by sphere with different *'Cord'* texture resolution.

	Total execution time (s)	
	Sofa	Sphere
VDP	7.256	4.654
VDM	0.340	0.152
SSIM	0.282	0.134
CWSSIM	0.929	0.432

Table 4: Total execution time in second. All the metrics run on the same machine.

Compared to the saliency maps shown in Figure 5, the ROI map is smoother. The saliency map and ROI are significantly correlated when $r = 0.560$ and $p < 0.001$.

Table 5 illustrates the correlation between IQM responses and ROI as well as the correlation between IQM responses and saliency map. As observed, the value between each IQM for ROI and saliency map is highly correlated when $r = 0.893$ and $p < 0.001$. The correlation coefficients between the adopted experimental subjective data set (ROI) and IQM responses exhibit that all models, except for VDM model, exhibit a good level of consistency with the subjective data.

The computation time is also another significant factor for selecting the image quality assessment. The computational complexity is measured in terms of time required by each of the metrics to assess the quality of a pair of images. In this step, each metric was computed for all pairs of images and then the average time was determined. The metrics were run on a computer with a 3.20 GHz Intel Xeon Six-Core processor.

In order to allow for a fair comparison, the publicly available Matlab implementation of each metric was used. The average performance of all the methods is provided in Table 4. SSIM, CWSSIM and VDM have a complexity of $O(N)$. This is due to the fact that these metrics work in the spatial domain avoiding the expensive FFT and FFT^{-1} transformations. This transformation can take up to 40% of the total execution time in VDP, and thus increase the complexity of this model to $O(NlogN)$ with an upper bound of $O(N^2)$ (see [25]).

To control the reaction of the metrics to different geometrical distortions the object in the scene (sofa) was shifted without any other quality distortions and then used as a distorted image. Additionally we applied the metrics to blurred, salt & pepper and Gaussian noise contaminated images. Figure 6 illustrates the output detection images of the metrics.

4 DISCUSSION

The differences between the metrics are caused by placing pressure on different aspects of human visual perception. Nevertheless the results show that all metrics can be an appropriate replacement for subjective quality measurement matrices.

The vision models have different ways to visualize the detected probability. While VDP uses a psychometric

ISBN 978-80-86943-40-4

	VDP		VDM		SSIM		CWSSIM	
	ROI	SM	ROI	SM	ROI	SM	ROI	SM
Cord-256 _ Cord- 64	0.71	0.79	0.21	0.19	0.75	0.61	0.72	0.84
Cord-128 _ Cord- 64	0.63	0.57	0.23	0.20	0.73	0.58	0.83	0.85
Cord-256 _ Cord- 128	0.013	0.011	0.19	0.21	0.15	0.12	0.35	0.47

Table 5: Correlation between objective image quality metrics; VDP, VDM, SSIM and CWSSIM with ROI and saliency map ($p > 0.0001$).

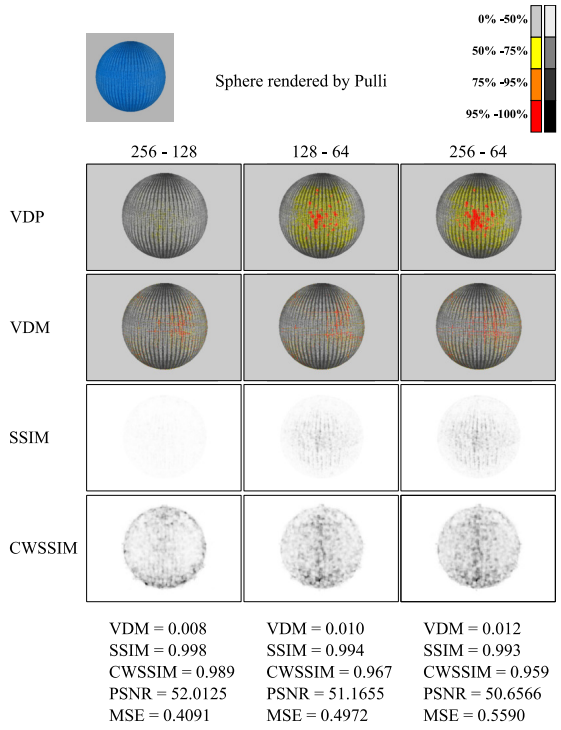


Figure 4: The output images of four IQMs by sphere with different *'Pulli'* texture resolution.

function, which describes the relationship between the threshold contrasts and detection probabilities, to convert the normalized threshold contrasts into detection probabilities, all other models make direct use of JND map and neglect the psychometric function.

An advantage of the output map is that the nature of the difference can be observed and this observation can be used for further rendering optimizations.

The results show a significant correlation between subjects' ability to perceive existing differences between the images and predictions of VDM/CWSSIM models. Based on this investigation, it seems that VDM and CWSSIM can well predict the differences between two images.

The responses of objective quality metrics to pixel depth for each image pair shows that all react similarly to depth but VDM is less sensitive than other metrics. Due to the textured pattern, the texture details for the parts of the sofa from the depth of 0.3 to 0.8 have

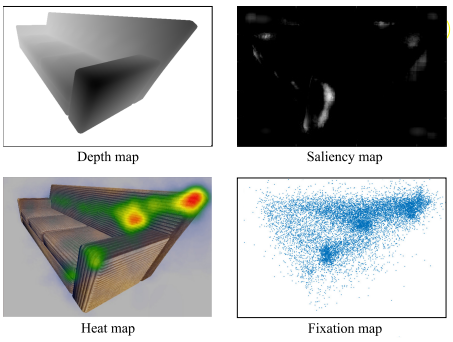


Figure 5: Depth map (left-up) ROI map (left-down), saliency map (right-up) and fixation map (right-down)

a 4 to 5 cycles per degree. HVS is most sensitive to intermediate ranges of spatial frequencies (around 4-6 cycles/degree), and is less sensitive to spatial frequencies both lower and higher than this. This explains why the metrics and the number of fixations have a higher rank in these depths.

The results of this experimental study showed that two groups of image comparisons exist. The first group consists of comparisons between Cord-256 and Cord-128. For this group, subjects are largely unable to perceive existing differences between the images. All models predict few visually perceivable differences for image pairs in this group.

The second group consists of comparisons between Cord-256 and Cord-64 as well as between Cord-128 and Cord-64. For this group, subjects are largely able to see the differences between the pairs. The models predict a larger number of differences which are also detectable with a higher probability.

Where strong correlations were observed between locations of predicted visually perceivable differences by VDP/CWSSIM and observed fixation patterns, a significant correlation was also observed between subjects' ability to perceive existing differences (the number of correct answers) and the results of VDM and CWSSIM tests (JND).

As observed, all models, expect VDM, are able to detect regions of interest in images. This feature is promising for future research on ROI issues. The computation of VDM, SSIM and CWSSIM does not require

ISBN 978-80-86943-40-4

time consuming Fourier transformations (as VDP does) and they are certainly faster than that of VDP model.

Second, it was observed that all metrics are highly sensitive to small translations, scaling and rotations, which lead to high predicted perceptability values in metrics, even though no quality differences are available in compared images. In the frequency domain, small translations, rotations and scalings lead to consistent phase changes. Due to the fact that VDP works in frequency domains, it reacts with greater sensitivity to geometrical distortions than other metrics. According to [16], this problem can be overcome by analyzing images in complex wavelet domains through Structural Similarity based metrics, but the results were not promising in the case of the presented study.

Another common problem shared by the models is the disregard for color perception by HVS as well as incorporation of just the contrast sensitivity and luminance adaptation. A promising direction in the future would be an analysis of full-colored images.

Additionally, there is a lack of no-reference perceptual picture quality metrics, since both of the metrics are relative (full-reference). It is supposed that more work could be done in the field of no-reference image quality assessment.

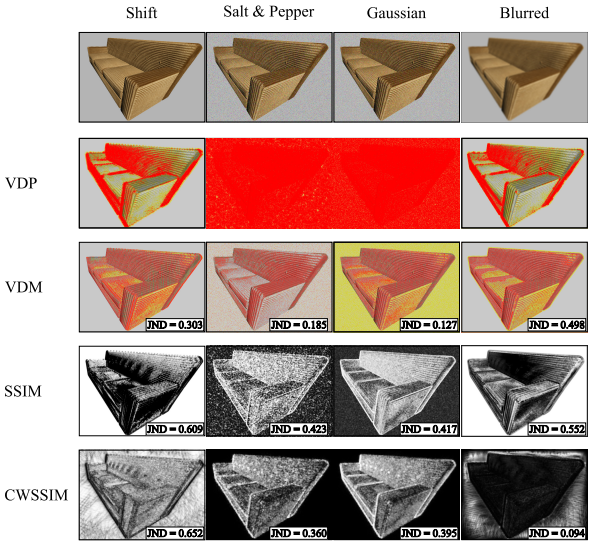


Figure 6: Objective quality metrics responses to shifted, salt & pepper and Gaussian noise contaminated and blurred images.

5 CONCLUSION

In the contribution, we investigated the suitability and integrity of certain image quality metrics, the traditional error-sensitivity and structural based to predict levels of perceptibility for compressed BTF textures. To confirm the validity of obtained results, they were compared

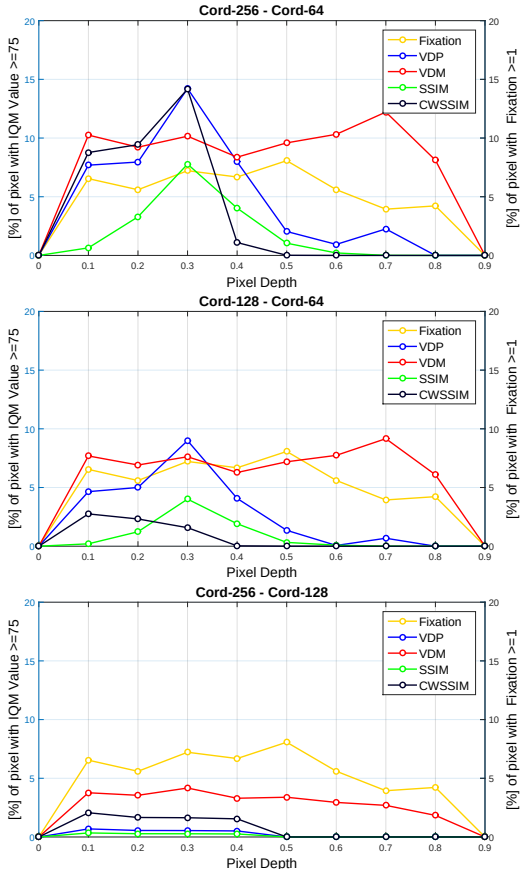


Figure 7: The percentage of fixation in each depth and the responses of VDP and VDM to the pixel depth between *Cord-256 _ Cord- 64* (top), *Cord-128 _ Cord- 64* (middle) and *Cord-256 _ Cord-128* (bottom).

with those obtained by an experimental study. In our validation experiment, it was observed that VDM and CWSSIM can in general better predict the differences between two images. On the other hand, VDP is better able to detect the location of visible differences in images.

Structural based IQMs are able to successfully predict image quality in close agreement with traditional error-sensitivity based IQMs.

The computation time is also another significant factor in image quality assessment, specially so when real-time image resolution changes need to be introduced as per the assessed quality of the rendered scene. In this scenario, all models, except VDP, prove to be proper options. This is because VDM, SSIM and CWSSIM operate in the spatial domain and unlike VDP, do not use the Fourier transform. However, in situations where one needs to improve the image quality of only parts of an object, only VDP can provide enough information on those areas requiring a higher resolution.

As observed, all models, expect VDM, are able to detect regions of interest in images. This feature is promising for future research on ROI issues.

ISBN 978-80-86943-40-4

6 REFERENCES

[1] K. J. Dana, B. Van Ginneken, S. K. Nayar, and J. J. Koenderink, "Reflectance and texture of real-world surfaces," *ACM Transactions on Graphics (TOG)*, vol. 18, no. 1, pp. 1–34, 1999.

[2] J. Filip and M. Haindl, "Bidirectional texture function modeling: A state of the art survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 11, pp. 1921–1940, 2009.

[3] J. Meseth, G. Müller, R. Klein, F. Röder, and M. Arnold, "Verification of rendering quality from measured btfs," in *Proceedings of the 3rd symposium on Applied perception in graphics and visualization*, pp. 127–134, ACM, 2006.

[4] G. Müller, J. Meseth, and R. Klein, "Compression and real-time rendering of measured btfs using local pca," in *Vision, Modeling, and Visualization: Proceedings*, p. 271, AKA, 2003.

[5] J. Filip, M. J. Chantler, and M. Haindl, "On optimal resampling of view and illumination dependent textures," in *Proceedings of the 5th symposium on Applied perception in graphics and visualization*, pp. 131–134, ACM, 2008.

[6] J. Filip, M. J. Chantler, P. R. Green, and M. Haindl, "A psychophysically validated metric for bidirectional texture data reduction.," *ACM Trans. Graph.*, vol. 27, no. 5, p. 138, 2008.

[7] S. Daly, "Digital images and human vision," ch. The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity, pp. 179–206, 1993.

[8] J. Lubin, "A visual discrimination model for imaging system design and evaluation," *Vision models for target detection and recognition*, vol. 2, pp. 245–357, 1995.

[9] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[10] Z. Wang and E. P. Simoncelli, "Translation insensitive image similarity in complex wavelet domain," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, vol. 2, pp. ii–573, IEEE, 2005.

[11] B. Azari, S. Bertel, and C. A. Wuethrich, "A perception-based threshold for bidirectional texture functions," *Proceedings of the 38th Annual Meeting of the Cognitive Science Society, CogSci 2016, Philadelphia , USA, Agust 10-13, 2016*, 2016.

[12] A. B. Watson, "The cortex transform: rapid computation of simulated neural images," *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 311–327, 1987.

[13] R. Mantiuk, S. J. Daly, K. Myszkowski, and H.-P. Seidel, "Predicting visible differences in high dynamic range images: model and its calibration," in *Electronic Imaging 2005*, pp. 204–214, International Society for Optics and Photonics, 2005.

[14] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on communications*, vol. 31, no. 4, pp. 532–540, 1983.

[15] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Transactions on Pattern analysis and machine intelligence*, vol. 13, no. 9, pp. 891–906, 1991.

[16] Z. Wang and A. C. Bovik, "Modern image quality assessment," *Synthesis Lectures on Image, Video, and Multimedia Processing*, vol. 2, no. 1, pp. 1–156, 2006.

[17] J. A. Solomon and D. G. Pelli, "The visual filter mediating letter identification," *Nature*, vol. 369, no. 6479, pp. 395–397, 1994.

[18] I. Ohzawa, G. C. DeAngelis, R. D. Freeman, *et al.*, "Stereoscopic depth discrimination in the visual cortex: neurons ideally suited as disparity detectors," *Science*, vol. 249, no. 4972, pp. 1037–1041, 1990.

[19] O. Schwartz and E. P. Simoncelli, "Natural signal statistics and sensory gain control," *Nature neuroscience*, vol. 4, no. 8, p. 819, 2001.

[20] K. Myszkowski, "The visible differences predictor: Applications to global illumination problems.," *Rendering Techniques*, vol. 98, pp. 223–236, 1998.

[21] H. Wilson, "Psychophysical models of spatial vision and hyperacuity," *Vision and Visual Dysfunction*, vol. 10, pp. 179–206, 1991.

[22] O. Le Meur, P. Le Callet, D. Barba, and D. Thoreau, "A coherent computational approach to model bottom-up visual attention," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 5, pp. 802–817, 2006.

[23] L. Itti and C. Koch, "Computational modelling of visual attention," *Nature reviews neuroscience*, vol. 2, no. 3, pp. 194–203, 2001.

[24] D. Walther, "Planes of the head." `http://www.saliencytoolbox.net/`, 2006.

[25] B. Li, G. W. Meyer, and R. V. Klassen, "Comparison of two image quality models," in *Human Vision and Electronic Imaging III, San Jose, CA, USA, January 24, 1998*, pp. 98–109, 1998.

 ISBN 978-80-86943-40-4

ISSN 2464-4617(print)
ISSN 2464-4625(CD)
Computer Science Research Notes
CSRN 2801
Full Papers Proceedings
http://www.WSCG.eu

Barycentric Combinations Based Subdivision Shaders

Lucas Morlet, Marc Neveu, Sandrine Lanquetin, and Christian Gentil
Laboratoire Electronique, Informatique et Image
UFR Sciences et Techniques, allée Alain Savary, 21000 Dijon, France
{lucas.morlet, marc.neveu, sandrine.lanquetin, christian.gentil}@u-bourgogne.fr

ABSTRACT

We present a new representation of uniform subdivision surfaces based on Iterated Functions Systems formalism. Main advantages of this new representation are the formalization of topological subdivision, multiscale representation of limit surface, separation of iterative space where the attractor is computed once for all and modeling space where the attractor is projected many times. An important consequence of this approach is that all uniform subdivision schemes are handled in the same way whatever there are primal or dual, approximating or interpolating.

Subdivision surfaces are no longer viewed as a set of rules but as a list of barycentric combinations to apply on neighborhoods of the coarse mesh. These combinations are representative subsets of the attractor which is deduced from a Controlled Iterated Functions System automaton. From this new point of view we present in this paper a straightforward implementation to directly compute a tessellation of the subdivision surface from a control mesh. This implementation takes full advantage of Graphics Processing Units high capability of computation and Tessellation Stage of OpenGL/GLSL rendering pipeline to generate on the fly a tessellation of the limit surface with a chosen Level of Details.

Keywords

Shape Modeling, Subdivision Surfaces, Iterated Functions Systems, Tessellation Shaders

1 INTRODUCTION

With their overpowered capability of computation, Graphics Processing Units (GPUs) became an unavoidable tool for parallel computing in the last ten years. Whenever a big amount of independent computations are required, an implementation GPU-based must be prioritized over Central Processing Unit (CPU) computations. During the graphics pipeline, several usual steps are highly parallelizable : the vertices positioning and the fragment colorization and blending.

Unfortunately, GPU dedicated memory is limited and transfer times may be longer than computation itself so transmitted information should be as compact as possible. A usual solution is to compress the information before sending them to the GPU and decompress it on the fly during the rendering. This kind of solution reduces occupied memory space and information transfer time but computations are added onto the GPU ; a compromise must be found to maximize the efficiency.

The main scope of this article is on the fly generation of geometry for subdivision surfaces. From a given control mesh, a tessellation of the limit surface is directly computed for a chosen Level of Details (LoD) by applying precomputed barycentric combinations on each patch of the coarse mesh. Patches are defined as in [Sta98] and combinations are deduced from a Controlled Iterated Function System (CIFS) automaton. Indeed barycentric combinations are a representative subset of the attractor of the CIFS which is unique in iterative space but can be projected multiple times in modeling space.

Usually, subdivision surfaces are computed with a set of subdivision rules, which are different for each scheme, applied iteratively on a control mesh. Thanks to this new representation, steps which are blended in these rules are separated : first the iterative process represents by the generation of barycentric combinations with a CIFS automaton, then the projection in modeling space which is the application of combinations on a patch. This separation enables to implement all uniform subdivision schemes in the same way only by changing the input list of precomputed barycentric combinations and the connectivity of input patches.

Another interest of our method is that a vertex position does not depend on the level of tessellation: whatever the chosen LoD, the vertex always belongs to the limit surface. This property is very useful in the CAGD context where usual tools work on limit surfaces rather than meshes. Moreover, isogeometric analysis using subdivision surfaces is more and more often integrated in CAD systems [PXXZ16] [BHU10]. This analysis relies on a compatible representation for geometric modeling and finite element simulation. A general representation of meshes and surfaces, with different LODs is highly recommended to handle isogeometric problems. Our model is quite well adapted for this purpose.

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.6

2 RELATED WORKS

Many subdivision schemes have been proposed in the last forty years as [CC78], [DS78], or [Loo87] for instance. From the beginning, authors have paid great attention to the limit surface. Reif [Rei95] proposed a general method to study convergence to the limit surface. Halstead [HKD93] and Stam [Sta98] gave methods to directly compute points on the limit surface. To cover the field of NURBS, extensions have been proposed to cope for non uniform cubic schemes, first by Sederberg [SZSS98], extended by Müller [MRF06] and to high degree surfaces by Cashman [CADS09]. All these schemes present a careful study of the convergence to the limit surface.

Other consideration has been paid on Iterative Function Systems (IFS). In [ZT96], the distinction between the iteration space, where attractors are defined, and the modeling space, where shapes are modeled, enables the generalization of IFS modeling and the definition of attractor projection. Free form fractals and usual free forms curves and surfaces (Bézier, uniform B-splines) can also be defined as an IFS. A link between IFS and subdivision surfaces has been proven very early by Warren and Weimer [WW01] and Shaefer et al [SLG05]. To our knowledge, the interest of the separation between iteration and modeling spaces has not been fully explored for subdivision surfaces.

Indeed, whatever the scheme, different aspects of subdivision surfaces are usually blended in subdivision rules: the connectivity of the control mesh, the subdivision in the parameterization space and the corresponding subdivision in the geometrical space. On the other hand, by using the IFS formalism, we can clearly separate these three aspects and enlarge the possibilities of subdivision surfaces processing. An example can be find in [PGSL14], where Podkorytov builds junctions between two subdivision surfaces, one defined from a primal scheme and the other from a dual scheme.

In video game context, real-time rendering of subdivision surface is an important challenge. Some works [NLMD12][BFK+16] devoted to GPU implementation focus on tessellation. Those works are mainly based on Stam's method [Sta98] with different adaptations. But, they essentially deal with Catmull-Clark subdivision without general formalism. To our knowledge, no proof of a possible extension of their methods to other schemes has been produced.

3 OVERVIEW

To begin, the Iterated Function Systems formalism is briefly presented in Section 4. In particular addresses notion and its influence on equivalence between parametric and barycentric spaces are highlighted as well as projection from barycentric to modeling space.

Then our method is explained step-by-step for the well-known Catmull-Clark subdivision scheme [CC78] in Section 5. First the mesh is cut in several patches. Then barycentric combinations are computed for both regular and irregular cases. To finish, corresponding combinations are applied on each patch to compute the limit surface of the control mesh.

In Section 6, application of our method on several subdivision schemes is presented. Doo-Sabin [DS78], Loop [Loo87], and Simplest [PR97] schemes are handled in the same way as Catmull-Clark scheme with differences only on patches construction and coefficients of barycentric combinations.

A simple and efficient implementation of our method is precisely described in Section 7. It relies on the OpenGL/GLSL rendering pipeline, in particular on the Tessellation Stage which is the core of our implementation. Thanks to our formalism, all subdivision schemes are handled in the same graphics pipeline. This implementation also manages mesh animation, trimming, and dynamic LoD without distinction between subdivision schemes. Some results are presented and comparisons with previous method are discussed.

In Section 8, a problem which appears along with patches size expansion is presented. High-degree or approximating schemes as Butterfly [DLG90] and Quads-interpolating [Kob96] are used as examples to highlight the combinatorial issue.

4 ITERATED FUNCTION SYSTEMS

Iterated Function Systems (IFS) [Hut81][Bar14] are a very efficient tool to represent self-similar objects like fractals, Bézier surfaces, B-spline surfaces... A self-similar object F is defined as an object composed of smaller copies of itself, which can be formalized by : $F = \bigcup_{i=0}^{N-1} T_i(F)$. Each transformation $T_i \in T$ maps F on a subpart of itself. The set of transformations $T = \{T_0 \dots T_{N-1}\}$ is called an IFS. The fixed point F is called the attractor associated to the IFS. According to some conditions, each transformation T_i is contractive for instance, F can be approximated from any initial compact set K_0 : $K_n = \bigcup_{i=0}^{N-1} T_i(K_{n-1})$.

Every point of the attractor corresponds to an infinite sequence of transformations. The sequence of transformation indexes is called the address of the point. The simplest way to obtain points belonging to the attractor is to compute the fixed point P_i of each contractive transformation T_i of the IFS. In the case of affine or linear contractive transformations this can be done by linear algebra. The address of P_i is i^ω (an infinite sequence of i). This means that any point of the attractor, whose address is σi^ω (where σ is a finite word) can be computed in a finite time.

 ISBN 978-80-86943-40-4

The address function ϕ, mapping an address to its corresponding point, defines a continuous parameterization of the attractor. Given two IFS $\mathscr{I}$ and $\mathscr{I}'$ with the same joining conditions (see [ZT96]), $\phi' \circ \phi^{-1}$ defines a morphism between the two attractors. The first attractor may be used as a parametric space. For a quadrangular surface (like a Bézier Patch), the first attractor could be the unit square defined with an appropriate IFS. The computation of ϕ^{-1} can be achieved with *the escape algorithm* [Bar14].

By choosing the right parameterizations, in a way their addresses have the form σi^{ω}, uniformly spaced and cover the whole parametric space, a tessellation of the attractor can be computed. The maximal authorized length of σ is the level of tessellation of the attractor. The higher it is, the closer from the real attractor the tessellation is. Examples of second tessellation level for two IFS in the parametric space are given in Figure 1.

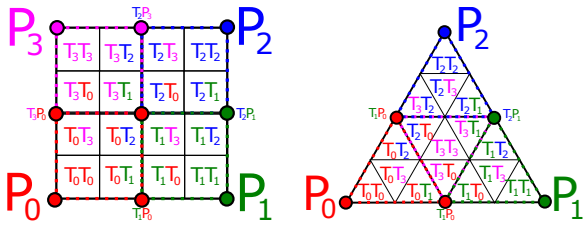


Figure 1: Two examples of IFS in parametric space : on the left, the unit square is cut in four squares ; on the right the "unit" equilateral triangle is cut in four equilateral triangles. Both spaces have four transformations labeled $T_0 \dots T_3$ and associated fixed points $P_0 \dots P_3$

Usually the iterative space, where a tessellation of the attractor is computed, is the modeling space, where the attractor is displayed. In our case, the iterative space is a barycentric space, where every point is a barycentric combination so that the sum of its coordinates is 1. The computed tessellation is then projected in the modeling space by applying the generated combination onto a control mesh. The whole process is summarized in Figure 2.

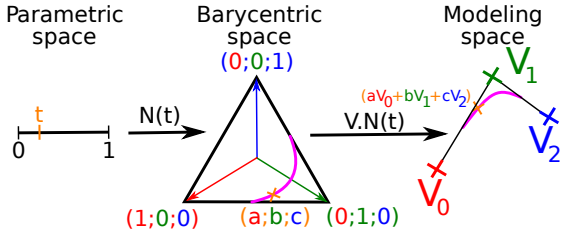


Figure 2: Morphisms between parametric, barycentric, and modeling spaces for a uniform quadratic B-Spline. N is the function that associate a point of the parametric space to the point of barycentric space of same address. V is the projection of a barycentric combination onto a control polygon

The subdivision process consisting in application of the same set of transformations at each level of iteration can be controlled using an automaton describing which transformations can be apply at each state. Such IFS are called Controlled Iterated function Systems (CIFS). The notion of address remains the same and correspond to the set of words accepted by the automaton.

Since subdivision surfaces are iterative models, an IFS and so a CIFS automaton can be created to generate them. For a given list of parametric points, addresses are generated. These addresses are words which are read by the CIFS automaton to compute a set of combinations in a barycentric space. Resulting combinations are then projected onto a control mesh : a tessellation of the attractor, which is the limit surface in the subdivision surface case, is computed.

5 CATMULL-CLARK

For sake of clarity, we first present our method with the well-known Catmull-Clark scheme.

5.1 Patch construction

A *mesh patch* (denoted hereafter *patch* for short) is a set of vertices necessary and sufficient to compute *a piece of the limit surface*. No confusion must be made between these two terms. According to the subdivision scheme, patches have different connectivities. In the Catmull-Clark case, patches are composed of a central face surrounded by a ring of faces [Sta98]. Some examples can be found in Figure 3.

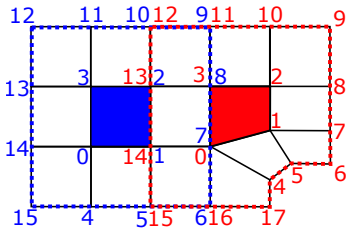


Figure 3: Two examples of Catmull-Clark patches and their indexation. In blue a regular patch (with all the central vertices of valence 4) and in red an irregular patch (with a central vertex of valence 5). Notice that the irregular central vertex, which is unique, is always indexed as 0.

Every central vertex of a regular patch has a regular valency. On the other hand, a patch containing at least one extraordinary vertex (with irregular valency) in its central face is considered as irregular. In this paper, the number of extraordinary vertices per face is limited to a single one to avoid a combinatorial explosion of the number of cases.

When a patch is subdivided, with respect to usual subdivision method, several subpatches are obtained. Each subpatch is also a patch and so can be subdivided at its turn. Every subdivision matrix associates a patch to one of its subpatches. It can be easily deduced from subdivision rules. The set of these transformations is a CIFS

ISBN 978-80-86943-40-4

whose the attractor is the piece of the limit surface defined by the patch. This CIFS depends on the valency of the irregular vertex, if it exists. For the sake of simplicity, regular patches are treated first, then generalized to irregular patches.

5.2 Regular patches

In the regular case, a Catmull-Clark subdivision surface is identical to a bicubic uniform B-Spline surface. So regular patches are grids of 4 by 4 vertices. Subpatches are four in number and each one is also a regular patch. The regular patch subdivision is presented in Figure 4.

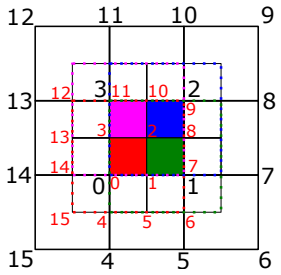


Figure 4: The regular patch subdivision of Catmull-Clark scheme into four regular subpatches. Notice that vertices are indexed in the same way in subpatches as in their parent patch.

Four square subdivision matrices $M_{i\in[0;3]}$ can be created to transform the parent patch in each regular subpatches. The M_0 subdivision matrix is given as an example in Figure 5.

$$M_0 = \left(\begin{array}{cccc|cc|cc|cc|ccc}
V & E & F & E & E & F & & & & & & F & E & F \\
B & B & C & C & C & C & & & & & & & & \\
A & A & A & A & & & & & & & & & & \\
B & C & C & B & & & & & & & & C & C & \\
\hline
B & C & & & B & C & & & & & & & C & C \\
A & A & & & A & A & & & & & & & & \\
C & B & & & C & B & C & C & & & & & & \\
E & V & E & F & F & E & F & E & F & & & & & \\
\hline
C & B & B & C & & & & C & C & & & & & \\
F & E & V & E & & & & F & E & F & E & F & & \\
C & C & B & B & & & & & & C & C & & & \\
E & F & E & V & & & & & & F & E & F & & \\
\hline
C & & & B & & & & & & & C & C & B & C \\
A & & & A & & & & & & & & A & A & \\
B & & & C & C & & & & & & & C & B & C \\
A & & & & A & & & & & & & & A & A
\end{array}\right)$$

$$A = \frac{1}{4} \quad B = \frac{3}{8} \quad C = \frac{1}{16}$$

$$V = \frac{9}{16} \quad E = \frac{3}{32} \quad F = \frac{1}{64}$$

Figure 5: The M_0 subdivision matrix for the Catmull-Clark subdivision scheme. Null coefficients are omitted.

These subdivision matrices are stochastic and have a unique eigenvalue equals to 1 and all others in interval $[0;1[$. This implies that the associated transformation is contractive and so apply an infinity of times the transformation on a patch will end in a unique fixed-point. As proven in [HKD93], this fixed-point can be computed directly by using the left eigenvector associated to the eigenvalue 1 as a barycentric combination B_i which associates a patch to a point of the limit surface.

From every given a point $(u;v) = T_aT_b \dots T_yP_z$ in the parametric space, its address is used to compute the corresponding barycentric combination $B_zM_y \dots M_bM_a$. This combination transforms a regular patch into the point of the limit surface of the local parameterization $(u;v)$. Computations of combinations can be expressed as a CIFS automaton whose an example is given in Figure 6.

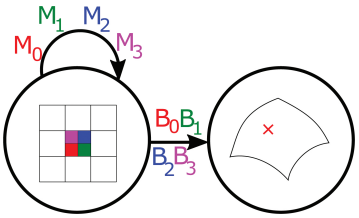


Figure 6: The CIFS automaton for the regular case of Catmull-Clark subdivision scheme.

For a given Level of Details, several parametric points are chosen to represent a discretization of parametric space. These points must be uniformly spaced, cover the whole parametric space, and correspond to a finite address. Then each barycentric combinations associated to these points are computed. Apply all these combinations on the same patch create a tessellation of the limit surface with the chosen Level of Details.

5.3 Irregular patches

A major interest of subdivision schemes is the management of irregular connectivity. For example, bicubic uniform B-Spline surfaces can not be computed on a non-grid mesh. Catmull-Clark scheme, in addition to generate bicubic uniform surfaces in regular case, permits to generate C1-continue surfaces around extraordinary vertices.

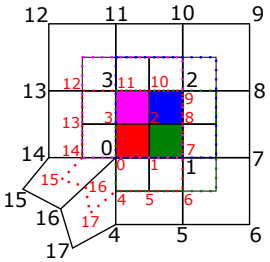


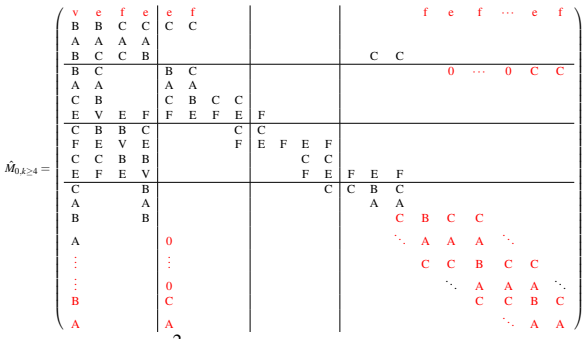
Figure 7: An irregular patch (valence 5) subdivision of Catmull-Clark scheme into four subpatches. Notice that the red subpatch has the same connectivity as its parent patch whereas the other ones become regular patches.

In presence of extraordinary vertices, patch connectivity and subdivision rules are different, so some barycentric combinations should be recomputed. As showed in Figure 7, subpatches connectivities are also different.

As long as the applied transformation is the one centered on the extraordinary vertex, the subpatch keeps the connectivity of its parent. As soon as another transformation is applied, the subpatch becomes regular.

ISBN 978-80-86943-40-4

To manage irregular transformations, irregular subdivision matrices $\hat{M}_{i,k}$, which depend on the valency k of the unique extraordinary vertex have to be introduced. $\hat{M}_{i>0,k}$ matrices are not so different from $M_{i>0}$ except there are not square anymore. On the other hand, $\hat{M}_{0,k}$ is square but depends on the valency k of the extraordinary vertex. This matrix is presented in Figure 8.



$$v = \frac{4k^2-7k}{4k^2} \quad e = \frac{6}{4k^2} \quad f = \frac{1}{4k^2}$$

Figure 8: The $\hat{M}_{0,k>4}$ subdivision matrix for the Catmull-Clark subdivision scheme. Black coefficients are the fixed ones and the red depends on the valency of the extraordinary vertex. A, B, C, V, E, and F coefficients are the same as in Figure 5.

Concerning computations of fixed points, $\hat{B}_{i>0,k}$ do not need to be computed because there are equal to $B_i\hat{M}_{i,k}$. Conversely every $\hat{B}_{0,k}$ has to be computed : it is the eigen-vector associated to the eigen-value 1 of $\hat{M}_{0,k}$. Everything is summerized in the automaton of irregular case given in Figure 9.

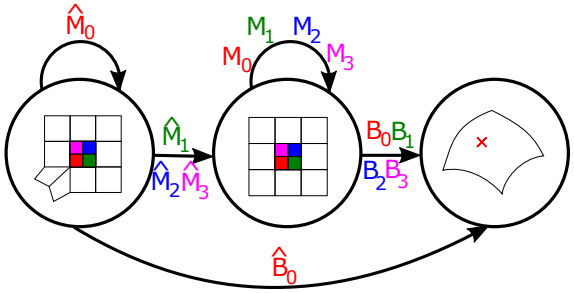


Figure 9: The CIFS automaton associated to irregular cases of Catmull-Clark subdivision scheme.

6 OTHER SUBDIVISION SCHEMES

As said before, our formalism handles any uniform subdivision schemes. In this section, some usual schemes are presented as CIFS automata.

6.1 Loop scheme

As for the Catmull-Clark scheme, the Loop subdivision scheme [Loo87] creates C2-continue surfaces. Patches are created in the same way : a central face and a ring of faces as shown in Figure 10.

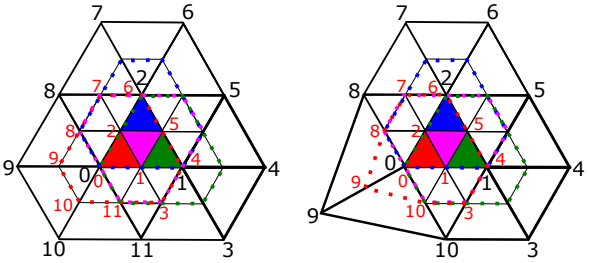


Figure 10: Patches and subpatches of the Loop subdivision scheme for the regular case (valence 6) and an irregular case (valence 5). Notice that the red subpatch is connectively-identical to the parent one and the other three are regular.

The structure of CIFS automaton associated to the Loop scheme is identical to the Catmull-Clark one. There are three states which correspond to irregular patches, the regular patch, and limit surface points and the transitions are labeled exactly in the same way.

Since the subdivision rules of the two schemes are different, the transformation matrices associated to the transitions are not the same in the two automata. Subdivision of the parametric space is also different so addresses are generated differently as shown on the right of Figure 1.

6.2 Doo-Sabin scheme

Unlike Catmull-Clark and Loop subdivision schemes, which are primal, Doo-Sabin [DS78] is a dual subdivision scheme which means that faces are not subdivided anymore but vertices. So corresponding patches are composed of a central vertex and a ring of adjacent faces. The valency of all the vertices is 4 ; irregular patches contain a unique non-quadrilateral face as shown in Figure 11.

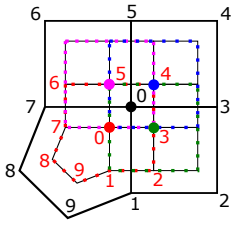


Figure 11: Irregular patch (with a pentagonal face) of the Doo-Sabin subdivision scheme and its four subpatches. Once more, the red subpatch is connectively identical to the parent one and the three others become regular.

At this point it is important to say that Doo-Sabin and Catmull-Clark subdivision schemes are identical from a topological subdivision point of view : it is a quadrangular subdivision with the same kind of connection between subdivided faces. As for the Loop scheme, the CIFS associated to Catmull-Clark and Doo-Sabin subdivision schemes have the same structure (states and transition labels). The only difference to highlight is the difference of transformation matrices.

ISBN 978-80-86943-40-4

6.3 Simplest scheme

Introduced by Peters and Reif [PR97] the Simplest subdivision scheme, also called Midedge scheme, inserts a vertex in the middle of each edge and creates new edges between new vertices. Then old vertices and edges are deleted.

As shown in Figure 12, applying the Simplest subdivision scheme twice is connectively-identical with Doo-Sabin but with different coefficients. So the CIFS automaton associated to the Simplest scheme is the same as Doo-Sabin one but coefficients of transformation matrices are different.

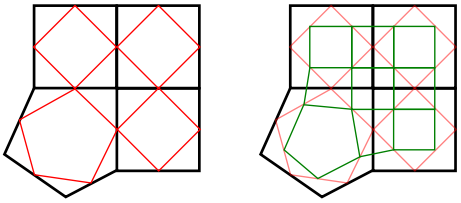


Figure 12: Subpatches of the Simplest subdivision scheme after one (red) and two (green) subdivisions. After two subdivisions, the result is connectively-identical with the Doo-Sabin scheme (see Figure 11).

7 IMPLEMENTATION

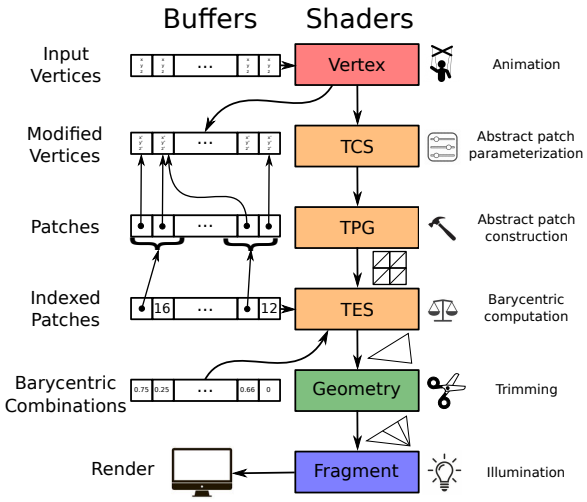


Figure 13: Implementation overview.

In this section, we suggest an implementation of our method in OpenGL/GLSL. Because our implementation needs Shader Storage Buffer Objects, the minimum version required for both core version is 4.3 or 4.0 with ARB_shader_storage_buffer_object extension. An overview is given in Figure 13.

7.1 First steps

For a given maximal Level of Details and a minimal and maximal valence of extraordinary vertices authorized, all combinations are computed for a chosen subdivision scheme. Then meshes are cut in several patches respecting inherent rules of the scheme. These two steps are done once for all and results are written in buffers.

7.2 Buffers

Buffers require to have an array of arbitrary length so they must be Shader Storage Buffer Objects (SSBOs). Referring to OpenGL specifications, SSBOs reads and writes use incoherent memory accesses and guarantee a minimum possibility of memory allocation of 128MB. Most GPU-implementations enable allocating a size up to the limit of GPU memory.

Vertices buffers

There are two vertices buffers : the *Input Vertices Buffer* which contains all vertices of the input control mesh and the *Modified Vertices Buffer* which contains the vertices after modification by the Vertex Shader.

Patches buffers

The GLSL Tessellation Shaders does not handle dynamic patch size. To bypass this constraint, two patches buffers are created. The first one, called *Patches Buffer*, contains all mesh patches written in a row, each patch containing index of its vertices. The second one, the *Indexed Patches Buffer*, indexes patches of *Patches Buffer* by a pointer to the begin and the length of each mesh patch.

Barycentric Combinations Buffer

For a given valence of extraordinary vertex, each barycentric combination contains as many coefficients as vertices in patch. The number of combinations of each patch size depends on the chosen maximal Level of Details. The *Barycentric Combinations Buffer* is a row containing all coefficients of all combinations of each kind of patch. By knowing the maximal Level of Details, the valence of extraordinary vertex, and the parameterization $(u;v)$ of the current limit surface point, the index of associated barycentric combinations can be computed.

7.3 Programmable rendering pipeline

Once all buffers are filled, the usual OpenGL/GLSL rendering pipeline with the succession of shaders is performed.

Vertex Shader

This shader treats one by one all the vertices of *Input Vertices Buffer* and write them into the*Modified Vertices Buffer*. Vertices are ordered in the same way in both

ISBN 978-80-86943-40-4

buffers. Vertex Shader can be simply a pass-through shader by copying the *Input Vertices Buffer* into the *Modified Vertices Buffer* but can also be the step where mesh is animated. Animation is discussed in Subsection 7.4.

Tessellation Control Shader (TCS)

This shader configures the Tessellation Primitive Generator (TPG) which creates an *abstract patch* and transfers it to Tessellation Evaluation Shader. An *abstract patch* is a set of parametric points connected with triangles. These triangles are chosen by the TPG in a way to cover all the parametric space. The parametric space can be of two types depending on the *abstract patch* configuration : the unit square or the "unit" equilateral triangle (cf Figure 1). The TCS describes how many points of the *abstract patch* are on each bounding-edge of parametric space and how many interior rings are created but can not choose how to rely them. The more detailed the limit surface is desired, the more important is the number of inserted points. Level of Details can be the same everywhere or adaptive and chosen on the fly for each patch. More information about adaptive LoD is given in Subsection 7.5.

Tessellation Evaluation Shader (TES)

This shader is the core of our implementation. From one side, it reconstitutes the current mesh patch by reading *Indexed Patches Buffer* which points to *Patches Buffer* which contains indexes of vertices in *Modified Vertices Buffer*. From the other side it reads the parameterization of all points of the *abstract patch* and deduced a list of associated barycentric combinations. To finish, it applies every combination on the patch to compute points of limit surface and rely them as describes by the *abstract patch*.

Geometry Shader

This shader takes an OpenGL primitive (a triangle in this case) as input and emits zero or more primitives (triangle-strip). It can be simply a shader which applies the Model-View-Projection matrix on each received triangle but can also be used to trim the limit surface. Subsection 7.6 is dedicated to the trimming of the limit surface by parametric space restriction.

Fragment Shader

As usual, this shader is in charge of coloration and illumination, with a Phong model for example. As proven in [BGN09], normal associate to every point of thelimit surface is the cross-product of the two half-tangents computed for the address of the point.

7.4 Animated meshes

The main idea is instead of animating the subdivide mesh, the coarse mesh is animated and then subdivided. Animation time is greatly reduce but time is spent to generate geometry. In the common case of animation by skeleton with an average of two bones by vertex geometry generation is slightly less two times longer than animation. Resulting surfaces are not the same in both methods : subdivide then animate deforms the limit surface whereas animate then subdivide assures the conservation of limit surface continuity.

7.5 Adaptive Level of Details

An efficient method to avoid the time-consuming display of very detailed distant objects is the LoDs strategy : the closer the object is, the more detailed it is ; the farther it is, the faster it is displayed. In most implementations, several meshes, with different LoDs, are generated and one of them is picked up at display time.

The limit of this approach is that the different meshes have to be predefined. On the contrary, with subdivision surfaces, only the coarse mesh is defined and the LoDs gives the number of subdivision. Iteratively generating geometry on the fly can be quite long with a multipass render. With our method, geometry is not iteratively generated but directly only by selecting the appropriate set of barycentric combinations to apply on the mesh patch.

Another advantage of our method is that an object does not need to be uniformly subdivided (i.e. all faces are subdivided the same number of times). Each face is independently treated from the others, with its own tessellation that can be non-uniform. To avoid cracks, a condition has to be imposed : even faces can be subdivided as wanted, edges are subdivided in the same way on both sides.

A naive method to ensure respect of edge uniform subdivision is to project every face of control mesh in image plane. Each edge is subdivided in function of its screen length and face is subdivided in function of its screen area. The OpenGL's invariance rules insures that multiple projection of the same edge results in the same screen length and so same subdivision for both adjacent sides.

7.6 Trimming

Trimming is a restriction of the parametric space that results in a restriction (holes for instance) on the limit surface. Quality of trimming is strongly dependent on the level of tessellation. To enhance quality, trimmed faces have to be more tessellated. Thanks to dynamic LoDs, untrimmed faces do not need to be as tessellated as trimmed ones.

 ISBN 978-80-86943-40-4

First, tessellation is traced in parametric space. If a point is outside the restriction it is tagged as valid otherwise not. For every triangle of tessellation, the number of valid vertices is counted. Each case is treated differently. All of them are presented in Figure 14.

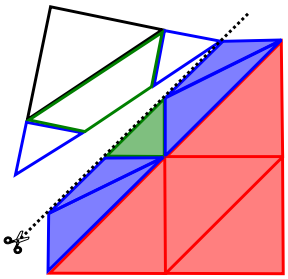


Figure 14: Example of trimming on a parametric space tessellated by 8 original triangles represented by their colored bounds. New triangles are filled if they are displayed, empty if not. Red triangles have three valid vertices so they are displayed as is. Blue triangles have an invalid vertex which becomes a vertex on each adjacent edge and the quadrilateral result is split into two triangles. Green triangle has two invalid vertices so both are recomputed. Black triangle is completely included in the restriction so they are discarded.

The intersections between the triangles edges are new point of parametric space. New points means new addresses and so new barycentric combinations. In order to get valid trimmed boundary points, new combinations have to be computed and applied on the patch.

7.7 Results and performance

In this section, several methods are compared in term of occupied memory space and computation time for the regular case of the Catmull-Clark subdivision scheme. These methods are three in number :

1. LoD meshes : one mesh is defined for each LoD. Only one is picked up and display.
2. Iterative subdivision : only patches of coarse mesh are transfered to the GPU and geometry is generated iteratively with respect to subdivision rules.
3. Ours : as Iterative subdivision, only patches are transfered to the GPU but geometry is generated directly by applying barycentric combinations.

Let n the maximal LoD, V_i the number of vertices and F_i the number of faces/patches of the mesh associated to the LoD of i. k is the number of existing valences. Two tables are given to compare these three methods in term of occupied memory space (Table 1) and computation time (Table 2).

Methods	LoD	It. sub.	Ours
Vertices	V_n	V_0	V_0
Indexed vertices	$4\sum_{i=0}^{n} 4^i F_0$	$18F_0$	$18F_0$
Combinations	0	0	$k*(2^n+1)^2$

Table 1: Comparisons of occupied memory space between the three methods. A face is defined by 4 pointers to vertices and an average patch by 16 pointers to vertices plus 2 pointers to index the patch.

Methods	LoD	It. sub.	Ours
Animated vertices	V_n	V_0	V_0
Subdivision	0	$\sum_{i=0}^{n-1} F_i$	0
Combinations	0	0	$(2^n+1)^2 F_0$

Table 2: Comparisons of computation time between the three methods. Animation of a vertex costs 25 basic operations (times or plus) per bones (matrix-vector product of size 4). A subdivision corresponds to the computation of 1 face-vertex, 4 edge-vertices and 4 vertex-vertices so 267 basics operations. Application of a barycentric combination costs an average of 93 basics operations (application of a 16 coefficients combination onto a patch of three coordinates vertices).

Name	Gaussian	Tetris	Head	Body
Vertices	100	74	4276	13 652
Faces	81	72	4249	13 650
LoD = 0	0.26	0.27	0.79	1.82
LoD = 1	0.26	0.28	1.24	3.27
LoD = 2	0.29	0.32	3.41	10.2
LoD = 3	0.42	0.57	16.4	47.6
LoD = 4	1.06	1.48	62.5	≈200
LoD = 5	2.53	3.58	167	≈500

Table 3: Computation times (in ms) of our method for different meshes and LoD. Tests are performed on a Dell Precision T7600 : Intel Xeon CPU E5-2609 @ 2.40 GHz x8 and a Nvidia Quadro K2000/PCIe/SSE2.

In term of occupied memory space, subdivision methods are obviously better than the LoD method because only the coarse mesh is needed. Because barycentric combinations have to be loaded in the memory, our method required a little more space than iterative method.

In term of computation time, our method is faster than iterative method because application of barycentric combination requires less computation than subdivision. On another side, our implementation is slower than LoD methods because animation is approximatively two times quicker than generation. Even if dynamic Level of Details permits to reduce the gap between the two methods, LoD method is still the fastest one.

Our method has been tested on some meshes : results are presented in Figure 15 and computation times are compared in Table 3.

ISBN 978-80-86943-40-4

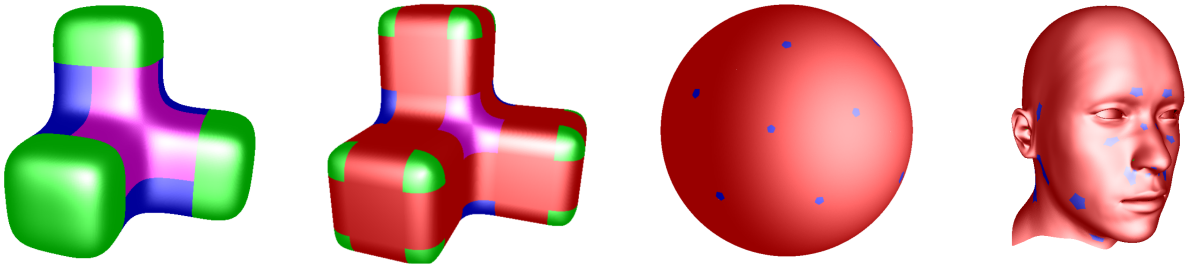


Figure 15: From left to right : Catmull-Clark and Doo-Sabin schemes applied on Tetris meshes ; Loop scheme applied on an Icosphere and Catmull-Clark scheme applied on Human Head mesh enlightened by a Blinn-Phong shader. Red faces correspond to regular patches, green to extraordinary valence of 3, blue of 5 and magenta of 6.

8 LIMITS

Even if our formalism can handle every uniform subdivision scheme, for certain schemes a combinatorial issue appears because of many possibilities of irregularities. In this case, the CIFS automaton still contains a unique regular state but also several irregular states that must be treated separately. Some examples are given in this section.

8.1 High degree schemes

The only condition for building patches is : "at most one extraordinary vertex inside a valid patch" (there is no condition on the exterior ring). The higher the degree of the surface is, the larger the patch is and so more possibilities of irregularities appear as shown in Figure 16.

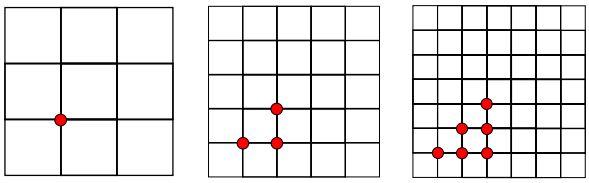


Figure 16: From left to right : subdivision patch for bicubic subdivision (Catmull-Clark), biquintic, and biheptic. Red vertices are the maintained possibilities of extraordinary vertices after reduction by symmetry and rotation. Notice the number of red vertices increases with the degree of surface.

A recurrence appears for odd degree regarding these three examples : the number of possibilities of extraordinary vertices for a surface of degree d is the sum of integer from 1 to $(d-1)/2$.

8.2 Interpolating schemes

Interpolating schemes need larger patches than approximating schemes so they suffer from the same issue as the high-degree schemes. For example Butterfly [DLG90] and Quads-interpolating [Kob96] schemes need a supplementary ring with respect to Catmull-Clark and Loop schemes in order to build a mesh patch. These patches and their subpatches are presented in Figure 17.

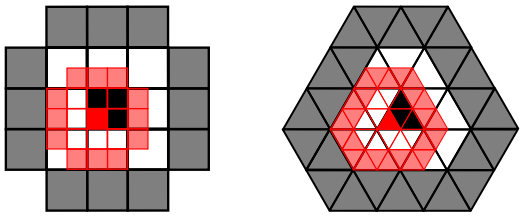


Figure 17: The parent patch and one of its subpatches for the regular case of Quads-interpolating (left) and Butterfly (right) subdivision schemes. Patches are centered on the filled face and the supplementary faces (compared to corresponding approximating schemes) are transparent.

9 CONCLUSION

In this article, subdivision surfaces are presented in the Controlled Iterated Functions Systems formalism. In this formalism, subdivision schemes are not viewed as a set of rules anymore but as a list of barycentric combinations. From this new point of view, all the uniform schemes, whatever they are approximating or interpolating, primal or dual, are handled in the same way. Usual tools as trimming and multi-resolution render are also independent from the chosen scheme.

An implementation based on this formalism is also suggested. This implementation generates directly and on the fly the limit surface from a control mesh faster than usual iterative subdivision surfaces but required a little more memory space. Compared to a usual LoD approach, our method is slower but saves a lot of memory space.

10 REFERENCES

[Bar14] Michael F Barnsley. *Fractals everywhere*. Academic press, 2014.

[BFK+16] Wade Brainerd, Tim Foley, Manuel Kraemer, Henry Moreton, and Matthias Nießner. Efficient gpu rendering of subdivision surfaces using adaptive quadtrees. *ACM Trans. Graph.*, 35(4):113:1–113:12, July 2016.

ISBN 978-80-86943-40-4

[BGN09] Hicham Bensoudane, Christian Gentil, and Marc Neveu. Fractional half-tangent of a curve described by Iterated Function Systems. *Journal of Applied Functional Analysis*, 4(2):311–326, April 2009.

[BHU10] Daniel Burkhart, Bernd Hamann, and Georg Umlauf. Iso-geometric Finite Element Analysis Based on Catmull-Clark Subdivision Solids. *Computer Graphics Forum*, 2010.

[CADS09] Thomas J Cashman, Ursula H Augsdörfer, Neil A Dodgson, and Malcolm A Sabin. Nurbs with extraordinary points: high-degree, non-uniform, rational subdivision schemes. In *ACM Transactions on Graphics (TOG)*, volume 28, page 46. ACM, 2009.

[CC78] Edwin Catmull and James Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design*, 10(6):350–355, 1978.

[DLG90] Nira Dyn, David Levine, and John A Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM transactions on Graphics (TOG)*, 9(2):160–169, 1990.

[DS78] Daniel Doo and Malcolm Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978.

[HKD93] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using catmull-clark surfaces. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 35–44, New York, NY, USA, 1993. ACM.

[Hut81] John E Hutchinson. Fractals and self similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.

[Kob96] Leif Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. In *Computer Graphics Forum*, volume 15, pages 409–420. Wiley Online Library, 1996.

[Loo87] Charles Loop. Smooth subdivision surfaces based on triangles. 1987.

[MRF06] Kerstin Müller, Lars Reusche, and Dieter Fellner. Extended subdivision surfaces: Building a bridge between nurbs and catmull-clark surfaces. *ACM Transactions on Graphics (TOG)*, 25(2):268–292, 2006.

[NLMD12] Matthias Nießner, Charles Loop, Mark Meyer, and Tony Derose. Feature-adaptive gpu rendering of catmull-clark subdivision surfaces. *ACM Transactions on Graphics (TOG)*, 31(1):6, 2012.

[PGSL14] Sergey Podkorytov, Christian Gentil, Dmitry Sokolov, and Sandrine Lanquetin. *Joining Primal/Dual Subdivision Surfaces*, pages 403–424. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[PR97] Jörg Peters and Ulrich Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Transactions on Graphics (TOG)*, 16(4):420–431, 1997.

[PXXZ16] Qing Pan, Guoliang Xu, Gang Xu, and Yongjie Zhang. Isogeometric analysis based on extended catmull-clark subdivision. *Computers & Mathematics with Applications*, 71(1):105 – 119, 2016.

[Rei95] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. volume 12, pages 153–174, 1995.

[SLG05] S. Schaefer, D. Levin, and R. Goldman. Subdivision schemes and attractors. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[Sta98] Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 395–404, New York, NY, USA, 1998. ACM.

[SZSS98] Thomas W Sederberg, Jianmin Zheng, David Sewell, and Malcolm Sabin. Non-uniform recursive subdivision surfaces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 387–394. ACM, 1998.

[WW01] Joe Warren and Henrik Weimer. *Subdivision Methods for Geometric Design: A Constructive Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2001.

[ZT96] Chems Eddine Zair and Eric Tosan. Fractal modeling using free form techniques. *Computer Graphics Forum*, 15(3), 1996.

 ISBN 978-80-86943-40-4

ISSN 2464-4617(print)
ISSN 2464-4625(CD)
Computer Science Research Notes
CSRN 2801
Full Papers Proceedings
http://www.WSCG.eu

Fuzzy image inpainting aimed to medical images

Pavel Vlašánek
Institute for Research
and Applications of Fuzzy
Modeling
30. dubna 22
701 03 Ostrava 1, Czech
Republic
pavel.vlasanek@osu.cz

ABSTRACT

This paper focuses on a reconstruction of low color depth images using fuzzy mathematics. For demonstration purposes, we chose medical images taken from magnetic resonance imaging (MRI) and computed tomography (CT). The proposed technique is based on idea of diffusion where pixels surrounding damaged region are used to determine the corrupted ones. As it is illustrated, the classical diffusion techniques are not so effective. In the paper, we describe the reason why and propose the solution in a form of the new algorithm. The algorithm is demonstrated and visually compared with another ones.

Keywords

image inpainting, fuzzy transform, clustering

1 INTRODUCTION

Image inpainting based on fuzzy mathematics is topic which belongs to *soft computing image processing*. The term stands for image processing tasks performed by soft computing techniques such as neural networks or fuzzy based approaches. The image inpainting stands for unwanted region removal followed by its recovery.

The image inpainting was introduced by Bertalmio et al. [1], who proposed to use partial differential equations (PDE). It consists in a propagation of the colors inward damaged area. The technique is successful for corruptions like scratches or thin inscriptions. In general, small damaged regions without any big hole. The diffusion based inpainting applied to the big hole leads to the unnaturally blurry reconstruction because of its inability to keep the patterns. The solution for this problem is in the *patch based approach* [2, 3]. For this approach, the algorithm searches for a square patches used latter as a replacement for the similar patches in the damaged region. This approach keeps the potential patterns and/or textures. From techniques using both, we can mention [4, 5].

As we stated above, the problem of diffusion is in blurring. The reason is in isotropic nature of the common algorithms which propagates the information to the all directions. We propose to segment the input image first, and apply the inpainting on the each segment separately. The algorithm is demonstrated on the low color depth images where successfully prevents creation of the unwanted artifacts.

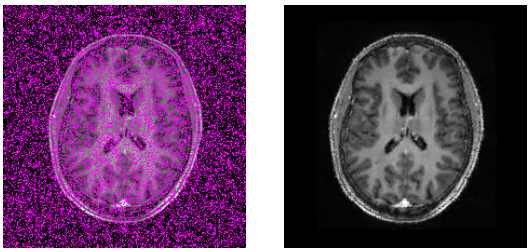
Structure of this paper is as follows. Notation and history of image inpanting is given in Section 2. Section 3 contains information about fuzzy transform and fuzzy clustering. The details of the proposed algorithm are described in Section 4. Section 5 includes experiments and comparison. The paper is concluded in Section 6.

2 NOTATION AND HISTORY

Notation used in the paper is as follows. Discrete image I is a 2D matrix function such as $I : [0,M]_Z \times [0,N]_Z \to [0,255]_Z^3$, where $[0,255]_Z^3$ stands for the pixel intensities in three color channels. We denote $[0,M]_Z = \{0,1,2,\ldots,M\}$, $[0,N]_Z = \{0,1,2,\ldots,N\}$ and $[0,255]_Z = \{0,1,2,\ldots,255\}$. Thus, image width is equal to $M+1$ and height to $N+1$. Image I is partially known. The region Φ stands for known (undamaged) pixels and Ω for unknown (undefined, damaged). Their border is denoted by $\delta\Omega$ and taken as unknown. We assume that $\Phi \cap \Omega = \emptyset$ and $\Phi \cup \Omega \cup \delta\Omega = [0,M]_Z \times [0,N]_Z$. To distinguish between Φ and Ω, a mask S is used. The mask is a binary image where black pixels denote unknown area $\Omega \cup \delta\Omega$. Let us remark that the mask must be given by user.

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.7

The digital image inpainting is demonstrated in Fig. 1.



(a) Input image I (b) Reconstructed image O

Figure 1: Demonstration of image inpainting. The noise was erased by proposed algorithm.

The general principle and formal description was given by pioneers Bertalmio et al. [1] and it is as follows

$$I^{z+1}(x,y) = I^z(x,y) + \Delta t I_t^z(x,y), \forall (x,y) \in \Omega, \qquad (1)$$

where z is the iteration step, (x,y) stands for pixel coordinates, Δt is the rate of improvement and $I_t^z(x,y)$ stands for the update of image $I^z(x,y)$. The $I_t^z(x,y)$ step is computed using a smoothness $L^z(x,y)$ estimated by Laplacian operator. The change of smoothness is propagated from known region Φ to the border $\delta\Omega$ in the direction $\vec{N}(x,y)$. This statement is shown in Fig. 2.

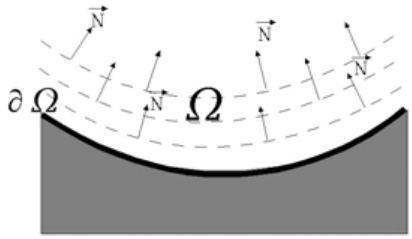


Figure 2: Illustration of the Bertalmio et. al. inpainting process. (Figure taken from [1])

Bertalmio et al. [1] proposed formula

$$I_t^z(x,y) = \vec{\delta L}^z(x,y) \cdot \vec{N}^z(x,y), \qquad (2)$$

where $\vec{\delta L}^z(x,y)$ is the measure of the change in smoothness $L^z(x,y)$ and direction defined as orthogonal to image gradient

$$\vec{N}^z(x,y) = (\bigtriangledown I^z(x,y))^{\perp}. \qquad (3)$$

Another principle was proposed by Ogden et al. [6]. The authors used Gaussian pyramid based technique. The technique builds the pyramid using convolution and sub-sampling followed by linear interpolation.

Elad et al. [7] recommended to separate image to the geometry part D_g and texture part D_t where inpainting is done separately for both. Image decomposition using D_g and D_t matrices is as follows

$$I = D_g \alpha_g + D_t \alpha_t, \qquad (4)$$

where α_g and α_t stands for geometry and texture coefficients. Its illustration is in Fig 3.

Figure 3: Image decomposed to the texture (bottom left) and geometry (bottom right) parts (figure taken from [7])

Their image representation is defined as follows

$$\min_{(\alpha_g,\alpha_t): I = D_g\alpha_g + D_t\alpha_t} \|\alpha_g\|_p + \|\alpha_t\|_p, \qquad (5)$$

where p is the coefficient of the ℓ-norm $\|\alpha\|_p = (\sum \|\alpha(g)\|^p)^{1/p}$. The model proposed by Elad et al. [7] is as follows

$$\min_{(\alpha_g,\alpha_t)} \|\alpha_g\|_1 + \|\alpha_t\|_1 + \lambda \|I - D_g\alpha_g - D_t\alpha_t\|_2^2 + \gamma TV(D_g\alpha_g).$$

TV stands for total variation, $p = 1$ and $\lambda, \gamma > 0$. Adaptation for image inpainting is

$$\min_{(\alpha_g,\alpha_t)} \|\alpha_g\|_1 + \|\alpha_t\|_1 + \lambda \|C(I - D_g\alpha_g - D_t\alpha_t)\|_2^2 + \gamma TV(D_g\alpha_g),$$

where $C = 1$ stands for the undamaged pixels and $C = 0$ for damaged ones. More information is in [7]. Our novel algorithm extends the idea of the separation and combines it with diffusion approach using fuzzy mathematics.

For sake of comparison, let us mention two other techniques[1]. First of them is based on Navier-Stokes equation [8]. The authors use physics of viscous fluid motion to propagate the information inward damaged region. Second one is based on fast marching method

[1] Both of them are implemented in OpenCV framework.

ISBN 978-80-86943-40-4

(FMM) [9] which highlights an importance of filling order.

3 PRELIMINARIES

First to describe is F-transform which is a technique for changing an image representation. Second one is clustering algorithm fuzzy C-Means.

3.1 F-transform

We propose a fuzzy based approach to solve image inpainting task [10]. Let us describe fuzzy transform (F-transform) [11] given by a fuzzy partition satisfying following definition. Fuzzy sets $A_0,\dots,A_m; m<M$ identified with their membership functions (basic functions) $A_0,\dots,A_m : [0,M] \to [0,1]$, establish a *fuzzy partition* of $[0,M]$ with nodes $0 = x_0 < x_1 < \cdots < x_m = M$ if the following conditions are fulfilled:

1) $A_k : [0,M] \to [0,1]$, $A_k(x_k) = 1$;

2) $A_k(x) = 0$ if $x \notin (x_{k-1}, x_{k+1})$, $k = 0,\dots,m$;

3) $A_k(x)$ is continuous;

4) $A_k(x)$ strictly increases on $[x_{k-1}, x_k]$, $k = 2,\dots,m$; and strictly decreases on $[x_k, x_{k+1}]$, $k = 1,\dots,m-1$;

5) $\sum_{k=0}^{m} A_k(x) = 1$, $x \in [0,M]$.

Assume that fuzzy sets $A_0,\dots,A_m$ establish a fuzzy partition of $[0,M]$. The vector of real numbers $\mathbf{F}_m[I] = (F_0,\dots,F_m)$ is the *(direct) discrete F-transform* of I w.r.t. $A_0,\dots,A_m$ where the component F_k is defined by

$$F_k = \frac{\sum_{x=0}^{M} A_k(x) I(x)}{\sum_{x=0}^{M} A_k(x)},\ k = 0,\dots,m. \tag{6}$$

Let us introduce F-transform of a 2D gray-scale image I. Let $A_0,\dots,A_m$ and $B_0,\dots,B_n$ be basic functions, $A_0,\dots,A_m : [0,M] \to [0,1]$ be fuzzy partition of $[0,M]$ and $B_0,\dots,B_n : [0,N] \to [0,1]$ be fuzzy partition of $[0,N]$. If for all $k \in 0,\dots,m (\exists x \in [0,M])\ A_k(x) > 0$, and for all $l \in 0,\dots,n (\exists y \in [0,N])\ B_l(y) > 0$ with respect to Φ, we say that the set of pixels Φ is *sufficiently dense with respect to the chosen partitions*.

We say that the $m \times n$-matrix of real numbers $[F_{kl}]$ is called *the (discrete) F-transform* of I with respect to $\{A_0,\dots,A_m\}$ and $\{B_0,\dots,B_n\}$ if for all $k = 0,\dots,m$, $l = 0,\dots,n$,

$$F_{kl} = \frac{\sum_{y=0}^{N}\sum_{x=0}^{M} I(x,y) A_k(x) B_l(y)}{\sum_{y=0}^{N}\sum_{x=0}^{M} A_k(x) B_l(y)}. \tag{7}$$

The coefficients F_{kl} are called *components of the F-transform*. The formula (7) is called *direct step*. In order to reconstruct the original function, it is usually followed by *inverse step* as follows

$$O(x,y) = \sum_{k=0}^{m}\sum_{l=0}^{n} F_{kl} A_k(x) B_l(y), \tag{8}$$

where O is the reconstructed image. According to formula (8), the computation takes particular component F_{kl} and spread it to the appropriate region of O with respect to A_k and B_l. For details see [10].

In order to omit the damaged pixel from the computation, mask S is used as follows

$$F_{kl} = \frac{\sum_{y=0}^{N}\sum_{x=0}^{M} I(x,y) A_k(x) B_l(y) S(x,y)}{\sum_{y=0}^{N}\sum_{x=0}^{M} A_k(x) B_l(y) S(x,y)}. \tag{9}$$

The mask is binary image where 0 denotes *damaged pixel*. According to formula (9), these pixels are not taken into consideration during the F-transform component computation.

This algorithm works well for photos [12, 13, 10, 14] but not so sufficiently for low depth color images. The reason is in isotropic nature of the algorithm where different regions are mixed together in the inpainted area Ω. The addressed issue is illustrated in Fig. 4.

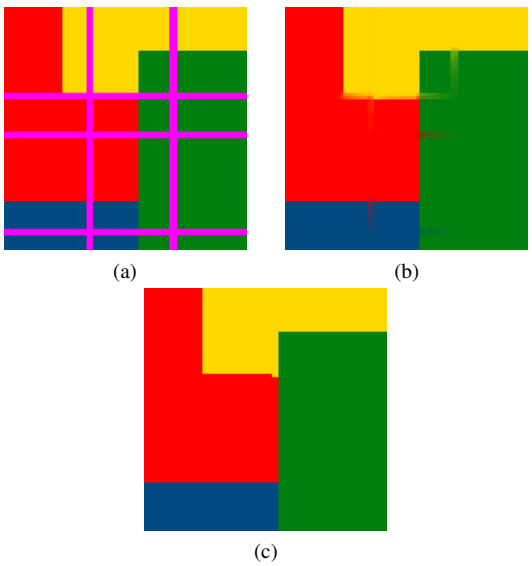


Figure 4: Demonstration of the unwanted blurriness in the reconstruction . a) Damaged image I; b) image inpainting using diffusion [10]; c) image inpainting using the proposed algorithm.

F-transform has been proven to work on various image processing tasks such as edge detection [15, 16], image fusion [17] or image compression [18].

ISBN 978-80-86943-40-4

3.2 Fuzzy clustering

Segmentation is important part of our technique. We propose to use fuzzy C-Means (FCM) which proved itself as very useful for image processing [19]. The algorithm was developed by J.C. Dunn [20] and improved by J.C. Bezdek [21]. It belongs to *soft clustering* which refers to the fact that each data point (pixel) belongs to the more than one cluster.

The input for the algorithm is set of elements $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ into a collection of K fuzzy clusters defined by centers $C = \{\mathbf{c}_1, ..., \mathbf{c}_K\}$ such as

$$c_k = \frac{\sum_x w_k(x)^m x}{\sum_x w_k(x)^m}, \tag{10}$$

where $w_k(x)$ stands for membership degree of element x to cluster c_k. Besides the clusters, the algorithm returns partition matrix $W = w_{i,j} \in [0,1]$, $i = 1, ..., n$, $j = 1, ..., K$ where w_{ij} stands for membership degree of x_i to cluster c_j.

The FCM minimize function

$$\arg\min_C \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^m \left\| \mathbf{x}_i - \mathbf{c}_j \right\|^2, \tag{11}$$

where

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\left\| \mathbf{x}_i - \mathbf{c}_j \right\|}{\left\| \mathbf{x}_i - \mathbf{c}_k \right\|} \right)^{\frac{2}{m-1}}}. \tag{12}$$

4 NOVEL APPROACH DESCRIPTION

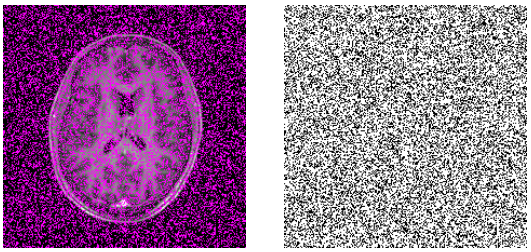
We propose to divide an image to the several independent parts and process them one by one where the processing consists in image inpainting. The division is performed using fuzzy C-Means algorithm. Let us define discrete binary image $V_i : [0,M]_Z \times [0,N]_Z \to 0,1$ where the image is identified with cluster c_i and pixels which belongs to it with maximum membership degree. All regions are inpainted using F-transform based algorithm and in the end, all of them are put together to create output image O.

4.1 Algorithm

Let us describe the algorithm on the image in Fig. 5.

First problem to solve is to determine to which cluster which damaged pixel belongs. For the rough estimation, we propose to use fuzzy image inpainting. Result is in Fig. 6.

In this step, the pixels from damaged region Ω are replaced by the blended colors of the surrounding ones.



(a) I (b) S

Figure 5: a) Image I; b) mask S.

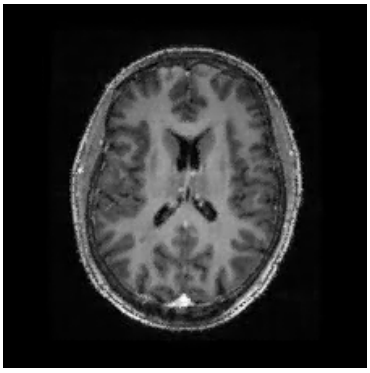


Figure 6: Image inpainting [10] of Fig. 5.

For small holes, the average of surrounding colors can be used. For bigger holes, it is better to use more advanced techniques. Because of a robustness, accessibility and a way of processing, we propose to use F-transform inpainting described above which can handle both.

We assume that inpainting fills in Ω using colors from the close neighborhood. Thus, we can estimate the region used later for the separated reconstruction.

Because of the nature of the low color depth images, each blurriness caused by reconstruction is highly visible. The blurriness is caused by mixing of colors from the clearly separated places. Thus, we propose the separation for future region-by-region processing. For that purpose, fuzzy C-Means is proposed as can be seen in Fig. 7. The inpainted image from previous step is used as an input.

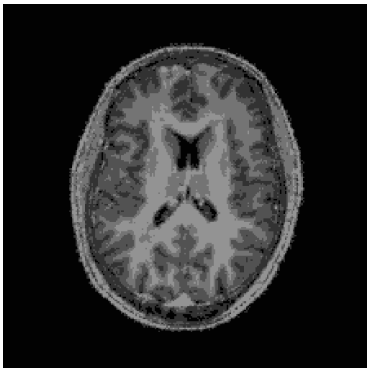


Figure 7: Fuzzy C-Means applied to Fig. 6.

ISBN 978-80-86943-40-4

The regions are recognized using binary images $V_1, V_2, ..., V_K$ identified with the clusters. Image V labels *valid pixels*. Therefore, the separated regions are reconstructed from pixels of the known part of themselves. This feature makes the inpainting algorithm anisotropic which prevents the unwanted color mixing. Demonstration of the segmented regions for $K = 5$ is in Fig. 8.

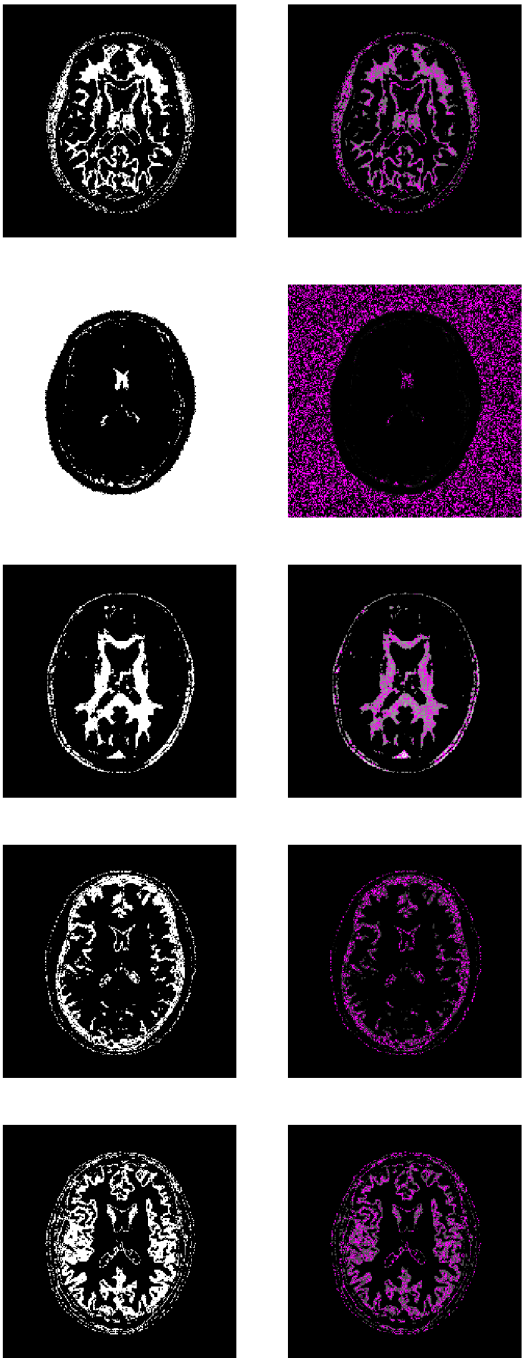


Figure 8: Left column contains images V_1, V_2, V_3, V_4, V_5 and right column respective parts of input image I. The division is based on Fig. 7.

Next step is to create a set of masks $S_1, S_2, ..., S_K$ to use together with $V_1, V_2, ..., V_K$. Each mask labels damaged pixels just in particular region. The image V influences the computation of the F-transform components as follows

$$F_{kl} = \frac{\sum_{y=0}^{N}\sum_{x=0}^{M} I(x,y)S(x,y)V(x,y)A_k(x)B_l(y)}{\sum_{y=0}^{N}\sum_{x=0}^{M} S(x,y)V(x,y)A_k(x)B_l(y)}. \quad (13)$$

Each region is inpainted independently using proper S and V. Example for $K = 5$ is illustrated in Fig. 9. For better illustration, Fig. 10 shows detail of a brain and comparison between our novel algorithm and an original one.

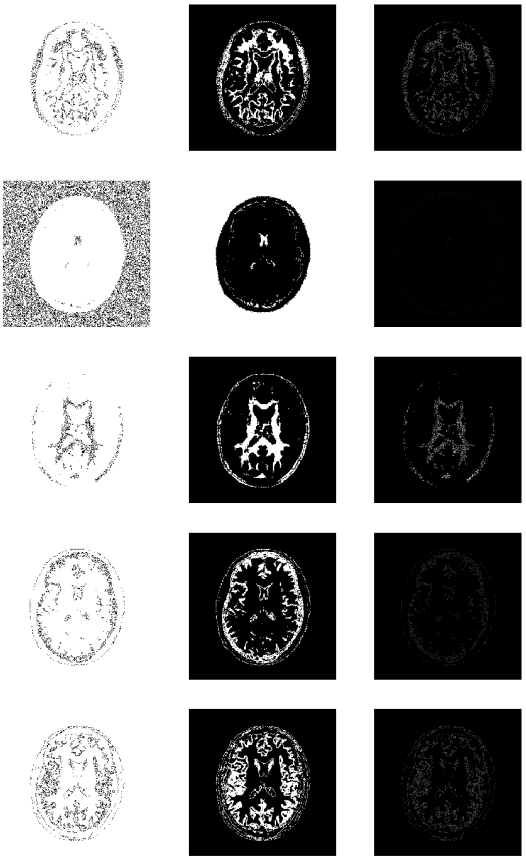


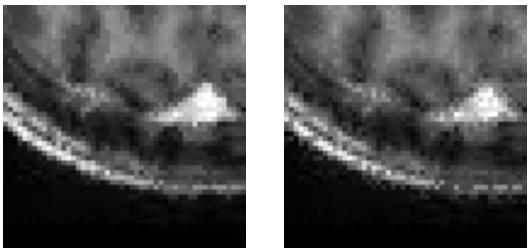
Figure 9: Left column contains masks S_1, S_2, S_3, S_4, S_5. Middle one contains valid pixels V_1, V_2, V_3, V_4, V_5 and right one reconstructed parts in each region. The division is based on Fig. 7.

The proposed algorithm is focused on the specific images which are very sensitive to blurred reconstruction due to their low depth colors. Thus, the algorithm is demonstrated on medical images in Fig. 11 and applied to each color channel.

Let us summarize all steps:

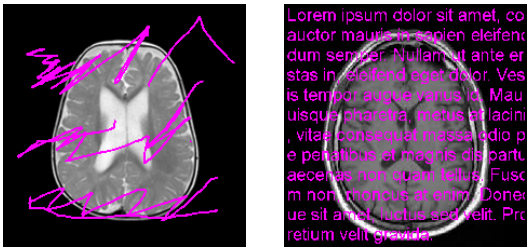
1. Inpaint input image using conventional F-transform image inpainting.

ISBN 978-80-86943-40-4

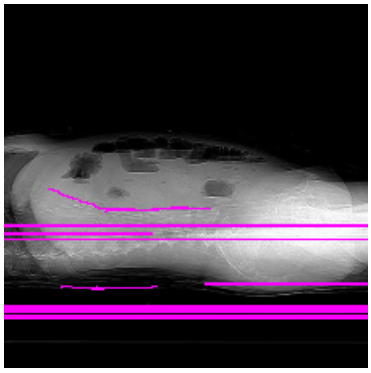


(a) (b)

Figure 10: a) Novel algorithm with more clear and unified output; b) original algorithm [10]. Adjacent colors are more uniform for figure a). Visible difference is in white spot region and even not so shattered edge.



(a) Brain (b) Brain



(c) Torso

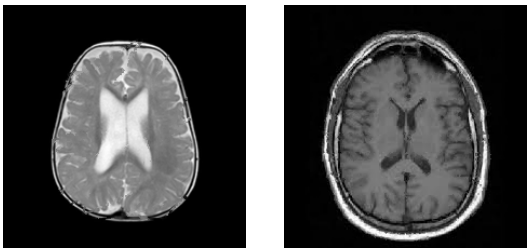
Figure 11: A set of images for demonstration purposes.

2. Cluster the inpainted image using fuzzy C-Means.
3. Identify each cluster region with binary image V_i.
4. Separate input mask S to the several regions S_i based on clusters.
5. Inpaint input image region by region using appropriate V_i and S_i.

Implementation of our new technique is based on publicly available F-transform source code from *OpenCV*[2]. For testing images, one reconstruction lasts few seconds on the average PC. Results are available in Fig. 12.

A deep comparison with plenty of techniques is not possible due to lack of their implementations and/or their inabilities to work with specific types such as noisy images[3].

[2] *opencv-contrib* framework, module *fuzzy*



(a) Brain (b) Brain

(c) Torso

Figure 12: Image inpainting of Fig. 11. Target was to erase purple text, lines and noise.

4.2 Qualitative comparison

Let us compare the novel technique with basic principle of the enhanced diffusion idea [10], Navier-Stokes [8] and FMM [9]. The various outputs can be seen in Fig. 13, 14 and 15.

Due to size limitation, the details are not obvious. Let us demonstrate the main feature of the proposed technique which is to keep sharp reconstruction without blurred artifacts. The details are given in Fig. 16.

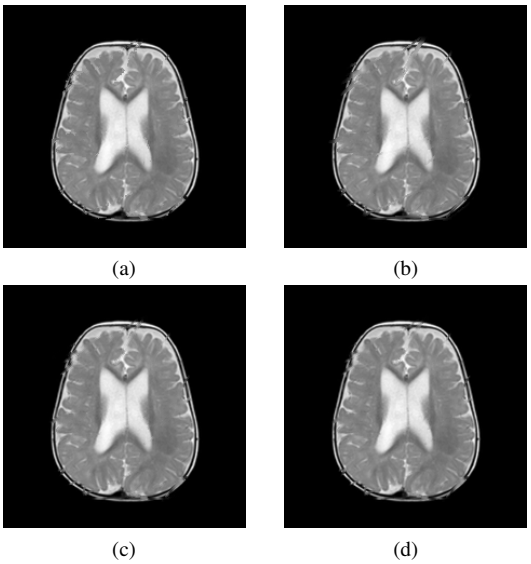
5 CONCLUSION

In this initial study, we proposed a technique aimed to low color depth images usable for medical ones from MRI or CT. Its novelty consists in separation according to colors followed by region-by-region processing. For the separation, the fuzzy C-Means is proposed and for the processing the fuzzy image inpainting. To make the separation possible, we need to estimate to what clusters all damaged pixels belong. For that purpose, it is necessary to do inpainting of the whole image as a first step.

Disadvantage of the common techniques lays in their isotropic way of processing. As was demonstrated, this

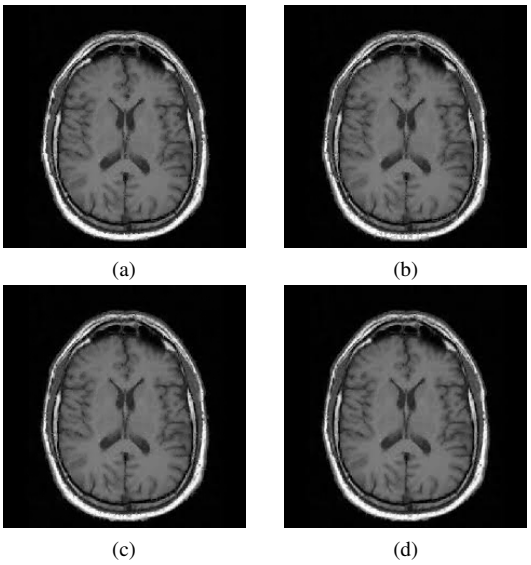
[3] For instance, the exemplar based techniques are not effective for highly damaged pictures, moreover with damage distributed all over them.

ISBN 978-80-86943-40-4



(a) (b) (c) (d)

Figure 13: Comparison of the various image inpainting techniques. a) Novel algorithm; b) fuzzy image inpaintg; c) Navier-Stokes; d) FMM.

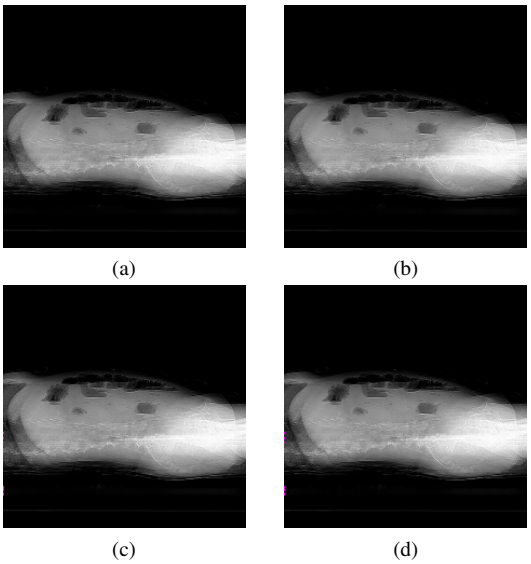


(a) (b) (c) (d)

Figure 14: Comparison of the various image inpainting techniques. a) Novel algorithm; b) fuzzy image inpaintg; c) Navier-Stokes; d) FMM.

fact leads to blurry reconstruction which is more obvious in low depth color images. The separation part of our algorithm successfully prevents this issue and in fact it could come after any common inpainting algorithm as a second step. To achieve this, we use *submask* labeling the damaged pixels of the particular region and *valid pixels* labeling pixels of the cluster dedicated to the same region. User should provide image, mask and specify a number of clusters. Everything else is performed automatically by algorithm and the implementation respectively.

We explained the way of working of our algorithm and compared it with another techniques. For future research, we would like to extend the idea for heuristic and probability estimation of the reconstructed part specifically aimed to medical usage.



(a) (b) (c) (d)

Figure 15: Comparison of the various image inpainting techniques. a) Novel algorithm; b) fuzzy image inpaintg; c) Navier-Stokes; d) FMM.

ACKNOWLEDGMENT

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602".

6 REFERENCES

[1] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[2] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 1033–1038.

[3] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *Image Processing, IEEE Transactions on*, vol. 13, no. 9, pp. 1200–1212, 2004.

[4] N. Komodakis and G. Tziritas, "Image completion using efficient belief propagation via priority scheduling and dynamic pruning," *Image Processing, IEEE Transactions on*, vol. 16, no. 11, pp. 2649–2661, 2007.

[5] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," in *ACM*

ISBN 978-80-86943-40-4

Transactions on Graphics (TOG), vol. 22, no. 3. ACM, 2003, pp. 303–312.

[6] J. M. Ogden, E. H. Adelson, J. R. Bergen, and P. J. Burt, "Pyramid-based computer graphics," *RCA Engineer*, vol. 30, no. 5, pp. 4–15, 1985.

[7] M. Elad, J.-L. Starck, P. Querre, and D. L. Donoho, "Simultaneous cartoon and texture image inpainting using morphological component analysis (mca)," *Applied and Computational Harmonic Analysis*, vol. 19, no. 3, pp. 340–358, 2005.

[8] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–355.

[9] A. Telea, "An image inpainting technique based on the fast marching method," *Journal of graphics tools*, vol. 9, no. 1, pp. 23–34, 2004.

[10] I. Perfilieva and P. Vlašánek, "Image reconstruction by means of F-transform," *Knowledge-Based Systems*, vol. 70, pp. 55–63, 2014.

[11] I. Perfilieva, "Fuzzy transforms: Theory and applications," *Fuzzy sets and systems*, vol. 157, no. 8, pp. 993–1023, 2006.

[12] I. Perfiljeva, P. Vlašánek, and M. Wrublova, "Fuzzy transform for image reconstruction," *Uncertainty Modeling in Knowledge Engineering and Decision Making*, pp. 615–620, 2012.

[13] P. Vlašánek and I. Perfilieva, "Image reconstruction with usage of the F-transform," in *International Joint Conference CISIS'12-ICEUTEÂ´12-SOCOÂ´12*. Springer, 2013, pp. 507–514.

[14] ——, "Interpolation techniques versus F-transform in application to image reconstruction," in *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2014, pp. 533–539.

[15] I. Perfilieva, P. Hodáková, and P. Hurtík, "Differentiation by the F-transform and application to edge detection," *Fuzzy Sets and Systems*, 2014.

[16] M. Danková, P. Hodáková, I. Perfilieva, and M. Vajgl, "Edge detection using F-transform." in *ISDA*, 2011, pp. 672–677.

[17] M. Vajgl, I. Perfilieva, and P. Hod'áková, "Advanced F-transform-based image fusion," *Advances in Fuzzy Systems*, vol. 2012, p. 4, 2012.

[18] I. Perfilieva and B. De Baets, "Fuzzy transforms of monotone functions with application to image compression," *Information Sciences*, vol. 180, no. 17, pp. 3304–3315, 2010.

[19] M. J. Christ and R. Parvathi, "Fuzzy c-means algorithm for medical image segmentation," in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, vol. 4. IEEE, 2011, pp. 33–36.

[20] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *Cybernetics*, vol. 3, pp. 32–57, 1973.

[21] J. C. Bezdek, "Objective function clustering," in *Pattern recognition with fuzzy objective function algorithms*. Springer, 1981, pp. 43–93.

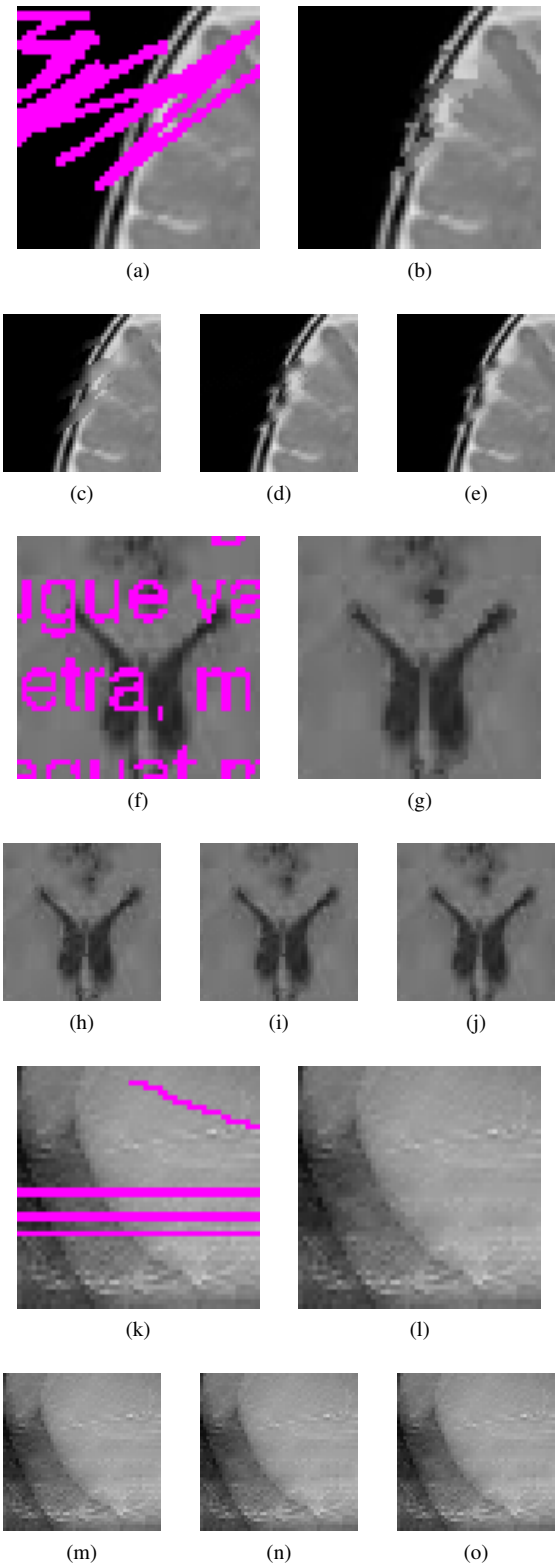


Figure 16: Details of the reconstruction in Fig. 15, 13 and 14. a) f) k) Damaged image; b) g) l) image inpainted by the novel algorithm; c) h) m) image inpainted by original fuzzy inpainting [10]; d) i) n) image inpainted by Navier-Stokes [8]; e) j) o) image inpainted by FMM [9].

ISBN 978-80-86943-40-4

ISSN 2464-4617(print)
ISSN 2464-4625(CD)

Computer Science Research Notes
CSRN 2801

Full Papers Proceedings
http://www.WSCG.eu

Single image summary of time-varying Earth-features

Gaurav Tripathi
Masters student
University of Calgary
Department of Computer Science
Calgary, AB T2N 1N4, Canada
gaurav.tripathi@ucalgary.ca

Katayoon Etemad
Postdoctoral Fellow
University of Calgary
Department of Computer Science
Calgary, AB T2N 1N4, Canada
ketemad@ucalgary.ca

Faramarz Samavati
Full Professor
University of Calgary
Department of Computer Science
Calgary, AB T2N 1N4, Canada
samavati@ucalgary.ca

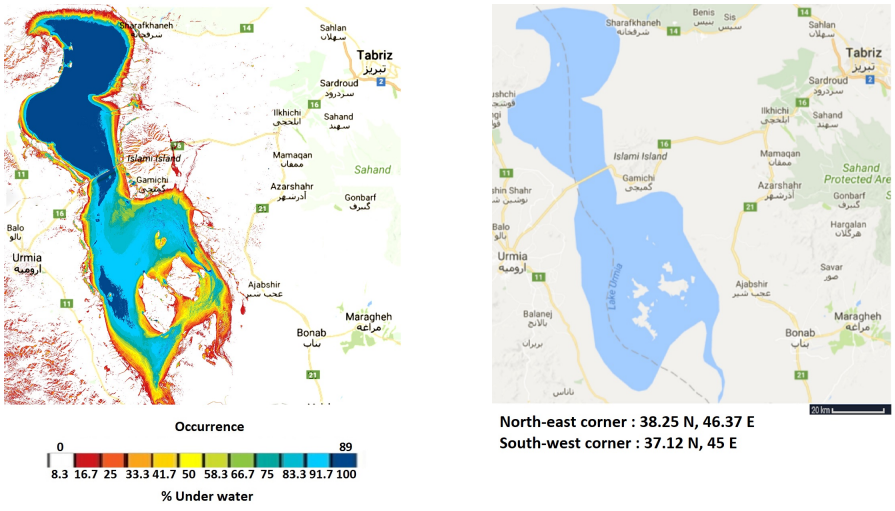


Figure 1: Lake Urmia 89 layers SIS from 2013-2017.

ABSTRACT

The Earth's surface is live and dynamic due to natural and manmade events. Tracking and visualizing Earth-features (e.g. water, snow, and vegetation) is an important problem. Earth observation satellite imagery like Landsat 8 makes the tracking feasible by providing detailed multispectral imagery at regular intervals. In this paper, we explore a single image summary approach to detecting changes in Earth-features by using the Landsat 8 dataset. In our system, we use appropriate thresholds for spectral indices to identify features, reference datasets, and combine multiple images using predefined color palettes to generate a single image summary of features for a region. Furthermore, we illustrate the benefit of our method over traditional visualizations with case-studies for the Lake Urmia, the Amazon Rainforest, and the Bering Glacier.

Keywords

Visualization, Remote sensing, Landsat 8, Spectral indices, Deforestation, Drought, Glacial melting

1 INTRODUCTION

Earth is dynamic and ever-changing planet. Natural resource managers, policymakers, researchers, and, the general public need information about these changes to detect environmental changes and assess the impacts of global warming [30]. Displaying a video or scientific charts are useful but not sufficient to clearly show the changes in the *Earth-features* (e.g. Vegetation, water, snow) present on the Earth's surface, due to phenomenon like desiccation of lakes, melting of glaciers, and deforestation. Changes in individual Earth-features are not always visible in regular RGB images. Moreover, it is hard to perceive significant changes in a short period of time. Capturing changes in the Earth-features and visualizing them in an appropriate way is a fundamental and important problem.

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.8

The regular collection of datasets performed by Earth observation satellites can play an important role in addressing the issue of capturing Earth-feature changes. In this paper, we use Landsat 8 satellite imagery datasets to identify and detect changes in Earth-features. In addition to regular RGB images, Landsat 8 data contains spectral bands (i.e. a range of frequencies along the electromagnetic spectrum) that can reveal important features of a region such as the prevalence of snow, water, and vegetation.

With respect to the visualization challenge, techniques like animated timelapses and image lineups are traditionally used. These techniques are used to represent temporal changes in Earth-features. One major problem with the animated timelpase approach, like Google Timelapse [11], is the loss of context. To identify Earth-feature changes using a video, the observer needs to move back and forth between video frames. In the second traditional approach, a number of images are placed side-by-side to allow for comparisons between multiple frames. The main drawback to the use of multiple images on screen (or on paper) is that the size of the individual images needs to be reduced to fit all of them in a single frame. The size reduction results in detail loss for high-resolution images, and is especially pronounced when there are many images. "small multiples large singles" mentioned in [35] is in congruence to using SIS and making it convenient for users to perform effective visual exploration.

To address the limitations of these traditional techniques, in this paper, we propose a novel single image summary (*SIS*) for Earth-features using Landsat 8 images (e.g. Figure 1). SIS represents temporal changes to the Earth-features in a given region over a fixed duration of time. To create a SIS, we determine the Earth-feature recurrence, e.g. water existence, for any location in the region of interest (ROI) and then map the resulting recurrence values to predefined colors. To provide a better context for the feature of interest, we call this location-based recurrence distribution a *recurrence map*. We also add an overlay of the traditional map of the surrounding areas. SIS representation resolves the loss of context issue and retains the resolution of the original image dataset.

To prepare a SIS, we download data for a region covering the duration of interest. The cloudy pixels are detected using the Quality Assurance band provided by Landsat 8. In the next step, relevant Earth-features are identified by using spectral band operations. After feature identification, we generate the recurrence map by counting the occurrence of that feature in every location in the region. To facilitate these operations, and as a proof of concept, we have implemented a software prototype. In our system, we use the recurrence map and predefined color palettes to generate single image summaries for three Earth-features: vegetation, water, and snow. We use our system in three case studies : (i) Lake Urmia , (ii) the Amazon Rainforest, and, (iii) the Bering Glacier. We also discuss the impact of different color palettes on results.

The paper is arranged as follows. Section 2 covers previous work and related material on Landsat 8, spectral indices and geospatial visualization. Section 3 details our approach and implementation. Section 4 discusses the case-studies. Section 5 concludes this article and suggests future works.

2 BACKGROUND AND RELATED WORK

2.1 Landsat 8

For over 40 years, seven Landsat satellites have collected spectral information regarding the Earth's surface. The latest satellite dataset in the series, Landsat 8, was made available for free public use on May 30, 2013 [30]. The near polar orbit of Landsat 8 allows it to regularly visit an area every 16 days. This temporal resolution allows researchers to track seasonal changes on the Earth's surface, as in this work, which tracks changes in time-varying Earth-features. The Landsat 8 sensors collect data at a spatial resolution of 30 meters (visible, NIR, SWIR); 100 meters (thermal); and 15 meters (panchromatic).

Landsat 8 measures the energy reflected by land surface across different frequency ranges from the electromagnetic spectrum. Each range of frequency is called a band. Land features like water, vegetation, and snow reflect energy based on their unique surface characteristics. Thus, measuring reflected energy helps one identify the observed feature. In multispectral satellite imagery, multiple band data is used to create spectral indices that make it possible to identify features. The Green (0.533 - 0.590 micrometers), Red (0.636 - 0.673 micrometers), Near-infrared (0.851 - 0.879 micrometers), and Shortwave infrared (1.566 - 1.651 micrometers) bands have been used in this research. These bands are combined to create metrics that assist in discriminating Earth-features.

2.2 Spectral Indices

A spectral index is a metric used to identify specific features or phenomena in remote sensing imagery. It is prepared by linear or nonlinear combinations of two or more bands. Vegetation, water, and snow indices are one of the most studied and commonly used spectral indices [37] in the field of remote sensing. An optimally designed spectral index is supposed to be as sensitive as possible to the essential feature of interest and insensitive to nonessential features in the observation area [36]. In this paper, we particularly focus on spectral indices for vegetation, water, and snow.

 ISBN 978-80-86943-40-4

The normalized difference vegetation index (NDVI) is a spectral index that can be used to analyze remote sensing measurements and assess whether the target contains live green vegetation [26, 29]. Live vegetation absorbs solar radiation in the photosynthetically active range with a wavelength between 400 to 700 nanometers. Wavelengths up to red can be used by leaves to synthesize organic molecules while NIR and longer wavelengths cannot be used for synthesis. Hence, the leaves reflect energy of the NIR range. NDVI utilizes these facts in a simple form as

$$NDVI(p) = \frac{NIR(p) - Red(p)}{NIR(p) + Red(p)} \quad (1)$$

where $NIR(p)$ and $Red(p)$ are respectively the near infrared and red reflectance at a pixel p in the band. It is clear that NDVI values occupy the range [-1,1].

For areas with dense vegetation, NDVI has a high value (between 0.3 to 0.8) [10]. Other vegetation indices are also utilised to detect vegetation but NDVI is the most frequently used in remote sensing studies [37].

NIR radiation is strongly absorbed by water bodies and strongly reflected by soil and other surfaces. Furthermore, visible radiation (Red, Green, and Blue) is strongly reflected by water bodies. Normalized Difference Water Index (NDWI) utilizes these facts in a simple form as

$$NDWI(p) = \frac{Green(p) - NIR(p)}{Green(p) + NIR(p)} \quad (2)$$

where $NIR(p)$ and $Green(p)$ are respectively the near infrared and green reflectance at a pixel p in the band. The positive values of NDWI segregate open water bodies. The spectral response of water bodies indicates that the Green band is more suited towards segregating water features and keeping suitable ranges of NDWI as compared to Red and Blue bands [17, 27].

Its challenging to identify snow in satellite images because snow and clouds are equally bright in the visible wavelength. While snow cover absorbs Shortwave infrared (SWIR) radiation, clouds reflect SWIR radiation strongly. To utilize this difference in contrast between different bands, a normalized difference snow index (NDSI) is formulated [22]. NDSI utilizes these facts in a simple form as

$$NDSI(p) = \frac{Green(p) - SWIR(p)}{Green(p) + SWIR(p)} \quad (3)$$

where $Green(p)$ and $SWIR(p)$ are respectively the green and short-wave infrared reflectance at a pixel p in the band. Again, values range from -1 to 1. The threshold for determining the presence of snow varies and needs to be decided on a per-region basis.

2.3 Geospatial visualization

Time-varying data visualization is a well-studied area in information visualization of abstract data [15, 38]. Visualizing time-varying geospatial data is more challenging because of its location dependency [20].

Videos are commonly used to visualize changes in time-varying datasets. In [28], video synopsis has been used to reduce video size while retaining dynamic activities in a video. Google has created a timelapse tool using satellite imagery datasets to visualize changes in the Earth over past three decades [11]. In [25], the authors use videos to visualize changes in the Great Salt Lake. In general, the use of videos for visualizing these changes provides a sense of the overall trends but comparisons between consecutive frames is hard. For geospatial datasets, the comparison becomes harder, as the position of geospatial features in time is also important. The videos of time-varying geospatial data can be visualized on a physical globe for easy understanding [19].

To enable better frame comparison, multiple image lineups can be used [16, 39]. For example, multiple image lineups have been used to show changes in Lake Urmia over time [34]. To prepare the lineup, the authors use Moderate Resolution Imaging Spectroradiometer (MODIS) data [8] from 2000 to 2014. There are three disadvantages of the image lineup approach. First, the size of an individual image is decreased to accommodate all in a lineup. Second, only a limited number of frames can be placed in a lineup while still retaining the important details of a region. Third, it is hard to compare more than two images at a time in order to understand the changes occurring at a location.

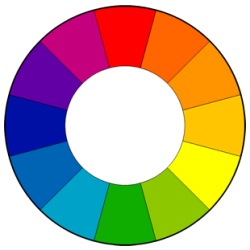


Figure 2: Basic color wheel [1].

Images can present and retain time-varying information in a single frame when using an appropriate color palette. An appropriate color-palette helps in logical organization of data and captures the trends and relationship within data. In addition to an appropriate palette, choosing the right number of colors for the color palette is an important aspect of identifying time-varying changes in Earth-features. In [23], two categories of color schemes, called sequential and diverging, were suggested. Furthermore, a low number of color palette is suggested for the proper representation of data classes in thematic maps. In sequential schemes, low data values are represented

ISBN 978-80-86943-40-4

by light colors and high data values are represented by dark colors. Sequential color schemes are used to represent data that changes from a high to low value. In diverging color schemes, the mid data value is represented using one color and two different sequential schemes diverge from this shared middle value. It is evidenced in [21] that palettes with fewer colors are more discriminable while more colors are harder for users to process.

The color wheel is a common tool used to design colors for a palette. Figure 2 shows one of the most commonly used wheels. Adjacent colors on the color wheel are called analogous colors. They are frequently found in nature and harmonize well to avoid jarring effects in the image [1, 18].

3 METHODOLOGY

The main goal of this work is to create a single image summary of Earth-features in a selected region that summarizes a certain duration of time. One challenge lies in how to capture changes in Earth-features. Another challenge is how to visualize these features. We use relevant bands from Landsat 8 datasets to create spectral indices that help in identifying the Earth-features. To visualize Earth-feature changes, we generate a recurrence map for the ROI and apply a color palette based on the recurrence of feature in any location. This leads to the generation of a SIS.

3.1 Capturing Earth-feature changes

Landsat 8 data can be accessed freely on Amazon Web Services (AWS) and Earthexplorer [6]. Earthexplorer only allows bulk data download for scenes, which results in long download times and more storage usage. We use AWS as the downloading source as it offers good speed and the option to download individual band data. AWS is also a suitable target for creating a web crawler because it provides a hyperlink for each file.

Landsat 8 images are separated into scenes for easy downloading. Each scene represents an area of approximately 185 km by 185 km and contains 11 bands and metadata files. The bands are delivered as 16-bit images in GeoTIFF file format. The hyperlinks for around 2 million Landsat 8 scenes are available in a file called *scene-list* that is provided by AWS. The file is updated daily with the latest scenes and each line in the file contains the details of the geographic location of the scene and a url for scene download.

We developed a downloader program that downloads scenes based on scene-list, date, bands, path, and row parameters. To download data in the range of 2013-2017 takes approximately an hour of time. After the downloading step, our system detects cloudy pixels using the Quality Assurance (QA) band provided by Landsat 8. The QA band contains bit-packed information about the surface conditions, which helps to indicate clouds. If the decimal value of a pixel is above 31744, the pixel is likely to be cloudy [5].

Due to the non-spherical shape of the Earth and near-polar orbit of Landsat 8, different day scenes from the same region (identical path and row) are not exactly aligned. Exact alignment is crucial for temporal stacking of images.

Landsat 8 scenes are projected using the Universal Transverse Mercator projection system [7]. There is a linear mapping between the image coordinate (rows and columns) and the (geographical) UTM coordinate. Our system uses this linear mapping for the purpose of aligning images. The system uses the image and UTM coordinate system values from metadata files to align pixels for all layers from the ROI. The system loads all the referenced layers in memory for the spectral index calculation used to identify the feature of interest (Section 2.2).

The calculated value is compared to a threshold and, based on whether the spectral index value is above or below that threshold, the pixel at point p in each of the layers is classified as belonging to a feature. The threshold for identifying features depends on a number of factors, including the physical properties of observed features or the analyst making the observation [22, 27]. For example, values of NDWI greater than zero commonly indicate water. In our system, an appropriate threshold is decided for each region based on visual inspection. Note that the same threshold is used for all layers in a single region.

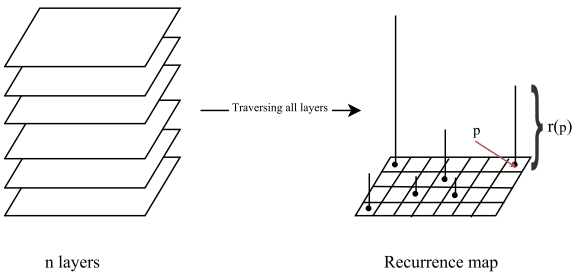


Figure 3: Recurrence map creation.

3.2 Recurrence map creation

After the stack of layers is classified into features, the recurrence map is prepared. A recurrence map represents the recurrence of a particular feature in the ROI. Our system has the capability to choose the ROI for the recurrence map preparation. For this research, the ROI is chosen to be inside the boundaries of a Landsat 8 scene.To better understand the recurrence map, n temporal layers in the image collection are considered. For each point p of the map, the repetition of the feature at p, $r(p)$, is stored in the recurrence map. Figure

 ISBN 978-80-86943-40-4

3 illustrates the recurrence map. The value of $r(p)$, is normalized by the total number of non-cloudy layers at that point. The recurrence map can be constructed via a one time traversal of all layers in the current time range and stored. Once a recurrence map is created, various color maps can be used on it without the need to traverse all layers again.

3.3 Color-mapping

Our system follows a color-mapping procedure to apply a particular color-palette to the recurrence map in order to generate a SIS. Let us denote the current color palette by $c = \{c_1, ..., c_m\}$ where m is the number of colors. Each recurrence value is mapped to a color c_i (for some i). The color-palette and range of the recurrence map are also indicated in the SIS results. To provide a stronger distinction among feature categories, we choose palettes with a small number of colors [33]. Since the range of $r(p)$, is normally larger than m, a lookup table L is needed to assign a color to any $r(p)$

$$L: \quad r(p) \rightarrow c.$$

In our system, L is a uniform sampling of color maps by default. For practical case studies, one may tweak this transfer function to customize the result. Based on user preferences, our system can easily apply predefined sequential, divergent, and user-defined custom color-palettes to the recurrence map. We can also apply natural colors present in the RGB satellite images of the region. The color schemes for each of the case studies have been chosen in order to highlight the changes in features and maintain the context of surrounding regions.

4 CASE STUDIES

To evaluate our method and the implemented prototype, we have experimented on three case studies to highlight changes in Lake Urmia in north west Iran, the Amazon Rainforest in Brazil and the Bering Glacier in Alaska.

4.1 Lake Urmia

There has been a decrease in the water level of lake Urmia since the 1990s [4]. Some of the major speculated reasons for this decline are dam construction, diverting water for irrigation, less precipitation, and warmer climates [24]. There are many repercussions of this phenomenon. Reduction in water level is causing the salt levels to increase, thus causing the native brine shrimp population to decrease [14]. Since brine shrimp is the major food source for the bird population in this region, it is causing significant ecological disruption. The current increase in salt content is also causing the surrounding plant population to decrease as the drying of the lake also leaves a huge salt trail around the lake boundary.

In this section, SIS have been prepared for lake Urmia in order to observe the time-varying changes in water recurrence. A description of the area of study, choices of relevant spectral indices, and color palette are crucial elements of generating the summary images.

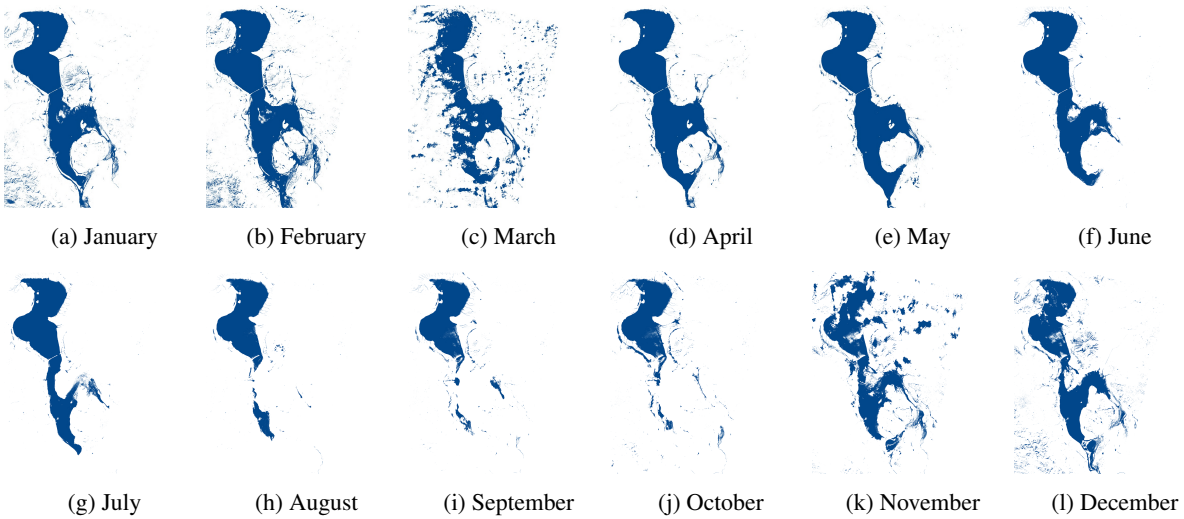
The study area is the Lake Urmia region in Iran lying between Urmia and Tabriz cities (as shown in Figure 1). According to the Landsat 8 operational orbit WRS-2 [13], scenes correspond to path 169 and row 34. Cloudy pixels in the dataset were detected using the QA band (as shown in Section 3.1). Positive values of NDWI are assumed to indicate the presence of water in this region. After classification of water in each layer, the recurrence map is prepared. In the next step a color-mapping is applied to the recurrence map. A color range from red to blue from the color wheel was used in order to represent Lake Urmia (as shown in Figure 2). Blue is assigned to the maximum recurrence value as water is typically represented as blue in pictures. Furthermore, a range of analogous colors and a variety of colors are used in order to represent close recurrences while at the same time distinguishing recurrences easily. Recurrences close to zero indicate feature absence and are assigned to a white color so that they don't distract one from observing nonzero feature presence. The objective is to maintain an intuitive understanding of the feature of interest with this color choice. Finally, the SIS is overlaid atop Google maps. In order to maintain a relevant location context in the SIS results, only important information such as roads, labels, and major city names have been kept. Figure 1 shows the result for Lake Urmia with a 12 color palette from a recurrence map created in the period 2013-2017.

There are some important observations evident from the result:

1. A desiccation in the wet region of the lake is clearly visible in the result. The inner region is blue while the boundary of the lake is colored otherwise. The upper part of Urmia has water present most of the time, as can be seen in the result image. Because of the difference in recurrence in the upper and lower parts, it looks like the lake could separate into upper and lower parts. Similar conclusions are obtained by combining elevation and surface water data from multiple sensors and satellites to track lake water level [34].

2. Another observation is the existence of several streams and rivers around the lake. It seems that they are important sources of water intake, however in the visualization they are usually presented in dark red. Dark red is assigned to places that rarely have water recurrence during the observation period. The limited sources of water inflow is one of the main reasons for the drying of lake over the years.

 ISBN 978-80-86943-40-4



(a) January (b) February (c) March (d) April (e) May (f) June

(g) July (h) August (i) September (j) October (k) November (l) December

Figure 4: Multiple image lineup of Lake Urmia from 2015.

3. The figure also shows that there is a large red area surrounding the lake boundary. The red color suggests that the region rarely has water present. In [32], it has been speculated that the boundary of the lake usually has salt deposits. Winds carry salt deposits to settlements near the lake which impacts vegetation and causes health problems for people living in these surrounding areas.

Here we discuss the advantages of SIS (Figure 1) over traditional methods of visualizing the layers frame-by-frame (timelapse or animation) or by placing them in a lineup (Figure 4).

- Only one layer at a time can be observed with the timelapse visualization, and the context of both previous and subsequent images are lost. With SIS, its possible to see all changes in the water recurrence over time without losing the context. The recurrence of a feature at p can be easily deciphered by the shade of color at p.
- Showing 12 layers as images side-by-side requires more space and, consequently, either the resolution of each layer has to be decreased or it needs to be presented on a larger screen. As the number of layers increase, both of the previous problems are exacerbated. The resolution of SIS is independent of the number of layers that need to be visualized. So, whether 89 layers or 12 layers are being visualized, the resolution of the summary image doesn't change.
- SIS captures the trends of changes happening in the region. By using SIS, one can evaluate how frequently a feature occurs in a region. The SIS with its color palette easily and immediately highlights which regions have a permanent occurrence of a feature. An example is shown in Figure 1, where one can observe that the upper part of Lake Urmia always has water while the lower part has lesser occurrence.

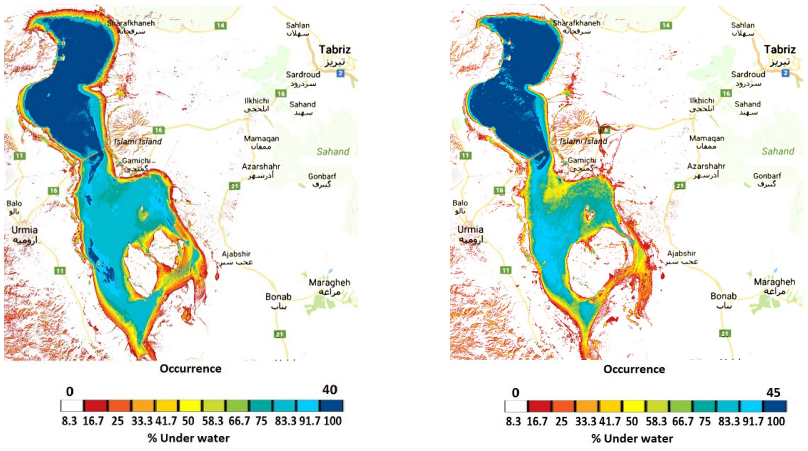
4.1.1 *Comparison of two SIS*

Although using one SIS illustrates a number of time-varying changes, experiments were conducted to observe the benefits of comparing two SIS (as shown in Figure 5). In our experiments, comparisons of two different time periods were done and the results have been shown in Figure 5 On comparing Figure 5a and Figure 5b, the island area in white near the lower right of the image seems to be increasing in size from 2013-2014 to 2015-2016. The islands have almost merged in 2015-2016. The yellow area near Gamichi is increasing in area over time. This indicates less water recurrence and drying of the lake, providing clear support for the increased salt deposition in the area as mentioned in [32].

4.1.2 *Experimentation with different color counts*

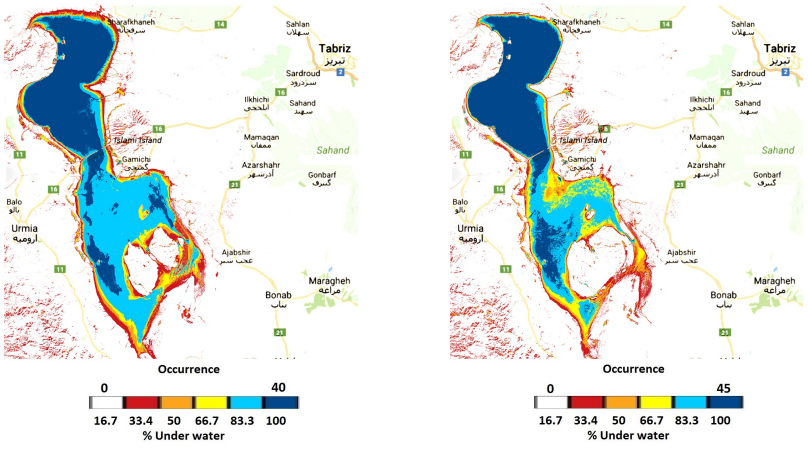
To test the impact of different color counts on results, summaries for 2 year durations have been prepared using a color palette with 6 colors (as shown in Figure 6). One can see similar trends in Figure 5 with 12 colors and Figure 6 with 6 colors. There are some advantages of using a larger number of colors. The color transition is smoother in case of 12 colors. The small number of colors in the palette can mask the underlying changes taking place in the region. For example, the difference between recurrences in the upper and lower parts of Urmia is more prominent in Figure 5b as compared to Figure 6b. There are some patchy deep blue colors that we observe in lower parts of Figure 6b because a wide range of recurrences have been assigned to a deep blue

ISBN 978-80-86943-40-4



(a) 2013-2014 (b) 2015-2016

Figure 5: Experimental results with 12 colors.



(a) 2013-2014 (b) 2015-2016

Figure 6: Experimental results with 6 colors.

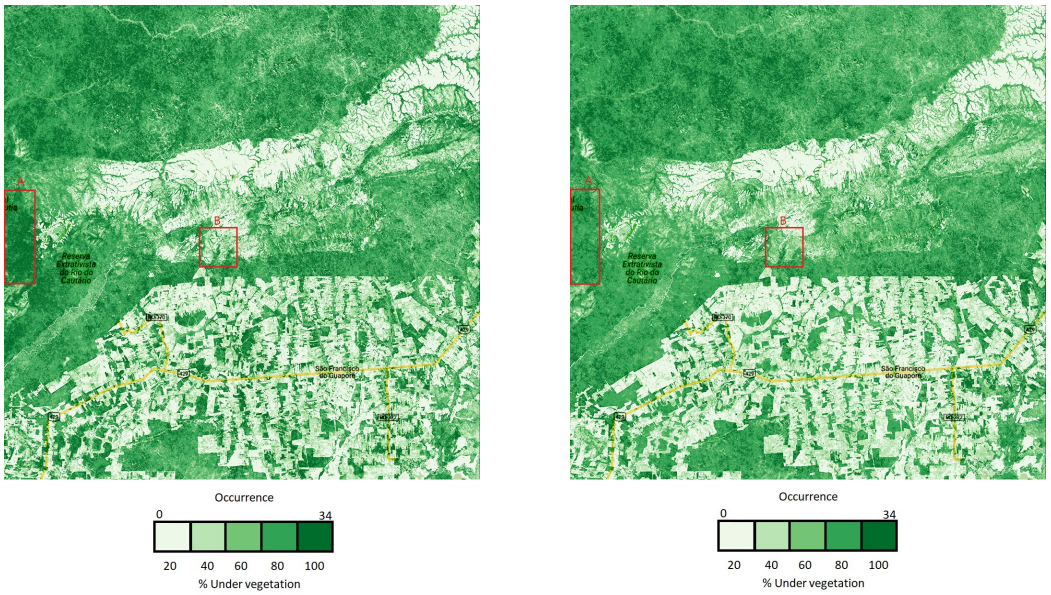
color so its possible to segregate these recurrences in Figure 5b but not in 6b. However, there are some disadvantages of choosing a large number of colors. Using a large number of colors increases redundant patterns in the images (red patches for low recurrences). Moreover, having a large number of colors can make it difficult to distinguish different recurrences.

4.2 Amazon Rainforest

Deforestation helps humans in some sense, but it has extreme negative impacts on climate change and the extinction of flora and fauna [12]. Brazil has recently made huge progress in reducing deforestation and in increasing reforestation [3]. As a result of this progress, heat trapping emissions have been lowered in Brazil as compared to other nations. The international effort known as "Reducing emissions from deforestation and forest degradation" (REDD+) contributed significantly to this achievement [9]. As part of this effort, developing nations reduce deforestation whilst wealthy nations compensate for economic loss. Norway pledged 2.5 billion dollars for the effort and Brazil pledged to reduce the rate of deforestation drastically by 2020.

Rondonia, a state in Brazil, was the most deforested part of the Amazon ecosystem in recent decades [12]. To observe the vegetation trends in the region, a SIS for the region has been created using path 232 and row 68 of Landsat 8 data. A natural and sequential color-palette has been used to generate the SIS. For example, dense vegetation usually looks dark-green in nature as well as in satellite imagery. Thus, a palette of dark-green to white has been used in the Amazon Rainforest case to represent high to low recurrences. A similar palette is used by Google Earth for showing regions with live and dead vegetation. In a recent work, a similar palette was adopted to apply cell shading to terrain features in order to represent trees [18]. The color palette was sampled from the paintings of famous panorama maps artist Heinrich Berann in order to replicate his hand-drawn style.

ISBN 978-80-86943-40-4



(a) 2013-2014

(b) 2015-2016

Figure 7: SIS for the Rondonia region (Amazon Rainforest).

The results are created using 34 layers from the duration of 2013-2014 and 34 layers from the duration of 2015-2016. On comparing Figure 7a and Figure 7b, one can observe that the amount of vegetation has generally decreased in the region. The dark green regions on left of Figure 7a have changed to light green in Figure 7b, which indicates decreased recurrence of vegetation (shown as A in Figure 7). White regions in the center have increased and green regions have decreased in Figure 7b when compared to Figure 7a, which again indicates a decrease in vegetation recurrence. In a few places, there is an increased recurrence too. These results suggest that the deforestation is still going on over the years (shown as A in Figure 7) while there has been some positive results from reforestation efforts (shown as B in Figure 7).

4.3 Bering Glacier

The Bering Glacier is one of the largest glaciers in North America and is located in Alaska. According to [2], the glacier has been retreating at an alarming rate of 10 miles per year. Ground measurements are hard to perform for glaciers due to the harsh conditions prevailing in glacial regions. Thus, remote sensing measurements make it easier to understand ongoing processes in glacial regions.

Using Landsat 8 data from 2013-2017 and Normalized Difference Snow index with an intuitive color palette, Figure 8 has been prepared. The results are shown in Figure 8. The scenes correspond to path 64 and row 18 and we use 5 colors to indicate the snow recurrence. 39 layers from 2013-2014 and 43 layers from 2015-2016 were used to prepare the SIS results. The green color corresponds to very low values of snow recurrence. The cyan color in the bottom half of the image indicates the North Pacific ocean.

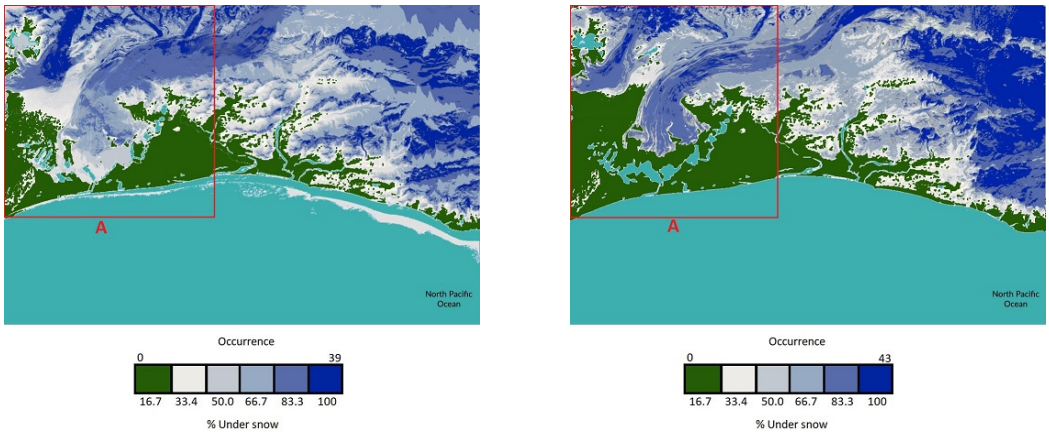
Most of the glacier region has snow present all around the observed duration, which is indicated by deep blue color. The image clearly shows the glacier retreating. The central medial moraine region [31] in the top left between the Bering and Steller Glaciers shows sparse and periodic snow cover (shown as A in Figure 8). On comparing Figure 8a to Figure 8b, it seems that the front of the Bering Glacier has retreated from 2013-2014 to 2015-2016. Furthermore, in 2015-2016, the occurrence of snow around the boundary of the glacier seems to be decreasing (vegetation-snow interface in Figure 8).

5 CONCLUSION

In this paper, we have explored the key aspects of generating single image summaries of Earth-features using Landsat 8 data. We have discussed techniques to identify Earth-features, detect and ignore clouds, reference layers, calculate recurrence of features, and apply a color palette to generate the result. We have presented results and discussed the implications of the results using different counts of colors and layers. We have shown the utility of summaries in detecting changes by describing case studies of several sample areas. Landsat 8 data was obtained from AWS for this research. Our results show change trends which are consistent with other studies and identify new trends which may be utilized by other researchers.

There is room for future work. Clouds hamper the ability to observe land cover in Landsat 8 and sometimes

ISBN 978-80-86943-40-4



(a) 2013-2014

(b) 2015-2016

Figure 8: SIS for the Bering Glacier region.

bring noise into the results. An automatic method for reconstructing data that is hidden by clouds would generate better results. Additionally, it would be interesting to use Landsat 8 along with MODIS and other satellite imagery datasets. Although MODIS has a coarse spatial resolution, its fine temporal resolution of 1 day seems promising towards capturing the change trends.

6 ACKNOWLEDGMENTS

This research received generous support from the Natural Sciences and Engineering Research Council of Canada - Collaborative Research and Development (NSERC-CRD) grant. We would like to thank Global Grid Systems and Shima Dadkhahfard for their insightful discussions and inputs. We would also like to thank Troy Alderson for his editorial comments.

7 REFERENCES

[1] *Basic color schemes: Color theory introduction*, http://www.tigercolor.com/color-lab/color-theory/color-theory-intro.htm, (Accessed on 11/13/2017).

[2] *Bering glacier melts faster | far north science*, http://www.farnorthscience.com/2007/06/04/climate-news/bering-glacier-melts-faster/, (Accessed on 11/10/2017).

[3] *Brazil's success in reducing deforestation (2011)*, https://www.ucsusa.org/global_warming/solutions/stop-deforestation/brazils-reduction-deforestation.html#.Wi7nPd-nFqR, (Accessed on 12/11/2017).

[4] *Lake urmia: how irans most famous lake is disappearing-world news-the guardian*, theguardian.com/world/iran-blog/2015/jan/23/, (Accessed on 12/18/2017).

[5] *Landsat 8 pre-collection quality assessment band | landsat missions*, https://landsat.usgs.gov/qualityband, (Accessed on 01/15/2018).

[6] *Landsat data access | landsat missions*, https://landsat.usgs.gov/landsat-data-access.

[7] *Landsat processing details | landsat missions*, https://landsat.usgs.gov/landsat-processing-details, (Accessed on 11/12/2017).

[8] *Modis web*, https://modis.gsfc.nasa.gov/data/, (Accessed on 12/19/2017).

[9] *Redd+ - home*, http://redd.unfccc.int/, (Accessed on 12/11/2017).

[10] *Remote sensing phenology*, https://phenology.cr.usgs.gov/ndvi_foundation.php, (Accessed on 12/15/2017).

[11] *Timelapse : Google earth engine*, https://earthengine.google.com/timelapse/, (Accessed on 12/05/2017).

[12] *World of change: Amazon deforestation : Feature articles*, https://earthobservatory.nasa.gov/Features/WorldOfChange/deforestation.php, (Accessed on 12/11/2017).

[13] *The worldwide reference system Â« landsat science*, https://landsat.gsfc.nasa.gov/the-worldwide-reference-system/, (Accessed on 11/21/2017).

[14] M Abbaspour and A Nazaridoust, *Determination of environmental water requirements of lake urmia, iran: an ecological approach*, International Journal of Environmental Studies **64** (2007), no. 2, 161–169.

[15] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski, *Visualization of time-oriented data*, Springer Science & Business Media, 2011.

ISBN 978-80-86943-40-4

[16] Roger Beecham, Jason Dykes, Wouter Meulemans, Aidan Slingsby, Cagatay Turkay, and Jo Wood, *Map lineups: effects of spatial structure on graphical inference*, IEEE transactions on visualization and computer graphics **23** (2017), no. 1, 391–400.

[17] Dale HP Boland, *Trophic classification of lakes using landsat-1 (erts-1) multispectral scanner data*, US Environmental Protection Agency, Office of Research and Development, Corvallis Environmental Research Laboratory, Assessment and Criteria Development, 1976.

[18] S. Alex Brown and Faramarz Samavati, *Real-time panorama maps*, Proceedings of the Symposium on Non-Photorealistic Animation and Rendering (New York, NY, USA), NPAR '17, ACM, 2017, pp. 6:1–6:11.

[19] Shima Dadkhahfard, Katayoon Etemad, John Brosz, and Faramarz Samavati, *Area preserving dynamic geospatial visualization on physical globe*, 9th International Conference on Information Visualization Theory and Applications, IVAPP 2018, 2018.

[20] Jason Dykes, Alan M MacEachren, and M-J Kraak, *Exploring geovisualization*, Elsevier, 2005.

[21] Connor C Gramazio, David H Laidlaw, and Karen B Schloss, *Colorgorical: Creating discriminable and preferable color palettes for information visualization*, IEEE transactions on visualization and computer graphics **23** (2017), no. 1, 521–530.

[22] Dorothy K Hall and George A Riggs, *Normalized-difference snow index (ndsi)*, Encyclopedia of snow, ice and glaciers, Springer, 2011, pp. 779–780.

[23] Mark Harrower and Cynthia A. Brewer, *Colorbrewer.org: An online tool for selecting colour schemes for maps*, The Cartographic Journal **40** (2003), no. 1, 27–37.

[24] M Hoseinpour, A Fakheri Fard, and R Naghili, *Death of urmia lake, a silent disaster investigating causes, results and solutions of urmia lake drying*, 1st International Applied Geological Congress, Department of Geology, Islamic Azad University, Islamic Azad University-Mashad Branch, Iran, 2010.

[25] M.C. Hung and Y.H. Wu, *Mapping and visualizing the great salt lake landscape dynamics using multitemporal satellite images, 1972 to 1996*, International Journal of Remote Sensing **26** (2005), no. 9, 1815–1834.

[26] FJ Kriegler, WA Malila, RF Nalepka, and W Richardson, *Preprocessing transformations and their effects on multispectral recognition*, Remote Sensing of Environment, VI, 1969, p. 97.

[27] S. K. McFEETERS, *The use of the normalized difference water index (ndwi) in the delineation of open water features*, International Journal of Remote Sensing **17** (1996), no. 7, 1425–1432.

[28] A. Rav-Acha, Y. Pritch, and S. Peleg, *Making a long video short: Dynamic video synopsis*, 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 1, June 2006, pp. 435–441.

[29] J_W Rouse Jr, RH Haas, JA Schell, and DW Deering, *Monitoring vegetation systems in the great plains with erts*, (1974).

[30] David P Roy, MA Wulder, TR Loveland, CE Woodcock, RG Allen, MC Anderson, D Helder, JR Irons, DM Johnson, R Kennedy, et al., *Landsat-8: Science and product vision for terrestrial global change research*, Remote Sensing of Environment **145** (2014), 154–172.

[31] Robert Allan Shuchman and Edward George Josberger, *Bering glacier: interdisciplinary studies of earth's largest temperate surging glacier*, vol. 462, Geological Society of America, 2010.

[32] Richard Stone, *Saving iran's great salt lake*, Science **349** (2015), no. 6252, 1044–1047.

[33] Alexandru C Telea, *Data visualization: principles and practice*, CRC Press, 2014.

[34] M.J. Tourian, O. Elmi, Q. Chen, B. Devaraju, Sh. Roohi, and N. Sneeuw, *A spaceborne multisensor approach to monitor the desiccation of lake urmia in iran*, Remote Sensing of Environment **156** (2015), no. Supplement C, 349 – 360.

[35] Stef van den Elzen and Jarke J van Wijk, *Small multiples, large singles: A new approach for visual data exploration*, Computer Graphics Forum, vol. 32, Wiley Online Library, 2013, pp. 191–200.

[36] M. M. Verstraete and B. Pinty, *Designing optimal spectral indexes for remote sensing applications*, IEEE Transactions on Geoscience and Remote Sensing **34** (1996), no. 5, 1254–1265.

[37] Andrés Viña, Anatoly A Gitelson, Anthony L Nguy-Robertson, and Yi Peng, *Comparison of different vegetation indices for the remote assessment of green leaf area index of crops*, Remote Sensing of Environment **115** (2011), no. 12, 3468–3478.

[38] Chaoli Wang, Hongfeng Yu, and Kwan-Liu Ma, *Importance-driven time-varying data visualization*, IEEE Transactions on Visualization and Computer Graphics **14** (2008), no. 6, 1547–1554.

[39] Hadley Wickham, Dianne Cook, Heike Hofmann, and Andreas Buja, *Graphical inference for infovis*, IEEE Transactions on Visualization and Computer Graphics **16** (2010), no. 6, 973–979.

 ISBN 978-80-86943-40-4

ISSN 2464-4617(print)
ISSN 2464-4625(CD)
Computer Science Research Notes
CSRN 2801
Full Papers Proceedings
http://www.WSCG.eu

Real-Time Visual Off-Road Path Detection

Marc Steven Krämer, Lars Kuhnert and Klaus-Dieter Kuhnert
Department of Electrical Engineering & Computer Science
Institute of Real-Time Learning Systems
University of Siegen, Germany
marc.kraemer@uni-siegen.de, lars.kuhnert@uni-siegen.de, kuhnert@fb12.uni-siegen.de

ABSTRACT

In this paper, we propose a fast and real-time capable system for visual off-road path detection. We equipped our robot AMOR with a single monocular camera and explored unstructured environments like woods. In these areas, it is almost harder to identify and classify drivable and non-drivable parts in an image. In urban regions, roads can be detected by lane markers or delimitations whereas the boundaries of a forest path blend into the environment almost seamlessly. In our work, we developed a software system that is based on mostly simple and low computationally intensive algorithms. We developed and tested the functions with a large dataset of camera images and also generated a manually Ground Truth for the evaluation.

Keywords

Road Detection, Off-road Autonomous Navigation, Image Segmentation, Terrain Classification, Mobile Robots

1 INTRODUCTION

In recent years, interest in autonomously operating robots has increased. The main task is the navigation from point A to point B as well as the necessary road detection. In general, a distinction can be made between structured roads and unstructured roads (off-road) [8]. Most of the problems in identifying roads in urban areas have widely been solved by the automotive industry[1]. The road boundaries and landmarks are a great help for the detection task in this environment [6]. Therefore, the current challenge is the path recognition in natural, off-road environments. Since it is not necessary to have roads available, the area is divided into passable and non-passable corridors.

The developed path recognition systems are based on various sensors. A laser scanner or stereo camera can detect areas close in front of the vehicle precisely in 3D. However, in order to reach higher speeds it is also necessary to cover other distances that radar or monocular cameras can detect [8]. Other previous projects mainly use stereo cameras .[8][7][4][9][6][5]. In the case of monocular camera-based methods, the main focus was on terrain classification by means of textures and obstacle detection. The classifier used for the path recognition is previously trained with different types of road images. This training is expensive because it has to be done for many types of roads [2] [5] [1]. Another approach splits the camera image into separate grid cells. The cells are now compared by their similarity and subdivided into drivable and non-drivable areas. This is based on a cell directly in front of the vehicle, which is assumed as being a part of the road [3].

The majority of the previously presented procedures have been developed for the use during the DARPA Challenge in America. This runs through the Mojave Desert. In this Paper, we also present a monocular camera based system. The algorithms used in this work have been created especially for our local areas and environments. These include forest areas with more or less fortified paths and different seasons.

This paper is organized as follows: In the next section, we describe the hardware setup and the generated test dataset. The general software architecture and some sub functions are explained in section 3. In section 4, the analyses algorithms for detecting roads are described and will be evaluated in the following section before the paper is finally concluded.

2 HARDWARE SETUP

In this section, we describe our experimental setup. First, we introduce our off-road robot AMOR followed by a description of the test areas and the test image data set.

2.1 Robot AMOR

For our tests, we used the Robot AMOR (Autonomous Mobile Outdoor Robot) which is shown in Figure 1. The outdoor suitable robot was built on the base of

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.9

a quad from Yamaha. The basic concept was the fusion of a large number of redundant sensors to achieve a robust function even in complex and variable environments. The robot is equipped with various different cameras, starting from simple USB webcams to industrial highly-sensitive monocular or stereo camera systems.

Figure 1: Autonomous Mobile Outdoor Robot.

2.2 Test Dataset

To achieve a robust functioning system we needed to collect a huge set of test data. Therefore, we selected five different test paths with a total length of 7.5km and changing road surfaces. The individual routes were approached at different seasons and weather conditions. Thus, we generated a dataset with 23 sequences and 43961 images.

Figure 2: Manual Ground Truth with binary mask.

To evaluate our results, we manually created a Ground Truth and labeled a part of our dataset. This resulted in 1884 images pairs of an original camera image and a corresponding binary mask, where we marked the drivable path (see Figure 2).

2.3 Auxiliary Rectangles

The recordings of the dataset were made with different cameras on various positions on the robot. In addition to the resolution and the aspect ratio even the opening angle of the lens and the position differs. In addition, there are other objects in the visual field, like a lidar sensor or parts of the robots frame. To deal with these problems, we used auxiliary rectangles. As a simple and good option, it has been found to use three rectangles. One rectangle, which points to the road directly in front of the vehicle and two more to the right and left environment. Figure 3 shows this.

Figure 3: Auxiliary rectangles.

These rectangles are not fixed, so the position and size are variables. In a future system, it is planned to use a three-dimensional sensor, like a lidar or stereo-camera, to precisely detect the near surroundings of the robot and classify them into drivable and non-drivable areas. Based on these informations we can adjust the road and environment rectangles to detect the distant surroundings in a fast image processing.

3 SOFTWARE ARCHITECTURE

This section describes the basic structure of the software. The development was based on the idea of a modular and extendible system. In order to understand this structure, we briefly go through the processing chain and describe the individual functions. Figure 4 shows the interaction of the software components in a flow chart.

At the beginning of the path recognition, a pre-test on the recorded image is done which will detect unusable and useless images. In this, for example, care is taken that the image is not extremely over- or underexposed, so that further processings would be impossible. A detailed description is given later in this chapter. If the pre-test fails, the image analysis is canceled and the software waits for the next image. Otherwise, the path type is determined. For this purpose, the current image is classified into a specific path type in a fast process. For example, differences are made between a forest path and a tarred road.

Based on the type of path thus determined, the analysis algorithms used for further analysis are selected. Each analysis algorithm tries to find the path by another method. For example, based on the saturation in the image, the natural environment can be distinguished from the road or, in another method, the edges of the street can be tracked. These chosen analysis algorithms are then executed in parallel.

Each of these functions provides a binary black-and-white mask as output, which specifies where is a path in the image and where not. This mask is evaluated with a plausible test. The results of all analysis algorithms are then merged. This results in a matrix in which parts of the path have a high and the environment a low value.

ISBN 978-80-86943-40-4

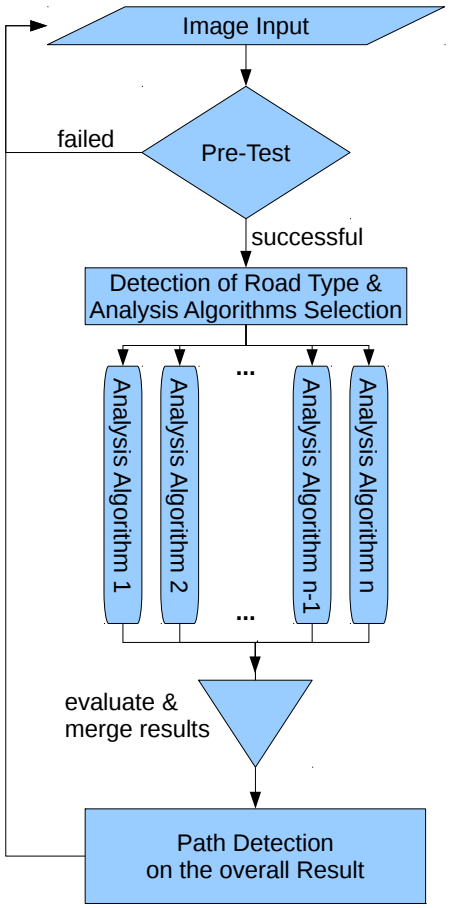


Figure 4: Software flow chart.

In other words, when expressed as an image, the result of the addition is a gray-scale image in which the recognized path has a color value greater than zero. The higher this value is in one pixel, the safer it is a drivable part.

3.1 Image Pre-Test

To detect unusable images we use a pre-test. These can occur, for example, while driving in places where individual shots are overexposed. One of the many other possibilities here is the position directly in front of a house wall, where there is no path to detect. To save computation time we recognize these images and prevent the execution of the analysis algorithms. The method developed here therefore tries to cautiously detect inappropriate images. We use the image areas within the rectangles to make a statement with as little effort as possible whether an image is usable or not. For this, we calculate the mean values of the saturation and blue channel in the segments and compare the ratio between road and environment with a threshold value. In the following equation, this is done by the example of the left environment rectangle and the blue channel.

$$\frac{max(mean_{blue}^{left}, mean_{blue}^{road})}{min(mean_{blue}^{left}, mean_{blue}^{road})} < \text{minDiff} \qquad (1)$$

Analogue for the right rectangle and the saturation of both ones. We found 1.4 for the best working minDiff value. If two or more of the four tests fail, the image is rejected.

During the image pre-test, additionally a binary mask is created which indicates bad points due to high exposure. These occur especially during trips in dark forest areas. In order to detect this, the brightness values (V) in the HSV color space are checked against a threshold value (250) up to the half of the image height, like Figure 5 shows. All image points that are contained in the mask are set to zero.

Figure 5: Highlight test with binary mask on the right.

3.2 Road Type Classification

After a successful pre-test, we detect the road type. Based on this type, the appropriate analysis algorithms will be selected and executed. The road type is distinguished between the drivable path and the surroundings. It is classified into normal road (for example tarred road), dirt road and dirt road with tracks. Figure 6 shows these different path types.

Figure 6: Path types: normal road(1), dirt road saturated environment(2), dirt road (3), dirt road with tracks(4).

We differentiate the environment only between saturated and unsaturated, where a saturated environment is characterized by a (natural) green vegetation. Figure 7 shows how to distinguish a road and a dirt road. On the bottom of the images a histogram of the road rectangle (see Figure 3) is drawn. The histogram of a dirt

ISBN 978-80-86943-40-4

road is usually relatively wide, whereas the histogram of normal roads are quite narrow. To make efficient use of this, we calculate the standard deviation of all RGB-channels. The mean of these values is used to assess the condition of the path in dirt road or normal road. A value greater than 12 means dirt road, a lower one normal road. In case of a dirt road, we also try to detect tracks for the sub type dirt road with tracks. This algorithm is explained in 4.2.

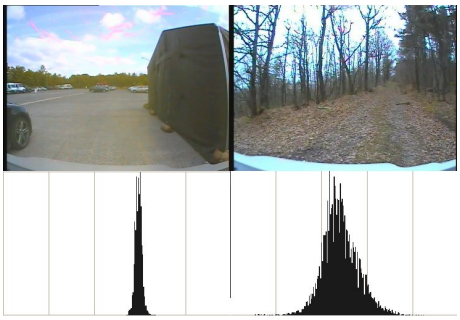


Figure 7: Road Types and Histograms.

In a final step, we have look at the environment. For this, we consider the standard deviation of the saturation channel. If this exceeds the threshold value of 20, the environment is saturated.

4 ANALYSE ALGORITHMS

The following section describes the individual analysis algorithms. Since the total number of algorithms is thirteen, they are only briefly discussed and not considered in detail. However, due to relatively simple procedures, these are easy to understand. Every algorithm gets as an input image a RGB, HSV or grayscale image and returns a binary mask. This mask separates the drivable from the non-drivable parts of the image. All following functions are named by their general idea.

4.1 Normal Road and Dirt Road

In the first part, we describe the analyses algorithms, which we use for the detection of normal and dirt roads.

Blur and Contours Detection

The algorithm BlurAndContours is a method to detect roads within a natural environment. After the image has been blurred, a contour recognition is performed. The result is then converted into a polygon. Figure 8 shows the processing steps. The blurring of the image eliminates fine structures that are common in forest roads, caused for example by fallen leaves. As a result, larger structures are detected during the following contour detection.

The contour detection is performed using the integrated OpenCV function. The necessary threshold value is determined iteratively. For this purpose, the contour

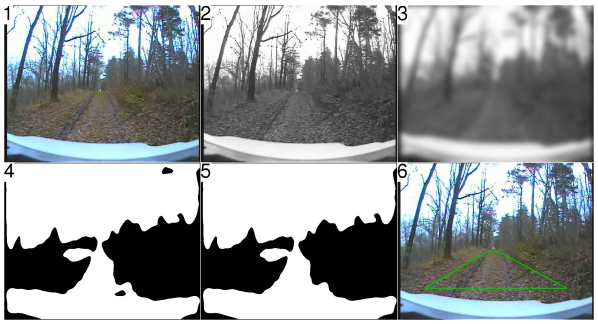


Figure 8: Algorithm Blur and Contours: Input (1), greyscale image (2), greyscale image after bluring (3), mask (4), mask without holes (5), fitted Polygon(6)

recognition is carried out until the suspected path at the top has a width of more than 75 pixels, the last threshold is accepted. The result is usually a large contour, where the narrowest point is the upper edge of the path. After this, we produce a polygon, which starts on two pre-defined origin points in front of the vehicle and ends in this point. The inner of this polygon represents the returned mask.

Equalize Saturation

The Equalize Saturation analysis method is based on changing the saturation values in the HSV color space and comparing the before and after RGB images. At first a histogram equalization in the saturation channel is done. In the histogram equalization process, the color values of a single channel (here saturation) are stretched so that they fill the entire range (0..255), as Figure 9 shows.

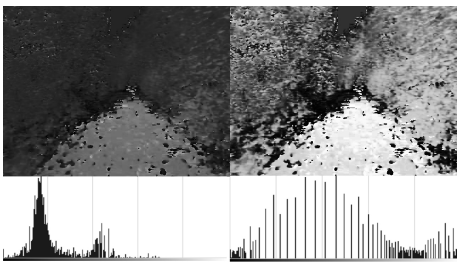


Figure 9: Saturation Histogram Equalization

Subsequently, a comparison of the blue and saturation channels is performed before and after the saturation equalization. Figure 10 shows the differences. In order to decide whether the difference images can be used for a road detection, we calculate the mean difference values in the road and both environment rectangles for the blue color channel. If the changes on the road are above a threshold and the ratio between road and environment changes is very low, the image is rejected, because the differences are not high enough. After the blue channel, the same process is repeated for the saturation channel and both results are added. Figure 11 gives a complete overview of the algorithm.

ISBN 978-80-86943-40-4

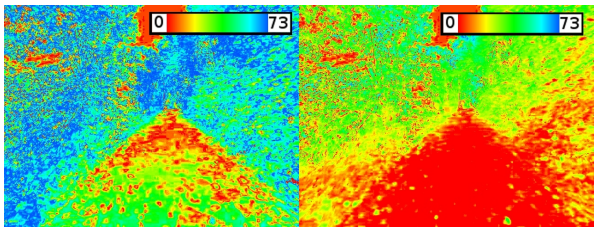


Figure 10: Differences before and after saturation equalization in blue (left) and saturation(right)-channel

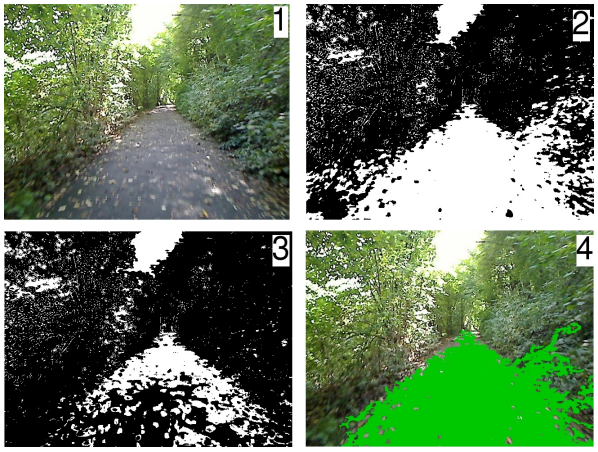


Figure 11: Equalize Saturation: Input(1), blue channel result(2), saturation channel result (3), final result(4).

Polygon Fitting

The polygon fitting method (see Figure 12) offers a very simple and fast, but often well-working way to recognize the road. Here the mask is represented by a polygon whose lower vertices are defined by the street rectangle and upper one by a simple procedure. The result of the algorithm is a mask into which a trapezoid is drawn. The trapezoid is determined by the two bottom and two top points. The two lower points are calculated using an auxiliary point in the center of the street rectangle, where a constant value is added to both sides. For the top points, another upper auxiliary point is required whose X coordinate determines the image column with the highest sum of color values and Y coordinate the image line with the smallest sum of color values as shown in Equation 2 and 3. In order to define two upper points for the trapezoid, here as well as at the lower points, a constant value is added to the left and right point.

$$P_x = x, \text{ with } max(\sum_{y=1}^{height}(I_{xy}^{blue})), x \in \{1,..,width\} \quad (2)$$

$$P_y = y, \text{ with } min(\sum_{x=1}^{width}(I_{xy}^{blue})), y \in \{1,..,height\} \quad (3)$$

Histogram Back Projection

The Histogram Back Projection is a very simple method, which is working on a single channel. The color values inside the road rectangle are stored. All image pixels are compared with this list. If the current value is listed, the pixel is marked as road, otherwise as environment. Before the execution, a pre-test is made on the road rectangle. The distance between the maximum and minimum pixel value has to be lower than a given threshold of 100. Figure 13 shows an example. An alternative method of this function uses the values in the range between the minimum and maximum value in the road values and not only the exact values.

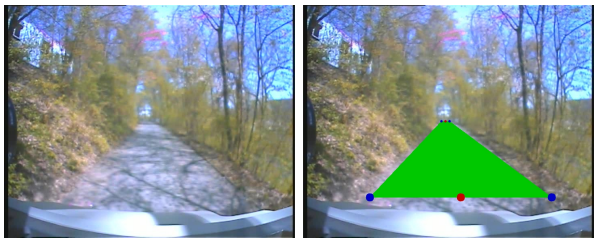


Figure 12: Polygon Fitting: Input(left), Resulting Polygon with auxiliary points(right).

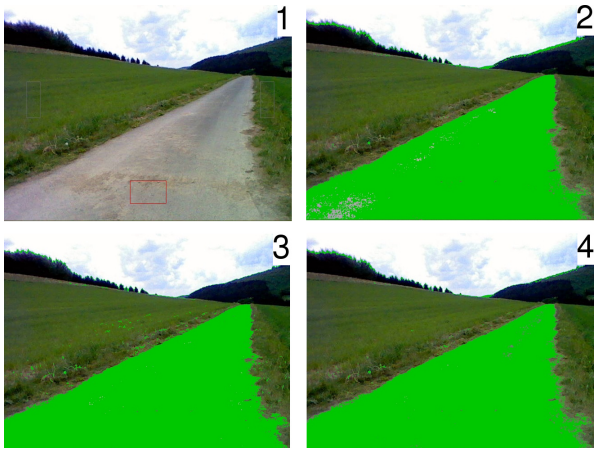


Figure 13: Histogram Back Projection: Input (with road rectangle)(1), Blue channel result (2), Green channel result (3), Val channel result (4).

Template Matching

The template matching method uses the standard OpenCV algorithm. As template area, we use the road rectangle. The following correlation function is used for the template matching process.

$$R_{ccorr}(x,y) = \sum([T(x',y') * I(x+x',y+y')]^2) \quad (4)$$

As result we get a 2-dimensional matrix, with the values of the match, like the third image in Figure 14 shows. To separate the drivable and non-drivable part, a threshold is needed. We calculate this one from the matrix values in the road rectangle as follows

$$threshold = minVal - \frac{meanVal - minVal}{2} \quad (5)$$

ISBN 978-80-86943-40-4

After a binarization of the matrix based on this threshold the resulting image is generated.

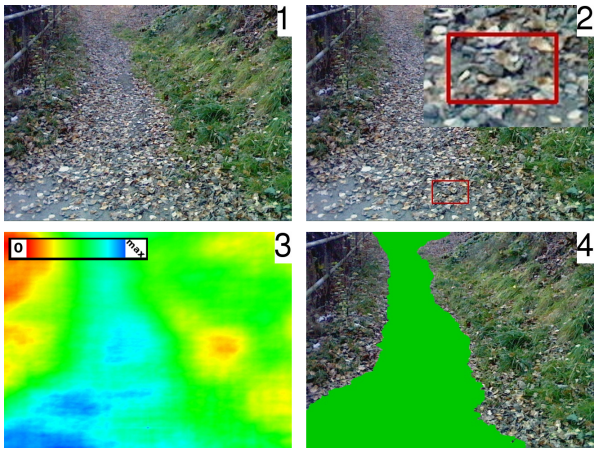


Figure 14: Template Matching:Input(1), Template(2), Template Matching Matrix(3), Result(4).

Region Growing

In a region growing procedure all neighboring points (starting from an initial point) whose colors are within a certain tolerance are marked. In our case, the initial point is in the middle of the road rectangle. As described above, the method sounds quite simple. The challenge is to determine the tolerance values. The easiest way to allow the whole color range within the street rectangle mostly misses. This is because even small disturbances, such as leaves, extend the color space too much. To solve this problem we calculated a histogram of the street rectangle and used only the values in the second and third quartile, as shown green highlighted in Figure 15.

Figure 15: Region Growing: Input with drawn road rectangle(left), histogram and quartile (right).

Before the region growing process, the interquartile range is checked against a threshold. If it is higher than 30, the image is rejected. Figure 16 shows the complete algorithm, which is very well working on normal roads. After the region growing, we eliminate holes in the resulting mask by walking through the lines and selecting the first and last pixel greater than zero.

Increase Saturation

Here we use, similar to the Equalize Saturation method, the difference in the RGB color space before and after

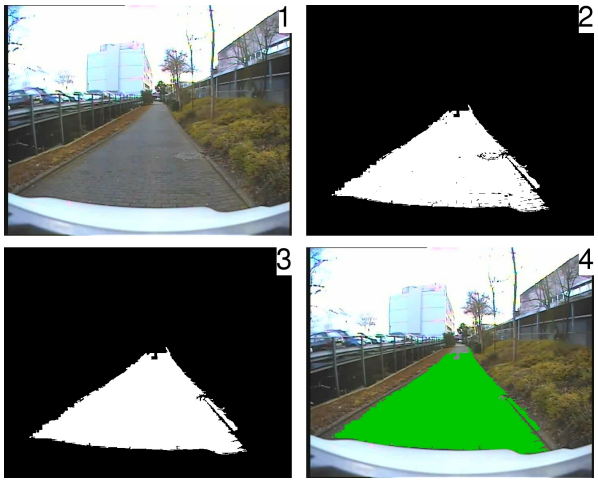


Figure 16: Region Growing: Input(1), Mask after Region Growing(2), Hole Elimination (3), Final Result (4).

a saturation increase. In the RGB color space, differences of the environment are higher than on the street. In contrast to the Equalize Histogram version, a factor for the saturation enhancement is determined in this method. This is done iterative by increasing the factor and comparing the average blue values in the road and environment rectangles until they differ clearly. We found 50 as a good working threshold.

In order to determine the bounds for the comparison, the average change in the road rectangle and the maximum change downwards and upwards between the original and higher saturated image are calculated. *road* means the road rectangle:

$$diffLow = min(I^{orig}_{x,y} - I^{highSat}_{x,y}), (x,y) \in road \quad (6)$$

$$diffHigh = max(I^{orig}_{x,y} - I^{highSat}_{x,y}), (x,y) \in road \quad (7)$$

$$meanDiff = \frac{1}{n}\sum(I^{orig}_{x,y} - I^{highSat}_{x,y}), (x,y) \in road \quad (8)$$

We found the following dynamic threshold values as a good choice for the bounds.

$$low = |(|meanDiff| - |diffLow|)| * 0.5 \quad (9)$$

$$high = |(|meanDiff| - |diffHigh|)| * 0.5 \quad (10)$$

Based on the lower and upper limits, the original image is compared with the higher saturated one. If the color values are within the limits, the point is marked as road, otherwise as environment. An Example is shown in Figure 17. To improve the result we only use parts connected to the road rectangle.

Simple Method Function

This method is a very simple and fast but good method for road detection. It is based on a comparison of all

ISBN 978-80-86943-40-4

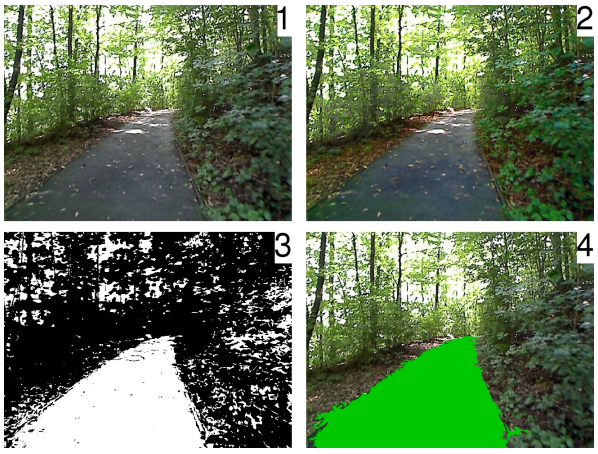


Figure 17: Increase Saturation: Input(1), increased saturation(2), difference between 1 and 2 (3), Final Result (4).

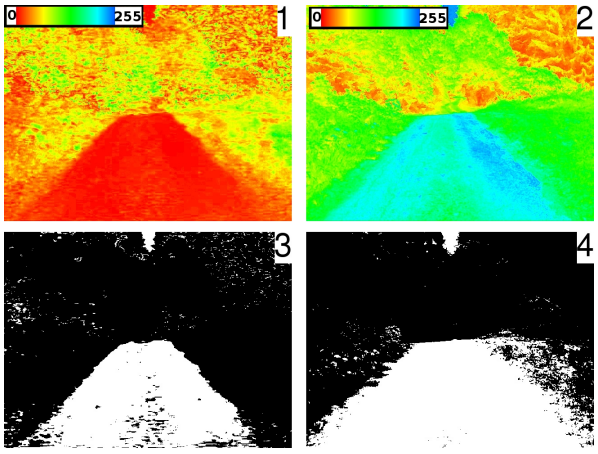


Figure 18: Simple Method Functions: Input saturation(1), Input Value(2), Result saturation (3), Result Value (4).

pixel values with a threshold. The first version is working in RGB color space with the blue channel. We calculate the minimum, maximum and mean blue values in the road rectangle of the image. Based on these values a threshold is determined as follows

$$thr_{blue} = \frac{max_{blue} + 3 * min_{blue} + mean_{blue}}{5} \quad (11)$$

Pixel values greater than this threshold will be selected as road, lower or equal as environment. An analogue function can be applied in HSV color space for the saturation and value channel. Here we use the average values of both environment rectangles for the threshold generation.

$$threshold_{sat} = \frac{max_{sat} + 2 * mean_{sat}}{3} \quad (12)$$

$$threshold_{val} = min_{val} \quad (13)$$

An example in the HSV-Color space is shown in Figure 18. Furthermore, all three simple functions perform a pre-test based on the comparison between road and environment rectangle to reject images.

Roadside Detector

The Roadside detector method searches the two roadsides to differentiate the road from the environment. Figure 19 shows the procedure. Beginning from a start point (center of the road rectangle), we follow scanlines and try to find the border. In order to detect it, we are going through each scanline and compare the mean color value in a rectangle with the color value at the starting point. This is shown for a single and for all scanlines in parts two and three of the Figure. The angles of the scanlines are calculated using the following angular step function. From the roadside points determined this way, a polygon is then assembled which indicates the road.

$$\alpha_{i+1} = \alpha_i + (10^\circ - |sin(2 * \alpha_i| * \alpha_{step}/2) \quad (14)$$

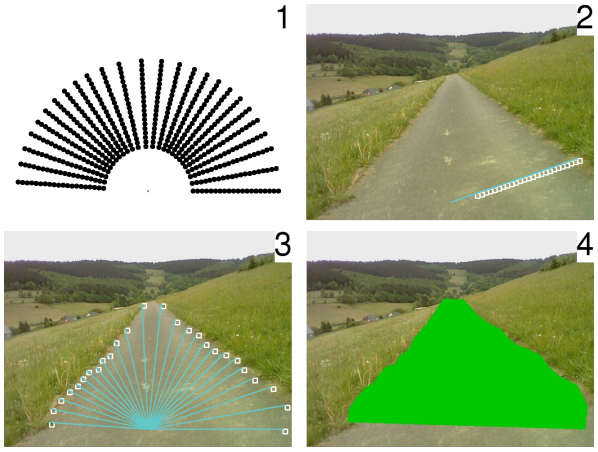


Figure 19: Roadside Detector: Scanline scheme(1), Single Scanline(2), Detected Roadside (3), Final Result (4).

4.2 Dirt Road with Tracks

Next to the previously mentioned roads, we developed special functions for the detection of dirt roads with tracks. At first, we developed a method to detect if tracks in a dirt road are present. The tracks are searched based on a horizontal baseline defined by the position of the road rectangle. This can be seen on the left in Figure 20. Rectangles are now formed on this line, within which the average color values (saturation and blue) are calculated and stored. Subsequently, starting from the middle point, the right and left rectangles are gradually compared with each other. If the saturation subsides, i.e. the green area in the middle has been exceeded and the color values of the rectangles match, these two determine the path tracks. In addition to the statement as to whether tracks are present, the function supplies one point each within the right and left track.

For the detection of dirt roads with tracks, we mostly adapted the previously used functions to this special

ISBN 978-80-86943-40-4

Figure 20: Find Tracks: Horizontal Scanline (1), Detected Tracks (2).

case. Thus, the basic idea is not new and we will only describe these new algorithms brief.

Follow Tracks

The core of the algorithm provides the functionality to follow a track from a given starting point. For this, we compare the color values inside rectangles (cells) as shown in Figure 21. The color values used are the mean values in the saturation, blue, green and red channel.

$$mean_{ch} = \frac{1}{n}\sum(I_{ch}(x,y)),\ (x,y) \in cell \qquad (15)$$

We try to find the best fitting cell in the next horizontal line. It starts with a cell drawn around the starting point and continues recursively until the color values differ too much (difference in two channels is more than 20). For the comparison, we calculate the following metric for each cell

$$diff^i = 2*mean^i_{sat} + 2*mean^i_{blue} + mean^i_{green} + mean^i_{red} \qquad (16)$$

Figure 21: Follow Tracks: Detected Tracks(1), Final Result(2).

Other Track Detection Functions

All other track detection methods are adapted versions of already existing functions and will only be listed and explained within a few sentences. We extended the roadside detection for tracks by running the origin analysis method twice, once in the right and once in the left track. For the result, a polygon with respective outer points is drawn. In the case of the histogram back projection, the reference values can be calculated from rectangles in each of the two tracks instead of the road rectangle, as Figure 22 shows. With the help of the highest points in the individual tracks, a mask can now be generated which describes the road.

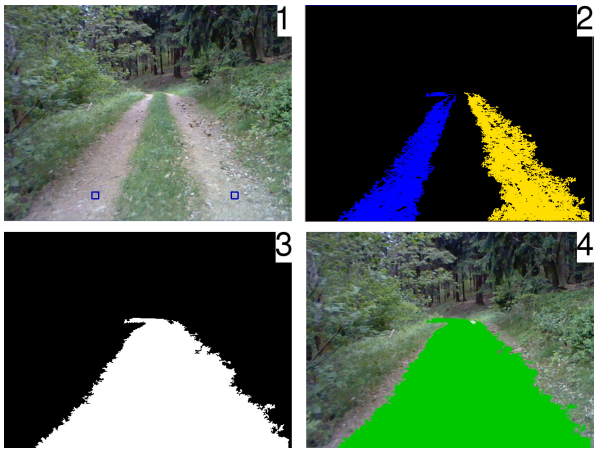


Figure 22: Histogram Back Projection Tracks: Detected Track Startpoints(1), Left and Right Tracks(2), connected mask(3), Final Result(4).

Analog to the Histogram Backprojection we adapted the template Matching and both Simple method functions with two seeding rectangles inside the tracks.

4.3 Evaluation Function

As already described in the last sections, some of the analysis functions detect in an additional pre-test whether the input image is suitable for the implemented method. Generally, we make a plausible test with the resulting mask for all algorithms. For this purpose, several conditions are checked and a scoring with a value between 0 and 100 is done. At first, we check the coverage of the rectangle in the mask. The road rectangle should be covered at least with 25 percent and the environment rectangles with a maximum of 20 percent. If one of these conditions fails, we devalue the score for this mask. The same if the whole mask (more than 95 percent) is covered.

4.4 Algorithm Selection

We made many tests with our dataset to generate a matrix, in which we assign the algorithms to the road and environment types. Some algorithms can be used with RGB or HSV input images, others only with single channels. Thus, the matrix in the following Figure 23 lists also the possible input data and provides as a result a lookup table for which method can be used in which conditions.

As can be seen in the matrix, a larger amount of analysis algorithms is available for the different road types. As already shown in the flow chart in Figure 4, the individual analysis algorithms are executed in parallel and the result masks are combined into a total result. A resulting image built with six algorithms is shown in Figure 24 with a simple implemented point structure for

ISBN 978-80-86943-40-4

	normal road	normal road, sat. Env	dirt road	dirt road sat. Env	dirt road sat. tracks
Simple Method	RGB	RGB,HSV	RGB,HSV	RGB,HSV	
Region Growing	RGB,B	B,S	RGB,B	B,S	
Road Border Detection	B	B,S	B	B,S	
Polygon Fitting	RGB	RGB	RGB	RGB	RGB
Hist. Back Projection	B	B	B	B,S	
Blur and Contours		RGB	RGB	RGB	RGB
Equalize Saturation		RGB		RGB	
Increase Saturation		RGB		RGB	
Track Algorithms					x

Figure 23: Analysis algorithm matrix.

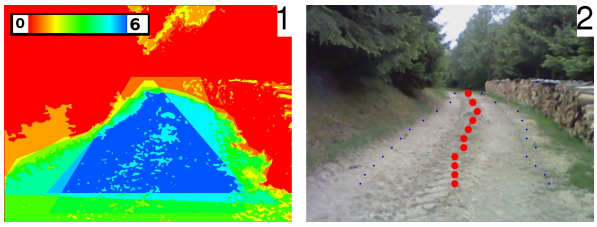


Figure 24: Total Result: pseudo-colored(1), path point structure(2).

the path description. The evaluation function returns a quality value after each analysis method. If the number of algorithms is restricted, only the best n methods of the previous run will be used. Since the rating is continuous, the used algorithms can change with each new input image. This ensures that only those methods are used that have proven themselves in past images. With the help of this option and the number of threads, it is now possible to adapt the complete software system to the processing power of the hardware.

In our software we describe the resulting road with a path-point structure, as shown in Figure 24 on the right. This structure is simply created on the total result mask of all analysis algorithms. To build the point structure, the algorithms walks, starting in the middle of the road rectangle, vertically and determines the width of the road in the total mask. Thus a path is generated where the points lie in the middle of the road.

5 EXPERIMENTAL RESULTS

In this section, we will verify the result of the analysis algorithms with the test dataset. In a first step, it makes sense to compare the resulting mask with the manually created ground truth mask. This comparison evaluates how many points are covered in the ground truth mask and how many additional ones are outside. The two values are given here in percent. So they reflect what percentage of the road was detected and what percentage of the environment was falsely classified as road.

The sole comparison of the mask values as described above would not be enough, because some algorithms (for example Simple Method Functions), usually generate road-pixels outside of the ground-truth mask. Since they occur only sporadically, these are not significant in the generation of the road point structure. Therefore, the main idea of how well the analysis worked can be best determined by the positions of the road points. So we compare the road structure generated on the ground truth mask with the one on the algorithm result mask and calculate the mean and maximum deviation (in pixels).

Our observation showed, that with an average difference of less than 85 pixels the result is useful and describes the path well. If the difference is higher, the result will be worse or unusable. If more road points are detected in one structure than in the other, we devaluate the result. In our evaluation we combined every analysis algorithm with every possible input data (RGB, HSV, red, blue, ...) and interpreted the results. Based on these informations we created the matrix in Figure 23.

In Figure 25 we show one plot as an example. Here it is the algorithm Histogram Back Projection with the blue channel as input data in a short 275 images sequence. At the beginning, the robot was heading towards a wall so that there was nothing to detect. The top plot shows, that during the drive in the later image sequence, the algorithm detects mostly more than 90 percent of the road (green graph). Nevertheless, there are also wrong detections outside the ground truth mask, but they are mostly low (blue graph). On the bottom of the Figure we compared the road point structure created from the algorithms result mask against the one from the ground truth mask. The mean deviation is drawn in red and the maximum (per point) in orange. Additionally there is a line at 85px. This represents the above mentioned border for useful path detections. Between image 200 and the end of the sequence the algorithm gets problems to detect the road. This can be seen by the graphs on the top. In most of these cases the mask evaluation function detected that the images are not suitable. Thus, we rated the deviation in the road point structure as -10, because our software was able to detect, that one function is not working and rejected the image instead of delivering wrong results.

All in all, using a variety of algorithms can cover a wide range of possible mobile robot operating areas. Roads are reliably recognized. Nevertheless, in some parts of the test data the detection unfortunately does not work satisfactorily with any of the algorithms. It is noteworthy, however, that the software detects this and thus does not provide any false results.

5.1 Runtime

An important goal in the implementation was the real-time capability of the software. We determined the runtime for each analysis algorithm on a single core with

ISBN 978-80-86943-40-4

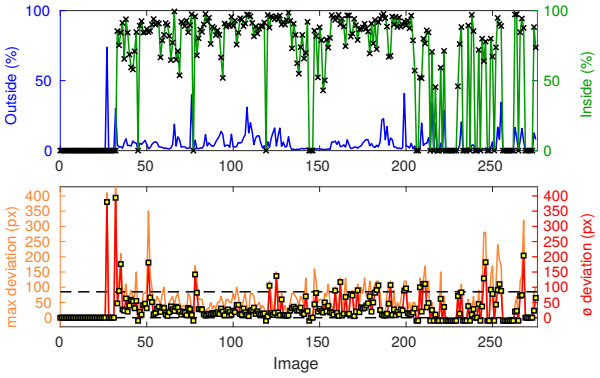


Figure 25: Evaluation of a single analysis algorithm.

the complete image set. The measurement was done on an AMD 635 X4 CPU and a 32-bit Linux operating system. Current high-end CPUs have about ten times the power. The runtimes of the individual algorithms are shown in Figure 26. The short runtime of the template matching is caused on the OpenCV multi-core implementation.

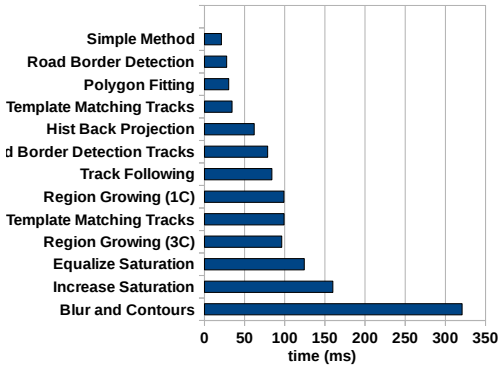


Figure 26: Runtimes of the analysis algorithms.

6 CONCLUSION

In this paper, we presented a software that can perform real-time path detection in different terrains based on monocular camera images using simple algorithms. The results have been obtained and validated with the help of a large test and ground truth dataset. Our software system can be adapted to the available computing resources by an intelligent scheduler, so that it can also be used on low power machines like ARM and no high-end machine is needed. Because of the required real-time capability, more complex approaches to path recognition were not pursued. In a further step, the auxiliary rectangles should be determined dynamically using 3D sensors. For example, a lidar scanner detects the drivable area in front of the vehicle with the spatial information. This road area is now used as a reference for the camera-based path recognition. With his extension, we plan to test the system for autonomous driving with our robot. In addition, the algorithms are to be tested with image data sets of other authors and compared with their results.

7 REFERENCES

[1] Y. Alon, A. Ferencz, and A. Shashua. Off-road path following using region classification and geometric projection constraints. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 1:689–696, 2006.

[2] A. Angelova, L. Matthies, D. Helmick, and P. Perona. Fast terrain classification using variable-length representation for autonomous navigation. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1 –8, june 2007.

[3] A. Broggi, C. Caraffi, S. Cattani, and R. Fedriga. A decision network based frame-work for visual off-road path detection problem. In *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pages 951 –956, 2006.

[4] A. Broggi, C. Caraffi, R. Fedriga, and P. Grisleri. Obstacle detection with stereo vision for off-road vehicle navigation. In *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, page 65, june 2005.

[5] S. Cattani, P. Medici, and G. Vezzoni. Path detection system for autonomous off-road navigation.

[6] H. Kong, J.-Y. Audibert, and J. Ponce. General road detection from a single image. *Image Processing, IEEE Transactions on*, 19(8):2211 –2220, aug. 2010.

[7] R. Manduchi, A. Castano, A. Talukder, and L. Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Auton. Robots*, 18(1):81–102, Jan. 2005.

[8] A. Nefian and G. Bradski. Detection of drivable corridors for off-road autonomous navigation. In *Image Processing, 2006 IEEE International Conference on*, pages 3025 –3028, oct. 2006.

[9] P. Vernaza, B. Taskar, and D. Lee. Online, self-supervised terrain classification via discriminatively trained submodular markov random fields. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2750 –2757, may 2008.

ISBN 978-80-86943-40-4

ISSN 2464-4617(print)
ISSN 2464-4625(CD)
Computer Science Research Notes
CSRN 2801
Full Papers Proceedings
http://www.WSCG.eu

Markerless Structure-based Multi-sensor Calibration for Free Viewpoint Video Capture

Alexandros Papachristou Nikolaos Zioulis Dimitrios Zarpalas Petros Daras
Information Technologies Institute, Centre for Research and Technology - Hellas
papachra@iti.gr nzioulis@iti.gr zarpalas@iti.gr daras@iti.gr

ABSTRACT

Free-viewpoint capture technologies have recently started demonstrating impressive results. Being able to capture human performances in full 3D is a very promising technology for a variety of applications. However, the setup of the capturing infrastructure is usually expensive and requires trained personnel. In this work we focus on one practical aspect of setting up a free-viewpoint capturing system, the spatial alignment of the sensors. Our work aims at simplifying the external calibration process that typically requires significant human intervention and technical knowledge. Our method uses an easy to assemble structure and unlike similar works, does not rely on markers or features. Instead, we exploit the a-priori knowledge of the structure's geometry to establish correspondences for the little-overlapping viewpoints typically found in free-viewpoint capture setups. These establish an initial sparse alignment that is then densely optimized. At the same time, our pipeline improves the robustness to assembly errors, allowing for non-technical users to calibrate multi-sensor setups. Our results showcase the feasibility of our approach that can make the tedious calibration process easier, and less error-prone.

Keywords

Spatial Alignment, External Multi-Sensor Calibration, Semantic Segmentation, Free-viewpoint Capture, RGBD

1 INTRODUCTION

Capturing the complete appearance of real people and general scenes has matured and attracted much interest lately. Be it either offline for high quality free viewpoint video [Ye13] and streamable 3D content [Col15], or in real-time for tele-presence [Esc16, Bec13] and tele-immersion [Zio16] scenarios, it can open up the potential for new immersive experiences in a variety of applications like gaming [Zio16] or remote interactions [Esc16, Bec13].

The backbone of these new experiences is the acquisition of a full 3D representation of general scenes or performances. While a variety of single sensor methods exist, some focusing only on geometry information [New15, Inn16, Zol14], and others also producing fully textured outputs [Guo17, Cao17], truly immersive experiences can only be facilitated by complete 360^o captures via multi-sensor systems. These systems present with both high-quality but expensive solutions [Dou16, Dou17], as well as lower cost ones [Ale17]. Either option utilizes color and depth (RGB-D) information acquired from multiple viewpoints that are spatially aligned, or otherwise externally calibrated, to a common coordinate system. Therefore, this external calibration step is a necessity for all performance capture methods alike.

However, multi-sensor calibration is typically a complex procedure that requires trained users, a requirement that inhibits the applicability of this technology to the consumer public. The complexity arises from the fact that most methods require capturing a calibration object in numerous poses into the captured area [Bec15, Bec17, Fur13, For17], and in some cases this is also performed in a sensor pairwise manner [Hei97]. Some recent methods utilize a static calibration structure to spatially align all viewpoints. In [Col15], calibrating a large amount of cameras is accomplished by using a very complex octagonal tower. There also exist lower complexity structures for setups with less sensors [Kow15, Ale17]. These structure-based multi-sensor calibration methods are more suitable for non-technical users as they require minimal human intervention apart from assembling the structure.

In this work, we lift the requirement of using markers or patterns when utilizing a known structure for calibrating multiple RGB-D sensors. Our main contribution is a correspondence identification step that requires no feature extraction or marker identification. We exploit only the structure's geometrical semantics and segment input depth maps into labeled regions. Therefore, lifting the requirement of markers or patterns, our pro-

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.10

posed method does not require color input and operates on a markerless basis.

In the remainder of the paper we review related work in Section 2 and present our method in detail in Section 3. Then, in Section 4 a detailed evaluation follows, with a concluding discussion presented in Section 5.

2 RELATED WORK

Precise spatial alignment of multi-sensor 3D capturing systems is essential for the creation of realistic 3D human models and assets. Preliminary methods relied only on distributed RGB cameras and were based on the simultaneous capture of a planar rigid printed checkerboard with known dimensions from at least 2 cameras [Hei97, Zha00]. This technique, which is typically used for stereo calibration, is the de facto method to estimate the intrinsics parameters along with the relative pose between neighboring cameras. Despite producing high accuracy results, it requires great effort from the user because the printed checkerboard must be slowly moved and (re-)positioned inside the capturing area. In addition, it also requires knowledge of the technical details behind the calibration process to avoid hard-to-detect checkerboard poses and partial views. Furthermore, it requires hardware synchronized sensors or otherwise, a precise synchronization step for all cameras should precede. Furthermore, the solution is anchored on a selected reference camera as it is not possible to transform all viewpoints to a common global coordinate system. In systems composed of more than 2 sensors, a potential erroneous estimation could be accumulated during the aggregation of relative transformations.

To address the aforementioned limitation, state-of-the-art systems based on the same checkerboard pattern, have incorporated additional optical tracking systems [Bec15, Bec17] or IMU sensors [Fur13]. These alternative methods rely heavily on the tracking systems which are mainly responsible to track the checkerboard's location and define the global coordinate system. Nonetheless, tracking systems require special technical knowledge to mount and operate. Moreover, capturing of the moving checkerboard still requires human intervention, which potentially introduces errors. Further, temporal alignment of the sensors is also needed to synchronously capture the input images. Another associated challenge is the motion blur introduced by the moving checkerboard that can deteriorate the calibration's overall performance.

Specifically for RGB-D sensors, some methods exploit the availability of depth measurements as well as the color information to detect a set of 2D features [Low04, Bay08, Rub11, Alc11] within the capturing area, which can then be converted to 3D points using the depth data. When matched between neighboring viewpoints 3D-to-3D correspondences are established [Dou14], that are used to estimate the relative pose between the sensors. Several works have attempted to enrich the capturing area with features or markers placed on a structure to establish robust 3D correspondences [Ale17, Kow15, Kai12]. Using a common structure offers the advantage of avoiding pairwise calibration and instead, spatially aligns all sensors onto the same coordinate system directly.

However, establishing only sparse correspondences based on detected 2D features or markers is frequently prone to errors due to measurement inaccuracies. To overcome this, dense alignment methods are used that exploit the overlap between viewpoints. Albeit, these still require a rough initial alignment that is given by sparse feature correspondences. Dense methods are usually developed using a variant of the Iterative Closest Point (ICP) algorithm [Kow15, Kin05], graph-based optimization [Ihr04] or bundle adjustment [Van17]. A comprehensive review of refinement methods can be found in [Pom13]. In a similar fashion, other approaches densely estimate the viewpoints of spatially distributed sensors by initially detecting lines and planes [Den14, Owe15, Xu17]. This is succeeded by a post-refinement step to find a globally optimal solution. More recently, a color-based object was utilized and tracked to simultaneously align multiple RGB-D sensors both in the spatial and temporal domain [For17]. It still remains though, a complex process that requires a user to move the object within the scene.

While machine learning algorithms are now abundantly used in various computer vision tasks due to their high performance, they have found little use in calibration tasks. They have been mostly used in localization tasks utilizing decision trees on pure color [Sho13] or RGB-D [Bra14, Bra16] information. Similarly, deep learning variants of these methods have emerged [Ken15, Zam16, Mel17, Nak17, Poi16]. Despite having displayed promising initial results, their accuracy and robustness have not been put to the test of multi-sensor alignment in order to demonstrate their applicability to this specific problem.

3 MARKERLESS STRUCTURE-BASED SPATIAL ALIGNMENT

Our goal is to perform a multi-sensor extrinsic calibration aiming to spatially align the generated point clouds into a common, global, coordinate system. We rely on an easy to deploy calibration structure that is assembled by four equally sized boxes, and more specifically, low-cost commercially available packaging boxes. This approach requires minimal human intervention and is inspired by [Ale17]. Unlike the structure assembled in [Ale17] though, we opt for a simpler assembly process where the boxes are positioned on top of each other,

 ISBN 978-80-86943-40-4

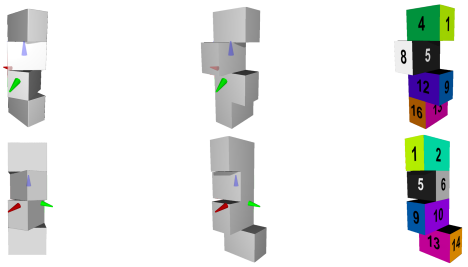


Figure 1: **Left**: The symmetric calibration structure used in [Ale17]. **Middle**: Our asymmetric structure with its corresponding semantic labels per face **(Right)**. Instead of aligning the boxes on top of each other using their diagonals, they are now snapping on their top sides' corners, following a 90° rotational pattern.

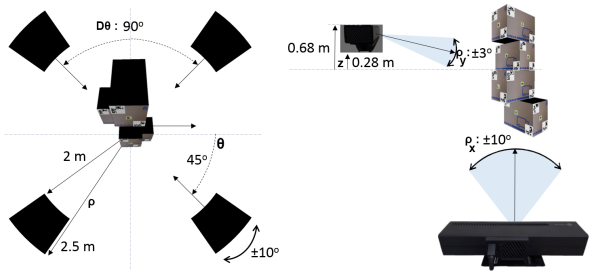


Figure 2: Pose generation process. The range limits of each parameter in equation (1) used to sample/generate the poses are visually presented, showcasing the possible sensor positions around the calibration structure.

while using their side corners instead of their diagonals to snap each box with the one placed on top of it. Fig. 1 showcases the structure of [Ale17], as well as our modified assembly. Another advantage associated to this modification is that the structure is now fully asymmetric, compared to the previous symmetric (i.e. mirrored) assembly. The calibration structure serves as a spatial anchor as all sensor viewpoints' relative pose to its coordinate system (depicted in Fig. 1) will be estimated. This simplifies the calibration process as it removes the necessity of complex pairwise alignments.

Our approach differs from similar approaches that utilize boxes [Kow15] or structures [Ale17], as it does not rely on feature extraction or marker detection. Instead, our correspondence estimation is only reliant on the structure's geometry, as observed by the depth sensor. We exploit the a-priori knowledge related to the structure's shape by training a Fully Convolutional Network (FCN) [Lon15] to identify the structure's boxes' sides. In this way, we perform an initial viewpoint estimation which we then densely refine via a global optimization.

3.1 Semantic Correspondences

Given the now asymmetric geometry, we assign a unique label to each distinct box side and train an FCN for a dense classification task that aims to identify each side in an input depth image. The updated structure's asymmetry allows for easier learning of unique feature descriptors for each viewing direction and is free of any ambiguities that would arise from a symmetric one. The multi-view spatial alignment process can potentially involve a very wide variety of different captured depth data as it involves the full 6 DOF of both the structure and sensor. Training an FCN means we don't have to rely on hand-crafted features or a customized methodology. Training a network for the task of labeling each side, given the numerous possible poses that a sensor can observe the structure, requires a very large dataset. We circumvent the difficult task of manually labeling such a large dataset by synthesizing it. This is accomplished by building a virtual model replica of the calibration structure using the boxes' known dimensions. Nonetheless, creating such a dataset requires a very large amount of storage. Therefore, we chose to simply generate the depth images and labels on-the-fly, using the graphics pipeline to render our data and simulate realistic depth data capturing conditions.

Pose Generation: The structure's coordinate system, and therefore the global coordinate system that all sensors' data will be transformed to, resides on the virtual structure's origin as shown in Fig. 1. Each sensor i'th pose $[\mathbf{R}|\mathbf{t}]^i \in \mathbb{SE}^3$ with respect to the structure, consists of a rotation $\mathbf{R}^i \in \mathbb{SO}^3$ and translation $\mathbf{t}^i \in \mathbb{R}^3$. To generate a large amount of poses we sample positions $\mathbf{t^i} = (t_x^i, t_y^i, t_z^i)$ in a circular pattern around the structure looking towards its origin. We use a cylindrical coordinates sampling (ρ, θ, z) for the position of each sampled viewpoint, omitting the superscripts i for the remainder of this section. A free viewpoint capture setup requires its sensors to look inwards towards its capturing space's center. In our case, this resides on the structure's center position, i.e. the global coordinate frame's origin. Therefore, the poses' rotations $\mathbf{R}$ are set to look at $(0,0,0)$. In practice, though, one cannot achieve such an accurate positioning of the sensor. To compensate for this, we compose additional rotational perturbations ρ_x and ρ_y to each sampled viewpoint, to further augment the variety of sampled poses and capture realistic positioning conditions. These are rotations around the x and y axis respectively, which essentially represent the sensor's pan(right/left) and tilt(up/down) rotations as shown in Fig. 2. For each of these variables we generate discrete samples from a uniform distribution $U(a,b,c)$ at the interval $[a,b]$ in steps of c units:

$$\begin{aligned} \theta &\overset{\alpha}{\sim} U(\alpha - 10^\circ, \alpha + 10^\circ, 2.5^\circ), \\ z &\sim U(0.28m, 0.68m, 0.02m), \\ \rho &\sim U(2.0m, 2.5m, 0.02m), \\ \rho_x &\sim U(-10^\circ, 10^\circ, 2.5^\circ), \\ \rho_y &\sim U(-3^\circ, 3^\circ, 3^\circ). \end{aligned} \tag{1}$$

ISBN 978-80-86943-40-4

where $\alpha = \{45^\circ, 135^\circ, 225^\circ, 315^\circ\}$. The bounds for range ρ and height z, confine the viewpoint within the limits of fully capturing a human subject given a reasonable vertical field of view. In addition, the sensor originated rotations ρ_x and ρ_y, are set in a range of values that are reasonable to position the captured subject close to the sensor's center. Regarding the distribution of the viewpoints around the structure as offered by the cylindrical angle θ, we restrict it around specific 90° intervals, thus focusing only on the case of 4 viewpoint capture. The 4 sensor case is the most optimal solution in terms of cost against quality when aiming for full 360° coverage with the least amount of sensors. By considering an approximate positioning of the sensors around the structure, offered by the selected range of θ angles, we add a restriction in order to decrease the number of input poses and increase the robustness of our predictions. This restriction is a structure placement guideline: "*to have the sides of all boxes looking in between of two sensors*", as illustrated in Fig. 2. The same figure also presents the aforementioned sampling spaces, as well as the relative to the structure positioning of the sensor poses that are generated for creating the training data. We generate a total of $N = 530712$ poses $[\mathbf{R}|\mathbf{t}]$.

Data Generation: The on-the-fly data generation process takes as input the 3D virtual model which is decomposed into parts, each part being one side of each box comprising the structure. We generate N samples using the poses $[\mathbf{R}|\mathbf{t}]$ to position the virtual camera and render the model, acquiring the generated z-buffer as the input data depth map $\mathbf{D}(u,v)$. Each part is also assigned a unique label for a total of 25 distinct labels, six sides for each box plus the background. Each labeled part is rendered with a unique color. By also acquiring the swapped color buffer we obtain the ground truth per pixel labeled image $\mathbf{L}(u,v)$. To simulate more realistic input, we add noise on the rendered depth map randomly choosing the noise function for each sample. We use a noise model better suited for disparity based depth maps (e.g. structured light) as presented in [Bar13] as well as a random noise simulation scaled with the depth value of each pixel:

$$\mathbf{D}_n(u,v) = \text{sign}(U(-1,1)) * \mathbf{D}(u,v) * \sigma_d * (1 - e^{\frac{-U(0,1)^2)}{2}}) \quad (2)$$

where $\mathbf{D}_n$ and $\mathbf{D}$ are the noisy and rendered depth maps respectively, U denote random uniform distributions, and σ_d is a depth scaling factor. In addition, we composite the rendered model onto random backgrounds. These are selected uniformly from various cases: i) white noise, ii) Gaussian noise, (both scaled appropriately to produce values within the expected depth ranges), iii) 159 backgrounds drawn from the database of [Was16] (selected one per 30 frames) and iv) 326 backgrounds drawn from in-house recordings with actual people being captured. Therefore, we augment our online generated training corpus using a mix of noisy and real backgrounds as well as two distinct depth noise models, with some examples presented in the supplementary material. It should be noted that we also generate a smaller test dataset from sensor positions not included in the train data, as a result of choosing different starting values and step units in the same ranges as those presented in (1).

Architecture: The detailed deep Fully Convolutional Network (FCN) configuration and architecture used is presented in Figure 3 (Left). It comprises of a multilayer convolution and a symmetric deconvolution network. The first part learns to extract various features from the input depth map, while the second learns to produce the semantic segmentation of the input into its distinct labels, i.e. box sides, out of the extracted features. The final densely predicted labels are computed out of a probability feature vector of size equal to the amount of labels (25) for each pixel, which constitutes the output of our network that is estimated via a softmax function. The resulting prediction map matches the resolution of the input depth map, as while the convolution part reduces the size of the activations, the following deconvolution part enlarges them back to their original size.

Correspondence Establishment: After segmenting the depth map into regions that correspond to each distinct box's sides, we can use this semantic information to establish correspondences between the acquired depth map and the virtual structure model. Initially, we discard the regions labeled as background or the box sides that are facing upwards/downwards. Then, for each remaining segmented region L, we back-project all depth map pixels to 3D (in the sensor's local coordinate system), and extract their median 3D position $\mathbf{m}_L$. Small area labeled regions are heuristically discarded when containing less than n elements. The 3D point $\mathbf{m}_L$ corresponds to the labeled box's rectangular side center point and therefore, we can establish a correspondence in 3D with the known point's coordinates in the asymmetric structure's virtual model. This 3D correspondence establishment is illustrated in Fig. 3 (Right), where the matching of the corresponding median points between two real views and the virtual structure model is presented.

3.2 Global Spatial Alignment

Given the 3D-to-3D correspondences as an input, we determine an initial alignment of all viewpoints by using the generalized Procrustes analysis [Ken05]. For each viewpoint s we obtain its pose $\mathbf{P}_s$ with respect to the global coordinate system that is originated in the structure's virtual model. However, this initial viewpoint estimation may often present slight errors as a result of the sparseness or inaccuracy of correspondences

 ISBN 978-80-86943-40-4

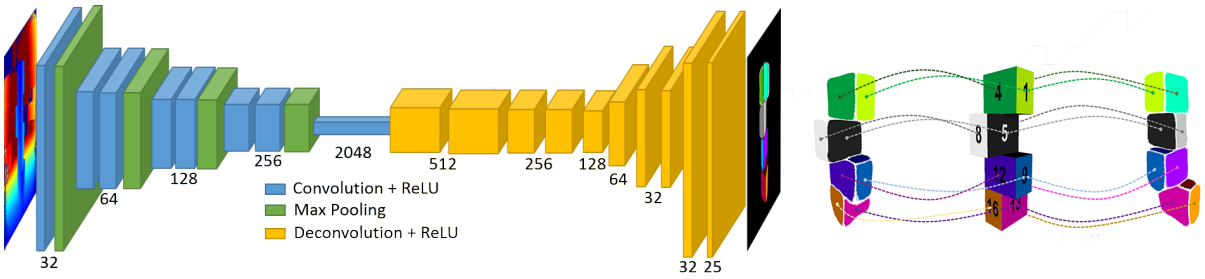


Figure 3: **Left:** The architecture of our semantic segmentation FCN. Having a single depth image as an input, it segments and densely classifies it in order to identify the per-pixel labels of the observed calibration structure. **Right:** Correspondences are established by extracting the 3D medians of the detected labeled regions. These 3D points are then matched against the midpoints of their corresponding virtual structure's box sides to establish 3D-to-3D correspondences. (best viewed in color)

used. This will lead to a visible drop of quality for the produced / captured content. Another reason for inaccurate correspondences is the possibility of an imperfect assembling of the structure, which is more typical when using markers (misplacement) or features (imprecise localization). Thereby, a second step is needed to refine the initial viewpoint estimations by densely aligning the point clouds of adjacent sensors. Instead of a simple pairwise optimization, we solve for an optimal global solution using all viewpoints simultaneously. We use a graph-based optimization where the spatial relationships between the sensor set S are represented by a graph $G = (P, E)$.

The nodes of the graph are the estimated poses $\mathbf{P}_s \in \mathbb{SE}^3$ of each sensor s in the global (virtual structure) coordinate system. The edges E_{ij} represent constraints in the poses between the nodes i and j in the form of observations of j from node i. These observations are established as correspondences $\mathbf{P_i v}_i \leftrightarrow \mathbf{P}_j \mathbf{v}_j$ with $\mathbf{v} \in \mathbb{R}^3$ being a point in the sensor's local coordinate system. These correspondences are acquired by nearest neighbor searches between viewpoints i and j after transformed to the common coordinate system. Each correspondence / edge is encoded as point-to-plane distance:

$$E_{ij} = \|(\mathbf{P}_i^{-1}\mathbf{P}_j\mathbf{v}_j - \mathbf{v}_i)^T \mathbf{n}_i\|_2 \qquad (3)$$

where $\mathbf{n}_i \in \mathbb{R}^3$ is the normal vector of $\mathbf{v}_i$. As depth maps can be noisy around edges, in order to reduce the effect of outliers, we only establish 3D point correspondences within a radius r_{cutoff} between adjacent sensors, with their adjacency estimated by their initial sparse spatial alignment. The graph-based optimization uses the Levenberg-Marquardt method [Mar63] to solve the underlying system, with an iterative scheme. We perform a fixed number of iterations while also dropping r_{cutoff} after a set number of iterations. Solving for all poses simultaneously instead of in a pairwise fashion, we get a globally optimum solution. The refined poses of the viewpoints effectively maximize the overlap between neighboring point clouds. Overall, this dense refinement step rectifies any human-related or systematic errors and improves the quality of the spatial alignment.

4 RESULTS AND DISCUSSION

We evaluate our multi-sensor external calibration method under a variety of 4-sensor setups all focused on free viewpoint capture of human performances. Consequently, the sensors are all looking inwards, towards the center of the capturing area.

Implementation details: Our experiments are based on the Microsoft Kinect 2.0, a Time-of-Flight RGB-D sensor. Our semantic labeling FCN was trained on a NVIDIA Titan X using the Caffe framework [Jia14]. We rendered the generated data using the average Kinect intrinsics parameters (512×424 resolution, a 366.66 focal length baseline and placed the principal point at the depth maps center) to create the projection matrix and trained the FCN on the full resolution images. We train our network for $100k$ iterations with an initial learning and batch size of 0.001 and 5 respectively. We increase the batch size to 15 after $50k$ iterations and linearly decay the learning rate with a gamma of 0.9 every $10k$ and $15k$ iterations when the batch size is 5 and 15 respectively. We use the ADAM optimizer [Kin14] with its standard momentum and epsilon parameters. The threshold for discarding labeled regions n_{pixel} was heuristically selected to be 2000 pixels to discard potential erroneous estimations predicted by the FCN. After training is over, our model achieves a mean Intersection over Union (mIoU) of 86.23% on the generated test dataset. For the refinement step we use the *g2o* framework [Kum11] for 10 iterations and initially set r_{cutoff} to $0.05m$ and drop it to $0.01m$ after 5 iterations.

Data acquisition: As the data acquisition requires capturing a static structure object, the process is free of temporal synchronization or motion blur issues. We exploit this to aggregate frame information within a time window of N frames. Thus, we obtain a median depth map out of 30 frames capturing the static structure. The

ISBN 978-80-86943-40-4

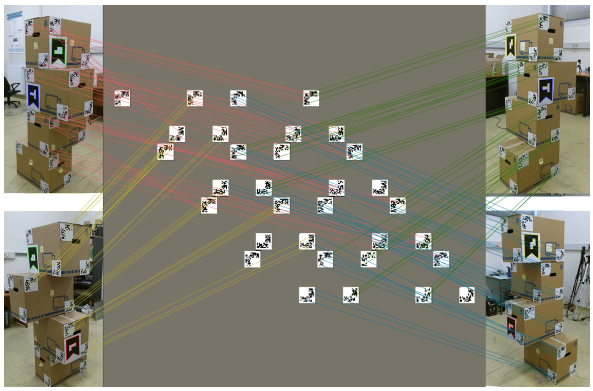


Figure 4: Example calibration views as captured by the color sensor. The SIFT correspondences are also shown, as well as the marker placement for [Ale17] and [Kow15]. SIFT features are matched against the texture applied to the virtual calibration structure model. LiveScan3D marker detections are highlighted on the color images.

median depth map cancels out noise and has less holes, while also being robust to any interference between the sensors. Further, the acquisition process requires minimal human intervention which is limited to assembling the structure near the center of the capturing area, so as to be visible from all sensors simultaneously.

Metric: We measure the accuracy of the registration using the Rooted Mean Squared Euclidean (RMSE) distance between the closest points of overlapping areas of adjacent point-clouds (back-projected from the corresponding depth maps). Given that we seek to measure how well the overlapping surfaces fit and since the viewpoints' overlap is limited as a result of their 90^o intervals placement around the capturing space, we only use those correspondences with distances less than $0.02m$ for each viewpoint pair. We calculate the RMSE error across all adjacent viewpoint pairs for each sensor and average the overall error.

4.1 Evaluation

We compare our method against the structure-based method [Ale17] and "LiveScan3D" [Kow15] that is similarly reliant on attaching markers on rigid surfaces (i.e. boxes). For [Ale17] we utilize the publicly offered markers offered that we attached on the structure following the available instructions. This method only performs spatial alignment based on sparse correspondences. For extracting the SIFT [Low04] correspondences we opt for a brute force matching strategy, instead of approximate versions as we are not bounded by timing constraints. The marker placement and feature matching process is shown in Fig. 4.

For [Kow15], we use the offered set of markers which are used to obtain initial pose estimates. These are then refined by a dense optimization step using pairwise ICP. The "LiveScan3D" markers were also attached on the same calibration structure to allow for simultaneous comparison between all methods as shown in Fig. 4. Markers' positions in the structure's (i.e. global) coordinate system were calculated as they are required as input by [Kow15] to drive the initial registration. For box assemblies that are not perpendicular (e.g. the structure of [Ale17]), this would require some effort by the users to calculate the markers' positions using trigonometry.

Both [Kow15] and our method utilize a post dense refinement step to improve the spatial alignment results, while [Ale17] does not. As a result, we also offer results for [Ale17] by adding a graph-based dense refinement step after the initial alignment obtained by the sparse feature correspondences. We refer to the sparse version as "Sparse-Only" and to the extended post-refinement version as "Sparse+Graph".

We conduct experiments for a variety of setups in order to evaluate all methods in terms of accuracy and robustness. We even purposefully include defective assemblies of the calibration structure to assess each method's efficacy with respect to mis-assemblies (namely f, g and h in Table 1). Given that our markerless correspondence estimation focuses on a particular capturing setup and was trained on these poses only, we use an approximate 4 sensor placement at 90^o intervals around a circle. It should be noted that markers for [Ale17] and [Kow15] were placed on the structure without overlapping as seen in Fig. 4.

Table 1 presents quantitative results of our experiments while also offering each setup's approximate sensor placements. Fig. 6 displays the qualitative results for the same setups. Even though experiments (a) and (b) included sensor poses that were out of the training range, there was no meaningful accuracy degradation compared to other setups, demonstrating how our model has generalized efficiently, well-behaving even in unseen poses. More importantly, while trained on synthetic data with an assortment of augmentations (noise and backgrounds) it has managed to produce high quality segmentation results in realistic data acquired from various sensors as seen in Fig. 5. Segmentation results for all experiments are available in the supplementary material.

Overall, the results presented in Table 1 show that our method outperforms others, except for the SIFT-based one enhanced with the graph-based refinement step. However, our method removes the need for markers, which is a cumbersome and error-prone procedure during the assembly of the structure. Moreover, in the mis-assembly experiments the semantic based method outperforms the marker-based one and, as seen in Fig. 6 it was able to converge in all cases despite the errors, compared to the other methods that did not converge in all cases. This is due to a robuster initial alignment (and

ISBN 978-80-86943-40-4

	a	b	c	d	e	f	g	h
LiveScan3D	6.2	6.42	7.07	6.44	7.35	9.57	**6.30**	10.40
Sparse-Only	10.85	11.94	7.50	8.42	8.66	11.44	11.01	12.36
Sparse+Graph	**5.8**	6.52	**5.84**	**6.18**	**6.29**	9.50	7.43	12.25
Ours	6.39	**6.30**	6.31	6.61	7.56	**6.67**	6.68	**7.15**

Table 1: RMSE results (in mm) of our method and the compared ones. Approximate sensors' placements were: $a \sim \{\rho : 1.7m, z : 0.5m\}$, $b \sim \{\rho : 1.7m, z : 0.28m\}$, $c \sim \{\rho : 2.0m, z : 0.28m\}$, $d \sim \{\rho : 2.0m, z : 0.5m\}$, $e \sim \{\rho : 2.0m, z : 0.5m$ globally rotated compared to d $\}$, $f \sim \{\rho : 2.0m, z : 0.5m$ with translational error$\}$, $g \sim \{\rho : 2.0m, z : 0.5m$ with rotational error$\}$ and $h \sim \{\rho : 2.0m, z : 0.5m$ with both rotation and translational errors$\}$

by extension correspondence estimation) that helps the dense post-refinement step rectify any potential errors.

5 CONCLUSIONS

In this work, we have presented a markerless structure-based external calibration method for multi-sensor setups oriented towards 3D performance capture. Instead of relying on markers to establish correspondences, we exploit the known structure's geometry and train a CNN to semantically label perspective depth maps acquired when viewing the calibration structure. It is an innovative alternative to sparse feature-based spatial alignment that only works with depth input instead of relying on color information. We have demonstrated that this is indeed an effective approach that minimizes human error when assembling the structure and simplifies the overall process. In addition, we showcase how machine learning can be used in the task of multi-sensor spatial alignment. Overall, our method offers an easier and more practical multi-sensor calibration process that is more appropriate for a wider offering of free-viewpoint capture technologies.

Regarding the limitations of our method, it cannot be used to spatially align viewpoints that are looking outwards like showcased in [Kow15]. Additionally, its effectiveness in greater distances is questionable, but that is also a concern in general for depth sensors, whose accuracy degrades proportionally to the measured depth. Further, generalization to any position around the structure is something that should be explored in the future to allow for arbitrary positioning of the sensors (e.g. setups focusing on frontal captures only). Moreover, sparse alignment is reliant on a good segmentation result as erroneous estimates would cause the median calculation to drift. Finally, given that training is coupled to the selected sensor, applicability to a variety of sensor types might require re-training using new intrinsic parameters, however re-training is an one-time requirement.

6 ACKNOWLEDGEMENTS

This work was supported and received funding from the European Union Horizon 2020 Framework Programme funded project *Hyper360*, under Grant Agreement no. 761934. We are also grateful and acknowledge the support of NVIDIA for a hardware donation.

REFERENCES

[Alc11] P. F. Alcantarilla and T. Solutions. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell*, 34(7):1281–1298, 2011.

[Ale17] D. S. Alexiadis, A. Chatzitofis, N. Zioulis, O. Zoidi, G. Louizis, D. Zarpalas, and P. Daras. An integrated platform for live 3d human reconstruction and motion capturing. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):798–813, 2017.

[Bar13] J. T. Barron and J. Malik. Intrinsic scene properties from a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 17–24, 2013.

[Bay08] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008.

[Bec13] S. Beck, A. Kunert, A. Kulik, and B. Froehlich. Immersive group-to-group telepresence. *IEEE Transactions on Visualization and Computer Graphics*, 19(4):616–625, 2013.

[Bec15] S. Beck and B. Froehlich. Volumetric calibration and registration of multiple rgbd-sensors into a joint coordinate system. In *2015 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 89–96, 2015.

[Bec17] S. Beck and B. Froehlich. Sweeping-based volumetric calibration and registration of multiple rgbd-sensors for 3d capturing systems. In *2017 IEEE Virtual Reality (VR)*, pp. 167–176, 2017.

[Bra14] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pp. 536–551. Springer, 2014.

[Bra16] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image.

 ISBN 978-80-86943-40-4

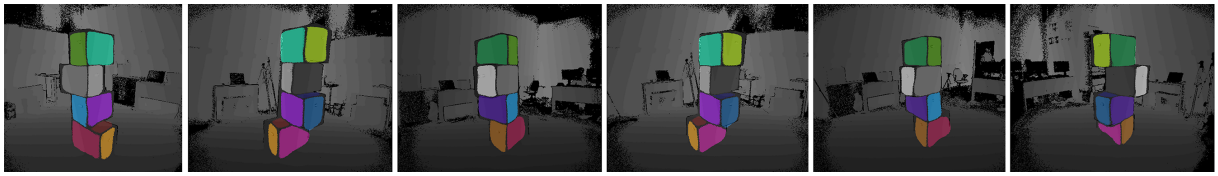


Figure 5: Qualitative results of our model on realistically acquired depth maps. (best viewed in color)

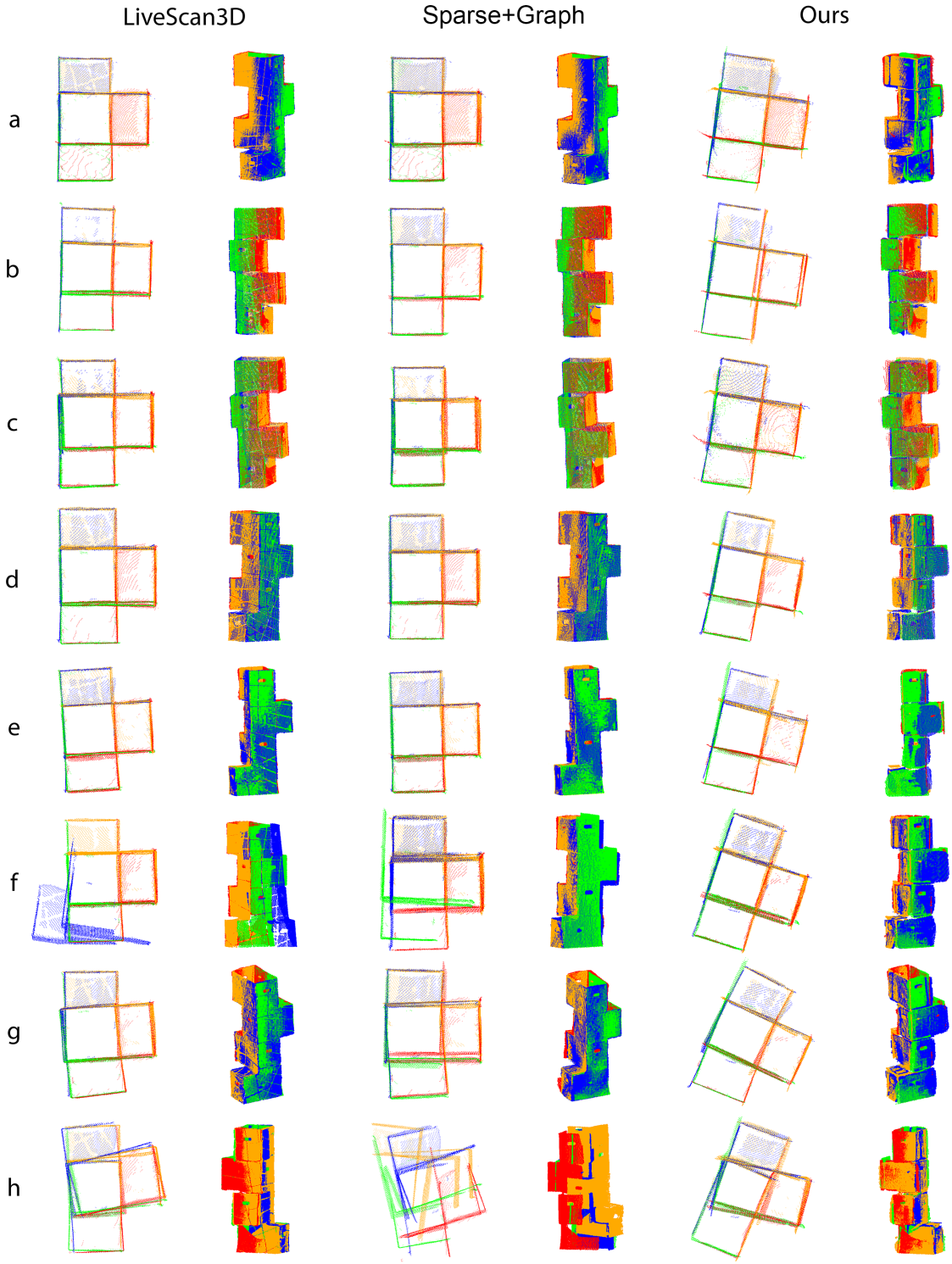


Figure 6: Qualitative results for all experiments. Each row depicts experiment (a-h), whilst each column shows the calibrated point clouds (color-per-sensor) for the evaluated methods in top and side views. (best viewed in color)

ISBN 978-80-86943-40-4

In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[Cao17] H. Zhu, Y. Liu, J. Fan, Q. Dai, and X. Cao. Video-based outdoor human reconstruction. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):760–770, 2017.

[Col15] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (TOG)*, 34(4):69, 2015.

[Den14] D. Teng, J. C. Bazin, T. Martin, C. Kuster, J. Cai, T. Popa, and M. Gross. Registration of multiple rgbd cameras via local rigid transformations. *IEEE International Conference on Multimedia & Expo*, 2014.

[Dou14] M. Dou and H. Fuchs. Temporally enhanced 3d capture of room-sized dynamic scenes with commodity depth cameras. In *2014 IEEE Virtual Reality (VR)*, pp. 39–44, 2014.

[Dou16] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. Orts-Escolano, C. Rhemann, D. Kim, and J. Taylor. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)*, 35(4):114, 2016.

[Dou17] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi. Motion2fusion: real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)*, 36(6):246, 2017.

[Esc16] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, V. Tankovich, C. Loop, Q. Cai, P. A. Chou, S. Mennicken, J. Valentin, V. Pradeep, S. Wang, S. B. Kang, P. Kohli, Y. Lutchyn, C. Keskin, and S. Izadi. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pp. 741–754. ACM, 2016.

[For17] A. Fornaser, P. Tomasin, M. D. Cecco, M. Tavernini, and M. Zanetti. Automatic graph based spatiotemporal extrinsic calibration of multiple kinect v2 tof cameras. *Robotics and Autonomous Systems*, 98(Supplement C):105 – 125, 2017.

[Fur13] P. Furgale, J. Rehder, and R. Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1280–1286, 2013.

[Guo17] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *ACM Transactions on Graphics (TOG)*, 36(3):32, 2017.

[Hei97] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1106–1112, 1997.

[Ihr04] I. Ihrke, L. Ahrenberg, and M. Magnor. External camera calibration for synchronized multi-video systems. In *WSCG 2004 : the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2004 ; short communication papers proceedings*, Journal of WSCG, pp. 537–544, Plzen, Czech Republic, 2004. UNION Agency.

[Inn16] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pp. 362–379. Springer, 2016.

[Jia14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[Kai12] B. Kainz, S. Hauswiesner, G. Reitmayr, M. Steinberger, R. Grasset, L. Gruber, E. Veas, D. Kalkofen, H. Seichter, and D. Schmalstieg. Omnikinect: Real-time dense volumetric data acquisition and applications. In *Proceedings of the 18th ACM Symposium on Virtual Reality Software and Technology*, VRST '12, pp. 25–32, New York, NY, USA, 2012. ACM.

[Ken05] D. G. Kendall. A survey of the statistical theory of shape. *Statist. Sci.*, 4(2):87–99, 05 1989.

[Ken15] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2938–2946, 2015.

[Kin05] B. J. King, T. Malisiewicz, C. V. Stewart, and R. J. Radke. Registration of multiple range scans as a location recognition problem: hypothesis generation, refinement and verification. In *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)*, pp. 180–187, 2005.

[Kin14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kow15] M. Kowalski, J. Naruniec, and M. Daniluk. Livescan3d: A fast and inexpensive 3d data acquisition system for multiple kinect v2 sensors. In *2015 International Conference on 3D Vision*, pp. 318–325, 2015.

[Kum11] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, 2011.

[Lon15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

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

[Mar63] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[Mel17] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu. *Relative Camera Pose Estimation Using Convolutional Neural Networks*, pp. 675–687. Springer International Publishing, 2017.

[Nak17] Y. Nakajima and H. Saito. Robust camera pose estimation by viewpoint classification using deep learning. *Computational Visual Media*, 3(2):189–198, 2017.

[New15] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 343–352, 2015.

[Owe15] J. L. Owens, P. R. Osteen, and K. Daniilidis. Msg-cal: Multi-sensor graph-based calibration. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3660–3667, 2015.

[Poi16] P. Poirson, P. Ammirato, C. Y. Fu, W. Liu, J. Kosecka, and A. C. Berg. Fast single shot detection and pose estimation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 676–684, 2016.

[Pom13] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. Comparing icp variants on real-world data sets. *Auton. Robots*, 34(3):133–148, 2013.

[Rub11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pp. 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[Sho13] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2930–2937, 2013.

[Van17] F. Vasconcelos, J. P. Barreto, and E. Boyer. Automatic camera calibration using multiple sets of pairwise correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2017.

[Was16] O. Wasenmüller, M. Meyer, and D. Stricker. Corbs: Comprehensive rgb-d benchmark for slam using kinect v2. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pp. 1–7. IEEE, 2016.

[Xu17] C. Xu, L. Zhang, L. Cheng, and R. Koch. Pose estimation from line correspondences: A complete analysis and a series of solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1209–1222, 2017.

[Ye13] G. Ye, Y. Liu, Y. Deng, N. Hasler, X. Ji, Q. Dai, and C. Theobalt. Free-viewpoint video of human actors using multiple handheld kinects. *IEEE Transactions on Cybernetics*, 43(5):1370–1382, 2013.

[Zam16] A. R. Zamir, T. Wekel, P. Agrawal, C. Wei, J. Malik, and S. Savarese. *Generic 3D Representation via Pose Estimation and Matching*, pp. 535–553. Springer International Publishing, Cham, 2016.

[Zha00] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[Zio16] N. Zioulis, D. Alexiadis, A. Doumanoglou, G. Louizis, K. Apostolakis, D. Zarpalas, and P. Daras. 3d tele-immersion platform for interactive immersive experiences between remote users. In *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 365–369, 2016.

[Zol14] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, and C. Theobalt. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG)*, 33(4):156, 2014.

 ISBN 978-80-86943-40-4