CSRN 2701

(Eds.)

Paul Bourke
University of Western Australia, Australia
Vaclav Skala
University of West Bohemia, Czech Republic

Computer Science Research Notes

25. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision
WSCG 2017
Plzen, Czech Republic
May 29 – June 2, 2017

Proceedings

WSCG 2017

Full Papers Proceedings

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

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

CSRN 2701

(Eds.)

Paul Bourke
University of Western Australia, Australia
Vaclav Skala
University of West Bohemia, Czech Republic

Computer Science Research Notes

25. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision
WSCG 2017
Plzen, Czech Republic
May 29 – June 2, 2017

Proceedings

WSCG 2017

Full Papers Proceedings

Vaclav Skala – Union Agency

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

Computer Science Research Notes
CSRN 2701

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

ISSN 2464-4617 (Print) ***ISSN 2464-4625*** (CD/DVD)
ISBN 978-80-86943-49-7 *(Print)* ***ISBN 978-80-86943-44-2*** *(CD/DVD)*

WSCG 2017

International Program Committee

Adzhiev, Valery (United Kingdom)
Anderson, Maciel (Brazil)
Benes, Bedrich (United States)
Bilbao, Javier,J. (Spain)
Bourke, Paul (Australia)
Daniel, Marc (France)
de Geus, Klaus (Brazil)
Drechsler, Klaus (Germany)
Feito, Francisco (Spain)
Ferguson, Stuart (United Kingdom)
Galo, Mauricio (Brazil)
Giannini, Franca (Italy)
Gobbetti, Enrico (Italy)
Gudukbay, Ugur (Turkey)
Juan, M.-Carmen (Spain)
Kenny, Erleben (Denmark)
Kim, Jinman (Australia)
Kim, HyungSeok (Korea)
Lobachev, Oleg (Germany)
Molla, Ramon (Spain)
Montrucchio, Bartolomeo (Italy)
Muller, Heinrich (Germany)
Murtagh, Fionn (United Kingdom)
Pan, Rongjiang (China)
Pedrini, Helio (Brazil)
Platis, Nikos (Greece)
Ramires Fernandes, Antonio (Portugal)
Richardson, John (United States)
Ritter, Marcel (Austria)
Rojas-Sola, Jose Ignacio (Spain)
Sanna, Andrea (Italy)
Segura, Rafael (Spain)
Skala, Vaclav (Czech Republic)
Sousa, A.Augusto (Portugal)
Szecsi, Laszlo (Hungary)
Teschner, Matthias (Germany)
Tokuta, Alade (United States)
Umetani, Nobuyuki (Japan)
Wu, Shin-Ting (Brazil)
Wuensche, Burkhard,C. (New Zealand)
Wuethrich, Charles (Germany)
Yao, Junfeng (China)

WSCG 2017

Board of Reviewers

Aburumman, Nadine (France)
Adzhiev, Valery (United Kingdom)
Anderson, Maciel (Brazil)
Assarsson, Ulf (Sweden)
Averkiou, Melinos (Cyprus)
Ayala, Dolors (Spain)
Benes, Bedrich (United States)
Bilbao, Javier,J. (Spain)
Capobianco, Antonio (France)
Carmo, Maria Beatriz (Portugal)
Charalambous, Panayiotis (Cyprus)
Cline, David (United States)
Daniel, Marc (France)
Daniels, Karen (United States)
de Geus, Klaus (Brazil)
De Martino, Jose Mario (Brazil)
de Souza Paiva, Jose Gustavo (Brazil)
Diehl, Alexandra (Germany)
Dokken, Tor (Norway)
Dong, Yue (China)
Drechsler, Klaus (Germany)
Eisemann, Martin (Germany)
Feito, Francisco (Spain)
Ferguson, Stuart (United Kingdom)
Galo, Mauricio (Brazil)
Garcia-Alonso, Alejandro (Spain)
Gdawiec, Krzysztof (Poland)
Giannini, Franca (Italy)
Gobbetti, Enrico (Italy)
Gobron, Stephane (Switzerland)
Goncalves, Alexandrino (Portugal)
Gonzalez, Pascual (Spain)
Gudukbay, Ugur (Turkey)
Hernandez, Benjamin (United States)
Jones, Mark (United Kingdom)
Juan, M.-Carmen (Spain)
Kenny, Erleben (Denmark)
Kim, HyungSeok (Korea)
Kim, Jinman (Australia)
Kurillo, Gregorij (United States)
Kurt, Murat (Turkey)
Lee, Jong Kwan Jake (United States)
Liu, Yang (China)
Liu, Beibei (United States)
Liu, SG (China)
Liu, Damon Shing-Min (Taiwan)
Lobachev, Oleg (Germany)
Luo, Shengzhou (Ireland)
Marques, Ricardo (Spain)
Mei, Gang (China)
Mellado, Nicolas (France)
Meng, Weiliang (China)
Mestre, Daniel,R. (France)
Meyer, Alexandre (France)
Molina Masso, Jose Pascual (Spain)
Molla, Ramon (Spain)
Montrucchio, Bartolomeo (Italy)
Muller, Heinrich (Germany)
Murtagh, Fionn (United Kingdom)
Nishio, Koji (Japan)
Oberweger, Markus (Austria)
Oyarzun Laura, Cristina (Germany)
Pan, Rongjiang (China)
Pedrini, Helio (Brazil)
Pereira, Joao Madeiras (Portugal)
Pina, Jose Luis (Spain)
Platis, Nikos (Greece)
Puig, Anna (Spain)
Raidou, Renata Georgia (Austria)
Ramires Fernandes, Antonio (Portugal)

Richardson, John (United States)
Ritter, Marcel (Austria)
Rodrigues, Joao (Portugal)
Rojas-Sola, Jose Ignacio (Spain)
Sanna, Andrea (Italy)
Schwaerzler, Michael (Austria)
Segura, Rafael (Spain)
Serano, Ana (Spain)
Sik-Lanyi, Cecilia (Hungary)
Sommer, Bjorn (Germany)
Sousa, A.Augusto (Portugal)
Szecsi, Laszlo (Hungary)
Teschner, Matthias (Germany)
Todt, Eduardo (Brazil)
Tokuta, Alade (United States)
Tytkowski, Krzysztof (Poland)
Umetani, Nobuyuki ()
Umlauf, Georg (Germany)
Vanderhaeghe, David (France)
Vidal, Vincent (France)
Vierjahn, Tom (Germany)
Wu, Shin-Ting (Brazil)
Wuensche, Burkhard,C. (New Zealand)
Wuethrich, Charles (Germany)
Yao, Junfeng (China)
Yoshizawa, Shin (Japan)
YU, Qizhi (United Kingdom)
Zhao, Qiang (China)

WSCG 2017

Full Papers Proceedings

Contents

Image-based information visualization
(or how to unify SciVis and InfoVis)

Alexandru Telea
Institute Johann Bernoulli
University of Groningen
The Netherlands

ABSTRACT

For decades, scientific visualization (SciVis) and information visualization (InfoVis) have been related, but still distinctly separated disciplines. Methods and techniques in the two areas have developed relatively separately, causing an arguably unnecessarily separation in the visualization field. Attempts for unification exist, but are largely based on heuristics, and subject to critique from both the SciVis and InfoVis angles. In this talk, we argue that this separation is not necessary, and, up to large extents, artificial. More specifically, we argue that the difference between SciVis and InfoVis is not a matter of design decisions only, but, more centrally, a matter of representing the structure of large data collections by means of smooth, continuous, encodings. We present a way to cast InfoVis along the same principles as the more classical SciVis, based on a continuous, multiscale, spatial representation of data. Putting it simply, we argue that visualizing large amounts of InfoVis data can use encoding techniques which share the same continuity and multiscale principles as most classical spatial SciVis (or image processing) methods use. In turn, we show how this is possible by means of defining appropriate similarity metrics and encoding principles for InfoVis data. This leverages a wealth of data simplification, encoding, and perception principles, since long available for SciVis data, for the richer realm of InfoVis data. We demonstrate our image-based paradigm by examples covering the visualization of relational, multidimensional, and time-dependent InfoVis.

.

Increasing diversity and usability of crowd animation systems

Ciprian Paduraru
University of Bucharest and Electronic Arts
Bucharest, Romania
ciprian.paduraru2009@gmail.com

ABSTRACT

Crowd systems are a vital part in virtual environment applications that are used in entertainment, education, training or different simulation systems such as evacuation planning. Because performance and scalability are key factors, the implementation of crowds poses many challenges in many of its aspects: behaviour simulation, animation, and rendering. This paper is focusing on a different model of animating crowd characters that support dynamically streaming of animation data between CPU and GPU. There are three main aspects that motivate this work. First, we want to provide a greater diversity of animations for crowd agents than was possible before, by not storing any predefined animation data. Another aspect that stems from the first improvement is that this new model allows the crowd simulation side to communicate more efficiently with the animation side by sending events back and forth at runtime, fulfilling this way use-cases that different crowd systems have. Finally, a novel technique implementation that blends between crowd agents' animations is presented. The results section shows that these improvements are added with negligible cost.

Keywords

animation, crowd, skeleton, GPU, blending, memory, compute shader, vertex shader

1 INTRODUCTION

Crowd simulations are becoming more and more common in many computer graphics applications. It is a critical component nowadays in video games industry (games like Fifa 2017®, Ubisoft's Assassin's Creed®, RockStar's Grand Theft Auto® series), evacuation planning software (e.g. Thunderhead's Pathfinder software®) or phobia treatments applications, where it is being used to create realistic environments. However, crowd simulation and rendering is a costly operation and several techniques were developed to optimize CPU, GPU or bandwidth usage to have as many agents as possible. In our paper, by agent, we mean an individual visible entity in the crowd. Entities can be humans, cars or every other instance the client desires to configure and use. We split a crowd system implementation into two logical parts: simulation and render side. The simulation usually deals with driving behaviours, actions, and events for crowd agents, while the render side deals with animation and rendering of the crowd's agents. The focus of this paper is on the render side, and especially on how to deal with the agents' animations. The main motivation to study animation systems for crowds is that realistic motion and diversity of animations are important aspects of user perception. There are three contributions that this paper adds to the field of animations for crowd systems:

- We do not require to store all animations pose data on the GPU side as similar solutions does. Instead, we use a streaming model of poses which has insignificant performance overhead; This allows crowd systems to have a greater diversity of animation data than before ([Rud05], [Ngu07], [Sho08], [Bea14]).
- Fulfill the requirements of modern crowd systems, where the simulation side often needs to request complex blending between the motion of groups or individual agents, by feeding dynamic input states or events. The inverse communication is also possible: animations can trigger events to the simulation side (more details on Section 3).
- At the moment of this paper, the first documented method to perform blending between animations of agents that share the same animation data streams with different time offsets.

Overall, we consider that by using our approach, animation systems for crowds can get more usability and variation with minimal performance loss.

The rest of the paper is organized as follows. Related work in the field is presented in Section 2. A deeper introduction into current techniques and some other technical requests that are driving our work are presented in Section 3. Section 4 describes the usability of our framework and strategies that we use to implement it. Comparative results are shown in Section 5. Finally, conclusions are given in the last section.

2 RELATED WORK

The ability to adjust the skin mesh of a character without having to redefine the animation is highly desirable, and this is possible using a technique called vertex skinning [Ryd05], [Bea16]. By using vertex shaders, a lot of the computation that was previously done on the CPU side is now moved on GPU side, which proves to be faster for this kind of computations [Sho08]. The main technique applied for rendering animated characters using vertex skinning in crowd systems is called skinned instancing, which significantly reduce the number of draw calls needed to render agents. The way skinned instancing work in [Rud05], [Ngu07], [Sho08], [Bea14] is by writing all the animations data on a GPU texture at once, then have the vertex shader perform skinning operations by extracting the correct pose from that texture for each individual instance. In our approach, the difference is that the GPU texture doesn't have to be filled at initialization time with all animations data. Instead, with some bandwidth cost, we send the poses of some template animations from CPU to the GPU, facilitating this way client driven blending between motions.

While the skinned instancing and skeletal animation are highly desirable for rendering high-quality agents for crowds, mainly because of the current GPUs capabilities nowadays, there are other techniques that can be used to limit the number of polygons rendered without having users notice artifacts. These methods are described below and are used in our framework to decide how to render characters at a lower level of details (LOD). Even if the focus of our paper is to use skeletal animations using instancing for crowds, these techniques can reduce substantially the resources needed to render and animate agents at lower LODs. Image-based approaches have been used to simplify rendering of large crowds in an urban simulation [Tec00]. A discrete set of possible views for a character is pre-rendered and stored in memory. A single quad is then used when rendering the character by using the closest view from the set. Also, as mentioned in [Mil07], impostors present great advantages in current graphics hardware when rendering crowd characters using instancing. In the shader program, the current viewing frustum and character heading can be obtained to compute the texture coordinates that are most similar to the current view and animation pose. In the same category of optimizations, but targeted for simplifying the work of content creators (animators), is the cage-based method mentioned in [Kim14]. This method can be used for lower level of details in parallel with our technique, to increase the number of animations available for lower LODs.

In our approach, we consider that blending between motions of individual or larger groups of agents can be requested by the simulation side (e.g an agent exiting from a boids behavior because an emergent event started). This means that a decision mechanism at simulation layer (e.g. a finite state machine, a behaviour tree or even a neural network) feeds our animation blending mechanism with concrete parameters. Several papers presented below provided an inspirational point for our work and can be used in conjunction with our techniques as plugins.

Motion warping [Wit95] is one of the core techniques used to create whole families of realistic motions which can be derived from a small subset of captured motion sequences and by modifying a few keyframes. [Kov02] presents a method that automatically blends motions in the form of a motion graph. This method can be used by the locomotion system of a crowd to choose the animations needed at runtime from an initial database of assets. The main strategy is to identify a portion between two motion frames that are sufficiently similar such that blending is almost certain to produce a nice looking transition. The identification process compares the distances between the pairs of frames of the two motions to identify the best points for transition. The concept of registration curve was introduced by [Kov03]. Considering the problem of blending two motions, the registration curve creates an alignment curve which aligns the frames for each point on the timewarp curve. A statistical model that can do unsupervised learning over a motion captured database and generate new motions or resynthesize data from it, is introduced by [Bra00] under the name of Style Machines. Finally, an interesting survey of the animation techniques used by several games engines is presented by [Greg14]. This proved to be a motivation for this paper, due to the complex requirements of different game engines.

Compression techniques have also been studied in several papers. One that can be adapted to the same use case as our paper, is [Ari06] where the author is mainly using PCA and clustering (along other tricks, such as virtual markers per bone instead of angle) to compress clips of a motion database that is to be streamed at runtime. The same problem is also tackled in [Hij00] and [Sat05]. Another technique to reduce the memory

needed to store animations is to use dual quaternions representation [Kav07]. These memory footprint optimization techniques can be used independently of the methods described in this paper to allow even more variety. A study for improving crowd simulations in VR is described in [Pel16].

3 METHODS

This section provides an introduction on how skeletal animation works, how the existing solutions are implementing this technique for crowd animations and the new requirements for different crowd systems that appeared over the few past years.

3.1 Basics

One of the most used techniques for animating 3D characters is skeletal animation [Mag88]. Using this technique, a skeleton (represented by a hierarchy of bones) and a skin mesh must be defined for an animated character (a skin mesh is a set of polygons where vertices are associated with their influencing bones and their weights; the process is called rigging). A pose is a specification that assigns for each bone of the skeleton a geometric transformation. The process of skeleton and mesh authoring is defined in a reference pose (usually called bind pose). An animation can be defined by a series of keyframes, each one defining a pose and other metadata attributes or events (as described below). When playing an animation for a given character, the vertices of its skin mesh will follow the associated bones (also called skin deformation). The formula for computing each vertex transform is given in Eq. 1. Consider that each vertex with the initial transform v in the reference pose is influenced by n bones. $M_{ref_i}^{-1}$ is the inverse transform matrix of the i^{th} bone's reference pose transform. This moves the vertex multiplied on the right from model space to the bone's local space. Multiplying this with M_i, the world space bone's transform in the current pose, returns the vertex transform with respect to bone i at the current pose. Finally, multiplying this with the associated weight w_i, and summing up all results give the final vertex transform in world space.

$$v' = \sum_{i}^{n} w_i M_i M_{ref_i}^{-1} v, \quad where \sum_{i}^{n} w_i = 1 \qquad (1)$$

3.2 Current Techniques

The common pattern in animating crowds is to store the entire animation data set on the GPU memory [Ngu07], [Sho08], [Bea16]. The memory representation is a texture where animations are contiguously stored on texture's rows, each row representing a single pose for a single animation. Since the bone's transformation can be stored as a $4x3$ float matrix, and knowing that a texel (a cell in a texture) can store 4 floats, then each bone transformation for a pose can be stored in 3 columns of a texture. The skin mesh geometry is stored on GPU memory, in vertex buffers. A vertex shader program created for skinning transforms each vertex according to equation 1. The weights and $M_{ref_i}^{-1}$ matrix are constant and provided at initialization time. Knowing the current time and animation index assigned to each vertex, transform M_i, is sampled from the texture mentioned above.

The animations must be shared between agents since the memory requirements and processing power won't allow playing individual animations per each agent on large crowds at a reasonable framerate (e.g. have a set of normal walk animations being shared by all agents having a walking speed). To break repetition, an offset system is usually used: if two agents A and B are sharing the same animation T, then agents can have different time offset in this animation, randomly assigned. If offsets are different, then the user could hardly notice the sharing since the postures of agents A and B are different at each moment of time [Ngu07].

3.3 Current limitations and requirements

In the past few years, the requirements for animation systems have evolved significantly [Greg14], and the current documented implementations of crowd systems described above are not able to satisfy these requirements. A collection of these are defined below:

1. Animation state machines instead of simple animation clips are more and more common, with transitions between clips decided by a decision-making layer on the simulation side.
2. Generalized two-dimensional blending depending on input parameters feed at runtime.
3. Animation clips can be authored with event tags such that when the playback gets at certain points on their timeline it triggers an event to the simulation side (e.g play a sound when an agent is hit at correct timing).
4. Partial skeleton blends depending on state and animation layering: agents should be able to play multiple animations at the same time (e.g. walking and waving hands only when observing the human user).

Since the animation data is statically stored in a GPU texture for performance reasons, the above requirements can't be satisfied because input feed and decisions dynamically taken from the simulation side can have only a limited effect on the agent's animations (i.e. the system could allow only simple blending between existing poses). Also, storing the entire animation data set on GPU would significantly limit the number of possible animations that a crowd system can use. These

two main limitations are addressed by our solution and described in Sections 4 and 4.3.

3.4 Animation controllers in our framework

As stated in Section 3.2, a visual animating character definition can be represented as a pair $AnimDef = (Skeleton, Skinmesh)$. Additionally, the implementation of its animation needs a pose buffer and a root animation controller: $Anim_A = (AnimDef_A, P_A, Ctrl_A)$

The pose buffer represents the transformation data for each bone at the current time of the animation: $P_A(i)$ = transform of the i^{th} bone of the animation A's skeleton. The concept of animation controller used in our framework is similar to the one presented in [Greg14] and [Uni16]. Usually, every animation framework system has a visual editor that let users customize a controller and its internal evaluation operation. A base use case is to define an animation controller as representing a single motion clip (no child). Its evaluation returns the pose data at the specified time parameter, and it could involve decoding the animation clip data and performing interpolations between keyframes. Controllers can be represented as trees of operations. Evaluating a controller at time t, means evaluating recursively its children nodes then combining the pose buffer from each child into its own pose buffer (the one attached to the animation it is controlling).

Listing 1: Pseudocode for evaluating a controller

```
ControllerA::Evaluate(t)
    ChildrenList = {C_i |
      C_i is the i^th child controller}
    Evaluate(t, ChildrenList)
    P_A = Combine( P_Ci ).
```

Another base use case is to use a controller to blend between two animations (A and B). Such a controller can have two children controllers: $Ctrl_A$ and $Ctrl_B$. As shown in Eq. 2, the resulting pose of evaluating $Ctrl_C$ is an interpolation between the resulting poses of its two children by variable s (normalized blend time; 0 means start, 1 end).

$$P_c(t) = P_a(t) + (P_b(t) - P_a(t)) * s, \quad s \in [0,1]. \quad (2)$$

Complex trees of controllers can be customized for an animation. For instance, one could use a blend mask to consider only parts of the bones from each children controller. Eq. 3 presents a controller evaluation with three children: from $Ctrl_A$ it takes only the pose for head, $Ctrl_B$ gives the pose for arms, and finally $Ctrl_C$ provides the pose for legs of a biped character. A blend mask is defined as an array of $0-1$ values and has the same length as the pose array. The dot product between the two cancels the pose for bones that are not interesting for the mask (e.g. only the set of bones $S = \{i | B_{mask}(i) = 1\}$ are considered).

$$P_r(t) = B_{head} * P_a(t) + B_{arms} * P_b(t) + B_{legs} * P_c(t). \quad (3)$$

Another example of common animation controller are state machines ([Greg14]) where the transitions are generated / evaluated by triggers / values set from the simulation side (e.g. an AI system). If the controllers above can be implemented on GPU side using shaders for optimizing performance, the ideal running place for the controller representing a state machine would be CPU because of the tight communication between simulations and online data that are usually provided on the CPU side. Another motivation for running controllers on the CPU side instead of GPU, is the use case of authoring motion clips with certain events on their timeline that need to be communicated back to the CPU side. The frequent communication from GPU to CPU would cause serious performance problems that could eliminate the benefits of doing the math operations in shader programs. Usually, modern animation frameworks allow users to customize their own controllers and inject them into the animation system using a provided editor.

4 IMPLEMENTATION

Our solution to solve the requirements 1-4 described in 3.3 and the memory limitation of having all animation data set in memory is addressed by using a mixed model between streaming animations data from CPU to GPU, and storing only a part of the animation data set on the GPU memory.

4.1 User authoring and control

In our framework, an animation stream represents an extension to the tuple definition of an animation (A), as given in Section 3.4: $AStream_A = (AnimDef_A, P_A, Ctrl_A, PH_A)$. The last parameter added is a circular buffer storing the history of the last M values evaluated for P_A. Parameter M can be configured by user and represents the number of frames to be saved in the PH_A buffer - many applications typically consider sampling at $30Hz$ or $60Hz$).

The user input for our framework system can be defined as a tuple: $UserConfig = (StreamPool : AStream[], UniquePool : Anim[], NT, M)$. The StreamPool array contains streams of animations that can be shared by multiple agents. As a demo setup example for a crowd animation system, three types of animation streams could be defined: one playing a walk controller

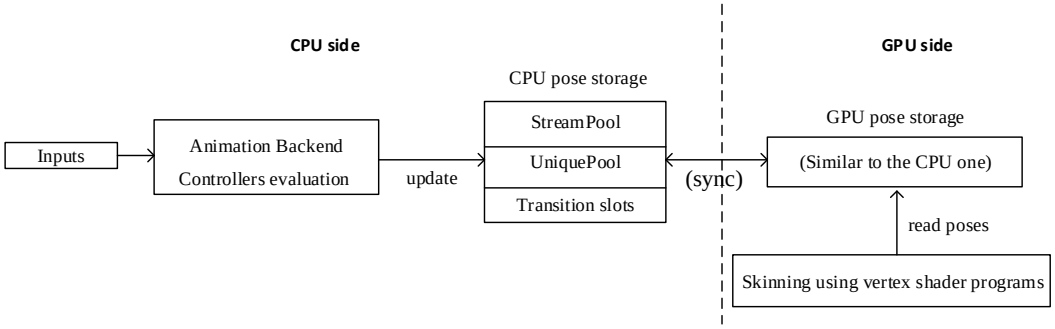


Figure 1: Overview of the CPU-GPU data flow on each frame.

animation, another playing an idle, and finally one waving hands to the user when approaching the camera. Initially, the agents are assigned to one of the streams in the *StreamPool* array, and one offset value in the space of the history poses of that animation stream (random normal distribution or customized by user).

More specifically, if each crowd agent has a unique id assigned, then its animation reference can be defined as a tuple: $AgentAnim_{ID} = (ID_{SI}, ID_O)$, where the first argument is its stream index in StreamPool, and the second is the offset in the history of that stream's saved poses (PH_{SI}). The offset value is relative to the PH's head (i.e. an offset value of 0, means the head of the ring buffer, while an offset N, represents N frames behind head - $PH(N)$). The purpose of the pose history buffer is to allow multiple agents that share the same animation stream look differently by having different offsets. From a quality perspective, having randomizations both at stream index and offset levels decreases significantly the probability of user observing agents that play the same animation at any point in time. This probability can be controlled by adding more or fewer streams of animations and by modifying parameter M.

UniquePool is an array of simple animation definitions that are not meant to be shared between agents - e.g. users can inject at runtime live motions recorded which they want to replicate for the agents in the crowd. Using a streamed animation for this use-case would generate a useless memory footprint for storing the PH parameter. Finally, parameter NT defines the number of preallocated transition slots. Transition slots are internal customizable components that can be used to blend the animation of a single agent from his current animation (and offset) to another one (more details about them in Section 4.3). At runtime, user can request a transition from the current animation of an agent to one of the animations from the stream or unique pools:

- $TransitionToStreamed(ID, streamIndex)$
- $TransitionToUnique(ID, uniqueIndex)$

4.2 CPU-GPU data flow

The data flow between CPU and GPU on each frame is presented in Figure 1. The *Inputs* block is responsible for gathering the inputs for the animation system (e.g. decision making systems deriving states for animation controllers executing state machines or live recorded motions providing online data). Component *AnimationBackend* executes all registered animation controllers and updates their pose information. The CPU pose storage object contains the history poses for *StreamPool*, and only the current pose for unique animations and transition slots. If the pose data for *UniquePool* and transition slots look like an array (one component for each used animation), the *StreamPool* has a more complex representation, presented in Figure 2.

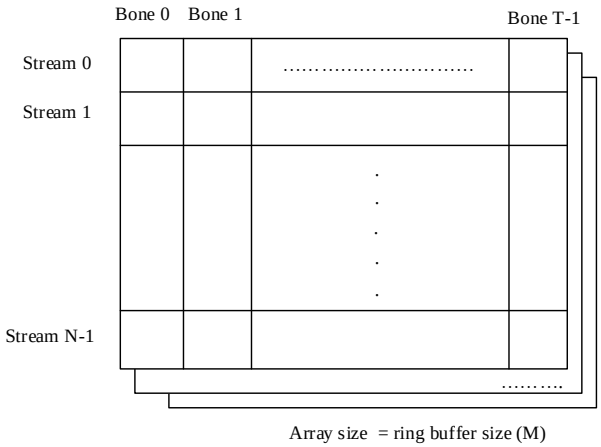


Figure 2: StreamPool storage representation for both CPU and GPU, having N pooled animations, T maximum number of bones and M slices. Each row of a slice represents a pose buffer and has enough capacity to store all skeletons used in the animation system.

At each frame F, for each streamed animation (S), the root controller attached to these animations is evaluated

and writes data on slice (array index) $F \bmod M$, on row $IndexOf(S)$. All these updates (plus the unique and transition slots' poses) are packed and sent to GPU to keep in sync both pose storages. The GPU needs the pose data for skinning purposes, since the shader knows for each vertex the global animation index (unified index system between streamed, unique pools and transition slots), the offset (valid in the case of streamed animations) and fetches the transformation matrix associated with the bones that affect that vertex. The CPU side needs the same pose data because of the blending system explained in the next section.

4.3 Blending between animations

The blending operations in the crowd animation systems is different from the traditional blending mainly because of the opportunities that appear in reusing computations, sharing strategies, and the offset system added to support variety. The transition slots from Figure 1 are responsible for blending between animations, and support transition requests between any animations A and B, with both of them $\in \{UniquePool, DynamicPool\}$. A transition slot T can be defined formally as a tuple:

$Tr_T = (P_T, Ctrl_T, A, B, O_A, O_B, T_0, L, StartP_T, TargetP_T)$. As with the previous definitions, $Ctrl_T$ is the controller attached to this animation and needs to be evaluated to write the output - P_T, representing the current pose of the transition. Skinning is performed using this current pose, similar to the system described in 4.2 (but considering the offset as 0 since only the current pose is written). A, B are the source respectively the target animation for blending, while O_A and O_B, are the offsets that the agent using this transition slot has in each animation (because animations can have different lengths, or for variety purposes, the user can request to blend the agent to a different offset than his current one). T_0 is the start time when the transition started, while L represents the transition duration. Finally, the last two components are explained in the next sub-section.

4.4 Evaluation dependency graph

A typical blend operation between animations A and B using controllers would follow the steps defined in Section 3.4: evaluate controllers for both animations then combine the resulted poses. In the crowd animation system presented in this paper, and knowing that most of the agents are transitioning between streamed animations, blending can be done faster by reusing evaluation results on each frame. The strategy used by our implementation is to create a dependency graph, where tasks (nodes) are the set of all active controllers, and links between them represent dependencies. There is a link between $Ctrl_1$ and $Ctrl_2$, if controller $Ctrl_1$ needs the evaluation results (pose) from $Ctrl_2$. In the case of animations $\in \{StreamPool, UniquePool\}$, there is no dependency. If the transition T is used to play a blending animation between A and B, then $Ctrl_T$ depends only on $Ctrl_B$. The reason why it doesn't depend on $Ctrl_A$ too is that a snapshot of A can be saved, as shown below. Parameter $TargetP_T$ represents the target pose and is evaluated per frame, as follows:

- If $B \in UniquePool$, then $TargetP_T = P_B$ (the pose evaluated in the current frame)
- If $B \in StreamPool$, it must take the value from the pose history of B corresponding to the agent offset in the target animation: $TargetP_T = PH_B(O_B)$.

Parameter $StartP$ represents the pose that the agent had at the moment of transition request in animation A (fixed during transition and internally initialized at the request moment of time):

- If $A \in UniquePool$, then $StartP_T = P_A$
- If $A \in StreamPool$, $StartP_T = PH_A(O_A)$.

Practically, the blending operation is done on each frame between $StartP_T$ and $TargetP_T$, by using an interpolation that considers the current time of transition $(t - T_0)$ and total transition time (L). Since the starting pose is static, an additive blending method is suitable. Listing 2 shows the pseudocode for evaluating the transition controllers.

Listing 2: Pseudocode for transition controller with S representing the normalized blend time, and P the result of evaluation. Note that, according to the definition given above, when transitioning from animation A to animation B, the $StartP$ variable holds the current pose of animation A at the moment of the transition request.

```
TransitionController::Evaluate(t)
    wait Ctrl_B job to finish
    S = (t-T_0)/L
    if B ∈ UniquePool then
        TargetP = P_B
    else
        TargetP = PH_B(O_B)

    DiffPose = (TargetP - StartP)*S
    P = StartP + DiffPose
```

4.5 Parallelization on CPU and GPU

The dependency graph described in the previous subsection can be parallelized efficiently since only the evaluations of transition controllers have dependencies. More, there is one subtle observation that can remove the dependency if the target animation is in $StreamPool$: the wait on $Ctrl_B$ is needed only if

$PH_B(O_B)$ is not already computed (there is a good change to avoid the wait since agent's offsets are statistically behind the head of the pose buffer). For the same use case, the biggest parallelization improvement in our implementation was to move the transition controllers evaluation on the GPU side (i.e using compute shaders). These computations are suitable for compute shaders since the only job of the evaluation is to perform a straight interpolation between poses, and these are already cached on the GPU memory in sync with the CPU storage (Section 4.2).

5 EVALUATION

The animation techniques described in this paper were used in the FIFA17® game (`https://www.easports.com/fifa`). The purpose was to animate a crowd of over 65000 agents displaced in a football stadium, representing different categories of football fans (home, away or ultras). Their animations involved: scoring celebrations, disappointing reactions, anticipation of goals, disagreeing the referee or players' decisions, walking around chairs, etc. Each of these were considered templates and in our tests, we had 256 animations defined in *StreamPool*, and a maximum offset (M) of 60 frames. The number of transition slots allocated was 800, and this number was mainly targeted to support the mexican wave celebration in the stadium (i.e. characters in a specified stadium's area were supposed to stand up and raise their hands at specified moments of time).

For the first purpose of this paper, we are analyzing below the variety of animations in the stadium's crowd that was possible using the techniques described in [Ngu07], [Sho08], [Bea16] (and used in the previous editions of our game) compared against our new implementation. The mixed technique between sending pose data from CPU to GPU and storing only a part of the animations data on the GPU memory increases the variety of supported animations for crowd agents.

On top of this optimization, our new animation system supported dynamic input events as the ones described in Section 3. We were able to make the crowd more realistic with event driven behaviors. One example was agents waving hands or performing different animations on specific bones when a goal scorer was close to them, which was not possible before using preprocessed animations data stored on the GPU memory.

For each skeletal motion clip, on each sampled frame, data must store the SQT (scale, quaternion and transform) of each bone. Denoting by *NBones* the number of bones of the skeleton and by *SizePerBone* the average compressed data for storing the SQT per bone, then the pose data size for a single frame is $PoseSize = NBones \times SizePerBone$. Denoting by $MotionLength$ the average number of motion keys per animation, the average size of a clip is: $ClipSize_{OLD} = MotionLength \times PoseSize$. The memory allocated for animation data must fall under a GPU budget specified by the application. If this variable is denoted by $MemBudget$, then the number of clips that can be used with the previous methods is: $NumClips_{OLD} = \lfloor \frac{MemBudget}{ClipSize_{OLD}} \rfloor$.

By using our new approach, the GPU data that needs to be stored for each clip size is $ClipSize_{NEW} = M \times PoseSize$ (recall that M represents the configurable parameter for the maximum offset value that an agent can have in its shared animation stream). This means that the number of clips that can be stored now relative to the old method is: $NumClips_{NEW} = \lfloor \frac{MotionLength}{M} \times NumClips_{OLD} \rfloor$. The left term is always in $(0,1]$ since the maximum offset cannot exceed the number of frames in the animation.

Figure 3: Image showing a side of a stadium in a football match

.

Analyzing this in the context of our application, where a skeleton with 82 bones was used and with the help of the techniques presented in [Ari06], [Hij00], [Sat05], the resulted *PoseSize* was reduced from $3.2KB$ to $0.8KB$. $MotionLength$ was 300 since the average clip length was 10 seconds, and the sampling rate 30 frames per seconds (fps). In these conditions, the $ClipSize_{OLD} = 246KB$. With a GPU memory budget of $520MB$, if the previous version of application supported only 216 clips, by using a particular maximum offset value M of 60 frames, resulted in supporting up to 1080 different animation clips at the same time. The value used for M was high enough to let the 65000 agents in the crowd look like they perform different animations with only 256 animation streams, and a maximum offset (M) of 60 frames (the number of bones used was 82). The quality of the animations (i.e. if agents animations look different from each other) was evaluated by a quality assurance team who also did some tuning over the variables such that we get good enough results without stressing performance.

In terms of performance, the only theoretical problem could be the bandwidth between CPU and GPU, to keep the pose buffers in sync for the animations driven from the CPU side. However, with modern GPUs and computer architectures, a theoretical bandwidth of 32 GB/s (as specified by PCIe 3.0 standard) between CPU and GPU, would allow our application to send data for more than 100.000 animations on each frame considering our current setup and the other resources competing for the same bandwidth. The techniques described are scalable and can be used on any GPU as long as it satisfies the memory requirements and computational power (shader units) given by the configuration parameters and application's budgets.

Regarding to reactions, consider the case when the ball hits the crowd (i.e one or more agents). In this case, the agents which are effected would blend between their current animation and a hit by ball animation by using a transition slot. The target blend animation could be either a pooled one (hit by ball animation being shared by multiple agents) or a unique one (specific to one agent, client application having custom for the agent using the animation).

6 CONCLUSION

This paper presented some techniques for increasing the diversity and usability of animation systems in applications that use crowds of agents. To increase the diversity, this work changed the strategies used in previous approaches by not storing all the animations data on GPU memory, and by creating a data flow between CPU and GPU. Having animation controllers that can be evaluated using this strategy provides several advantages for the client system in terms of usability: the client is now able to send events back and forth between a simulation layer (e.g. a decision-making system) and an animation system. The techniques presented are used for skeletal animations that target a high level of details and can be combined with existing rendering and animation techniques targeted for a lower level of details. Then, the paper describes a blending technique that is able to perform an efficient transition between agents' animations, considering the sharing strategies specific to crowd systems. Also, a way to parallelize the controllers' evaluation both on CPU and GPU side was sketched. The crowd source code is currently inside a commercial package that we plan to decouple and make it open source for future research. We also want to invest more time in improving the parallelism of the computations and on the AI side of the agents since now we support better animations and instant reaction events.

7 REFERENCES

[Ari06] Arikan, O. Compression of motion capture databases, In Proceedings of SIGGRAPH, ACM, pp. 890-897, 2006.

[Bea14] Beacco A, Pelechano N, CAVAST: The Crowd Animation, Visalisation, and Simulation Testbed, Proceedings of Spanish Computer Graphics Conference, CEIG, pp: 1-10, 2014.

[Bea16] Beacco A, Pelechano N, Andujar C, A Survey of Real-Time Crowd Rendering, Computer Graphics Forum 35(8), pp: 32-50, 2016.

[Bra00] Brand M, Hertzmann A. Style machines, Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co, pp: 183-192, 2000.

[Greg14] Gregory, J. Game Engine Architecture, Second edition, Chapter 11: Animation Systems, CRC Press, pp. 543-647, 2014.

[Hij00] Hijiri T., Nishitani K., Cornish T., Naka T., Asahara S. A spatial hierarchical compression method for 3d streaming animation. Proceedings of Web3D-VRML ACM, pp. 95-101, 2000.

[Kav07] Kavan L., Collins S., Zara J., O'Sullivan C. Skinnig with dual quaternion. In Proceedings of the Symposium on Interactive 3D Graphics (SI3D), pp 39-46, 2007.

[Kov02] Kovar Lucas, Gleicher Michael, Pighin F. Motion graphs, ACM Transactions on Graphics (TOG)., 21(3), pp: 473-482, 2002.

[Kov03] Kovar Lucas and Gleicher Michael. Flexible Automatic Motion Blending with Registration Curves. In Proceedings of ACM SIGGRAPH, Eurographics Symposium on Computer Animation, pp. 214-224, 2003.

[Kim14] Kim J, Seol Y, Kwon T, Lee J, Interactive manipulation of large-scale crowd animation, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2014, 33(4), 2014.

[Mag88] Magnenat-Thalmann N, Laperrire R, Thalmann D, Montreal U. D. Joint-dependent local deformations for hand animation and object grasping, Proceedings on Graphics interface'88, pp. 26-33, 1988.

[Mil07] Millan E, Isaac Rudomin. Impostors, pseudo-instancing and image maps for GPU crowd rendering. Proceedings of the The International Journal of Virtual Reality, Volume 6, Issue 1, pp. 35-44, 2007.

[Ngu07] Nguyen H. GPU Gems 3, Chapter 2: Animated Crowd Rendering. Addison-Wesley, pp. 39-52, 2007.

[Poi09] Poirier M., Paquette E. Rig retargeting for 3d

animation. In Proceedings of the Graphics Interface Conference ACM Press, pp. 103-110, 2009.

[Rud05] Rudomin I, Millan E, Hernandez Z. Fragment shaders for agent animation using finite state machines.Simulation Modelling Practice and Theory, Volume 13, Issue 8 (November), Elsevier, pp. 741-751, 2005.

[Ryd05] Ryder G, and Day A. M, Survey of Real-Time Rendering Techniques for Crowds, Computer Graphics forum 24(2), Wiley, pp: 203-215, 2005.

[Sat05] Sattler M., Sarlette R., Klein R. Simple and efficient compression of animation sequences. In Proceedings of Eurographics Symposium on Computer Animation, pp. 209-217, 2005.

[Sho08] Shopf Jeremy, Joshua Barczak, Christopher Oat, Natalya Tatarchuk , March of the Froblins: simulation and rendering massive crowds of intelligent and detailed creatures on GPU, Proceeding of ACM SIGGRAPH, pp 52-101, 2008.

[Tec00] Tecchia F, Chrysanthou Y. Real-time rendering of densely populated urban environments. In Proceedings of the Eurographics Workshop on Rendering Techniques, Springer, pp. 83-88, 2000.

[Uni16] Unity engine manual - animation section https://docs.unity3d.com/Manual/AnimationSection.html

[Wit95] Witkin A, Popovic Z. Motion warping, Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. ACM, pp. 105-108, 1995.

[Pel16] Pelechano N. and Allbecky J. M., In Proceedings of IEEE Virtual Humans and Crowds for Immersive Environments (VHCIE), pp. 17-21, 2016.

Flexible Moment Invariant Bases for 2D Scalar and Vector Fields

Roxana Bujack
Los Alamos National Laboratory
P.O. Box 1663
USA, 87544 Los Alamos, NM
bujack@lanl.gov

Jan Flusser
Institute of Information Theory and Automation
Pod Vodarenskou vezi 4
Czech Republic, 182 08 Praha 8
flusser@utia.cas.cz

ABSTRACT

Complex moments have been successfully applied to pattern detection tasks in two-dimensional real, complex, and vector valued functions.
In this paper, we review the different bases of rotational moment invariants based on the generator approach with complex monomials. We analyze their properties with respect to independence, completeness, and existence and present superior bases that are optimal with respect to all three criteria for both scalar and vector fields.

Keywords

Pattern detection, moment invariants, scalar fields, vector fields, flow fields, generator, basis, complex, monomial

1 INTRODUCTION

Pattern detection is an important tool for the generation of expressive scientific visualizations. Scientific datasets are ever increasing in size, yet the bandwidth of the human visual channel remains constant. Pattern detection algorithms allow us to reduce this abundance of information to simply features in which the scientist is interested.

One of the challanges in pattern detection is that physical phenomena expressed in coordinates usually come with some degrees of freedom that make the search more complex and time-consuming than inherently necessary. The underlying feature is present no matter how it is oriented. Likewise, the exact position or the scale in which a pattern occurs should not change whether or not it is detected. Using pattern detection algorithms that are independent with respect to these coordinate transformations can therefore significantly accelerate the process.

A common and successful class of such algorithms is based on moment invariants. These are characteristic descriptors of functions that do not change under certain transformations. They can be constructed from moments in two different ways: the generator approach and normalization. Moments are the projections of a function onto a function space basis.

During normalization, certain moments are put into a predefined standard position. The remaining moments are then automatically invariant with respect to this transformation. In contrast, the generator approach uses algebraic relations to explicitly define a set of moment invariants that are constructed from the moments through addition, multiplication, or other arithmetic operations.

Each of these approaches comes with its own advantages and disadvantages. Depending on the application, one may be superior to the other. In this paper, we will concentrate on the generator approach. We begin with a review of generators currently in the literature for two-dimensional scalar and vector fields, demonstrating their differences and dicussing shortcomings; we present a flexible basis able to overcome them.

A set of moment invariants should have the following three important qualities:

Completeness: The set is complete if any arbitrary moment invariant can be constructed from it.

Independence.: The set is independent if none of its elements can be constructed from its other elements.

Existence: The set is existent, in other words flexible, if it is generally defined[1] without requiring any specific moments[2] to be non-zero.

[1] We use the arithmetic meaning of defined. For example, the operation $1/x$ is defined for $x \neq 0$ and undefined if $x = 0$.

[2] As a counter example, the so far suggested basis for real valued functions requires at least one moment to be non-zero that suffices $p_0 - q_0 = 1$.

Completeness ensures that the set has the power to discriminate two objects that differ by something other than only a rotation. Independence accelerates feature detection by preventing comparison of redundant values. Finally, existence guarantees that the set can detect any pattern and does not have restrictions to its specific form, such as having a non-vanishing linear component.

In the real-valued case, a complete and independent set of moment invariants was proposed by Flusser in [1]. We build upon his results to construct a basis that generally exists. Since our basis is flexible, it can be adapted, making it robust even if all moments that correspond to rotational non-symmetric complex monomials are close to zero. Further, it is automatically suitable for the detection of symmetric patterns without prior knowledge of the specific symmetry.

Schlemmer et al. [2] were pioneers in the field, being the first to extend the concept of moment invariants to vector fields. Their suggested generator falls short of being a bona fide basis, according to their own definition, as it does not meet the requirements of completeness and independence. A proof can be found in Section 5.1. Later, Flusser et al. [3] proposed the first complete and independent basis of moment invariants for flow fields. In this paper, we build upon these efforts and introduce a novel basis that meets the full set of standards for a basis. As in the real-valued case, our suggested basis is independent, complete, solves the inverse problem, and additionally is generally existent.

2 RELATED WORK

In 1962, moment invariants were introduced to the image processing society by Hu [4]. He used a set of seven rotation invariants.

Teague [5] and Mostafa and Psaltis citeAMP84 advocated for the use of complex moments. This particularly simplifies the construction of rotation invariants as rotations take the simple form of products with complex exponentials.

In 2000, Flusser [1] presented a calculation rule to compute a complete and independent basis of moment invariants of arbitrary order for 2D scalar functions. He also showed that the invariants by Hu [4] are not independent and that his basis solves the inverse problem [6].

Building on Flusser's work, Schlemmer et al. [2] were the first to derive moment invariants for vector fields. In their pioneering work in 2007, they provided a set of five invariants. Later, in his thesis, Schlemmer also presented a general rule for moments of arbitrary order [7].

Apart from the use of complex numbers, moment tensors are the other common framework for the construction of moment invariants. They were suggested by Dirilten and Newman in 1977 [8]. The principal idea is that tensor contractions to zeroth order are naturally invariant with respect to rotation. It is more difficult to answer questions of completeness or independence in the tensor setting [9], but in contrast to the complex appproach, it generalizes more easily to three-dimensional functions. Pinjo et al. [10], for example, estimated 3D orientations from the contractions to first order, which behave like vectors. Another path that has been successfully taken uses spherical harmonics [11, 12, 13, 14] and their irreducible representation of the rotation group. A generalization of the tensor approach to vector fields was suggetsed by Langbein and Hagen [15].

In contrast to the derivation of explicit calculation rules that generate invariants, normalization can be used. A description of normalization for scalar fields can be found in [3]. Bujack et al. followed the normalization approach to construct moment invariants for two-dimensional [16] and three-dimensional [17] vector fields. Additionally, while Liu and Ribeiro [18] do not call it moment normalization, they follow a very similar approach.

The interested reader can find a detailed introduction to the theory of moment invariants in [3] and an overview of feature-based flow visualization in [19].

3 REAL-VALUED FUNCTIONS

Two-dimensional real valued functions $\mathbb{R}^2 \to \mathbb{R}$ are often embedded into the complex plane $\mathbb{C} \sim \mathbb{R}^2 \to \mathbb{R} \subset \mathbb{C}$ to make use of the easy representation of rotations in the setting of complex numbers. We briefly revisit the foundation of moment invariant bases of complex monomials. A more detailed introduction can be found in [3].

For a function $f : \mathbb{C} \to \mathbb{C}$ and $p,q \in \mathbb{N}$, the complex moments $c_{p,q}$ are defined by

$$c_{p,q} = \int_{\mathbb{C}} z^p \overline{z}^q f(z)\,\mathrm{d}z. \tag{3.1}$$

Let $f'(z) : \mathbb{C} \to \mathbb{C}$ differ from f by an inner rotation by the angle $\alpha \in (-\pi, \pi]$

$$f'(z) = f(e^{-i\alpha} z), \tag{3.2}$$

then, the moments $c'_{p,q}$ of f' satisfy

$$c'_{p,q} = e^{i\alpha(p-q)} c_{p,q}. \tag{3.3}$$

Starting with (3.3), Flusser [1] shows that a rotational invariant can be constructed by choosing $n \in \mathbb{N}$ and for $i = 1,..,n$ integers $k_i, p_i, q_i \in \mathbb{N}_0$. If they satisfy

$$\sum_{i=1}^{n} k_i (p_i - q_i) = 0, \tag{3.4}$$

then, the expression

$$I = \prod_{i=1}^{n} c_{p_i,q_i}^{k_i} \tag{3.5}$$

is invariant with respect to rotation. From this formula, infinitely many rotation invariants can be generated, but most of them are redundant. In order to minimize redundancy, Flusser constructs a basis of independent invariants. The following definitions and the theorem stem from [1].

Definition 3.1. An invariant J of the shape (3.5) is considered to be dependent on a set $I_1,...,I_k$ if there is a function F containing the operations multiplication, involution with an integer exponent and complex conjugation, such that $J = F(I_1,...,I_k)$.

Definition 3.2. A basis of a set of rotation invariants is an independent subset such that any other element depends on this subset.

3.1 Flusser's Basis

The following basis was suggested by Flusser in [1], where the proof of the theorem can be found.

Theorem 3.3. *Cited from [1]. Let M be a set of complex moments of a real-valued function, $\bar{M}$ the set of their complex conjugates and $c_{p_0,q_0} \in M \cup \bar{M}$ such that $p_0 - q_0 = 1$ and $c_{p_0,q_0} \neq 0$. Let $\mathscr{I}$ be the set of all rotation invariants created from the moments of $M \cup \bar{M}$ according to (3.5) and $\mathscr{B}$ be constructed by*

$$\forall p,q,p \geq q \wedge c_{p,q} \in M \cup \bar{M} : \phi(p,q) := c_{p,q} c_{q_0,p_0}^{p-q} \in \mathscr{B}, \tag{3.6}$$

then $\mathscr{B}$ is a basis of $\mathscr{I}$.

This basis satisfies another important property as it solves the inverse problem, meaning up to the one degree of freedom stemming from the rotational invariance, the original moments can be unambiguously reconstructed from the basis [6].

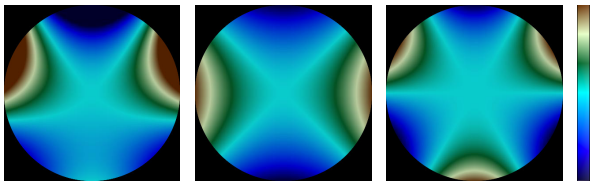
In certain situations, it may occur that no non-zero moment with $p_0 - q_0 = 1$, required for Theorem 3.3, can be found. In this case, Flusser's basis is undefined. However, it is sufficient for c_{q_0,p_0} to have a value close to zero to make the produced invariants unstable and therefore unusable.

Example 3.4. The function

$$f(x,y) = (-y^3 + 3x^2y + x^2 - y^2)\chi(x^2 + y^2 \leq 1) \tag{3.7}$$

with χ corresponding to the characteristic function, has the complex moments $c_{2,0} = \pi/6$, $c_{0,2} = \pi/6$, $c_{3,0} = i\pi/8$, $c_{0,3} = -i\pi/8$, $c_{3,1} = \pi/8$, $c_{1,3} = \pi/8$.

All other moments up to fourth order are zero. There is no $p_0 - q_0 = 1$ with $c_{p_0,q_0} \neq 0$. Therefore, the basis from Theorem 3.3 does not exist. Still, it would be possible to construct moment invariants for f, for example, $c_{3,1}c_{0,2} = \pi^2/48$.



Function (3.7) without rotational symmetry. Its quadratic part with two-fold symmetry. Its cubic part with three-fold rotational symmetry.

Figure 1: The function (3.7) from Example 3.4 and its components visualized using the height colormap.

It should be noted that the situation of vanishing moments always occurs with symmetric functions. In this case, Flusser et al. [20] provide a different basis, tailored toward the specific n-fold rotational symmetry, which needs to be known in advance. However, as can be seen in Example 3.4, all moments with $p_0 - q_0 = 1$ can be zero for non-symmetric functions, too.

3.2 Flexible Basis

Motivated by Example 3.4, we propose the following basis. Since it is adaptive, it exists for any pattern.

Theorem 3.5. *Let $M = \{c_{p,q}, p+q \leq o\}$ be the set of complex moments of an arbitrary real-valued function $f : \mathbb{R}^2 \to \mathbb{R}$ up to a given order $o \in \mathbb{N}$. If there is a $0 \neq c_{p_0,q_0} \in M$ with $p_0 - q_0 < 0$, we define the set $\mathscr{B}$ by $\mathscr{B} := \{\phi(p,q), p+q \leq o, p \geq q\}$ with*

$$\phi(p,q) := c_{p,q} c_{p_0,q_0}^{-\frac{p-q}{p_0-q_0}}, \tag{3.8}$$

and otherwise by $\mathscr{B} := \{c_{p,p}, p+p \leq o\}$. Then $\mathscr{B}$ is a basis of all rotation invariants of M, which is generally existent independent of f.

Before embarking on the proof of this theorem, we would like to provide useful context towards a better understanding of the proof.

We start by noting that this basis is tailored toward a given function. Different functions may result in different bases and a basis that exists for one function may not exist for another function. In order to maximize stability, we suggest choosing the lowest order moment, c_{p_0,q_0}, with a magnitude above the average:

$$|c_{p_0,q_0}| \geq \frac{\sum_{p+q<o} |c_{p,q}|}{\sum_{p+q<o}}. \tag{3.9}$$

The fraction in the exponent of (3.8) corresponds to a root of a complex number, which has $|p_0 - q_0|$ solutions. It is not necessary to store the invariants for all complex roots, but only for a single arbitrary but consistent one. However, during the comparison step with the pattern, we need to take this ambiguity into account

and compare the arbitrary root of the function to each of the multiple roots of the pattern. We do not need to store the multiple roots of the pattern either as we can compute the missing ones if we know just one invariant $\phi(p,q)$ and the chosen p_0, q_0 from (3.9) using the rule

$$\phi(p,q)e^{\frac{2i\pi k}{p_0-q_0}} \tag{3.10}$$

for $k = 1, ..., p_0 - q_0$. Please note though it is crucial that all elements $\phi(p,q)$ of the set of stored invariants were generated using the same complex root. We show in detail why it is necessary to work with this ambiguity in Subsection 3.3.

Proof. This proof consists of four parts.

Invariance. We can see from (3.5) and (3.4) that the elements $\phi(p,q)$ are rotation invariant, because of $1(p-q)+(p_0-q_0)(-(p-q)/(p_0-q_0)) = 0$. The elements $c_{p,p}$ are naturally invariant with respect to arbitrary rotations, because of (3.2).

Completeness. We will solve the inverse problem. The assertion then follows from the fundamental theorem of moment invariants [21]. Analogous to [6], we can pick one orientation to remove the degree of freedom that comes from the rotation invariance. We assume $c_{p_0,q_0} \in \mathbb{R}^+$. Firstly, since $c_{p_0,q_0} \in \mathbb{R}^+$, it coincides with its absolute value, which can be constructed from $\phi(q_0,p_0)$ via

$$\begin{aligned} c_{p_0,q_0} &= |c_{p_0,q_0}| = \sqrt{\overline{c_{p_0,q_0}} c_{p_0,q_0}} = \sqrt{c_{q_0,p_0} c_{p_0,q_0}} \\ &= \sqrt{c_{q_0,p_0} c_{p_0,q_0}^{-\frac{q_0-p_0}{p_0-q_0}}} = \sqrt{\phi(q_0,p_0)} \end{aligned} \tag{3.11}$$

because real valued functions suffice

$$c_{p,q} = \overline{c_{q,p}}. \tag{3.12}$$

Please note that the invariant $\phi(q_0,p_0)$ is part of the basis, because from the restriction on the normalizer $p_0 - q_0 < 0$ follows the restriction for the elements of the basis $p > q$ with $p = q_0, q = p_0$. Secondly, for all $p > q$, the original moment $c_{p,q}$ can be reconstructed from any of the possibly multiple $\phi(p,q)$ using the calculation rule

$$c_{p,q} = \phi(p,q) c_{p_0,q_0}^{\frac{p-q}{p_0-q_0}}. \tag{3.13}$$

Then, for all $p < q$, the original moments can afterwards be reconstructed from $c_{q,p}$ using the relation (3.12). Finally, for $p = q$, the moments are already part of the basis.

Existence. If all moments with $p_0 - q_0 \neq 0$ are zero, the basis reduces to $\{c_{p,p}, p+p \leq o\}$. It is known from [20] that this is a basis for circular symmetric functions[3]. For all other functions, a non-zero non-symmetric moment c_{p_0,q_0} with $p_0 - q_0 \neq 0$ can be chosen. If it should suffice $p_0 - q_0 > 0$, then we automatically know from (3.12), that $c_{q_0,p_0} \neq 0$, too. It satisfies the constraint $q_0 - p_0 < 0$ and the basis exists as defined.

Independence. We use the polar representation $c_{p_0,q_0} = re^{i\phi}$ of the normalizer of a function f to construct the new function

$$f'(z) := r^{\frac{1}{p_0-q_0}} f(e^{\frac{i\phi}{p_0-q_0}} z). \tag{3.14}$$

Using (3.2), we see that moments of f' suffice $c'_{p,q} = c_{p,q} c_{p_0,q_0}^{-(p-q)/(p_0-q_0)}$ and therefore coincide with the basis elements $\phi(p,q)$ of f. Since the moments of f' are independent, so is the basis. If no normalizer c_{p_0,q_0} can be found, the basis consists solely of moments and is therefore independent, too. □

Example 3.6. The flexible basis exists for the function (3.7) from Example 3.4 and Figure 1. In agreement with (3.9) among the moments up to fourth order, we pick $p_0 = 0, q_0 = 2$. Then, the non-zero elements of the basis are

$$\begin{aligned} \phi(2,0) &= c_{2,0} c_{0,2} = \frac{\pi^2}{36}, \\ \phi(3,0) &= c_{3,0} c_{0,2}^{\frac{3}{2}} = \pm \frac{i\pi\sqrt{\pi}^3}{8\sqrt{6}^3}, \\ \phi(3,1) &= c_{3,1} c_{0,2} = \frac{\pi^2}{48}. \end{aligned} \tag{3.15}$$

Pleas note that during the pattern recognition task, the flexible basis that is tailored toward the pattern will be evaluated on the field where the chosen normalizer c_{p_0,q_0} may vanish. The moment invariants always become unstable if the moment c_{p_0,q_0} is close to zero, which leads to very high values in the invariants. But because of 3.9 these areas must be very different from the pattern. So this kind of instability does not influence the result of the pattern matching.

3.3 Multiple Complex Roots

In this subsection, we will show why the proposed treatment of the multiple complex roots is necessary in order to guarantee independence, invariance, completeness, and existence. It may be skipped on first reading.

Invariance. If we restrict the basis from Theorem 3.5 to one representative of the possibly multiple complex roots, the resulting set is no longer invariant with respect to rotation. Without loss of generality, let us choose the root with the lowest non-negative angle to

[3] We call a function circular symmetric or completely rotationally symmetric if its rotated version coincides with the original function independent from the rotation angle α, meaning it suffices $\forall \alpha \in [0, 2\pi) : f(z) = R_\alpha f(z)$. One could say, it is n-fold symmetric with $n = \infty$.

the positive real axis. Then, using function f from (3.7) as in Example 3.6, we would pick $\sqrt{\pi/6}$ as the representative complex root of $c_{0,2} = \pi/6$. The generated set would have the form $\phi(2,0) = c_{2,0}c_{0,2} = \pi^2/36$, $\phi(3,0) = c_{3,0}c_{0,2}^{3/2} = i\pi\sqrt{\pi}^3/8\sqrt{6}^3$, $\phi(3,1) = c_{3,1}c_{0,2} = \pi^2/48$. Let f' be f if we rotate it by π, then the moments of

$$f'(x,y) = (y^3 - 3x^2y + x^2 - y^2)\chi(x^2+y^2 \leq 1) \quad (3.16)$$

are the same as in Example (3.4) except that the ones of odd order in the middle row change their sign. As a result, the chosen representative root of $c_{0,2}$ is still $\sqrt{\pi/6}$, and the new generated set differs from the previous, because $\phi(3,0) = c_{3,0}c_{0,2}^{3/2} = -i\pi\sqrt{\pi}^3/8\sqrt{6}^3$ has the opposite sign.

Completeness. In many applications, the full discriminative power of a complete basis is not necessarily required. In these cases, we can replace $\phi(p,q)$ from Theorem 3.5 by the simpler formula

$$\phi'(p,q) := c_{p,q}^{p_0-q_0} c_{p_0,q_0}^{-(p-q)}. \quad (3.17)$$

The resulting generator $\mathscr{B}$ can be used instead of the basis from Theorem 3.5. It has only one unique element for each p,q because it does not contain complex roots. But note that this set is not generally complete. To prove that, we revisit the function from Example 3.6 with moments calculated up to fourth order. If we use the basis from (3.8), the invariant $c_{3,1}c_{0,2} = \pi^2/48$ is part of the basis and can therefore be constructed from the basis trivially.

However, if we use $\phi'(p,q)$ from (3.17), we get $\phi'(2,0) = c_{2,0}^2 c_{0,2}^2 = \pi^4/6^4$, $\phi'(3,0) = c_{3,0}^2 c_{0,2}^3 = -\pi^5/8^2 6^3$, $\phi'(3,1 = c_{3,1}^2 c_{0,2}^2 = \pi^4/8^2 6^2$, from which $c_{3,1}c_{0,2}$ cannot be constructed. We can only use $\phi'(3,1) = (c_{3,1}c_{0,2})^2$, which does not contain the more detailed information that $c_{3,1}c_{0,2} = \pi^2/48$ was actually positive. As an example, the function

$$g(x,y) = (31(x^2-y^2) - 40(x^4-y^4) - y^3 + 3x^2y)\,\chi(x^2+y^2 \leq 1) \quad (3.18)$$

shown in Figure 2 has the moments $c_{2,0} = \pi/6$, $c_{0,2} = \pi/6$, $c_{3,0} = -i\pi/8$, $c_{0,3} = i\pi/8$, $c_{3,1} = -\pi/8$, $c_{1,3} = -\pi/8$. The basis from Theorem 3.5 shows the difference between g and f, because here $\phi_g(3,1) = c_{3,1}c_{0,2} = -\pi^2/48$ has opposite sign than $\phi_f(3,1) = \pi^2/48$ in (3.15). In contrast to that, the generator defined in (3.17) assumes the exact same values $\phi'_g(3,1) = c_{3,1}^2 c_{0,2}^2 = \pi^4/8^2 6^2 = \phi'_f(3,1)$ for g as for f.

Existence. If we restrict ourselves to moments that have no symmetry with respect to rotation whatsoever, i.e. $p_0 - q_0 = 1$, then we have no complex roots and get one unique solution for each p,q. In this case, the basis reduces to the one suggested by Flusser and it may not exist even for non-symmetric functions as was already seen in Example 3.4.

Independence. Considering the multiplicity of the complex roots does not violate the independence if we interpret them in the following way. The multiple roots of an invariant are not independent invariants themselves, but merely manifestations of the same invariant. We do not have to store them separately, because we can construct all roots from one representative using formula (3.10).

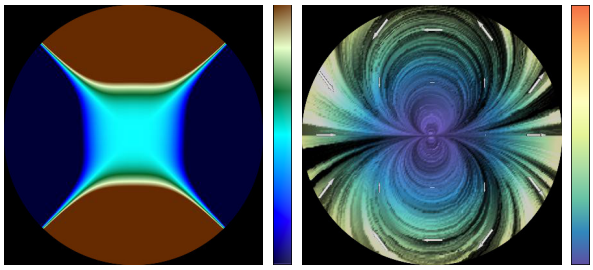


Figure 2: The function $g(x,y)$ from (3.18) visualized using the height color map. The generator (3.17) produces the same invariants as for $f(x,y)$ from Figure 1, even though they are clearly different.

Figure 3: Arrow glyphs and line integral convolution (LIC) [22] of the function (5.10) from Example 5.2. Color and size of the arrows represent the speed. The generator (5.5) does not exist for this pattern.

4 COMPLEX FUNCTIONS

The bases from the previous section were tailored towards real valued functions. Since they satisfy $c_{p,q} = \overline{c_{q,p}}$, it was sufficient to only include $\phi(p,q)$ for $p > q$. Analogous to Theorem 3.5, a flexible basis for arbitrary complex functions that behave under rotations as given in (3.3) can be constructed using the following theorem.

Theorem 4.1. *Let $M = \{c_{p,q}, p+q \leq o\}$ be the set of complex moments of a complex function up to a given order $o \in \mathbb{N}$. If there is a $0 \neq c_{p_0,q_0} \in M$ with $p_0 - q_0 \neq 0$, we define the set $\mathscr{B}$ by $\mathscr{B} := \{\phi(p,q), p+q \leq o\} \setminus \{\phi(p_0,q_0)\} \cup \{|c_{p_0,q_0}|\}$ with*

$$\phi(p,q) := c_{p,q} c_{p_0,q_0}^{-\frac{p-q}{p_0-q_0}}, \quad (4.1)$$

and otherwise by $\mathscr{B} := \{c_{p,p}, p+p \leq o\}$. Then $\mathscr{B}$ is a basis of all rotation invariants of M that exists for any arbitrary complex function.

Proof. The proof works analogously to the proof of Theorem 3.5. □

5 FLOW FIELDS

We can interpret a complex function $f : \mathbb{C} \to \mathbb{C}$ as a two-dimensional vector field by means of the isomorphism between the complex and the Euclidean plane. Analogously to scalar functions, we can make use of the complex moments $c_{p,q}$ as defined in (3.1).

In contrast to the scalar case, flow fields transform by a total rotation. Therefore, we assume that $f'(z) : \mathbb{C} \to \mathbb{C}$ suffices

$$f'(z) = e^{i\alpha} f(e^{-i\alpha} z). \tag{5.1}$$

In this case, the moments $c'_{p,q}$ of f' are related to the moments of f by

$$c'_{p,q} = e^{i\alpha(p-q+1)} c_{p,q}. \tag{5.2}$$

A proof can, for example, be found in [16].

Schlemmer and Heringer [2] showed that analogously to (3.5), any expression of the shape

$$I = \prod_{i=1}^{n} c_{p_i,q_i}^{k_i} \tag{5.3}$$

with $n \in \mathbb{N}$ and for $i = 1,...,n : k_i, p_i, q_i \in \mathbb{N}_0$ is invariant to total rotation, if

$$\sum_{i=1}^{n} k_i (p_i - q_i + 1) = 0, \tag{5.4}$$

because of (5.2).

5.1 Schlemmer's Generator

The first moment invariants for vector fields were suggested by Schlemmer et al. in 2007 [2]. In that paper, instead of presenting a rule for the generation of moment invariants of arbitrary order, a set of five invariants was explicitly stated. Two years later, in his thesis [7], Schlemmer provided the general formula with which invariants of arbitrary order can be produced. The five moments from [2] are exactly the invariants that are produced from this formula if the maximal order of the moments is restricted to two. We therefore assume that Schlemmer at al. used this formula in their 2007 paper [2], although not explicitly stated.

Theorem 5.1. *Cited from [7]. Let M be the set or a subset of all complex moments $c_{p,q}$ of order $(p+q) \in \{0,...,o\}$, $o \geq 2$. Let $\mathscr{I}$ be the set of all moment invariants being constructed according to (3.5) from the elements of M. Let $c_{\dot{p},\dot{q}}$ and $c_{\ddot{p},\ddot{q}} \in M$, with $\dot{p} - \dot{q} = \ddot{q} - \ddot{p} = 2$ and $c_{\dot{p},\dot{q}}$ as well as $c_{\ddot{p},\ddot{q}} \neq 0$ If the set $\mathscr{B}$ is constructed as follows:*

$$\mathscr{B} = \{\phi(p,q) := c_{p,q} c_{\dot{p},\dot{q}}^{a_{p-q}} c_{\ddot{p},\ddot{q}}^{b_{p-q}}, c_{p,q} \in M\}, \tag{5.5}$$

with

$$a_m = \begin{cases} 0, & \text{if } m \geq -1 \\ (|m|+1) \operatorname{div} 3, & \text{if } m \leq -2 \end{cases} \tag{5.6}$$

and

$$b_m = \begin{cases} m+1, & \text{if } m \geq -1 \\ (m+1) \bmod 3, & \text{if } m \leq -2 \end{cases} \tag{5.7}$$

then $\mathscr{B}$ is a basis of $\mathscr{I}$.

This theorem in fact happens to be slightly incorrect. Schlemmer's generator is neither independent nor complete and therefore no basis in the sense of Definition 3.2. We prove why in the two following paragraphs and give two explicit examples. In our opinion, this minor inaccurateness does not lessen the impact of their contribution to the pattern detection and flow visualization communities.

Independence. This generator is not independent, because the invariant $\phi(\dot{p},\dot{q})$ and $\phi(\ddot{p},\ddot{q})$ are identical. We can see that from $\dot{p} - \dot{q} = 2, \ddot{p} - \ddot{q} = -2$, and

$$\begin{aligned} \phi(\dot{p},\dot{q}) &\overset{(5.5)}{=} c_{\dot{p},\dot{q}} c_{\dot{p},\dot{q}}^{a_2} c_{\ddot{p},\ddot{q}}^{b_2} \overset{(5.6),(5.7)}{=} c_{\dot{p},\dot{q}} c_{\dot{p},\dot{q}}^{0} c_{\ddot{p},\ddot{q}}^{3} = c_{\dot{p},\dot{q}} c_{\ddot{p},\ddot{q}}^{3}, \\ \phi(\ddot{p},\ddot{q}) &\overset{(5.5)}{=} c_{\ddot{p},\ddot{q}} c_{\dot{p},\dot{q}}^{a_{-2}} c_{\ddot{p},\ddot{q}}^{b_{-2}} \overset{(5.6),(5.7)}{=} c_{\ddot{p},\ddot{q}} c_{\dot{p},\dot{q}}^{1} c_{\ddot{p},\ddot{q}}^{2} = c_{\dot{p},\dot{q}} c_{\ddot{p},\ddot{q}}^{3}. \end{aligned} \tag{5.8}$$

Completeness. This generator is not complete, because the magnitudes $|c_{\dot{p},\dot{q}}|$ and $|c_{\ddot{p},\ddot{q}}|$ cannot be reconstructed from its elements. That follows from the fact that given the moments $c_{\dot{p},\dot{q}}$ and $c_{\ddot{p},\ddot{q}}$ of a function f, any function f' with $c'_{\dot{p},\dot{q}} = s^3 c_{\dot{p},\dot{q}}$ and $c'_{\ddot{p},\ddot{q}} = c_{\ddot{p},\ddot{q}}/s$ with arbitrary $s \in R^+$ will produce the same $\phi(\dot{p},\dot{q}) = \phi(\ddot{p},\ddot{q})$, because of

$$\phi'(\dot{p},\dot{q}) \overset{(5.8)}{=} c'_{\dot{p},\dot{q}} c'^{3}_{\ddot{p},\ddot{q}} = s^3 c_{\dot{p},\dot{q}} (\frac{1}{s} c_{\ddot{p},\ddot{q}})^3 = \phi(\dot{p},\dot{q}). \tag{5.9}$$

The generator can be transformed into a basis via $\mathscr{B} \setminus \{\phi(\dot{p},\dot{q})\} \cup \{|c_{\dot{p},\dot{q}}|\}$. But even with this correction, the basis is not well-chosen. For one, it is unnecessarily complicated, because it requires evaluation of the two auxiliary functions (5.6) and (5.7) and each element can consist of up to thee factors. Further, it does not exist for functions that do not have non-zero $c_{\dot{p},\dot{q}} \neq 0$ as well as $c_{\ddot{p},\ddot{q}} \neq 0$ with $\dot{p} - \dot{q} = \ddot{q} - \ddot{p} = 2$. This situation is similar to the one in Subsection 3.1. But in this case, even two non-vanishing moments of specific orders need to be present, which increases the number of cases in which the generator does not exist.

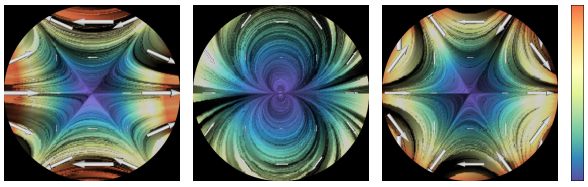
Example 5.2. The vector field given by the function

$$f(z) = z^2 \chi(|z| \leq 1) \tag{5.10}$$

has only one non-zero moment up to third order $c_{0,2} = \pi/3$. It is visualized in Figure 3. Even though it is not symmetric, Schlemmer's generator does not exist, because $c_{\dot{p},\dot{q}} \neq 0$ cannot be found to suffice $\dot{p} - \dot{q} = 2$.

Example 5.3. The vector field given by the function

$$f(z) = (z^2 + 2\bar{z}^2)\chi(|z| \leq 1), \tag{5.11}$$



The function has no symmetry. Its two-fold symmetric part. Its three-fold symmetric part.

Figure 4: Arrow glyphs and LIC of the function (5.11) from Example 5.3 and its components. The color and the size of the arrows represent the speed of the flow.

with χ being the characteristic function, is visualized in Figure 4. It has two non-zero moments up to third order

$$c_{0,2} = \frac{\pi}{3}, \quad c_{2,0} = \frac{2\pi}{3}. \tag{5.12}$$

Here, Schlemmer's generator does exist, because we can choose $c_{\dot{p},\dot{q}} = c_{2,0}$ and $c_{\ddot{p},\ddot{q}} = c_{0,2}$, but it contains only the redundant information

$$\begin{aligned} \phi(0,2) &= c_{0,2}c_{2,0}^{a-2}c_{0,2}^{b-2} = c_{0,2}c_{2,0}^{1}c_{0,2}^{2} = 2(\frac{\pi}{3})^4, \\ \phi(2,0) &= c_{2,0}c_{2,0}^{a_2}c_{0,2}^{b_2} = c_{2,0}c_{2,0}^{0}c_{0,2}^{3} = 2(\frac{\pi}{3})^4, \end{aligned} \tag{5.13}$$

from which we cannot reconstruct the magnitudes of the moments.

5.2 Flusser et al.'s Basis

A straight forward approach to generate a basis of moment invariants for vector fields was suggested by Flusser et al. in [3].

Theorem 5.4. *Let M be the set of moments up to the order $o \in \mathbb{N}$ and $c_{p_0,q_0} \neq 0$ satisfying $p_0 - q_0 = -2$. Further let $\mathscr{I}$ be the set of all rotation invariants created from the moments of M according to (5.3) and $\mathscr{B}$ be constructed by*

$$\forall p,q, p+q \leq o : \phi(p,q) := c_{p,q}c_{p_0,q_0}^{(p-q+1)} \in \mathscr{B}, \tag{5.14}$$

then $\mathscr{B} \setminus \{\phi(p_0,q_0)\} \cup \{|\phi(p_0,q_0)|\}$ is a basis of $\mathscr{I}$.

This produces not only an independent and complete set, but is also more flexible than Schlemmer's generator as it only needs a single specific non-zero moment, not two. Further, it is simpler and more intuitive because it does not need any additional series such as (5.6) and (5.7).

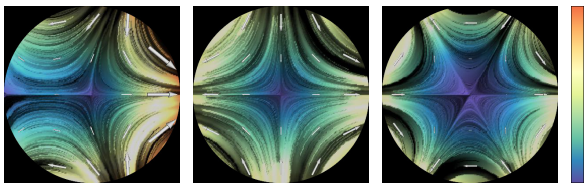
Example 5.5. Flusser's basis exists for the vector field given by the function (5.10) from Example 5.2 and Figure 3. It has one non-zero element $|c_{0,2}| = 2\pi/3$.

Example 5.6. Flusser's basis exists for the vector field given by the function (5.11) from Example 5.3, visualized in Figure 4, and, up to one degree of freedom, the moments can be reconstructed from the basis

$$|c_{0,2}| = \frac{2\pi}{3}, \quad \phi(2,0) = c_{2,0}c_{0,2}^3 = 8(\frac{\pi}{3})^4. \tag{5.15}$$

To show that, we fix the rotational degree of freedom by setting $c_{0,2} \in \mathbb{R}^+$ and get

$$c_{0,2} = |c_{0,2}| = 2\frac{\pi}{3}, \quad c_{2,0} = \phi(2,0)c_{0,2}^{-3} = \frac{\pi}{3}. \tag{5.16}$$



The function has no rotational symmetry. Its linear part with two-fold symmetry. Its quadratic part with three-fold symmetry.

Figure 5: Arrow glyphs and LIC of the function (5.17) from Example 5.7. The color and the size of the arrows represent the speed of the flow.

Example 5.7. The vector field given by the function

$$f(z) = (\bar{z} + \bar{z}^2)\chi(|z| \leq 1) \tag{5.17}$$

has three non-zero moments up to third order

$$c_{1,0} = \frac{\pi}{2}, \quad c_{2,0} = \frac{\pi}{3}, \quad c_{2,1} = \frac{\pi}{4} \tag{5.18}$$

and is visualized in Figure 5. Here, Flusser's basis does not exist because we cannot find any $c_{p_0,q_0} \neq 0$ with $p_0 - q_0 = -2$, even though the function is not symmetric.

5.3 Flexible Basis

Analogous to the scalar case, we can derive a robust basis even for patterns that do not have a numerically significant moment of one-fold symmetry.

Theorem 5.8. *Let $M = \{c_{p,q}, p+q \leq o\}$ be the set of complex moments of a vector field $f : \mathbb{R}^2 \to \mathbb{R}^2$ up to a given order $o \in \mathbb{N}$. If there is a $0 \neq c_{p_0,q_0} \in M$ with $p_0 - q_0 + 1 \neq 0$, we define the set $\mathscr{B}$ by $\mathscr{B} := \{\phi(p,q), p+q \leq o\} \setminus \{\phi(p_0,q_0)\} \cup \{|c_{p_0,q_0}|\}$ with*

$$\phi(p,q) := \phi(p,q) := c_{p,q}c_{p_0,q_0}^{-\frac{p-q+1}{p_0-q_0+1}}, \tag{5.19}$$

and otherwise by $\mathscr{B} := \{c_{p,p+1}, p+p+1 \leq o\}$. Then $\mathscr{B}$ is a basis of all rotation invariants of M, which generally exists independent of f.

Proof. The proof works analogously to the proof of Theorem 3.5. □

Remark 5.9. This last basis of invariants is equivalent to the normalization approach proposed by Bujack et al. [23].

Algorithm 1 Pattern Detection with Flexible Basis.

Input: $N_x \times N_y$ scalar field: f, $B_r(0)$ pattern: g, scales: $\{s_1,..,s_{N_s}\}$, maximum moment order: n,

for $p+q \leq n$ **do**
  moments of pattern: $c^g_{p,q} \overset{(3.1)}{=} \int_{B_r(0)} z^p \bar{z}^q g(z)\,\mathrm{d}z$,
end for
for $o = 0,..,n, p = 0,..,o, q = p,..,o-p$, **do**
  if $|c_{p,q}| \geq \frac{\sum_{p+q<o}|c_{p,q}|}{\sum_{p+q<o}}$ **then**
    choose normalizer (3.9) $p_0 = p, q_0 = q$
    break
  end if
end for
for $p+q \leq n$ **do**
  basis for pattern: $\phi^g(p,q) \overset{(3.8)}{=} c^g_{p,q}(c^g_{p,q})^{-\frac{p-q}{p_0-q_0}}$,
end for
for $x \in N_x \times N_y, s = s_1,..,s_{N_s}$ **do**
  for $p+q \leq n$ **do**
    field mom.: $c^f_{p,q}(x,s) = \int_{B_s(x)} z^p \bar{z}^q f(z)\,\mathrm{d}z$,
  end for
  for $p+q \leq n$ **do**
    basis: $\phi^f(p,q)(x,s) \overset{(3.8)}{=} c^f_{p,q}(x,s)(c^f_{p,q})^{-\frac{p-q}{p_0-q_0}}$,
  end for
  Euclidean distance over $|p_0-q_0|$ roots (3.10): $D(x,s) = \min_{k=1,..,|p_0-q_0|}(\sum_{p+q\leq n}(\phi^f(p,q)(x,s) - \phi^g(p,q)e^{\frac{2i\pi k}{p_0-q_0}})^2)^{\frac{1}{2}}$,
end for

Output: similarity of the pattern p to the field f at position x and scale s: $S(x,s) = D(x,s)^{-1}$.

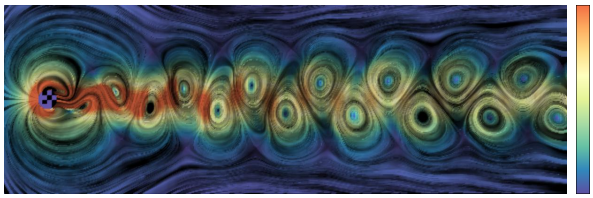
Example 5.10. The flexible basis exists for the vector field (5.17) from Example 5.7, visualized in Figure 5. Any of the three non-zero moments up to third order (5.18) can be chosen as normalizer c_{p_0,q_0}. In order to maximize stability, the proposed algorithm would choose $c_{p_0,q_0} = c_{1,0}$, resulting in two solutions of the complex square root $c_{1,0}^{-1/2} = \pm\sqrt{\frac{\pi}{2}}$ and the basis

$$|c_{1,0}| = \frac{\pi}{2}, \quad \phi(2,0) = \pm\frac{\sqrt{2\pi}}{3}, \quad \phi(2,1) = \frac{1}{2}. \tag{5.20}$$

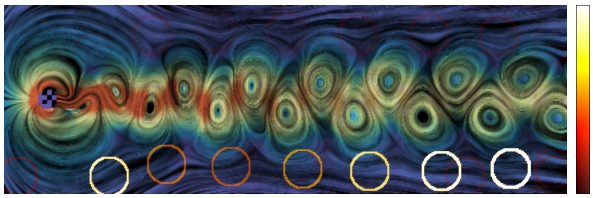
The algorithmic description of the pattern detection for the scalar case can be found in Algorithm 1.

6 EXPERIMENT

We apply the different vector field bases to a pattern detection task in a vector field. The dataset is a computational fluid dynamics simulation of the flow behind a cylinder. The characteristic pattern of the fluid is called the von Kármán vortex street. A visualization of the vortices with removed average flow can be found in Figure 6a. The direction of the flow is visualizaed using line integral convolution [22] and the speed is color coded using the colormap from Figure 7.



The non-flexible bases do not exist for moments up to first order. The algorithm does not produce any output.



The flexible basis does exist with normalizer $c_{1,0}$. The pattern from Figure 7 and its repetitions are correctly detected.

Figure 6: Result of the pattern detetction task using only moments up to first order. The speed of the flow is encoded using the colorbar on the top, the similarity of the field to the pattern using the colorbar on the bottom.

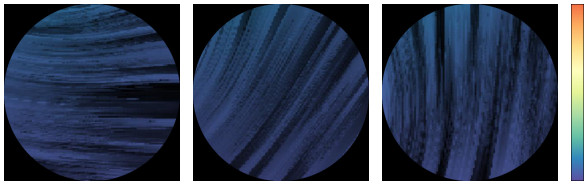
In our experiments, we consider moments up to first order in Figure 6 and moments up to second order in Figure 8. Please note that the basis suggested by Schlemmer [7] from Theorem 5.1 and the one suggested by Flusser [3] from Theorem 3.3 do not exist for moments calculated only up to first order, because a moment c_{p_0,q_0} with $p_0 - q_0 = -2$ cannot be found using only $c_{0,0}, c_{1,0}$, and $c_{0,1}$. For moments up to second order, there is only one potential moment $c_{p_0,q_0} = c_{0,2}$ satifying $p_0 - q_0 = -2$, which is why there is only one basis configuration for these two approaches. They coincide for the moments up to second order, except for the magnitude of the normalizer $|c_{0,2}|$. The remaining moment invariants are

$$c_{0,0}c_{0,2}, \quad c_{0,1}, \quad c_{1,0}c_{0,2}^2, \quad c_{1,1}c_{0,2}, \quad c_{2,0}c_{0,2}^3, \tag{6.1}$$

as already presented in [2].

Then, as long as the normalizer $c_{0,2}$ is numerically non-zero, all three bases will produce stable and identical results up to minor numerical differences. To show the difference between the flexible and non-flexible bases, we therefore use the pattern from Figure 7a, which satisfies $|c_{0,2}| < 0.01$. This pattern was extracted from the dataset itself. Its position in the von Kármán vortex street can be found in the lower, rightmost circle of Figure 6b. Since the only element which differs in the two non-flexible bases is close to zero, the results of the two are almost identical. The differences are numerically small and cannot be perceived by the human eye. To

save space, we plot only the instance that corresponds to Schlemmer's basis. The other is identical.

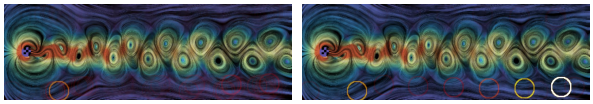


(a) Pattern cut out from the dataset. (b) The pattern rotated by $\pi/3$. (c) The pattern rotated by $\pi/2$.

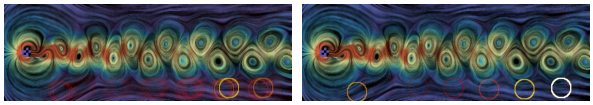
Figure 7: The pattern in different orientations. It was cut out from the dataset at the position of the lower right-most white circle in Figure 6b.

The output of our pattern detection algorithm are circles that indicate the position, the size, and the similarity of the matches. Similarity is encoded in the colormap in the bottom row of Figure 6. The higher the similarity, the brighter the color of the corresponding circle. The color white applies to all matches that have a Euclidean distance of all the moment invariants of less than 0.02. A more detailed description of the algorithm and the visualization can be found in [16].

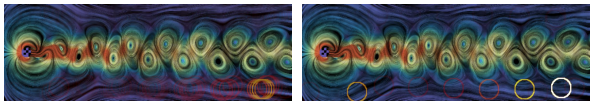
In Figure 6b, we can see that the flexible basis exists even for this pattern and that it correctly finds the pattern's original position. It further detects similar occurrences as it repeats itself in the periodic von Kármán street. As expected, the further we move towards the obstacle, the similarity in each repetition decreases, as indicated by the decreasing brightness of the circles.



non-flexible bases for the pattern oriented as in Figure 7a. Flexible bases for the pattern oriented as in Figure 7a.



non-flexible bases for the pattern oriented as in Figure 7b. Flexible bases for the pattern oriented as in Figure 7b.



non-flexible bases for the pattern oriented as in Figure 7c. Flexible bases for the pattern oriented as in Figure 7c.

Figure 8: Result of the pattern detetction task using moments up to second order. The result of the algorithm using the non-flexible bases is unstable (left). It depends on the orientation of input pattern. In contrast to that, the flexible basis produces consistent results (right).

Figure 8 compares the output of the algorithm using the flexible basis from Theorem 5.8 and the two non-flexible bases for moments up to second order. To show the instability of the non-flexible bases, we used three different instances of the pattern. They differ solely by their orientation. Theoretically, the invariants of all three bases should be invariant with respect to this degree of freedom and produce the same results for all three instances. But as can be seen in the left column of Figure 8, this is not true for the non-flexible bases. Depending on the orientation of the pattern, the similarity of the exact location of the pattern in the field is rather low. Sometimes its position is not the match with the highest similarity, or multiple fuzzy matches occur. On the right side, we can see that the flexible basis produces coherent, stable, and correct results independent from the orientation of the pattern.

7 DISCUSSION

We have reviewed the different bases of moment invariants built from complex monomials using the generator approach and compared their behavior with respect to three important qualities such a basis should suffice: independence, completeness and general existence.

For scalar fields, the basis suggested by Flusser [1] is complete and independent, but it only exists if the pattern has a non-zero moment that is not rotationally symmetric. We have extended his basis to one that always exists, no matter how the values of the moments of a function are distributed.

For vector fields, the first generator approach was suggested by Schlemmer [7]. We show that his set of moment invariants is neither complete nor independent and therefore does not satisfy the properties of a basis. As a result, Flusser et al. [3] were the first to provide a basis of moment invariants for vector fields using the generator approach. As in the scalar case, their basis is complete and independent, but requires a non-zero moment that has no rotational symmetry. We have derived an extension that exists for arbitrary vector fields and found it to coincide with the normalization approach by Bujack et al. [16].

One of the most interesting observations in this work is the equivalence of the optimal generator approach with the optimal normalization approach. Assuming that this fact should also be true for three-dimensional fields, it might be used for the study of 3D moment invariants. The 3D situation is much more complex and neither the generator nor the normalization approach have so far resulted in a set of moment invariants that is complete, independent, and generally existing. Assuming equivalence might guide future research to improve both methods.

8 ACKNOWLEDGEMENTS

We would like to thank Sebastian Volke and the FAnToM development group for the visualization tool, Mario Hlawitschka for the dataset, and Terece Turton for editing assistance. This work is published under LA-UR-17-20144. It was funded by the National Nuclear Security Administration (NNSA) Advanced Simulation and Computing (ASC) Program and by the Czech Science Foundation under Grant GA15-16928S.

9 REFERENCES

[1] Jan Flusser. On the independence of rotation moment invariants. *Pattern Recognition*, 33(9):1405–1410, 2000.

[2] Michael Schlemmer, Manuel Heringer, Florian Morr, Ingrid Hotz, Martin Hering-Bertram, Christoph Garth, Wolfgang Kollmann, Bernd Hamann, and Hans Hagen. Moment Invariants for the Analysis of 2D Flow Fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1743–1750, 2007.

[3] J. Flusser, B. Zitova, and T. Suk. *2D and 3D Image Analysis by Moments*. John Wiley & Sons, 2016.

[4] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.

[5] Michael Reed Teague. Image analysis via the general theory of moments∗. *Journal of the Optical Society of America*, 70(8):920–930, 1980.

[6] Jan Flusser. On the inverse problem of rotation moment invariants. *Pattern Recognition*, 35:3015–3017, 2002.

[7] Michael Schlemmer. *Pattern Recognition for Feature Based and Comparative Visualization*. PhD thesis, Universität Kaiserslautern, Germany, 2011.

[8] Hudai Dirilten and Thomas G Newman. Pattern matching under affine transformations. *Computers, IEEE Transactions on*, 100(3):314–317, 1977.

[9] Tomas Suk and Jan Flusser. Tensor Method for Constructing 3D Moment Invariants. In *Computer Analysis of Images and Patterns*, volume 6855 of *Lecture Notes in Computer Science*, pages 212–219. Springer Berlin, Heidelberg, 2011.

[10] Ziha Pinjo, David Cyganski, and John A Orr. Determination of 3-D object orientation from projections. *Pattern Recognition Letters*, 3(5):351–356, 1985.

[11] C.H. Lo and H.S. Don. 3-D Moment Forms: Their Construction and Application to Object Identification and Positioning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(10):1053–1064, 1989.

[12] Gilles Burel and Hugues Henocq. 3D Invariants and their Application to Object Recognition. *Signal procesing*, 45(1):1–22, 1995.

[13] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors. In *Symposium on Geometry Processing*, 2003.

[14] Nikolaos Canterakis. Complete moment invariants and pose determination for orthogonal transformations of 3D objects. In *Mustererkennung 1996, 18. DAGM Symposium, Informatik aktuell*, pages 339–350. Springer, 1996.

[15] Max Langbein and Hans Hagen. A generalization of moment invariants on 2d vector fields to tensor fields of arbitrary order and dimension. In *International Symposium on Visual Computing*, pages 1151–1160. Springer, 2009.

[16] Roxana Bujack, Ingrid Hotz, Gerik Scheuermann, and Eckhard Hitzer. Moment Invariants for 2D Flow Fields via Normalization in Detail. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 21(8):916–929, Aug 2015.

[17] Roxana Bujack, Jens Kasten, Ingrid Hotz, Gerik Scheuermann, and Eckhard Hitzer. Moment Invariants for 3D Flow Fields via Normalization. In *IEEE PacificVis in Hangzhou, China*, 2015.

[18] Wei Liu and Eraldo Ribeiro. Scale and Rotation Invariant Detection of Singular Patterns in Vector Flow Fields. In *IAPR International Workshop on Structural Syntactic Pattern Recognition (S-SSPR)*, pages 522–531, 2010.

[19] Frits H. Post, Benjamin Vrolijk, Helwig Hauser, Robert S. Laramee, and Helmut Doleisch. The State of the Art in Flow Visualisation: Feature Extraction and Tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

[20] Jan Flusser and Tomas Suk. Rotation Moment Invariants for Recognition of Symmetric Objects. *Image Processing, IEEE Trans.on*, 15(12):3784–3790, 2006.

[21] Thomas H. Reiss. The revised fundamental theorem of moment invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):830–834, 1991.

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

[23] Roxana Bujack, Ingrid Hotz, Gerik Scheuermann, and Eckhard Hitzer. Moment Invariants for 2D Flow Fields Using Normalization. In *IEEE PacificVis in Yokohama, Japan*, pages 41–48, 2014.

Performance Analysis of Corner Detection Algorithms Based on Edge Detectors

Naurin Afrin
Department of Computer Science and Software Engineering, Swinburne University of Technology, Australia
nafrin@swin.edu.au

Wei Lai
Department of Computer Science and Software Engineering, Swinburne University of Technology, Australia
wlai@swin.edu.au

Nabeel Mohammed
Department of Computer Science and Engineering, University of Liberal Arts, Bangladesh
nabeel.mohammed@ulab.edu.bd

ABSTRACT

Detecting corner locations in images plays a significant role in several computer vision applications. Among the different approaches to corner detection, contour-based techniques are specifically interesting as they rely on edges detected from an image, and for such corner detectors, edge detection is the first step. Almost all the contour-based corner detectors proposed in the last few years use the Canny edge detector. There is no comparative study that explores the effect of using different edge detection method on the performance of these corner detectors. This paper fills that gap by carrying out a performance analysis of different contour-based corner detectors when using different edge detectors. We studied four recently developed corner detectors, which are considered as current state of the art and found that the Canny edge detector should not be taken as a default choice and in fact the choice of edge detector can have a profound effect on the corner detection performance. We examined commonly used predefined threshold-based Canny detector with the adaptive Canny detector and found that adaptive Canny detector gives better results to work with.

Keywords

corners, edge detector, Canny, adaptive Canny

1 INTRODUCTION

Corners play an important role in different computer vision applications such as image matching and pattern recognition. Among different types of corner detectors, contour-based corner detectors are more stable and less sensitive to noise [Fmo01, Xia04, Moh07, Xia07, RMN11b]. The primary step of these detectors [Moh08, RMN11a, Moh09, Fmo01, Moh07, Zha10] is to extract the edges that are relevant for corner detection. A few applications like medical imaging requires perfect edge identification which is time-consuming, while different applications like mobile robot vision requires real time vision calculations and do not rely on impeccable edge recognition.

For contour-based corner detection, researchers have been using the Canny edge detector since its popularisation by [Moh08] and this trend has continued without question in [Moh08, RMN11a, Moh09, Fmo01] and others. As a part of our work, we analyse the role of edge detection method on the current state of the art chord-based corner detectors and what role, if any, different edge detectors can play in this process.

We considered the performances of very popular chord-to-point distance accumulation (CPDA) corner detector [Moh08], Chord to Triangular Arms Ratio (CTAR) [RMN11a], Difference of Gaussian(DoG) [Xia09] and Curve to Chord Ratio (CCR) [Ten15]. The DoG detector is not a chord-based detector, but it is presented to compare against a popular non-chord-based corner detector. Previous studies like [Ten15] have performed a comparative study on multiple edge detection techniques i.e. Canny [Can86], Sobel, Roberts, Prewitt [Pre70], LoG [Kam98] and Zerocross [Avl13] from an edge-quality perspective. In this paper we compare these techniques in the context of corner detection, more specifically, we tried to investigate its role on corner detection techniques based on some questions for the diverse nature of the different techniques:

1. Does Canny edge detector give best result in all conditions?

2. If not, then which edge detector performs better for detecting corners under which situations?

3. Which edge detector results in the maximum number of repeatably found corners?

4. Which edge detector 'works best' for which transformation?

5. Which edge detector is fast for which corner detector?

6. Which detector finds and extracts the edges quickly?

We observe that most of the contour-based corner detectors use Canny edge detector with threshold 0.2 and 0.7, which is not suitable to find corners in natural images. Thus, instead of following the trend, we examined the performance using the adaptive Canny edge detector and found that it gives excellent results for extracting edges, which results in detecting more corners.

This paper is organised as follows. Section 2 discusses about some classic edge detection techniques. Section 3 explains the importance of edge detection methods for detecting corners, while section 4 discusses about some current state of the art corner detectors. The performance analysis is presented in section 6. Finally, section 7 concludes the paper.

2 EDGE DETECTION

Generally, Edges refer to the sharp change in image brightness. So, if there is a high difference between two neighbouring pixels, a possible edge is detected. The edge detector determines the transition between these two regions based on grey level discontinuity. Edge detectors can be classified into two classes: First, the classical operators such as Roberts, Prewitt and Sobel operators and then Gaussian operators like Canny. Gaussian operator is used to blur images and remove noise. Both classes of edge detectors apply some simple convolution masks on the entire image in order to compute the first order (Gradient) and/or second order derivatives (Laplacian).

In the following sections we will present a few popular edge operators.

2.1 Sobel Operator

The Sobel operator is a pair of 3×3 convolution kernels as shown in Figure 1. These kernels are orthogonal to each other and is perfect for the edges that existed vertically and horizontally. This two masks are convolved with the image to calculate the gradient magnitude and gradient orientation.

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

Figure 1: Masks used by Sobel Operator.

2.2 Robert Cross operator

The Roberts Cross operator consists of a pair of 2×2 convolution kernels as shown in Figure 2. These kernels respond to edges that existed at 45° to the pixel grid. One kernel is used for each of the two perpendicular orientations.

-1	0	0	-1
0	1	1	0

Figure 2: Masks used for Robert Operator.

2.3 Prewitt operator

Similar to Sobel Operator, Prewitt Operator also uses two 3×3 matrix which are convolved with the original image to find vertical and horizontal edges [Pre70]. This operator calculates the gradient of the image intensity at each point and gradient orientation shows how abruptly the image changes at that point.

-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

Figure 3: Masks used for Prewitt Operator.

2.4 Laplacian of Gaussian (LoG) operator

The Laplacian of Gaussian operatorcalculates the second derivative of an image and does not require the edge direction [Kam98]. Commonly used kernels for LoG operator is shown in Figure 4.

2.5 Canny Operator

The Canny edge detector is one of the most popular methods to find edges by separating noise from input image. [Can86]. The steps of the Canny edge detection algorithm are filtering, hysteresis thresholding,

0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1

Figure 4: Masks used for LoG

edge tracking and non-maximal suppression. It uses Gaussian filter G_σ to smooth the image in order to remove the noise.

$$g(m,n) = G_\sigma(m,n) * f(m,n) \quad (1)$$

where $G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{m^2+n^2}{2\sigma^2})$

We have used Canny operator along with Canny using predefined high and low threshold, 0.7 and 0.2 respectively. We referred Canny with this predefined threshold as Canny_threshold in our experiment. Most of the corner detectors in the literature are using Canny_threshold edge detectors [Moh08, RMN11a, Moh09, Fmo01, Moh07, Zha10].

2.6 Zero cross

The Zerocross Operator finds the location where the Laplacian value goes through zero.The main disadvantage is susceptibility to noise [Avl13].

3 IMPORTANCE OF EDGE DETECTION FOR DETECTING CORNERS

A corner can be defined as the intersection of two edges or, as a point for which there are two dominant and different edge directions in a local neighbourhood. Therefore, corner detection process is closely related to edge detection.

The goal of an edge detection process is to mark the points at which the intensity changes sharply. Different effects, such as change of direction, or poor focus can result in change in the intensity values, resulting in errors such as false edge detection, loss of true edges, poor edge localization, as well as high computational time and problem due to noise.

Edge detectors that depend on Gaussian smoothing, leads to poorer localization of corner position for the rounding effect at corner neighbourhood. Moreover, the non-maximum suppression used in common edge detectors can make the straight lines curved.

Therefore, the choice of an edge detection process has significance of chord-based corner detectors. It may be obvious that the number of detected corners is depends on the number of edges extracted. However, it is not just how many edges are detected, but which edges are detected, that may be more important in the actual application of corner detectors.

4 CORNER DETECTION

A corner is one of the most stable features in a 2D image. In this section we will discuss four recent contour-based corner detectors [Moh08, RMN11a, Xia09]. All such detectors first extract the image edges, which they call contours, and then traverse these edges to search for points at which the curvature values are locally maximum or minimum [Fmo01, Xia04, Moh08]. As almost all methods in this category apply a Gaussian denoising step, the actual curvature value estimation is relatively robust against noise.

4.1 CPDA: Distance accumulation with multiple chord lengths

Chord to Point Distance Accumulation technique (CPDA) was proposed in [Moh08] and is one of the most instructive contour-based corner detectors. The method uses the distance accumulation technique to measure the curvature of every point on an edge.

CPDA detector uses three different chords of length 10, 20 and 30. These chords are moved along each curve. Before calculating the curvature values, each curve is smoothed with an appropriate Gaussian kernel (i.e. σ = 1, 2 or 3) in order to remove quantization noises. The accumulated curvature values for each chord are then normalized.

Next, CPDA finds the candidate corners by rejecting weak corners using local maxima of absolute curvature by comparing the curvature values with threshold T_h, which the authors set to 0.2. Based on the hypothesis that a well defined corner should have a relatively sharp angle [Xia04], CPDA calculates angle from a candidate corner to its two neighbouring candidate corners from the previous step, and compare with the angle threshold δ to remove false corners. The angle-threshold δ is set to 157°.

4.2 CCR: Distance accumulation using distance ratio

CCR first puts a chord along the curve and then calculates the flatness by using the ratio of the length of the curve to the length of the chord. Before that it uses Gaussian smoothing to remove the noise. The number of pixels within the curve segment intersected by the chord is 7 and Gaussian smoothing σ = 3 have been used to detect the corners. The threshold for the corners is defined as Th = 0.986.

4.3 CTAR: Chord to Triangular Arms Ratio

CTAR uses triangular measurement theory to estimate the curvature values. First it places a chord that is moved along the curve and a triangle is formed using the two intersection points between the chord and the

curve, and the middle point within those two intersected points. The ratio between the Euclidean distance of the two intersected points and the summation of the other two arms length of that triangle is computed. The main advantage of CTAR method is that it is not sensitive to noise as it does not use any derivative based measurements.

Like CCR detector, CTAR also used only one chord of length 7 along the curve. After estimating the curvature values, the local minima that are found from each curve estimation, is considered as corner location based on a threshold which is set to 0.989. The angle-threshold δ is set to 163°

4.4 DoG: Difference of Gaussian detector

DoG detector [Xia09] applies multiple levels of Difference of Gaussian (DoG) on a curve to obtain several corresponding planar curves. These planar curves are then convolved with Difference of Gaussian (DoG) filters for detecting the corners. The main advantage of DoG detector is that it uses two scales, a low and a high scale, and then combines them into the detection of the candidate corner so that the coarse-to-fine tracking may be supplanted

5 USING ADAPTIVE CANNY EDGE DETECTOR

The choice of an edge detection process has a great significance in chord-based corner detectors. It may be obvious that the number of detected corners depends on the number of edges extracted. Most of the contour-based corner detection process uses Canny edge detector [Moh08], [RMN11a], [Moh07] for the initial edge extraction step. CPDA corner detector [Moh08] first uses Canny edge detector with thresholds low = 0.2 and high = 0.7 and this trend continues in [RMN11a], [Moh07] and other recent chord-based corner detectors. Instead of following the trend, we analyse the role of Canny edge detection method with both adaptive and pre-defined threshold on the current state of the art chord-based corner detectors. We use adaptive Canny edge detection method that follows the most popular Otsu method to calculate the thresholds which is deduced by least square(LS) method based on gray histogram. We use the adaptive Canny edge detector from the implementation of MATLAB 2012b. The result has been discussed in Section 6.

6 PERFORMANCE STUDY

In this section, we discuss the performance of the edge detectors while applying them to detect corners using the corner detectors. First, the dataset is described. Next, the evaluation method and Finally the results are shown.

6.1 Dataset

We have used an image dataset of 23 different types of grey scale images to evaluate the performance of the corner detectors using different edge detectors. Seven different transformations have been applied to these base 23 images that includes Scaling, Shearing, Rotation, Rotation-Scale, Non-uniform Scale, JPEG Compression, Gaussian Noise to obtain more than 8000 transformed test images (see Table 1) . All the experiments were run on Matlab 2012b on an Windows 7 (64bit) machine with an Intel Core i5-3470 processor and 8GB of RAM.

Transformations	**Transformation factors**	**Number of images**
Scaling	Scale factors s_x=s_y in [0.5,2.0] at 0.1 apart, excluding 1.0	345
Shearing	Shear factors sh_x and sh_y in [0, 0.012]at 0.002 apart.	1081
Rotation	18 different angles of range $-90°$ to $+90°$ at $10°$	437
Rotation-Scale	in [-30 , +30] at $10°$ apart, followed by uniform and non uniform scale factors s_x and s_y in [0.8, 1.2] at 0.1 apart.	4025
Nonuniform Scale	Scale factors s_x in [0.7, 1.3] and s_y in [0.5, 1.5] at 0.1 apart.	1772
JPEG compression	Compression at 20 quality factors in [5, 100] at 5 apart.	460
Gaussian noise	Gaussian (G) noise at 10 variances in [0.005,0.05] at 0.005 apart.	230

Table 1: Image Transformations applied on 23 base images

6.2 Evaluation Method

We have applied automatic corner detection evaluation process proposed by Awarangjeb [Moh08] to examine the number of repeated corners. In this process the detected corner locations of an image are referred to as the reference corners and then compared the locations of the detected corners in the transformed image of the former one with the reference corners. If a reference corner is detected in a corresponding transformed location, then that corner is considered as repeated. The repeatability is the process of detecting the same corner locations in two or more different images of the same scene. The main advantage of this process is that

there is no limit on the number of images in the dataset. Moreover, this process does not require any human intervention.

6.3 Results and Discusssion

We studied the most commonly used Canny [Can86], Sobel, Roberts, Prewitt [Pre70], LoG [Kam98] and Zerocross [Avl13] edge detection methods and conducted our experiment to find out the answers of the questions mentioned earlier.

Figure 5 shows the detected corner locations for only CPDA corner detector after using different edge detectors to an image. It is clearly seen that each edge detector gives different corner locations for the same image because of the discrete edge extraction structures. As Canny, LoG and Zerocross extracts a good number of edges, the number of detected corners are also high. Prewitt, Roberts and Sobel derives less edges, resulting in low numbers of corner locations.

Initially, we have conducted our test to find out the effects of different geometrical transformations for finding edges, corner and repeatable corners. The comparative results of the edge detectors in terms of number of extracted edges, detected corners and repeatable corners under various conditions are presented in Figure 6, 7, 8 respectively.

First, we tried to find out the average number of edges retrieved using different corner detectors with different edge detectors after applying the transformations mentioned earlier. Our first experiment is conducted to notice the effects of different geometrical transformations on the images for detecting edges. Figure 6 shows that the Canny edge detector lefts others behind for detecting edges in almost every conditions. Each edge detectors perform differently in various geometrical changes. Evaluation of the images showed that under several conditions, Canny, LoG, Zerocross, Sobel, Prewitt, Roberts exhibit better performance, respectively.

Numbers of detected corners also depend on the number of extracted edges. However, if an edge detector extracts a good number of loosely connected edges, the detected corners will be few and not suitable for practical application (see Figure 5). We performed our experiment to figure out the average numbers of corners using different edge detectors under several transfromations and from Figure 7, we found that Canny edge detector results best under most of the geometrical transformations for finding corners. This happened for the same reason as Canny finds more edges results in finding more corners. However, Zerocross and LoG operators performs better in scale and shear transformations than Canny edge detector.

We analyzed how the different edge detectors effect the performances of finding repeatable corners under geometrical changes in Figure 8. It is noticeable that though Canny edge detector finds a large number of edges, resulting more corners, it is not best for finding repeatable corners. LoG operator is best followed by Zerocross operator for finding average repeatable corners. Though LoG and Zerocross operators give better result than Canny, it malfunctions at corners and curves. The edges are not connected like Canny, thus it results more edges and corner locations which may not be significant for practical applications.

Corner Detector	Edge Detector	Edge Detection time	Curve extraction time	Corner Detection time	Total time
CPDA	Canny	2.128	37.622	3.728	43.477
	Canny_th	2.09	4.456	1.314	7.861
	Prewitt	0.282	39.126	1.039	40.447
	log	0.76	29.869	2.921	33.551
	Roberts	0.309	46.604	0.586	47.499
	Sobel	0.282	38.601	1.078	39.962
	Zerocross	0.74	29.374	2.837	32.951
CTAR	Canny	2.188	37.574	0.537	40.299
	Canny_th	2.077	4.33	0.168	6.575
	Prewitt	0.302	39.437	0.181	39.92
	log	0.764	30.408	0.423	31.596
	Roberts	0.31	46.865	0.113	47.289
	Sobel	0.302	39.078	0.179	39.559
	Zerocross	0.759	29.506	0.413	30.678
CCR	Canny	2.587	46.644	0.632	49.864
	Canny_th	2.414	5.098	0.194	7.705
	Prewitt	0.305	39.706	0.172	40.183
	log	0.926	37.597	0.568	39.091
	Roberts	0.399	59.435	0.139	59.974
	Sobel	0.309	40.901	0.176	41.386
	Zerocross	0.768	30.337	0.428	31.533
DoG	Canny	0.059	0.55	0.082	0.692
	Canny_th	0.057	0.048	0.024	0.128
	Prewitt	0.006	0.309	0.027	0.342
	log	0.024	0.374	0.053	0.451
	Roberts	0.007	0.398	0.028	0.432
	Sobel	0.006	0.301	0.029	0.336
	Zerocross	0.023	0.368	0.053	0.443

Table 2: Time computation for different detectors (in seconds)

To find which detector is more efficient, we have examined the execution time for each of the four corner detectors using different edge detectors showed in table 2. We have found that Prewitt and Sobel detectors are fast compared to others to detect edges and Robert operator is quicker than others for curve extractions. However, Canny edge detector using thresholds is best for finding corners followed by Zerocross and log operator.

Now from figure 9, we found that Canny edge detector using adaptive threshold extracts more edges, results in finding a good number of corners, instead of using pre-defined threshold values. We evaluate the performance of these two edge detectors after applying seven different transformations and from figure 10 we find that Adaptive Canny edge detector performs better than Canny using pre-defined threshold in terms of the number of edge extractions and finding corners and repeated corners. So we use adaptive Canny edge detection method in the primary edge extraction step before detecting corners.

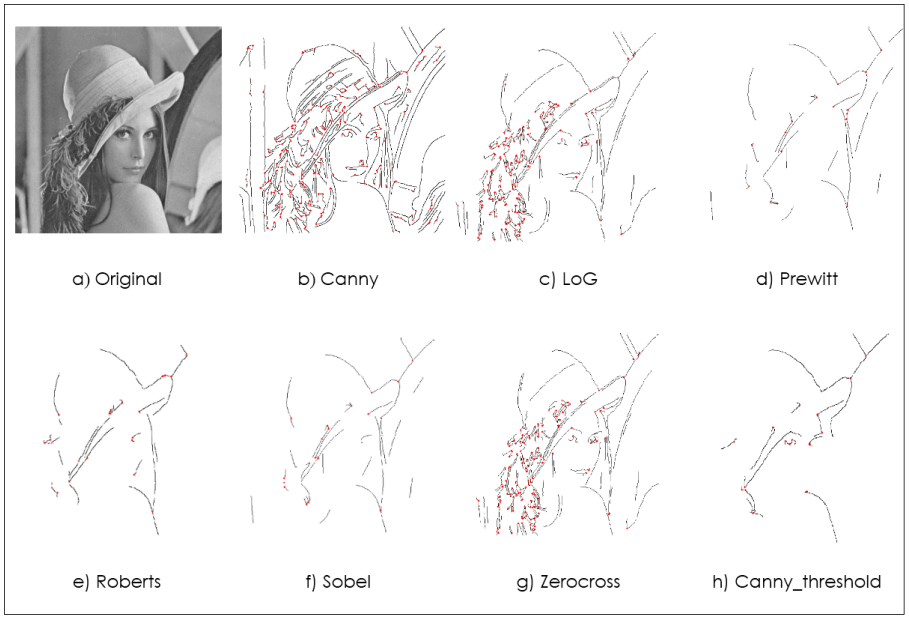


Figure 5: Corner detected by CPDA method using different edge operators

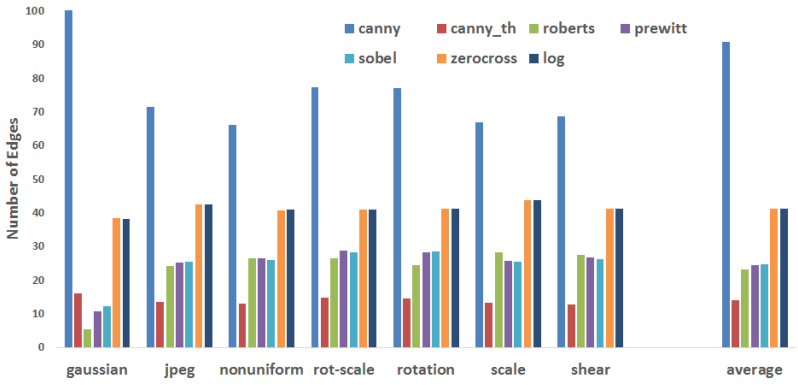


Figure 6: Number of extracted edges after applying different transformations

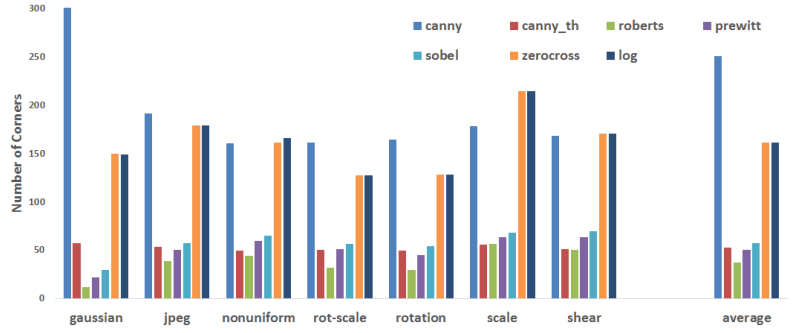


Figure 7: Number of corners after applying different transformations

7 CONCLUSION

We have analysed the performance of different edge operators on different contour-based corner detectors and investigate the performance under different transformations. Since edge detection is the early step in of contour-based corner detection, it is significant to know the performance of different edge detection techniques. In this research paper, the relative performance of various edge detection techniques is carried out with four contour-based corner detectors. It has been observed

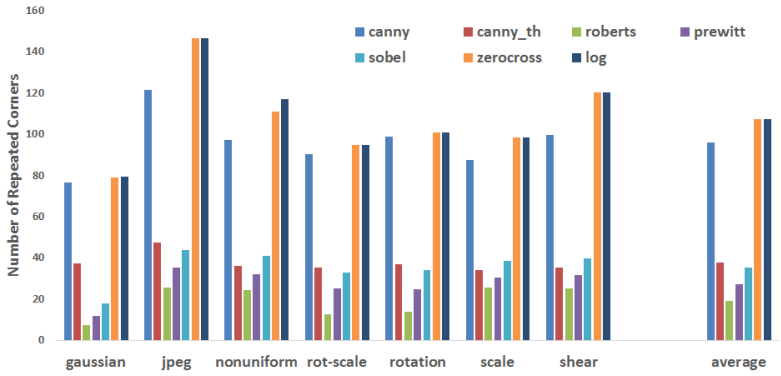


Figure 8: Number of repeated corners after applying different transformations

(a) Original (b)Adaptive Canny (c) Canny (low = 0.2 , high = 0.7)

Figure 9: Extracted edges and detected corners using Canny adaptive and Canny (0.2-0.7)

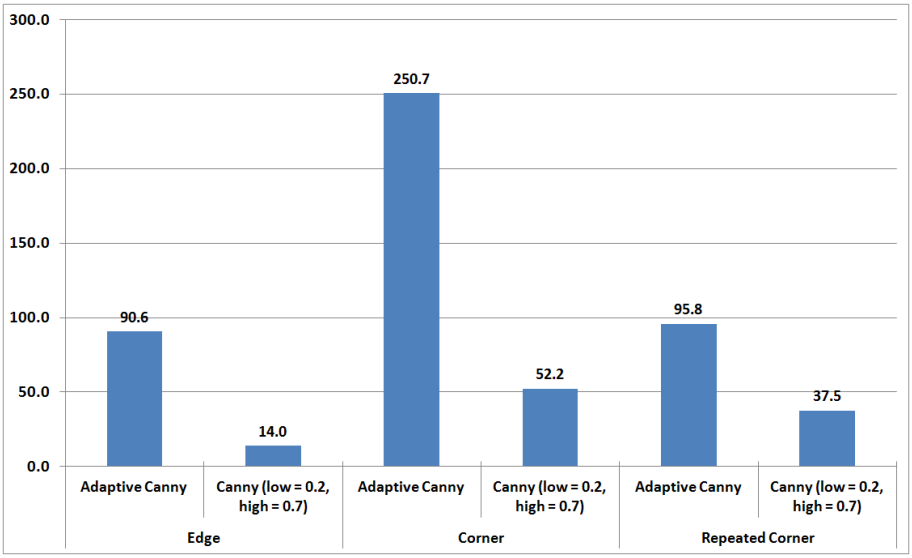


Figure 10: Performance comparison of Canny adaptive and Canny(0.2-0.7)

that Canny edge detection algorithm results higher accuracy in detection of edges and corners, but it is not best for finding repeatable corners, which is considered as one of the most important criterion to evaluate the performance of corner detection. Instead, LoG operator gives best results. In terms of efficiency, Prewitt, Roberts and Sobel operators are fast compared to others to detect edges. Therefore, we can choose different edge detectors, rather than choosing Canny edge detector as an ideal for each scenario. More importantly, we

observed the limitations of commonly used Canny edge detector using predefined threshold and applied adaptive Canny detector instead, which shows better results.

8 REFERENCES

[Dav80] David Marr, Ellen Hildreth. *Theory of edge detection*, Proceedings of the Royal Society of London. Series B. Biological Sciences, 207(1167):187–217, 1980.

[Dmi03] Chetverikov, Dmitry. *A simple and efficient algorithm for detection of high curvature points in planar curves*, International Conference on Computer Analysis of Images and Patterns. Springer Berlin Heidelberg, 2003.

[Dum99] Du-Ming Tsai, H-T Hou, H-J Su. *Boundary-based corner detection using eigenvalues of covariance matrices*,Pattern Recognition Letters, 1999.

[Fan88] Fang-Hsuan Cheng, Wen-Hsing Hsu. *Parallel algorithm for corner finding on digital curves*, Pattern Recognition Letters, 8(1):47–53, 1988.

[Fmo01] F Mokhtarian, F Mohanna. *Enhancing the curvature scale space corner detector*, pages O–M4B, 2001.

[Joh86] John Canny. *A computational approach to edge detection*,Pattern Analysis and Machine Intelligence, IEEE Transactions,(6):679–698, 1986.

[Moh07] Mohammad Awrangjeb, Guojun Lu. *A robust corner matching technique*, Multimedia and Expo,2007 IEEE International Conference on, pages 1483–1486. IEEE, 2007.

[Moh08] Mohammad Awrangjeb, Guojun Lu. *Robust image corner detection based on the chord-to-point distance accumulation technique*, Multimedia, IEEE Transactions, 10(6):1059–1072, 2008.

[Moh10] Mohammad Awrangjeb, Guojun Lu, Clive S Fraser. *A comparative study on contour-based corner detectors*, Digital Image Computing: Techniques and Applications (DICTA), 2010.

[Moh12] Mohammad Awrangjeb, Guojun Lu, Clive S Fraser. *Performance comparisons of contour-based corner detectors*, Image Processing, IEEE Transactions, 21(9):4167–4179, 2012.

[Pen13] Peng-Lang Shui, Wei-Chuan Zhang. *Corner detection and classification using anisotropic directional derivative representations*, Image Processing, IEEE Transactions, 22(8):3204–3218, 2013.

[RMN11a] RM Najmus Sadat, Shyh Wei Teng, Guojun Lu. *An effective and efficient contour-based corner detector using simple triangular theory*, Pacific Graphics Short Papers, The Eurographics Association, 2011.

[RMN11b] RMN Sadat, Zinat Sayeeda, MM Salehin, Naurin Afrin. *A corner detection method using angle accumulation*, Computer and Information Technology (ICCIT), 2011 14th International Conference, pages 95–99. IEEE, 2011.

[Xia04] Xiao Chen He, Nelson HC Yung. *Curvature scale space corner detector with adaptive threshold and dynamic region of support*,Pattern Recognition,Proceedings of the 17th International Conference, IEEE, 2004.

[Xia07] Xiaohong Zhang, Ming Lei, Dan Yang, Yuzhu Wang, Litao Ma. *Multi-scale curvature product for robust image corner detection in curvature scale space*, Pattern Recognition Letter, 28(5):545–554, 2007.

[Xia09] Xiaohong Zhang, Honxing Wang, Mingjian Hong, Ling Xu, Dan Yang, Brian C Lovell. *Robust image corner detection based on scale evolution difference of planar curves*, Pattern Recognition Letters, 2009.

[Zha10] X Zhang, H Wang, A.W.B Smith, X Ling, B.C Lovell, D Yang. *Corner detection based on gradient correlation matrices of planar curve*, Pattern Recognition 43, April 2010.

[Moh09] M Awrangjeb, Guojun Lu, C.S. Fraser, M.Ravanbakhsh. *A fast corner detector based on the chord-to-point distance accumulation technique*, in Proceedings of the 2009 Digital Image Computing: Techniques and Applications, ser. DICTA, Washington, DC, USA: IEEE Computer Society, 2009, pp. 519–525.

[Ten15] Teng S, Sadat R, Lu G. *Effective and efficient contour-based corner detectors*, Pattern Recognition, vol. 48, no. 7, pp. 2185 – 2197, 2015. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0031320315000357`

[Can86] J. Canny. *A computational approach to edge detection*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. PAMI-8, no. 6, pp. 679 –698, nov. 1986.

[Pre70] J. M. Prewitt. *Object enhancement and extraction*, Picture processing and Psychopictorics, vol. 10, no. 1, pp. 15–19, 1970.

[Kam98] B. Kamgar-Parsi, A. Rosenfeld. *Optimally isotropic laplacian operator*, IEEE transactions on image processing: a publication of the IEEE Signal Processing Society, vol. 8, no. 10, pp. 1467–1472, 1998.

[Avl13] M. Avlash, D. L. Kaur. *Performances analysis of different edge detection methods on road images*, International Journal of Advanced Research in Engineering and Applied Sciences, vol. 2, no. 6, 2013.

A Framework for Visually Realistic Multi-robot Simulation in Natural Environment

Ori Ganoni
Department of Computer Science
University of Canterbury
Christchurch, New Zealand
ori.ganoni@pg.canterbury.ac.nz

Ramakrishnan Mukundan
Department of Computer Science
University of Canterbury
Christchurch, New Zealand
mukundan@canterbury.ac.nz

ABSTRACT

This paper presents a generalized framework for the simulation of multiple robots and drones in highly realistic models of natural environments. The proposed simulation architecture uses the Unreal Engine4 for generating both optical and depth sensor outputs from any position and orientation within the environment and provides several key domain specific simulation capabilities. Various components and functionalities of the system have been discussed in detail. The simulation engine also allows users to test and validate a wide range of computer vision algorithms involving different drone configurations under many types of environmental effects such as wind gusts. The paper demonstrates the effectiveness of the system by giving experimental results for a test scenario where one drone tracks the simulated motion of another in a complex natural environment.

Keywords

Robot simulation, Drone simulation, Natural environment models, Natural feature tracking, Unreal Engine 4 (UE4).

1 INTRODUCTION

Graphical models of realistic natural environments are extensively used in games, notably simulation games and those that use immersive environments. These virtual environments provide a high degree of interactive experience and realism in simulations. Modern game engines provide tools for prototyping realistic, complex and detailed virtual environments. Recently, this capability of game engines has been harnessed to the advantage of computer vision community to develop frameworks that can be used in scientific applications where vision based algorithms for detection, tracking and navigation could be effectively tested and evaluated with various types of sensor inputs and environmental conditions. This paper focuses on the development of a comprehensive standalone framework for multi-robot simulation (specifically, multi-drone simulation) in complex natural environments, and proposes suitable configurations of tools, simulation architectures and also looks at key performance issues.

Several robot simulation engines exist which simulate different robots and vehicles e.g. multicopters, rovers, fixed wing UAV, etc. Each engine has its advantages. The engines use large simulation environments consisting of models, sceneries, etc. generated by other simulation packages and frameworks. Following are some examples of such engines with dependencies on other simulation packages:

- Standalone robot simulation engines using a flight simulator for models, sceneries and functions for visualization and simulation. Examples of such packages are: (i) ArduPilot [1] which communicates with Xplane [12] and Flightgear [4] (ii) PX4 [8] communicates with jmavsim [7]. Flight simulators are usually much larger projects than robot simulation projects. They are more focused on user experience and interaction, but they also have visualization and dynamic simulation capabilities which are useful characteristics for drone projects.
- Standalone simulation packages that use physics engines, graphical interfaces and simulation capabilities provided by other simulation tools: for example PX4 [8] with Gazebo [5]. Robot simulation environments are dedicated simulation environments. They are focused on giving proper tools for modeling and simulating robots but are less focused on visualization.
- Stand alone robot simulation environment: those environments include the robots and flying vehi-

cle models. An example of that kind of environment is: Modular Open Robots Simulation Engine (MORSE). Those environments are suitable for testing and evaluating ideas, but they don't have roots in real robot projects specifically in drone projects.

- Game engine stand alone environment: the robot is simulated inside a game engine. For example a benchmark for tracking based on UE4 [22]. Similarly to robot simulation environment, the drones inside game engines don't have roots in real drone projects. Additionally drone simulated in game engines don't share the dynamics of real drones. For example, they don't have to deal with wind gusts and vibrations.

In this paper, we propose a novel configuration that use game engines for the simulation environment, the primary motivation being the enhanced capabilities of a game engine such as UE4 in providing highly realistic environments and various modes of visualization. One of the primary advantages of this type of a configuration is that a game engine such as UE4 can provide realtime videos of camera output based on the position and attitude information of the robot. This paper also gives an overview of the DroneSimLab [3] developed by us, which has constantly evolved with the analysis of various requirements and concepts related to the simulation architecture presented later in this work. The design and implementation aspects of the key components of this simulation engine have been presented in detail.

This paper is organized as follows: in section 2 we give an overview of the DroneSimLab project. In section 3 we describe previous related work with game engines. Section 4 gives detailed information about the framework simulation architecture and design goals. Section 5 focuses on modifications needed to be made to meet the simulation design goals. Section 6 describes experimental results and performance. Section 7 provide information on future research directions as well as references to online demos of this research.

2 THE DRONESIMLAB PROJECT

We developed DroneSimLab as an opensource project to foster collaborative development of drone simulation packages that use the power and capabilities of the UE4 as discussed in the previous sections. Some of the main functionalities which the current implementation provides are:

- Multi-robot - can handle more than one robot and create visual interaction.
- Software In The Loop (SITL) driven - can simulate two drone models: ArduPilot and PX4
- Based on Game Engine - Uses UE4 as an optical and depth sensor
- Realtime - depends on the hardware but can run at 30 fps.
- Natural environments - can simulate trees, wind grass, etc. (comes with Game Engines assets).

3 RELATED WORKS

Some image processing and image-based algorithms have already been integrated into game engine/image simulation environments, although using game engines dedicated for games (like UE4, CryEngine, and Unity) in simulations is much less common. We believe the reason for that is due that they are more focused on game experience as opposed to any real-world scientific applications involving simulations with associated mathematical and physical models and computer vision algorithms.

One example of such an approach is the autonomous landing of a Vertical Takeoff and Landing (VTOL) Unmanned Aerial Vehicle (UAV) on a moving platform using image based visual servoing [25], where they used gazebo simulation simultaneous localization and mapping. Simultaneous Localization and Mapping (SLAM) [13] is also tested and developed for indoor scenarios using gazebo simulation [5] [20]. Some environments are combined to create a more powerful engine. For example, MORSE [17] combined with BGE (blender game engine) [16] and JSBsim (an open source flight dynamics model)

Lately, game engines have been increasingly used for simulations. Successful attempts have been made to evaluate the stability of structure using UE4, creating photo-realistic scenes of stacks of blocks and applying deep learning methods [19]. A series of towers made from wooden cubes were created in a simulated environment using UE4 [18]. Some of the towers were stable structures, and some collapsed when the dynamic simulation was run. A network was trained to detect the outcome of the experiments. Testing the network on real environments achieved equal performance compared to human subjects in predicting whether the tower will fall. The most important aspect of this research is the fact that they could train the network on 180,000 scenarios which seems not feasible in a real life environment.

A more recent work connected UE4 with OpenCV [15], the project is called UnrealCV [23]. It extends the UE4 with a set of commands to interact with the virtual world. Another work [22] proposed a new aerial video dataset and benchmark for low altitude UAV target tracking, as well as a photorealistic UAV simulator that can be coupled with tracking methods. Skinner

[27] proposed a high-fidelity simulation for evaluating robotic vision performance for repeating robotic vision experiments under identical conditions. Similarly, we are providing a sandbox for high-fidelity simulations not only for algorithms but also for full SITL simulations.

Recently Microsoft released AirSim [26], an open source simulator based on Unreal Engine for autonomous vehicles from Microsoft AI & Research which has a similar architecture as our proposed architecture. In their released implementation they are using their physics engine and control libraries.

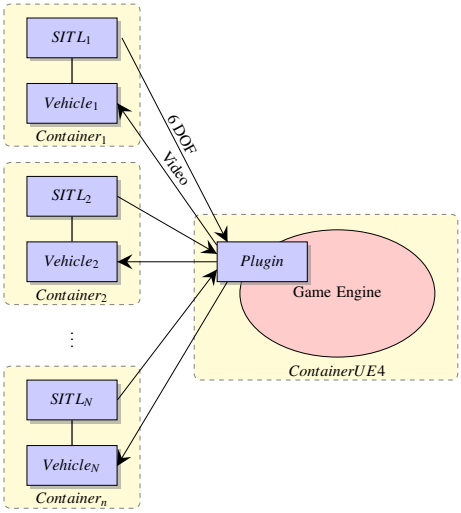


Figure 1: Simulation architecture

4 SIMULATION ARCHITECTURE

In this paper, we propose a simulation architecture designed to meet the following primary goals: (i) ability to generate realtime camera outputs for any arbitrary position and orientation in a natural environment, (ii) ability to integrate software and hardware in the loop simulations (iii) ability to combine multiple simulations and (iv) ability to reproduce results. These aspects are elaborated below.

4.1 Domain Specific Simulation Engine

We focused on three simulation engines for the framework.

- The game engine provides video, depth data and additional visual environmental effects like wind and dust.
- The physical model engine, usually supplied by the robot development framework.
- Supplimental simulation objects like communication channels models , computation power restrictions and additional simulation filters (for example a lens distortion filter).

Engines can create an environment for a single robot. For instance in the case of SITL, the simulation engine interacts with only one vehicle and produces sensory information for only one robot. On the other hand, the game engine can provide visual information for multiple robots as described in Figure 1, this is especially necessary if a visual interaction exists, for example; one robot can block the field of view for another robot, and this aspect should be implemented in the simulation. By embedding SITL into our framework we ensure that the simulation is highly correlated with the real robot architecture (since it is used for the robot development). Hardware In The Loop (HIL) can be later used for further validation.

4.2 Simulated sensor architecture

We identified three types of simulated sensors that can be used.

- Single domain sensor - lives only in one engine. For example a simulated RGB camera from UR4 [18]. Another example is the gyro sensor, which is simulated only in the SITL software e.g. gazebo, jmavsim, jsbsim etc.
- Multi domain sensor - lives in more then one engine. For example such a sensor can be seen in Figure 2. In this example the simulated distance sensor gets information in from various sources like an external Digital Elevation Model (DEM).
- Complex sensor - lives in both the physical domain and in the simulated domain. An example of such sensor is a camera in front of a screen. The display provides the visual information and the camera is used just as in the real system, enabling monitoring real system performance and hardware issues. This concept is an extension of the HIL mode which combines hardware testing and software testing.

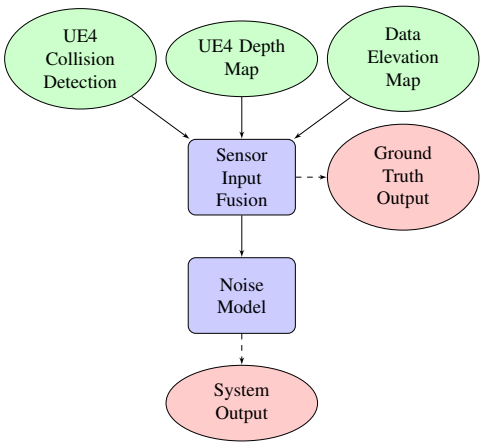


Figure 2: Example of a multi-domain distance sensor, The simulated sensor accepts inputs from different engine resources and produces ground truth information and noisy output.

4.3 Containers as vehicles

We used a container approach to combine simulations not originally intended to work alongside each other. The container approach enables us to use different operating systems and libraries on the same machine. We can also control the network configuration in each container. This approach is different from other frameworks like AirSim [26] by maintaining the original firmware developed by the Robot Simulation Framework. Our architecture can utilize the benefits of new features and continuing development of those environments.

4.4 Reproducibility

The ability to reproduce results under differing or constant conditions is vital in system development as well as in research, and becomes more and more difficult as the complexity of the system increases. To realize this concept, we are using several existing software tools. We are using the Docker engine to manage system configuration, and Git version control to handle the software development. Since the experiments are in a simulation, other reproducibility aspects such as high-fidelity are built-in. In section 6, we demonstrate testing of algorithms in complex natural environments by controlling simulation parameters. In this simulation, we can see that outdoor natural environment can be problematic for testing visual algorithms since we don't have full control of the environment. It seems that true reproducibility in an outdoor natural environment may be achieved only in simulation [27].

4.5 Build system & configuration management

The simulation environment uses these software tools:

- Version Control - All files of this project are managed by Git version control under GitHub servers [6]. The only exceptions are the UE4 projects which are managed locally due to the large file sizes. The UE4 [18] source code is still managed by git in dedicated GitHub repository. For the purpose of sending realtime ground truth position, changes have been made both to ArduPilot Project and to PX4 and are managed in separate forks. Those changes are not compatible with the design and purpose of the original projects. Changes that were compatible (e.g. a turbulence model) were returned to the community as pull requests and then pulled back into our local fork.
- Containers - Created with Docker engine.
- ArduPilot Fork [1] (Drone Project)
- ROS - Supporting firmware for the PX4 project.
- PX4 Fork [8] (Drone Project)
- UE4PyServer plugin [10]

5 ENGINE MODIFICATIONS

5.1 UE4 Plugin

Game engines are not dedicated research tools, obviously, but conveniently for our usage scenario they supply mechanisms like plugins to extend the capabilities of the engine. The plugin we used for the UE4 [18] is called UE4PyServer [10] Plugin and was developed for the purpose of this research. The main concepts behind the plugin development were:

- Realtime: For this simulation, we took advantage of the realtime capabilities of the game engine. Realtime simulation (RT) is important when you want to run many tests in a short period. RT simulation is also necessary when human interaction is involved because users expect realtime or near realtime behavior. To maintain RT behavior, the UE4 plugin was developed with minimal processioning on the UE4 side. The primary purpose of the plugin is to communicate with other parts of the simulation. e.g. receiving 6 DOF information and sending video data.
- Multi-Robot support - UE4 enables capture of the viewable screen to a file or a buffer, but this provides us with only one camera feed. To allow multiple cameras in the simulation, we used rendering-to-texture technique with object ScreenCapture2D [9]. The method is used in the game engine usually to render surfaces like security cameras, billboards, mirrors, etc. We used it to simulate a camera robot and depth sensors using the depth map provided by the ScreenCapture2D Object.
- Synchronization - We wanted the sampling to be synchronized for all the visual objects in the simulation. It is an important concept and might be critical for some applications, for example, simulating stereo camera. For this purpose, we used coroutines which are a light version of synchronized pseudo-threads.

5.2 Building realistic environment inside game engine for computer vision

There are some special considerations when building virtual environments in game engines for computer vision purposes.

- Level of Details (LOD) - in game engines using multiple LOD [21, Chapter 3] in order to maintain graphics performance especially frame rate. This may create unnatural textures changes which can be

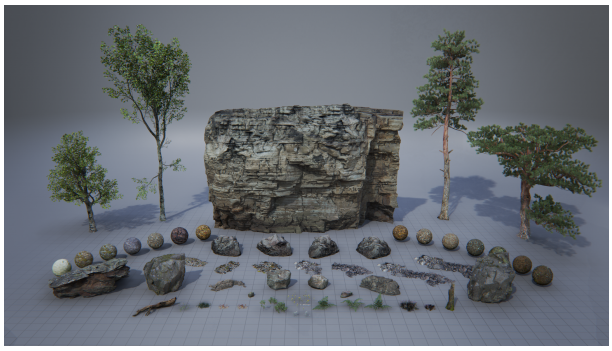


Figure 3: Open world Overview - these are the assets used to build a forest environment

destructive for computer vision algorithms. For the purpose of this research, we can control the environment and the simulation, and we can use that to create a scene with only one LOD.

- Repeating patterns - In Figure 3 we can see the meshes used to build the realistic scene. To reduce the effect of repeating patterns, each element is positioned in a different orientation and slightly different scaling. Also, the elements are positioned with some overlap with other objects which reduces the repeating effect.

- Culling adjustments - the area rendered in the scene also known as frustum should be large enough for all the objects in the scene to be rendered, so we will avoid popping effects due to movements of the cameras or the objects themselves.

- Dynamic shadows adjustments - moving objects in the scene like trees and robots should always cast dynamic shadows to imitate real scenarios.

5.3 SITL

The SITL engine needs to send 6 DOF information at a high rate to the game engine (at least 30 fps) to maintain realtime constraints. For that purpose, some modifications are needed to the engine, so the SITL engine will send ground truth information directly to the game engine, and also to logging mechanisms for later analysis as described in Figure 1.

6 EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

6.1 Plugin tests in natural environment

Running visual algorithms in a natural environment can be very challenging. Relative to artificial environments, natural scenes can by highly dynamic due to atmospheric conditions such as wind, and usually will not have distinct characteristics like straight lines, circles

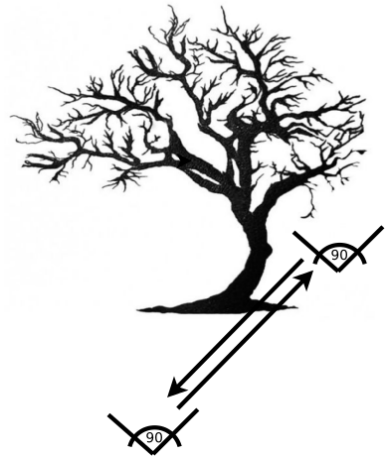


Figure 4: Scene architecture - In the UE4 Editor, we placed a camera at an initial position, in front of a tree. We moved the camera diagonally away from the tree, and then returned to the same point. Camera maneuvers which starts and ends in the same position are ideal for tracking tests. Ideally, the tracked points should get back to the same original coordinates.

corners, etc. Using UE4PyServer [10] (which was developed as part of the simulation framework) and UE4 [18] we developed a tracking simulation (live video can be found here [11]) to demonstrate the uniqueness of natural environment. The simulation is based on the Lucas-Kanade Optical Flow tracker implemented in the OpenCV library [14] which we use it to track an ordered grid of points (no feature extraction). The maneuver is a simple camera facing forward and moving diagonally back and then return to the original position as described in Figure 4. Ideally, we would expect that the tracked points will return to the same coordinates when the simulation cycles back to the starting frame. Since this is a complex 3D scene, not all the points will return to the same location due to the loss of tracking, but in Figure 5 we can see that running the experiment twice produces similar results. Similar but not exact, since there is still some randomness in the scene due to movement of leaves that might cause slight differences. In Figure 6 we conducted two experiments with the same setup, but in the second test, we add the wind to the scene by adding to the UE4 a Wind Direction Force object. We can see that the results are now very different. We repeated the experiment under various conditions and calculated the following MSE grade to quantify the tracking quality:

$$G = \frac{1}{N} \sum_{P \in T_p} |P_e - P_s|^2 \quad (1)$$

where: G is the tracking error, T_p is the tracked points, P_s an P_e are the start and end coordinates of the tracked

points and N is the number of tracked points. Summarized results are in the following table:

	low-wind	high-wind
low-alt	96.36 (98.54%)	219.99 (97.40%)
	81.15 (98.54%)	225.48 (98.05%)
	87.69 (98.54%)	234.41 (97.89%)
high-alt	55.72 (97.56%)	243.59 (98.54%)
	53.82 (98.21%)	757.61 (97.73%)
	48.94 (97.40%)	382.78 (98.38%)

Table 1: Tracking error (MSE) values in $pixels^2$. The numbers in brackets give the percentage of correctly tracked points.

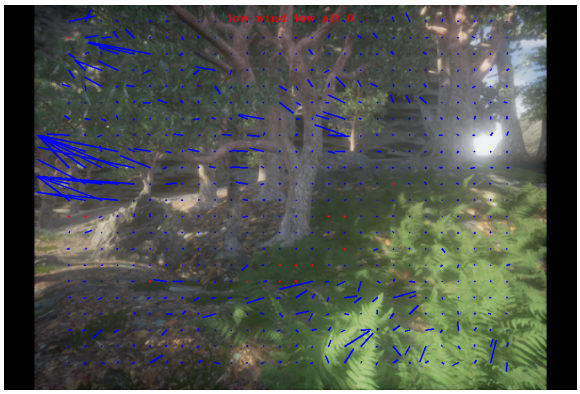
In Table 1, we can see the behavior of the tracking algorithm under different environmental conditions. As expected in high altitude (near the tree tops) with the combination of strong wind will be the most challenging scenario. As seen in the first column, the tracking error under low wind conditions is larger at low altitudes compared to high altitudes due to the presence of a higher density of objects such as leaves and branches that occlude the camera view at low altitudes. On the other hand, when the wind speed increases, the trend is reversed because leaves and branches tend to move more than the objects closer to the ground like tree trunk rocks, etc.

We developed a tool for profiling and monitoring the framework in addition to the existing tools in the UE4 editor. This is a high-level profiling tool that gives us the summary of the system utilization. An example of a test case is presented in Figures 4 and 7. fig 4 explains the scene architecture and Figure 7 the corresponding profiling graph. The peaks in the GPU utilization are due to camera IO-intensive movements as would be expected. In this example, the GPU peaks in the graphs correspond to camera movement. When the camera is moving, we can see that the GPU is fully utilized (it reaches 100%) resulting in reduced framerate and when the frame rate is reduced the CPU was less occupied because it was processing the images at a lower frame rate.

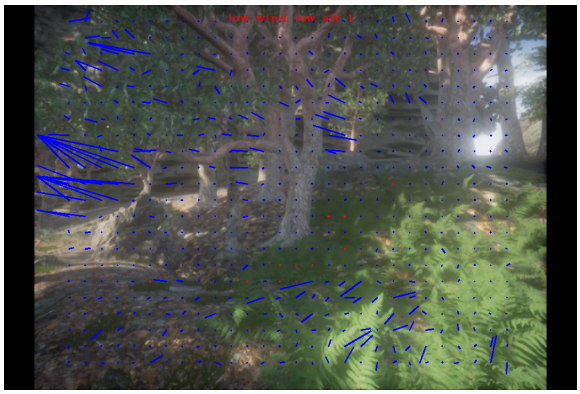
6.2 DroneSimLab tests

We created a setup in DroneSimLab for an experiment of one drone tracking another drone. The drones are Ardupilot drones simulated using their internal SITL engine. We simulated the wind in the UE4 as well as in the SITL engine including wind gusts. The two drones fly into the forest and then return to the original position [2]. One of the drones is using HSV tracker [24] to track the other drone (Figure 9). We repeated this experiment four times and the results are presented in Figure 8.

In all four scenarios, we can observe the loss of tracking capabilities when the drones enter the forest (black dots between frames 300 to 500 in all four scenarios). When the drones enter the woods, the shades from the forest canopy affects the color and brightness components of the drone as can be seen in this demo video [2]. The threshold for tracking is not updated dynamically to demonstrate this behavior. Other interesting phenomena is the high frequencies observed in the graph. These high frequencies result from the continuous maneuvering and changes in the 3D orientation of the drone to compensate for the high wind forces, which in turn results in variations in the estimation of the drones center position. In the last two experiments, we can see especially large amplitude in the beginning and at the end of the experiments as a result of the takeoff and landing process which provided different angles of viewing of the drone body led to a different estimation of the drone center.



(a)

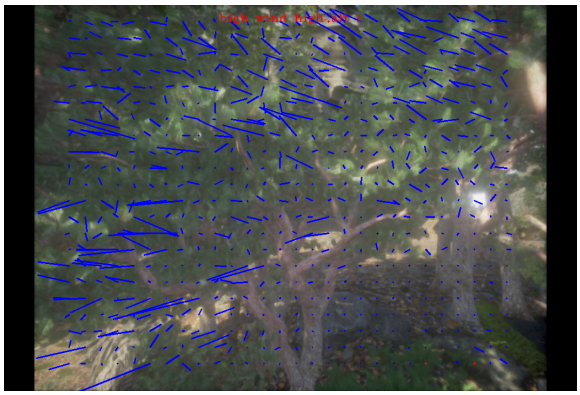


(b)

Figure 5: Simulation outputs (a) and (b) showing reproducability of results under similar environmental conditions.

The above results have clearly demonstrated the usefulness of a game engine in not only producing realistic natural environments and their camera outputs, but also providing the ability to add and modify realistic environmental effects such as changes in wind parameters and illumination conditions. These features allow us to

(a)



(b)

Figure 6: Simulation outputs showing the variations in results under differing environmental conditions. The output in (a) was generated without simulated wind, while in (b), wind was added to the simulation. We can observe degradation in the tracking quality of the grid features.

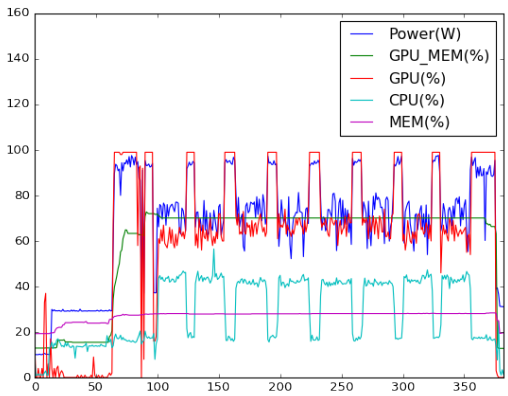


Figure 7: Profiling scene - high level profiling during scene playback created with nvidia-smi and Python psutil.

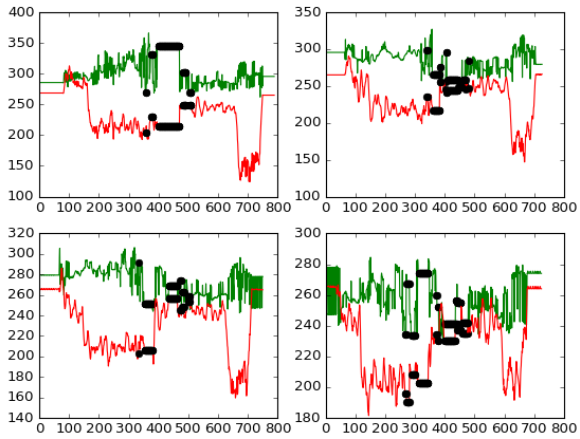


Figure 8: Tracking results - Four consecutive tracking experiments results. The black dots represents tracking failures. The X axis is frame number and the Y axis is pixel position. Green and red lines are the X,Y pixel coordinates respectively.

Figure 9: Tracking Experiment - A drone following another drone in DroneSimLab.

generate ground truth data for various test conditions and to evaluate machine vision algorithms.

7 CONCLUSIONS

The creation of visually realistic environments is a very powerful tool for computer vision research as can be seen in section 6 and this corresponding video demo [11]. The DroneSimLab project [3] aims to be a tool which adds game engines capabilities to the current existing robot simulation environments. The current work mainly focuses on UE4, but adding another game engine may increase the dimensionality of modifiable parameters in our systems. For instance, training deep learning algorithms on multiple worlds each created by a different game engine may more accurately generalize to the real world domain. This paper has presented a new framework for simulating multi-robot (specifically, multi-drone) motion in such environments, where environmental effects can be easily incorporated, and complex computer vision tasks evaluated. The simulation architecture along with the key functionalities of the simulation engine have been discussed in detail.

8 REFERENCES

[1] Ardupilot fork. https://github.com/orig74/ardupilot.

[2] Drone tracking drone in dronesimlab. https://youtu.be/Mj9xZECG40Q.

[3] Dronesimlab. https://github.com/orig74/DroneSimLab.

[4] flightgear. http://www.flightgear.org.

[5] gazebo. http://gazebosim.org.

[6] Github. https://github.com.

[7] jmavsim. https://pixhawk.org/dev/hil/jmavsim.

[8] Px4 firmware fork. https://github.com/orig74/Firmware.

[9] Scene capture 2d. https://docs.unrealengine.com/latest/INT/Resources/ContentExamples/Reflections/1_7.

[10] Ue4pyserver. https://github.com/orig74/UE4PyServer.

[11] Unreal engine 4 with python & opencv. https://youtu.be/q8kAooRaf7g.

[12] X-plane. http://www.x-plane.com/.

[13] Ilya Afanasyev, Artur Sagitov, and Evgeni Magid. Ros-based slam for a gazebo-simulated mobile robot in image-based 3d model of indoor environment. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 273–283. Springer, 2015.

[14] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.

[15] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[16] Arnaud Degroote, Pierrick Koch, and Simon Lacroix. Integrating Realistic Simulation Engines within the MORSE Framework. In *Workshop on Rapid and Repeatable Robot Simulation (R4 SIM), at Robotics: Science and Systems*, Roma, Italy, July 2015.

[17] Gilberto Echeverria, Nicolas Lassabe, Arnaud Degroote, and Séverin Lemaignan. Modular open robots simulation engine: Morse. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 46–51. IEEE, 2011.

[18] Epic Games. Unreal engine 4. http://www.unrealengine.com.

[19] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. *CoRR*, abs/1603.01312, 2016.

[20] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, and Oskar Von Stryk. Comprehensive simulation of quadrotor uavs using ros and gazebo. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 400–411. Springer, 2012.

[21] Thomas Mooney. *Unreal Development Kit Game Design Cookbook*. Packt Publishing Ltd, 2012.

[22] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *European Conference on Computer Vision*, pages 445–461. Springer, 2016.

[23] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. *arXiv preprint arXiv:1609.01326*, 2016.

[24] Adrian Rosebrock. Ball tracking with opencv. http://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/.

[25] Pedro Serra, Rita Cunha, Tarek Hamel, David Cabecinhas, and Carlos Silvestre. Landing on a moving target using image-based visual servo control. In *53rd IEEE Conference on Decision and Control*, pages 2179–2184. IEEE, 2014.

[26] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Aerial Informatics and Robotics platform. Technical Report MSR-TR-2017-9, Microsoft Research, 2017.

[27] John Skinner, Sourav Garg, Niko Sünderhauf, Peter Corke, Ben Upcroft, and Michael Milford. High-fidelity simulation for evaluating robotic vision performance. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2737–2744. IEEE, 2016.

A Novel Force Feedback Haptics System with Applications in Phobia Treatment

Daniel Brice
Queen's University
Belfast
Northern Ireland
dbrice01@qub.ac.uk

Scott Devine
Queen's University
Belfast
Northern Ireland
sdevine08@qub.ac.uk

Karen Rafferty
Queen's University
Belfast
Northern Ireland
K.Rafferty@qub.ac.uk

ABSTRACT

It is well known that multi-sensory stimulation can enhance immersion within virtual environments. Whilst there has been rapid development of devices which can enhance the visual immersion, technology to stimulate other senses, such as touch, is still under developed. Currently there is a problem wherein a surface in a virtual environment, such as a wall, cannot replicate the physical properties of a solid object. In this paper a novel system is proposed utilising the HTC VIVE and Rethink Robotics' Baxter Robot to replicate surfaces. A demonstration has been created whereby a user climbs a wall in a virtual environment by grabbing onto ledges which exist as a physical body located on Baxter's end effector. The system uses bi-directional TCP communication between an environment developed in Epic Games' Unreal Engine and the Baxter robot running the Robot Operating System framework. When an ascending user reaches out and grabs a ledge on the virtual wall they will be applying a torque to the Baxter arm which can be measured and the intended movement of the user inferred, resulting in the ledge being moved through a suitable Inverse Kinematics path. This has provided the user with the ability to climb a wall in VR in the absence of any hand tracking methods whilst receiving force feedback from the ledges they grasp onto. Current alternative systems only exist as wearables or operate in small spaces. The increased immersion in this VR demo can be used to assist those with phobias of heights.

Keywords

Encounter Haptics, Baxter, Virtual Reality, Phobia, Psychotherapy, HTC Vive, Unreal;

1 INTRODUCTION

Within recent years there has been a large rise in the popularity of Virtual Reality (VR). This has been due to the release of consumer available VR platforms, HTC Vive[1], Oculus Rift and PlayStation VR, which are now being used by early adopters. This increase in the availability of higher visual quality VR systems has resulted in industries investigating the feasibility of improving processes, such as training or demonstrating solutions. Some usage of VR has been in psychotherapy, where the ability to immerse one into any type of environment has been found to be beneficial [2]. However, it is known that multi sensorial stimulation can enhance presence within a VR environment. In particular, the ability to feel and touch something directly impacts our feeling of immersion. This paper proposes a haptic system with applications for psychotherapy of people with acrophobia (fear of heights). It is desirable to maximise the immersion of an individual during psychotherapy in VR [3]. The motivation is therefore to enable more effective treatment by improving the immersion of the individual into VR through the use of haptic feedback.

The current haptic feedback from both the Oculus Rift and the HTC Vive is limited to vibrations in the handheld controllers. In the event where a user wants to interact with a rigid body in VR, a wall for instance, they will only be able to receive feedback in the form of vibrations when the controller moves through the surface of the object. One significant problem in room-scale VR haptics is that the user is currently not able to receive a force feedback preventing them from moving through the object, therefore replication of the surface's physical presence is not achieved. This force feedback style of haptics, where the user is unable to penetrate a rendered surface, is the style used by the industry standard Sensable Phantom Omni [4] haptics device. The Phantom, commonly used for surgical training [5], provides the desired force feedback, but is limited to a workspace area of 160mm x 120mm x 70mm ($0.001344m^3$), insufficient for room-scale VR. The proposed system will demonstrate the ability to provide

force feedback haptics in a workspace volume of over $1m^3$, by utilising Rethink Robotics' Baxter robot's large 7 Degree-Of-Freedom (DOF) arms [6] as a proof of concept.

A second common limitation in VR haptics is the tethering of the user to the haptic device. The Phantom requires that the user be holding a stylus at all times for force feedback. Similarly for vibrational feedback in VR systems the user will need to be holding the controllers. This limits the user's immersion in VR as they are unable to fully interact with the object using their hands. The proposed system is able to provide force feedback in the absence of any tethered objects, such as controllers, wearables or styluses. This is done by creating an encounter haptic system, whereby the Baxter robot will produce force feedback for objects the user will be interacting with in VR.

The novel force feedback haptic system with 1:1 mapping of objects between VR and the physical world is presented in this paper. This system is comprised of an HTC Vive Head Mounted Display(HMD), Rethink Robitics' Baxter robot and two workstations. Additional functionality in the form of hand tremor measurements, measured by the Baxter robot, is explored. Applications for phobia treatment through a wall climbing demonstration are discussed, with suitability of the system explained.

2 BACKGROUND

Current Haptic Systems

One of the most popular force feedback haptic systems for professionals is the Omni Phantom. The phantom is best suited when the intention is to interact with a small object where fine manipulation and accuracy are essential. The Phantom will not permit the user to move through surfaces, applying a maximum resistive force of 3.3N. As previously mentioned the Phantom is limited to a small workspace. This is far from ideal when the user wants to perform tasks that are performed with great ranges of movement, such as scaling a wall or throwing a ball. The Phantom also requires that the user constantly holds a stylus. This stylus is attached to a series of mechanical linkages which is where the force feedback is provided. This is acceptable in the case where a surgeon will be training with a scalpel being replicated by the stylus. This does not however allow the user to contact surfaces with their hands alone. This haptic interface provided by the Phantom is common, i.e. using an intermediary object to transfer force feedback, such as in the Novint Falcon Haptic System. The Falcon has had successful results from users [7], but again is also not room scale.

The Dexmo F2 VR exoskeleton [8] is an electromechanical device attached to the hand of a user to allow haptics by means of holding virtual objects. The Dexmo is mounted onto the backside of the hand and fixed to each of the fingers. This system applies a mechanical brake on the finger joints when the user attempts to close their hand through a surface. This is effective in replicating the physical boundaries of an object held in the hand, i.e. one cannot close their fingers through a virtual ball being held. There is also no limited workspace using this design technique, it is theoretically room-scale. The force feedback of the device is limited to objects' surfaces which are held within the hand. This means that the braking system on the Dexmo will not be able to replicate a wall being pushed against by the user. The Dexmo is also a tethered device, requiring that the user must wear it at all times to receive force feedback.

In the work of Covarrubias and Bordegoni [9] researchers created a haptic device which is able to replicate curved geometry in a VE by manipulating a servo-actuated metallic strip with mechatronics mounted to a desk. In their work the geometry of a shape existing in the Unity game engine was passed through to an Arduino board connected to a series of servo motors. This enabled the curvature of shapes in Unity to be mimicked by the metallic strip. The results of this proposed system was the ability to feel the curvature of surfaces. This is a system which is also free hand, there are no intermediary objects such as a stylus or controller, increasing the haptic immersion. This system is however limited in a couple of ways. Though the curvature of a shape can be replicated, it can only be done over the narrow metallic strip, therefore the full geometry is not realized. Similar to other systems the workspace is limited, it is clear the system is only suitable for small objects and is not room scale.

Both the HTC Vive and Oculus Rift utilise vibrations in controllers to stimulate the sense of touch to the user. In the case of the user interacting with an object in VR they will be made aware of the contact between their hands and an object in a virtual environment (VE) through the use of these vibrations. This is sufficient in indicating the collision of surfaces in VR, but does not prevent users from moving their hands through surfaces they shouldn't be able to using force feedback. The users are also required to constantly be tethered to the controller to be able to feel any of these vibrations, denying the freedom of movement of the hand.

Current Phobia Treatment

Clinical psychology describes a phobia as an anxiety disorder, characterized by intense irrational fear of specific objects or situations. Many researchers have shown their progress in performing psychotherapy with the assistance of VR, to treating these phobias

[10]. The majority of treatments involve creating 3d models related to the phobia, such as city skylines in the case of acrophobia, 360 degree panorama images and sounds or videos in VR.

Acrophobia is the specific fear of heights, one which can be highly impactful on the quality of life a person can achieve if not treated. Work has been undertaken in the university of Ontario to create a VE room where people could receive treatment [11] . In this treatment patients would train in a small VE room, called the cave, using projectors to create the VE. The patients in this treatment would have physiological sensing equipment attached to them to observe their stress levels. The patients would progress through increasingly stressful acrophobic exercises as their tolerance increased. They would always retain the option to take a break from the VE if anxiety levels were too much to handle. What is notable about this study is that there is a focus on immersion, safety and objective data on the stress levels of the patient. These are aspects which have been retained in the proposed phobia treatment system.

Some have attempted to create games for the treatment of phobias. A system was designed with using the Microsoft Kinect controller with phobia treatment games on a pc for patients to work through [12],[13]. A system such as this with objectives and rewards for movement outside of the comfort zone for the phobic receiving treatment can be highly motivating and have a positive impact on their progress. This concept of measured progress and milestones has also been retained in the proposed phobia treatment system.

Research has shown how critical immersion is for inducing anxiety in patients for psychotherapy of phobias [2]. This need for immersion, alongside the absence of haptics in most VR therapy, is the motivation for the proposed system.

3 WALL CLIMBING DEMONSTRATION

Concept Features

We propose a wall climbing demonstration in VR with haptic force feedback to be used in psychotherapy treatment of those with acrophobia. The primary function of the proposed system is the ability to use the HTC Vive Head Mounted Display (HMD) as a VR viewer into a VE where the user can climb a tall wall, shown in Figure 1. By using VR the effect of ascending a wall and therefore increasing height can be achieved. This is essential in being able to trigger the fear of the phobic for treatment. The Baxter robot's role in this system will be to allow force feedback during wall climbing and enable tremor of the hand to be measured. This demonstration is absent of any hand held controller. Instead the Baxter robot's end effector will be used for input,

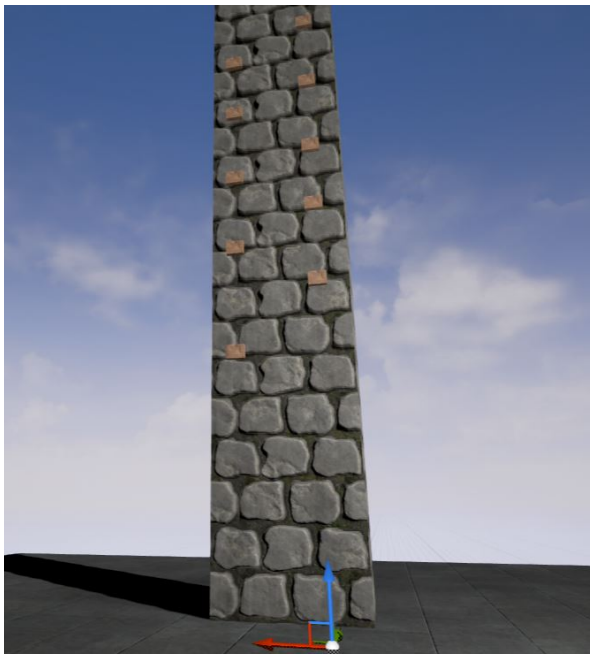


Figure 1: Wall in Unreal.

in the form of hand tremor and torque application. It is similarly used as an output, providing the force feedback of ledges in the VE. The Baxter robot itself is also referred to as a 'safe robot', due to the robots elastic actuators and torque sensing limbs. It can therefore operate alongside humans without the need for cages as most non-compliant robots require. This is what makes it suitable for haptics [6]. The sacrifice with this safety is that the arm is never entirely rigid and can be moved passively by a small amount (<15mm) by the user. This also leads to a tolerance of approx. +-5mm on the end effector accuracy[6].

Using VR allows for data to be collected from treatments, such as the height to which the patient ascends the wall or the time taken to perform a task. The user Point Of View (POV) can also be recorded, using software such as NVIDIA GeForce Experience ShadowPlay, during treatment, alongside external cameras recordings with timestamps. This means that areas of interest during the timestamped data, such as wall height position, can be further analysed by seeing exactly what the participant was seeing at the time. Similarly a camera recording the user allows for their body language to be observed retrospectively. Features such as this can be beneficial in indicating the progress of the users throughout treatment, as well as analysis to identify areas they may have been more or less comfortable with. Using a HMD for VR as opposed to creating a projector powered VE allows the system to exist in a much smaller working volume, though the Baxter is a large robot with reach of 1.21m [6]. This solves one of the big issues clinics have with VR treatment, being the amount of space used for their VE [14].

Using a VE allows for panoramic images to be stitched onto the Unreal Skybox, the VR background behind the wall. This means that patients can perform their treatment in urban city conditions with tall buildings, or more outdoor environments. This can be used for more specific treatment of participants, some of whom may be more acrophobic in certain settings.

The Baxter robot's end effector is used in an encounter haptics system, providing the ledges that the participants will be grasping onto, as shown in Figure 2. This will enable the users to move up and down the wall. The users are not required to be tethered to a device, other than the HMD for viewing purposes. This enables more natural and intuitive interactions between user and ledges. Another reason for designing the system where the users hands are free is to increase the immersion of the participant by providing them full sense of touch on their hands. One drawback of this design choice where the users hands directly contact objects is that the geometry of the object in a VE must match the geometry and scale of the physical object on the Baxter end effector.

Shaking or trembling of the hands is a common physical symptom experienced by people with phobias during anxiety inducing tasks. Links between tremor and anxiety have been made in a number of studies and it has also been recognized in a study of phobics in VR treatment [15], [3]. The Baxter is capable of both positional feedback, in the form of Cartesian coordinates, and torque sensing at the end effector. Using this torque feedback from the end effector hand tremor can be measured throughout the treatment of the user, providing objective feedback of stress levels. This can show not only progress throughout an entire treatment program, but also indicate the exact parts of the demonstration where the phobia was most stressful.

Safety and comfort are essential when performing any form of psychotherapy. By allowing free hand haptics absent of any tethering to a device the user is able to completely remove themselves from the environment simply by taking off their headset. The fact that the treatment is delivered through a VE allows users to push themselves to accomplish anxiety inducing tasks whilst retaining the option to completely remove themselves from the scene at any moment. As the participants are inside a VE they are also in no physical danger when performing tasks, such as wall climbing, as they would be if they were to perform the task outside of VR.

It is proposed that a user study be conducted, whereby people with acrophobia experience the demo with haptic feedback, as well as without haptic feedback (using controllers to climb the VR wall). Comparisons of the system with and without the haptic feedback can be made subjectively from user opinion as well as observed behaviour to validate the impact of the immersion, through haptics, on the anxiety levels of the phobic.

4 CONCEPT IMPLEMENTATION

The entire system, shown in Figure 4, is physically comprised of a Baxter robot, a Vive HMD and a Windows development machine, with a guest Ubuntu Virtual Machine (VM). The software used within the system are Ubuntu 14.04, Windows 10, ROS Indigo , MoveIt!, Unreal Engine 4 and Steam VR. The Ubuntu VM is the machine within which communication to the Baxter Robot is undertaken. The Windows PC is where the VR is controlled, using a VE created in Unreal with a plugin for the HTC Vive to be used as a viewer.

Figure 2: Example of user interacting with ledge.

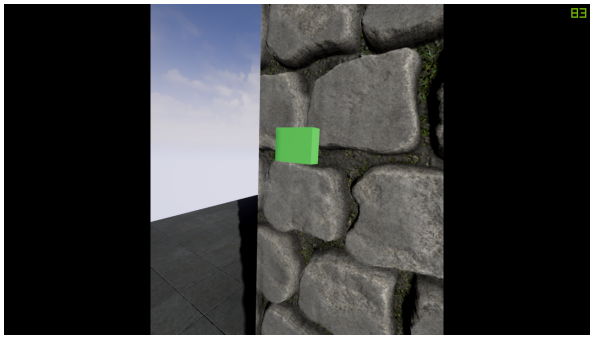


Figure 3: Example of user's view of ledge.

Setup

There were a small number of calibrations in preparing the haptic system. Initially an end effector, Figure 5,was manufactured of a pine wood cut (100mm x 130mm x 40mm), this geometry was mapped 1:1 with a corresponding Unreal model which would act as the ledges for users to grab onto on the wall. Once this was attached to the Baxter robot the robot gripper's weight was recalibrated to handle the additional mass.

To use the HTC Vive a Steam VR room setup was required. This calibration procedure was done carefully

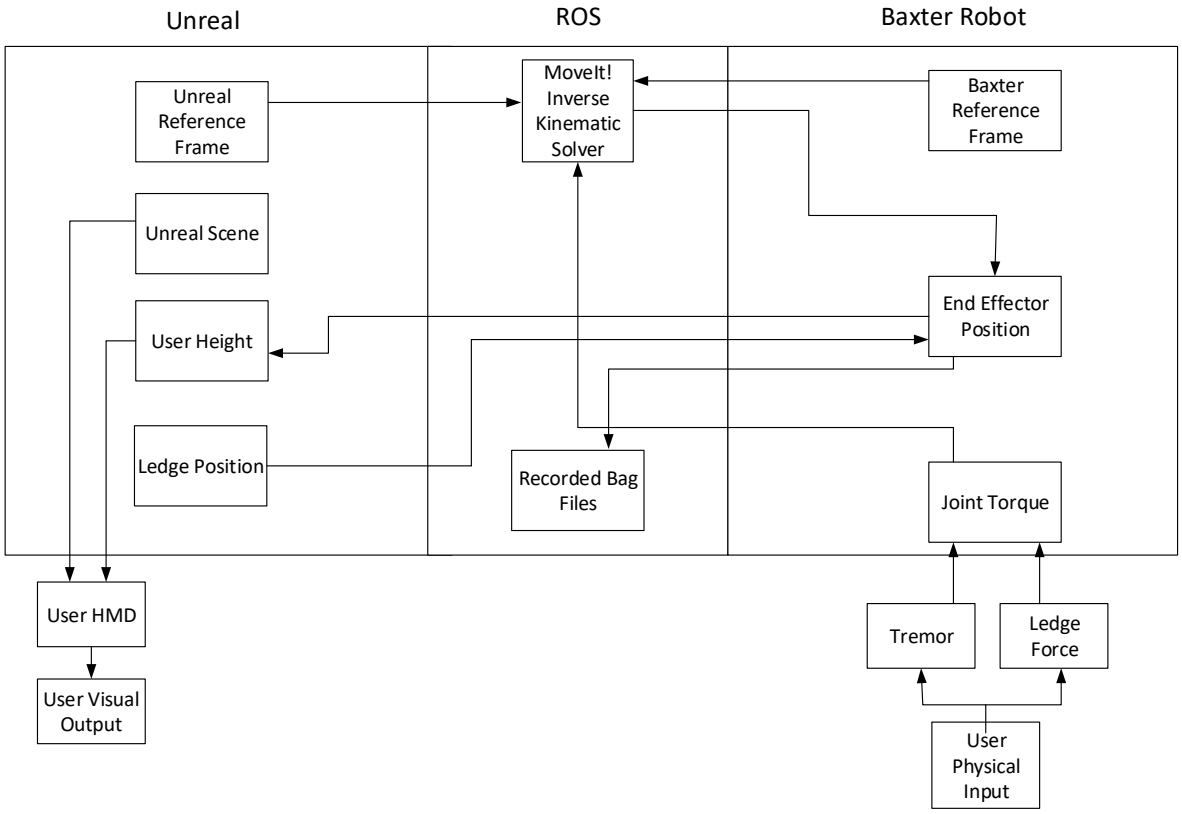


Figure 4: Information flow for Unreal, ROS and the Baxter Robot.

in an effort to align the coordinate reference frame of the Baxter with the coordinate reference frame of the HTC Vive. The distances between these axes were then derived using the Vive controllers in Unreal and sending the Baxter arm end effector to a known Cartesian coordinate, where this would be compared to a corresponding Cartesian coordinate sent to the PC by the controller occupying the same space.

Figure 5: Manufactured physical ledge.

MoveIt! is a common library used in robotics for path planning. In this case the library is used for its Inverse Kinematics (IK) solver. In such a solver cartesian coordinates with quaternions can be provided as input. These are used to determine joint angles resulting in the robot end effector being in the desired place. Not all IK solutions paths can be achieved smoothly by the Baxter robot. It is therefore important that the VEs be designed with consideration to the optimal workspace of the Baxter and movement of the Baxter end effectors be similarly limited to optimal IK solution paths.

A bi-directional TCP socket connection is formed between Unreal and ROS on the Ubuntu VM, using Rosserial Windows, a library required for ROS communication over Microsoft Windows. This permits the passing of messages from Unreal to ROS, such as ledge positions, to be used with the robot. There are a number of different types of ROS messages being passed between the Ubuntu and Windows workstations, in the form of Booleans, ROS Pose messages (Cartesian coordinates in the form of floats) and ROS Publishers and Subscribers, to enable positional data to be taken from Unreal and have the Baxter robot's end effector move to the same location in space.

Runtime

At start up a VE developed within Unreal is shown to the user through the Vive headset, shown in Figure 3.

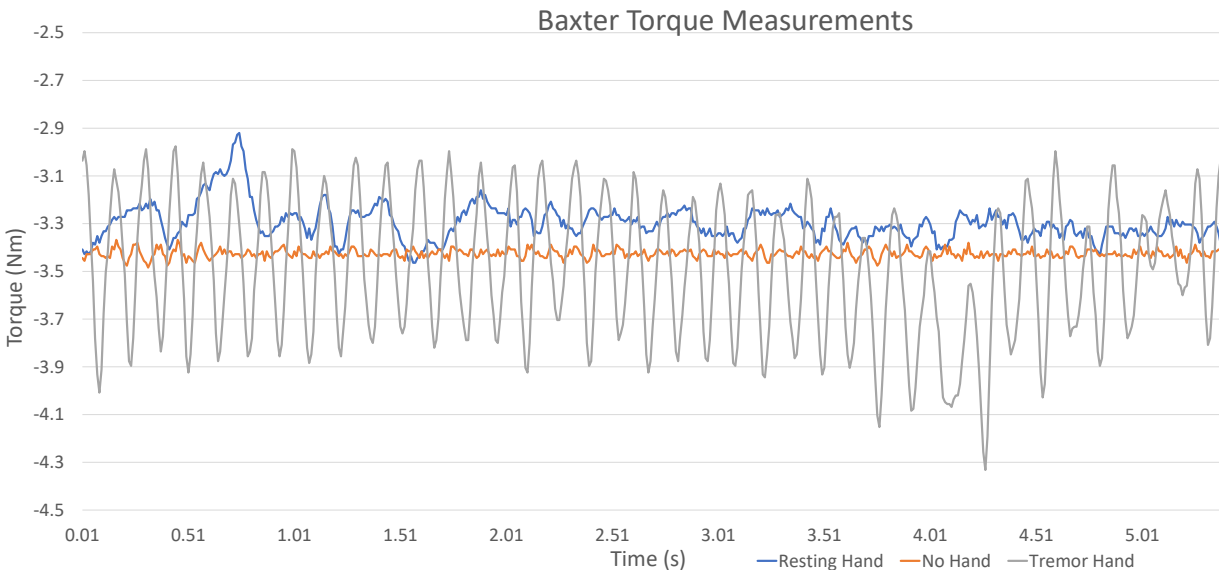


Figure 6: Tremor detection data plot.

There is an initial ledge shown through colour indication for the user grasp. As soon as this VE starts positional data is stored in ROS Pose messages which are then altered in Unreal for offsets calculated during the calibration procedure. These messages are then passed to ROS on the Ubuntu VM containing information regarding the location of the ledge. A ROS Node, an executable on ROS, will receive these Pose messages, using functions and scripts from the MoveIt! libraries to produce an IK solution which is forwarded to the Baxter robot. This results in the initial ledge which the user sees in VR being matched by a physical ledge held by the Baxter arm at the same location in space.

Once the IK solution has been met and the Baxter robot end effector is in the correct location the user is notified by a colour change in the VE, this is when they can grasp the ledge. When the user holds and moves the ledge downward torque sensing in the joints of the Baxter robot arm is used to deduce which direction the user is attempting to move the ledge in, currently limited to 2-DOF, as well as the magnitude of torque applied to each joint. With the direction of movement deduced an IK solution is provided for a new set of Cartesian coordinates, translated by 1mm in the direction of desired movement. The translation distance is adjusted in runtime based on the torque measured from the end effector, with a greater torque resulting in larger IK coordinate changes. The overall effect of this is that as the user moves the ledge the robot arm will move with them at the desired speed, whilst retaining the correct orientation of its end effector. The user is unable to move the end effector in other DOFs, such as towards or away from themselves, as the Baxter robot will be in a control mode called Positional Mode which requires IK solutions to be published for the arm to move.

As the user moves a ledge the Pose messages, containing the coordinates for the end effector, are passed from ROS to Unreal through the use of the ROS network. These are then used to adjust the height of the user in the VE, resulting in a 1:1 mapping of the ledge maintained between real world and VE during transit. At the same time torque measurements of the end effector are saved in timestamped logs. Both wall height and small tremor forces can be later analysed, inferring times when the user was most anxious and how high up the wall they were at the time. Throughout this movement of the ledge the robot arm will constantly be providing the rigid body of the ledge with force feedback. It should be noted that the force feedback will never be greater than 5N at the end effector.

Whilst one end effector is being moved downwards the other robot arm aligns itself to the next ledge for the user to grasp. This second ledge will then be ready to be grabbed irrelevant of the position of the first ledge due to the continuous 1:1 mapping and height changes of end effector and VR user height in the VE.

Measuring Hand Tremor

Hand tremor is measured through torque sensing in the Baxter robot's joints. As a user's hand tremors whilst holding the ledge they apply a force to the robot's end effector. This is then transmitted as a torque through to the closest joint. This can be recorded at 95Hz. Corresponding timestamps can then be used to align the tremor data with the Unreal recording.

To evaluate tremor detecting capabilities of the system a simple test was performed. Torques were recorded

in three scenarios:(1) no user contact with the ledge; (2) when the user held the ledge with minimum hand tremor; (3) when the user replicated mild hand tremor.

With the naked eye the tremoring hand was not visibly distinguishable from the resting hand, it was however very distinguishable in the data, Figure 6. The smallest tremor amplitude during the test corresponded to a torque change of approx. 0.2Nm, which could still be defined from the resting hand or no hand contact. In the case where a user's hand will tremor more during high levels of anxiety it will be clearly identifiable from the regions of lighter tremor. To more effectively evaluate the capabilities of the proposed system in determining stress levels. Results of the hand tremor will be compared to heart rate recordings for a participant during a future user study.

5 CONCLUSION

A novel, large volume, encounter haptics system is proposed. The system has features suitable for VR psychotherapy, in the form of offering the ability to record the POV of a user during treatment and objective feedback of anxiety levels. This haptic system is limited by the shapes available to add to the end effector of the Baxter robot and is most effective when VEs are designed with optimal IK solutions for the Baxter in mind. The increased immersion of the treatment through the use of haptic force feedback could help improve current VR treatment. The author believes that the most suitable users of such a system would be a VR treatment clinic wanting to conduct highly immersive and interactive phobia treatment. A user study to analyse the impact of these haptics in a VR treatment by comparing anxiety levels, objectively and subjectively inferred, is necessary in future work to compare a VR treatment with and without haptics.

6 REFERENCES

[1] HTC, "HTC Vive," https://www.vive.com/uk/, 2011, [Online; accessed 14-February-2017].

[2] D. P. Jang, J. H. Ku, Y. H. Choi, B. K. Wiederhold, S. W. Nam, I. Y. Kim, and S. I. Kim, "The development of virtual reality therapy (vrt) system for the treatment of acrophobia and therapeutic case," *IEEE Transactions on Information Technology in Biomedicine*, vol. 6, no. 3, pp. 213–217, Sept 2002.

[3] S. Shunnaq and M. Raeder, "Virtualphobia: A model for virtual therapy of phobias," in *2016 XVIII Symposium on Virtual and Augmented Reality (SVR)*, June 2016, pp. 59–63.

[4] Geomagic, "Phantom Omni," http://www.geomagic.com/en/products/phantom-omni/overview, 2011, [Online; accessed 14-February-2017].

[5] M. Mortimer, B. Horan, and A. Stojcevski, "4 degree-of-freedom haptic device for surgical simulation," in *2014 World Automation Congress (WAC)*, Aug 2014, pp. 735–740.

[6] R. Robotics, "Baxter Specifications," http://www.rethinkrobotics.com/baxter/tech-specs/, 2008, [Online; accessed 14-February-2017].

[7] E. Tanhua-Piiroinen, J. Pystynen, and R. Raisamo, "Haptic applications as physics teaching tools," in *2010 IEEE International Symposium on Haptic Audio Visual Environments and Games*, Oct 2010, pp. 1–6.

[8] Dexmo, "Dexmo," http://www.dextarobotics.com/, 2008, [Online; accessed 14-February-2017].

[9] M. Covarrubias and M. Bordegoni, "Immersive vr for natural interaction with a haptic interface for shape rendering," in *2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, Sept 2015, pp. 82–89.

[10] D. HorvÃ¡thovÃ¡, V. SilÃ¡di, and E. LackovÃ¡, "Phobia treatment with the help of virtual reality," in *2015 IEEE 13th International Scientific Conference on Informatics*, Nov 2015, pp. 114–119.

[11] J. P. Costa, J. Robb, and L. E. Nacke, "Physiological acrophobia evaluation through in vivo exposure in a vr cave," in *2014 IEEE Games Media Entertainment*, Oct 2014, pp. 1–4.

[12] M. B. Haworth, M. Baljko, and P. Faloutsos, *Treating Phobias with Computer Games*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 374–377. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34710-8_36

[13] S. Corbett-Davies, A. DÃ¼nser, and A. Clark, "An interactive augmented reality system for exposure treatment," in *2012 IEEE International Symposium on Mixed and Augmented Reality - Arts, Media, and Humanities (ISMAR-AMH)*, Nov 2012, pp. 95–96.

[14] B. Wiederhold, "Keynote address: Ten years of virtual reality clinical practice - lessons learned," in *2006 International Workshop on Virtual Rehabilitation*, 2006, pp. 35–35.

[15] S. J. K. Prasad, D. C. Priyanka, and V. Talasila, "A frame work for classifying physiological tremor variants employing principal component analysis," in *International Conference on Circuits, Communication, Control and Computing*, Nov 2014, pp. 173–176.

A generic approach for sunlight and shadow impact computation on large city models

Vincent Jaillot, Frédéric Pedrinis, Sylvie Servigne, Gilles Gesquière
Univ Lyon, LIRIS, UMR-CNRS 5205, F-69622, LYON, France
{vincent.jaillot, frederic.pedrinis, sylvie.servigne, gilles.gesquiere}@liris.cnrs.fr

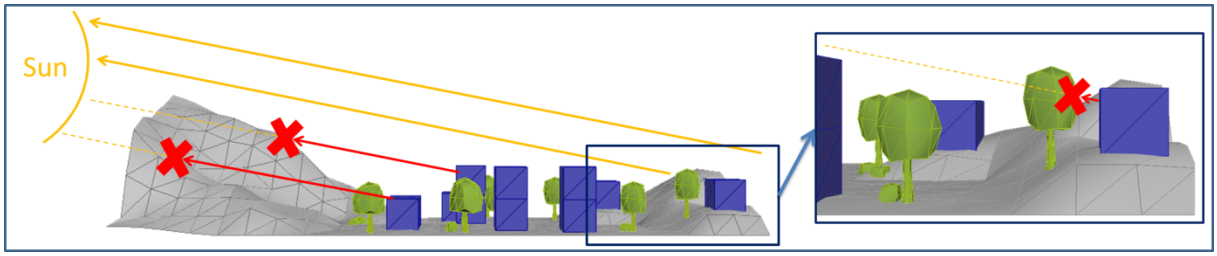


Figure 1 - Multi-scale and multi-object sunlight and shadow computation

ABSTRACT

Study of sunlight and shadow effects on the city has become more accessible with the development of 3D city models. It allows measuring when and how an object is exposed to the sunlight, which enables conducting many related studies such as energy analyses or urban planning. While many works have been done for this purpose, it may be interesting to know which objects (terrain, buildings, trees, etc.) prevent other objects from being exposed to the sunlight. In this paper we propose a method which detects the sunlit zones on a city model and the shadow impact of its objects. As these objects can be of various natures and as the acquisition processes varies from one city to another, they are not all necessarily available in each city model. Since an object's shadow can impact other very distant objects, we must have a method that handles efficiently large areas, especially knowing that city models can have fine geometric and semantic definitions. The generic approach we propose can manage these different city models by supporting every type of the above-mentioned objects and by relying on the use of standards.

This paper presents a generic method which allows sunlight and shadow computation on arbitrarily large 3D city models for impact analyses of each city object on its surroundings (close and far). This means that besides checking if a city object is shaded or not, we know which objects are responsible for the shade, thus allowing various impact analyses on cities.

Keywords

Sunlight and Shadow Computation; 3D City Models; Generic Approach; Different Scales; Large areas; Impact study

1. INTRODUCTION

More than half of the people on earth live in cities and this number should continue growing over the next few years. It implies that cities' size is constantly evolving. Governments and urban planners have thus a lot of responsibilities regarding renovation and construction projects. With this responsibility comes an increase in the will of citizens to understand their city by accessing the data describing it. Cities now offer open accesses to their 3D numerical models or to other data such as orthographies, maps, etc. For decades, 3D mostly had a visual role, but these past years, various other applications emerged [Bil15].

Sunlight computation on a 3D city model, as illustrated in Figure 1, is one of these new emerging topics. For example, it can help choosing the best area for a specific project such as a cafe terrace, photovoltaic panels [Dia11], urban agriculture [Joh15], etc. However, if many studies focus on the impact of the sun on city objects, none really considers the impact of their shadow on other city objects (shadow impact). We indeed do not only want information about which city objects are

illuminated or in the shade but we also need to know which objects create these shadows, in order to quantify the impact of a given object or a region (e.g. a well-known mountain).

City objects of virtual city models can be aggregated in layers according to their nature (buildings, vegetation, transportation, etc.). Every city model does not always have the same layers. If buildings and terrain are the most frequent, other layers such as vegetation, urban furniture or monuments can also have a significant shadow impact on the city. We thus want a method which is adaptable to all layers available in city models.

Furthermore, 3D city models can represent hundreds of km² of data (which can correspond to millions of triangles). It is necessary to be able to process it entirely because high towers or big mountains can have a very large shading impact. Our method must therefore be able to handle large scale data.

The temporal aspect must also be addressed because we want to compute the sunlight and shadow at different dates and times corresponding to different sun positions. This could for example be used to study the shadow impact brought by changes in the city between two dates.

The results of our method must be usable in different contexts by practitioners such as urban planners or geographers. Our objective is to be able to produce complete results allowing them to make different analyses according to their needs.

The method should be generic to be used with different city models across the world. We thus have to use international standards in order to make our process interoperable.

In this paper, we will first present a state of the art on this subject. We will then propose a new method to compute sunlight and shadow on large city models. Finally, we will present several possible applications on the city of Lyon in France before concluding.

2. RELATED WORKS

Real time shadow computation is a well-studied problem in video games and visual rendering oriented applications. McGuire *et al.* present several methods for computing real time shadow rendering by rasterization [McG03]. These methods allow fast shadow computation but do not allow knowing which object caused the shadow (they only give information about which pixels are in the shadow) and we need this information for quantifying the impact of objects on the city. Moreover, as these methods focus on visualisation, they only work within the frustum of the camera and the level of detail depends on the distance to the camera.

Most of the projects interested in solar analyses focus on solar radiation computation with several possible applications such as energy planning or evaluation of photovoltaic potential. Industrial solutions, such as *CiberCity*[1], *GTA GeoService GmbH*[2] or *I-Scope*[3], as well as projects like *OpenSolarMap*[4] propose solutions to compute the solar radiation of roofs in order to study their solar potential. However, they only address one part of our needs as they only focus on roof surfaces for studying the deployment of photovoltaic panels.

Freitas *et al.* present a detailed state-of-the-art review on modelling solar potential in the urban environment [Fre15]. They present and compare several methods based on numerical radiation algorithms coupled with GIS tools allowing 2D representation, analyses and visualisation, but also some more complex methods involving 3D models. The v.sun module [Hof12] for GRASS GIS is one of the latter. It offers a method to compute the solar radiation of 3D vector data using a novel vector-voxel approach allowing computing shadowing effects of city objects. However, they only focus on solar radiation of buildings on small areas (0.5 km²) and do not address the impact of city objects on their surroundings. Most of other methods presented by Freitas *et al.* [Fre15] are meant for 2D or 2.5D raster data. However, the one proposed by Redweik *et al.* allows computing the solar radiation on horizontal, tilted and vertical surfaces of LIDAR data [Red13]. Even if the results are precise, it is a quite complex approach which is meant for small areas (160 m² composed of 9 main buildings in their case). In addition, it is difficult to have semantic information linked to LIDAR data.

Alam *et al.* [Ala12] and Strzalka *et al.* [Str12], which are part of Simstadt project [Nou15], are also interested in the study of photovoltaic potentiality and integration in cities. They both propose an interesting algorithm for computing shadows in cities based on a ray-tracing process with a triangulated 3D city model. The rays go from the centroid of the triangles of the model to the sun positions during the period of computation, and if an intersection with another object is found, the triangle at the origin of the ray is set as in the shadow. In order to have more precise results, if a triangle is detected in the shadow, it is subdivided and other rays are thrown from the centroid of the newly created triangles until a predefined resolution is reached. Even if this algorithm may answer some of our needs (shadow computation of city objects), it would need to be extended to fulfil all of them. It is indeed only

[1] CiberCity : http://www.cybercity3d.com/

[2] GTA GeoService GmbH : http://www.gta-geoservice.de/

[3] I-Scope : http://www.iscopeproject.net/

[4] OpenSolarMap : opensolarmap.org

applied on a small area (1.5 km²) and only with roofs having a high photovoltaic potential (found in a pre-processing step). In addition, only the shadows casted by buildings are addressed here and other objects such as terrain or vegetation are not considered. Results do not provide information about which objects casted the shadow and thus about the shadow impact of city objects. Wieland *et al.* [Wie15] also propose a way of computing solar radiation on 3D city models using a ray-tracing method. However, instead of triangulating the 3D models until reaching a predefined resolution, they create a regular grid on each building face (walls and roofs) and generate rays from the points of this grid. This method is also only focusing on buildings and even if the shadow is computed along a regular grid with a resolution which can be modified, it does not allow linking the shadow with city objects having a semantic definition.

Alam *et al.* propose another way of computing the shadow of a 3D city model, in order to study the influence of its levels of details on the computation of the photovoltaic potential [Ala16]. Their method is highly adaptable as it allows choosing between different time intervals for sun positions, different resolutions of objects (which can be different between shadow receiver objects and shadow caster objects) and different sky resolutions. They indeed consider the sky as being a dome and divide it in patches. In a first step, they compute the visible part of the sky for each point of the buildings. In order to do that, they perform a ray-tracing process per triangle and for each sky-patch using a kd-Tree, which is very efficient for accelerating ray-tracing when looking for intersections with close neighbours. They compute and store a sky view factor [Wat87] for each sky-patch and each triangle for which the solar radiation will be computed. After doing that, they compute the sun positions and get the sky view factors of the active sky patch (where the sun is) in order to compute the solar radiation. Even if this method is flexible, interoperable and proposes a solution for accelerating the computation process, it only focuses on buildings, and on their photovoltaic potential, and does not address visibility and shadow impact issues.

To sum up, most methods only tackle problems related to shadow visualisation (mainly in visual rendering) or to solar radiation computation for energy analyses or photovoltaic potential evaluations. None is interested in computing and analysing the shadow impact of city objects. Moreover, most of them only consider buildings, plus terrain for some. None proposes a generic way for handling all city objects. In addition, most of the applications are applied on small areas as they mainly focus on neighbouring objects and not on the entire city. However, the temporal aspect is frequently considered as sunlight computations are often made on time periods. Standards are not always used but they are required for having an interoperable method, especially if the results are generated for usage in further processes.

3. SUNLIGHT AND SHADOW COMPUTATION PROCESS

We use the CityGML standard [Kol05] for describing our city models. Even if our method is not dependant on this standard, its use is spreading among cities and meets our needs. It allows describing 3D city models according to different layers of city objects which can have geometric and semantic information.

Loading an entire city model can be problematic since it costs a lot in terms of memory. To be able to manage arbitrarily large scale city models, we use a tiling process [Ped17]. This automatic process splits the 3D city model according to a regular grid with a cell size defined by the user (Figure 2). A tiled city model allows controlling the memory cost of the process since we can then load one tile at a time. Our method can thus cover entire city models without memory limitations.

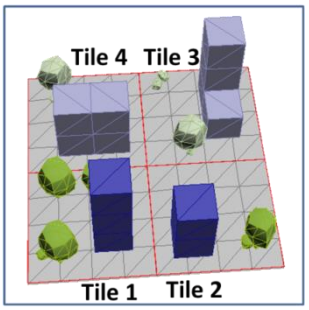


Figure 2 - A city model tiled according to a regular grid.

In order to compute the sunlight and shadow of a city model, we first need to compute the position of the sun corresponding to the dates and times of the study. We consider the sun's rays as parallel beams so we only need to compute the azimuth and elevation angles of the sun to know the direction of the rays. Michalsky [Mic88] presents an algorithm to compute these angles from the year 1950 to 2050 with uncertainties of +- 0.01°, which is acceptable for our application. We use this method to compute the N sun's positions corresponding to the N dates and times for which we want to compute sunlight and shadow.

With this information, we want to generate rays from each object of the city model toward the desired sun positions (corresponding to multiple dates and times). We then have to detect for every ray if it intersects another object of the city model or if it is exposed to sunlight. Each city object intersected by a ray is

identified as an object shading the origin of the ray, and the object corresponding to this origin is thus considered in the shadow. All our computations are made assuming a clear sky.

We implemented two simple tests to simplify the process by avoiding unnecessary computations. First, if we detect that a face is not oriented toward the sun, we directly set it as in the shadow since it is necessarily shaded by other faces of the city object. Then, based on the fact that sun rays always come from above, we do not compute the intersection between a ray and a face if this one is below the origin of the ray.

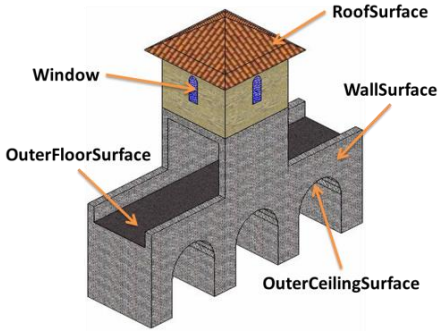


Figure 3 - Different semantically defined objects that may compose a bridge according to the CityGML standard. (Image extracted from CityGML 2.0 documentation).

To avoid testing the intersections with the 3D geometry of every city model object, we set up a **semantic Bounding Volume Hierarchy (sBVH)**. This is a Bounding Volume Hierarchy where each level corresponds to a semantic level of the city model: for each semantically defined object of a city model, a bounding volume will be computed and stored in the hierarchy. For example, in the CityGML standard, a bridge is a semantically defined object of a city model (see Figure 3) and will thus have a bounding box and correspond to a node in our sBVH. In figure 3, we can see that in this standard, a bridge can be decomposed in various objects that have a semantic definition (Window, OuterCeilingSurface, etc.). All of them will then also have a bounding box and be children nodes of the bridge in our sBVH. This principle is applied to all city objects and sub-objects until it reaches the last level of defined semantic objects in the city model.

We quickly navigate through the city model by testing intersections with bounding boxes instead of geometries. We then only have to load the 3D geometries of the objects of the lowest levels of the hierarchy which bounding boxes are intersected. The sBVH of a city model is presented in Figure 4: it is organized in several layers that have been tiled according to a regular grid, and each tile is composed of multiple levels of city objects having a semantic definition.

The use of the CityGML standard is important since it allows many possibilities in terms of semantic definition of city objects like buildings. Moreover, some of the current development of the standard (CityGML 3.0) concerns the addition of new semantic structures such as storeys for buildings, which will feed the sBVH. Some layers such as terrain are however rarely decomposed in multiple distinct objects so the use of the hierarchy would not be very effective in this case. The contribution of the sBVH thus depends on the semantic precision of the city model and is not the same for each layer.

The tile level of the sBVH is only defined using geometric information and not semantic. It is required for processing large areas because loading a complete layer at once can cause memory issues. Since there is no available semantic information

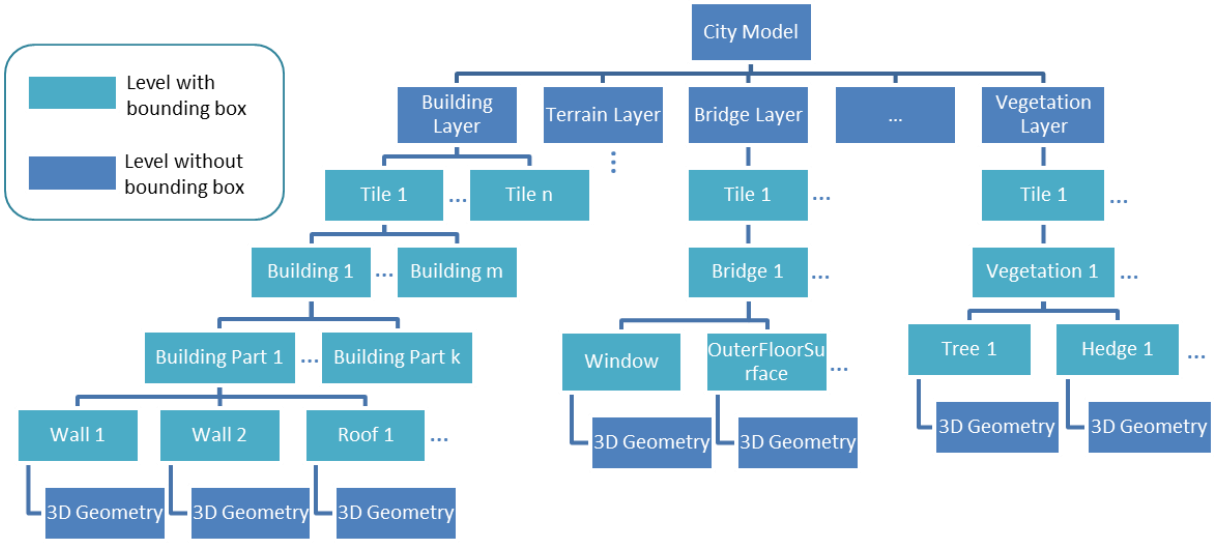


Figure 4 - sBVH of a city model composed of different tiled layers and of semantically defined city objects. A bounding box is precomputed for each of them.

allowing partitioning the city model, we chose to use an existing method based on geometry to partition the city [Ped17].

For each point of the city model, we consider *N* rays going toward the *N* precomputed sun's positions. We test intersections between the rays and the bounding boxes of the model by going through the sBVH presented in Figure 4. For each object of the lowest level of the hierarchy, we store a list of the rays intersecting its bounding box. Note that each ray holds a link to its origin and the date and time corresponding to a sun position. This enables us to store this in the intersected objects.

After having generated every ray and identified the possible intersected city objects (without having to load any 3D geometry, besides for initializing the rays), we browse them, load their 3D geometry and make intersection tests with every ray contained in their list. This way, we only have to parse and load the geometry of each intersected object once (just before computing intersection with every ray that has intersected its bounding box).

For a given ray generated from an object O1, if we find an intersection with the geometry of an object O2, we store the information that O2 shades O1 at the corresponding date and time. After processing the entire sBVH of the city model, the shadow impact and the sunlight information of every object can be measured.

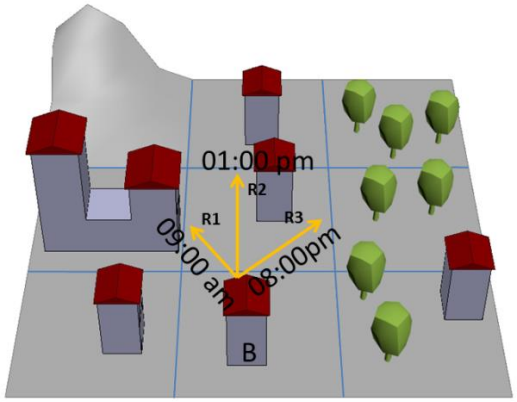


Figure 5 - 3 rays, which correspond to 3 hours, generated from a building toward the sun position, in a simple city model composed of 9 tiles and 3 layers (Terrain, Building and Vegetation).

Figure 5 shows an example of 3 rays generated for 3 sun positions from a point of a building B. Based on the first level of sBVH, for each ray {R1, R2, R3}, we search all tiles whose bounding boxes are intersected, as illustrated in Figure 6. The bounding boxes BB2, BB3, BB4 and BB7 are intersected by the rays. This means that we are going to go down in the sBVH for these 4 tiles. The other tiles are not intersected so we will not consider them for the rest of the computation for this point of B.

We then test the next level of the sBVH by computing intersections between each ray and the bounding boxes contained in the tiles previously intersected by these (Figure 7). Since R2 does not intersect any bounding box in the only tile it goes through, it intersects nothing in the city model. It is then directly going to the sun. In other words, the tested point of the building B is illuminated by sunlight at 01:00 pm.

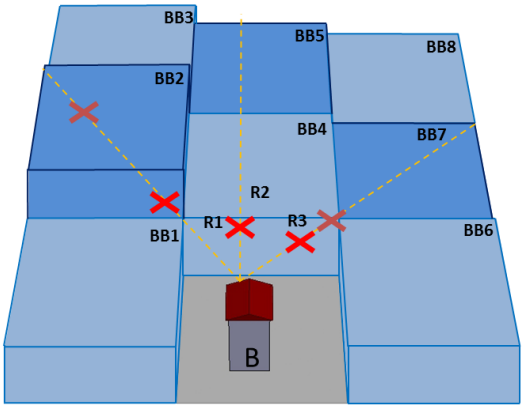


Figure 6 - Computation of intersections between the 3 rays and the bounding boxes of the tiles.

For rays R1 and R3, we need to continue browsing the sBVH because bounding boxes of the current level are intersected: T1 (corresponding to a terrain object) and B1 (corresponding to a building) are crossed by R1 while V1 (corresponding to vegetation) is crossed by R3.

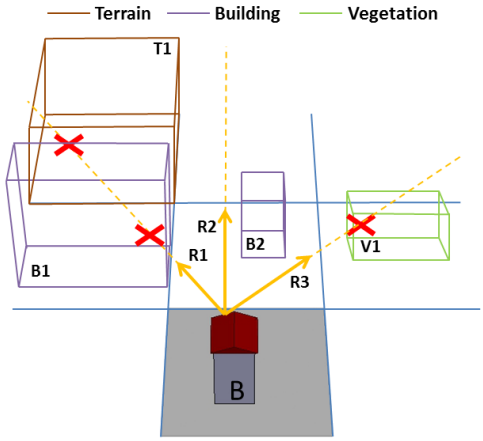


Figure 7 - Computation of intersections between the 3 rays and the bounding boxes of the objects of the intersected tiles of Figure 6.

The next level of intersections tests is represented in Figure 8. The terrain object T1 does not possess any sub object (it corresponds to the lowest level of the sBVH in its branch) so no more tests are needed. We just link it to the ray R1 (and its information: the point of the building B it comes from and the corresponding date and time), and we put it aside pending the next step.

We detect that R1 only intersects the bounding box of the wall part W1 of the building B1, and that R3 intersects the bounding box of the tree Tr1. These two objects are at the lowest level of the hierarchy in their respective branch of the sBVH, so we link them to the corresponding rays.

The next and final step consists in loading one by one the geometries of the city model whose bounding boxes possess at least one link with a ray in order to compute the last intersection tests (Figure 9).

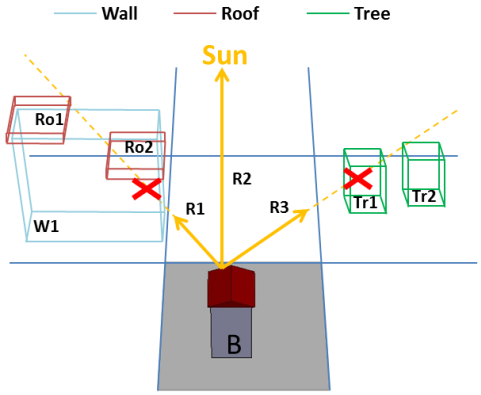


Figure 8 - Computation of intersections between the 2 remaining rays and the bounding boxes of the sub objects of the intersected objects of Figure 7.

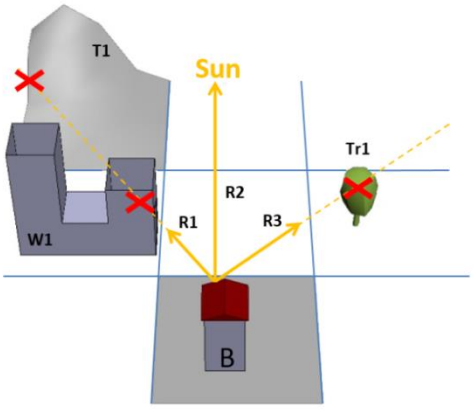


Figure 9 - Computation of intersections between the 3D geometries and the linked rays.

Finally, we can conclude that the tested point from the building B is shaded at 09:00 am by the wall part W1 of the building B1 and by the terrain T1, is illuminated at 01:00 pm and is shaded by the tree Tr1 at 08:00 pm. For all of them, the 3D position(s) that actually create the shade (corresponding to the intersection between the ray and the 3D geometry) are also known.

By querying the results, it is then easy to know that the wall W1 shades this point of the building B at 09:00 pm (at the 3D position intersected the ray R1).

We can also do the same for the terrain T1 or the tree Tr1.

As presented in this section, we perform a ray tracing process with rays going from points of the city model to the sun in order to know if they are sunlit or if they are shaded by city objects. To compute such an analysis on the entire city model, we then need to propose a discretization process in order to have a set of points that describes the entire 3D geometry of city objects.

In our datasets, we already have a triangulated 3D city model and we chose to keep it: we generate a ray for each triangle initially existing in the triangulated city model. Its origin is at the centroid of the triangle and it is oriented toward the sun positions. This induces some imprecisions since the triangulation is not necessarily homogeneous: some triangles are large and they can only store a single sunlight result even if they cover large areas. We should also address the fact that the triangles shapes may vary: an elongated triangle will produce imprecise results with our sunlight computation method even if it has a small area. A triangle subdivision process should thus also take into account the elongation. However, it was not a point we wanted to address in this paper since it concerns input files quality and many methods already exist to generate a precise triangulation of 3D models, especially for this kind of application. For example, before processing their sunlight computation, Alam *et al.* [Ala12] and Strzalka *et al.* [Str12] compute a more precise triangulation until the area and the elongation of each triangle are below threshold values.

The results are stored in a database in a way that offers possibilities to aggregate them at the user's convenience (for micro or macro analyses) in order to make them more workable for further computations. Each sunlight and shadow result is linked to the concerned objects in the city model. This allows retrieving all information linked to the objects: the geometric ones (e.g. the area, the perimeter, etc.) as well as the semantic ones (e.g. the address of a building, the name of a road, etc.). This permits users to aggregate various information allowing diverse applications.

4. APPLICATIONS

4.1 One method, multiple outputs

The implementation work has been done using the features of *3D-Use*[5] platform (3D Urban Scene Editor) in which we implemented the process presented in this paper. This tool supports various GIS (Geographic Information System) data and permits to elaborate and validate new processes. 3D-Use can open many file formats like CityGML, 3ds,

[5] 3D-Use : liris.cnrs.fr/vcity/wiki/doku.php?id=3duse_en

obj or Shapefile and proposes a 3D visualization of data coming from these files.

Figure 10 presents the total workflow of the sunlight and shadow computation: given the sun positions and a 3D city model, 3D-Use computes the sunlight and shadow information and adds them to a database. This data can then be fetched in order to generate outputs depending on what one would like to analyse. It is for instance possible to generate 2D shadow maps which can be useful for analysing the shadow impact on non-vertical surfaces. It is also possible to generate various types of charts depending on what one would like to know in term of sunlight and shadow impacts. Examples and uses of shadow maps and charts will be presented in the next sections. In addition, we will propose a temporal visualisation of the results in 3D-Use platform in section 4.5.

The purpose of the applications presented in this section is to illustrate the type of results that our method allows. However, domain specialists like urban planners will elaborate more pertinent usages of our method (e.g. comparison of the shadow impact of concurrent construction projects, understanding why a square or a park has been created in a certain area, etc.). In this goal, 3D-Use has been made available in open source[6] to our partners in order to make these dedicated studies.

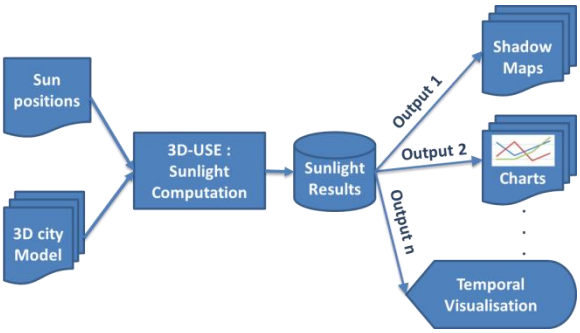


Figure 10 - Multiple outputs generation.

4.2 Application to the city of Lyon dataset

More than 500 Km² of data are available for the city of Lyon (France) and its surroundings. They are composed of 3D models stored in CityGML files already organized in different layers: LoD2 buildings, water and Digital Terrain Model (DTM). In addition to this 3D data, a large number of vectorial 2D datasets describe the territory (nearly 600 different datasets are available in the *Lyon open data*[7]). For example, we have an access to the road network, to forested areas or to the trees database and we can use them to generate or enhance 3D geometry in order to improve the virtual model of the city of Lyon and to get more relevant results.

[6] http://liris.cnrs.fr/~vcity/wiki/doku.php?id=3duse_en

[7] Lyon open data: https://data.grandlyon.com/

Figure 11 shows three tiles: one from a sparse district of Quincieux (small city near Lyon in France - on the left), one from a residential district of Francheville (another city close to Lyon - in the middle) and one really dense from the city centre of Lyon (on the right). We computed the sunlight and shadow on these three tiles of the same size (500m*500m) but of different densities (within these areas and without), on an i7-4770 @ 3.40GHz CPU.

Table 1 presents the computation results of the 3 different tiles presented in Figure 11 with two layers (LoD2 building and terrain) and for two different periods of time (one day and one month) with a time step of one hour.

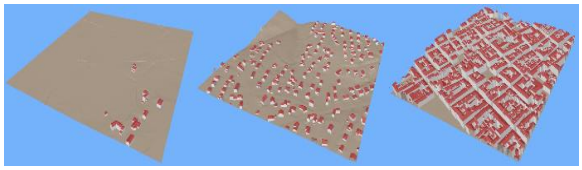


Figure 11 - Three tiles (500m*500m) of various urban densities: sparse tile of a district of Quincieux (on the left), residential district of Francheville (in the centre) and dense district of the city centre of Lyon (on the right).

Tile	Layer	Triangles	Time Period	Process Time
Quincieux	Buildings	460	1 day	43 s
			1 month	54 s
	Terrain	2840	1 day	8 min 30 s
			1 month	11 min 22 s
Francheville	Buildings	8095	1 day	1h 52 min 12 s
			1 month	3 h 8 min 43 s
	Terrain	13057	1 day	3h 48 min 34 s
			1 month	6 h 35 min 2 s
Lyon	Buildings	35455	1 day	22 h 42 s
			1 month	35 h 56 min 13 s
	Terrain	8767	1 day	6 h 19 min 39 s
			1 month	12h 39 min 20 s

Table 1 - Computation time of our method for three tiles (500m*500m) of various urban densities on two different time periods: 1 day (the 07th of April 2017) and 1 month (October 2016).

We can note that it can take more than one day to compute the sunlight and shadow of a tile for a period of one day in the case of a very dense area but it can also be very quick in some less populated regions such as the districts of Quincieux or Francheville. However, we can clearly see that the computation for one month takes a lot less than 30x more the time of computation for one day. This is due to our way of managing data presented in section 3: we only open 3D geometry once to test its intersection with all rays coming through its bounding box. Thus, the complexity of our method is linear but with a small factor depending on the dataset of the city. This means that increasing the

number of sun positions has a limited influence in terms of computation time.

Since our goal is to generate results more semantically precise than usual methods for computing shadows, our solution is mostly slower. However, our method is highly parallelizable as the computation process is the same for each triangle which means that we could use computing grids or GPGPU to reduce computational times.

4.3 Impact of a tower on its surroundings

The genericity of the method presented in section 3, the output possibilities detailed in section 4.1 and the available data described in the previous section allow a lot of different applications to our process. In this section, we will present an example of one of these possible applications: an analysis of the impact of the shadow of 'Tour Part-Dieu', a tower of Lyon on its surroundings in terms of distance and surface during two different days of the year (18/02/2016 and 04/07/2016). This tower, shown in Figure 12, is 165m high and its footprint covers 1 115 m². Our way of storing data presented in section 3 and the information about the object which casts the shadow allow to easily extract information allowing an impact analysis.

Figure 12 - The 'Tour Part-Dieu', a tower of Lyon (165m high).

Figure 13 shows the evolution of the maximum length of the shadow of this tower on the 18/02/2016 (in dark blue) and on the 04/07/2016 (in light blue). The curves have a similar shape: a spike at sunrise, a slowly decreasing path until the middle of the day and a slowly increasing path during the afternoon followed by another spike just before sunset. We can notice the big maximum lengths at sunrise and sunset. Actually, when the sun is low the impacted city objects situated far from the tower are also in the shade due to other closer objects, but this measure gives the theoretical impact of the tower. This information about which other city objects shade this particular object can also be extracted from the results of our method. During the other hours of sunlight of the day, we can note that the impact is of several hundred meters. This justifies considering entire territories for sunlight and shadow computation, allowing computing the full shadow impact of high-rise buildings and mountains. Large scale data management is fundamental to provide complete results.

We generated charts representing the evolution of the shadowed area caused by the tower at different times of the day. These graphs are presented in Figure 14. On the top, we can see the evolution of the shadow area on buildings, and on the bottom, the one on the terrain (without the buildings). In the two charts, the curve in dark blue represents the values on the 18/02/2016 and the one in light blue the values on the 04/07/2016.

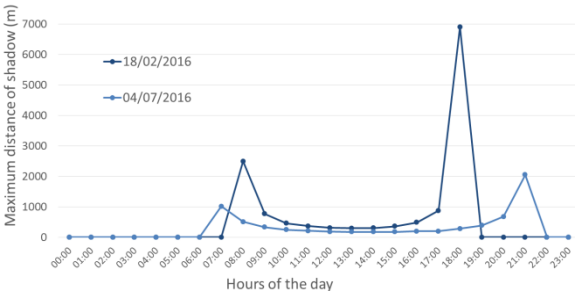


Figure 13 - Maximum length (in meters) of the shadow of 'Tour Part-Dieu' on its surroundings on the 18/02/2016 (in black blue) and the 04/07/2016 (in light blue).

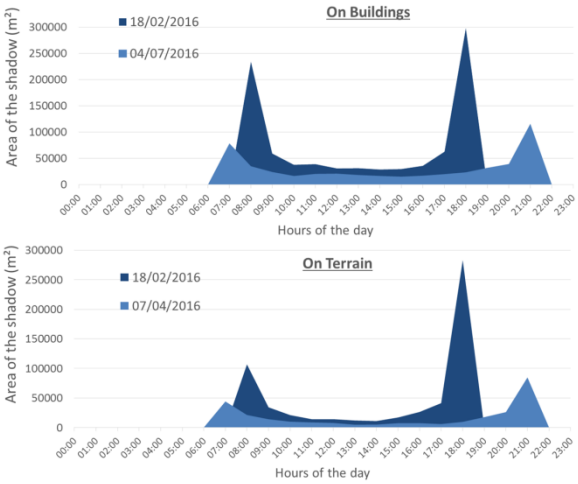


Figure 14 - Area (in square meters) of the shadow of 'Tour Part-Dieu' on the buildings (top) and on the terrain (bottom) of its surroundings on the 18/02/2016 (in dark blue) and the 04/07/2016 (in light blue).

We could pair the results shown in Figures 13 and 14 with other output information which our process allows to generate such as the semantic information of the shadowed parts of the model (roof, wall, owner, etc.). Moreover, we can generate these charts for other towers and compare their respective shadow impacts. We could also generate such analyses for concurrent construction projects of a tower. This would allow urban planners to easily take into

account the shadow impact of concurrent projects. For example, if the results produced by our process shows that one of the construction project shades 80% of an urban agriculture farm, urban planners will probably not keep this project or will propose modifications before the construction to avoid conflicts of interests.

4.4 Sunlight and Shadow Map

Our results can also be exploited to generate 2D sunlight and shadow maps representing the number of hours of sunlight of non-vertical surfaces of city models (such as roofs or terrain). Figure 15 shows the sunlight and shadow map of two tiles (see Figure 11): a district of Francheville on the left and a district of the centre of the city of Lyon on the right, both on the 17th of April 2017. On both figures, the triangles of the models are coloured from blue to red, depending on the number of hours they are exposed to the sun during this day.

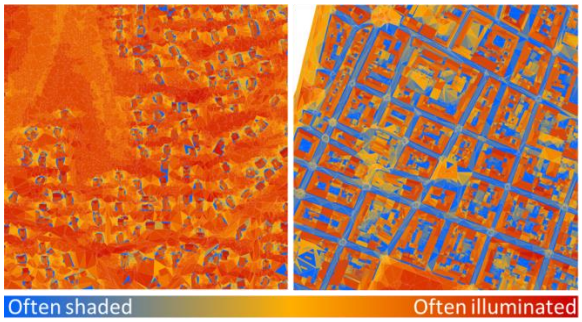


Figure 15 - Sunlight and shadow map of a district of Francheville (on the left) and of a district of the center of Lyon (on the right), both on the 17/04/2017.

On the shadow map of a district of Francheville (left side of Figure 15), we can distinguish the houses surrounded by small areas with little sun (red shapes surrounded by blue and yellow zones) and the two valleys on the upper left corner of the picture (light orange zones). On the shadow map of the district of the centre of Lyon (right side of figure 15), we can clearly see the roofs of the buildings which are a lot more illuminated than the terrain in their surroundings, indicating that the buildings are quite high and close to each other, unlike the houses of the district of Francheville. These sunlight and shadow maps can, for instance, help identifying which roofs or which terrain areas have a strong photovoltaic potential. We could also pair these results with the solar irradiance values of roofs and terrain which we could easily compute using one of the methods presented, analysed and compared by Loutzenhiser et *al.* [Lou07]. Once this solar irradiance values computed, we could store them with the information already computed.

In this application case, the sunlight and shadow maps represent the results for a day but it is of course possible to generate the same maps for a longer (or shorter) period depending on what one needs, and to choose the time step between two measures. Moreover, it is also possible to generate more macro results than one value per triangle by colouring for example each building in only one colour depending on the mean value of hours of sunlight of its triangles.

4.5 Temporal visualisation of the sunlight and shadow in Lyon and its surroundings

Another possible output is the temporal visualisation of sunlight and shadow on a 3D urban model. In order to do that, we improved some of the features of 3D-Use (allowing to manage temporal changes of cities [Cha17]) to be able to visualise the evolution of the shadow during a time period chosen by the user.

In Figure 16, the sunlight and shadow visualisation of a city district of Quincieux (presented in section 4.1, figure 11) at the same time (15:05) but at different dates: the 7th of January 2017 (on the left) and the 17th of July 2017 (on the right). On these images, we can clearly see the change of sun position between January and July.

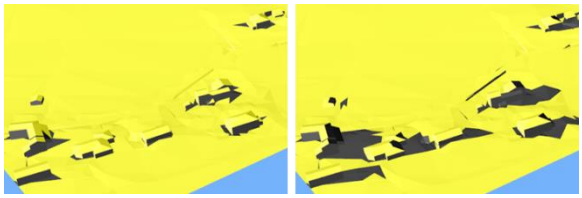


Figure 16 - Sunlight and shadow visualisation of a district of Quincieux on the 7th of January 2017 at 15:05 (on the left) and on the 17th of July 2017 at 15:05 (on the right).

In Figure 17, we show the visualisation of the sunlight results computed for a district of Francheville (see Figure 11) on the 17th of April 2017 at different times: 08:00 in the upper left corner, 10:00 in the upper right corner, 14:00 in the lower left corner and 19:00 in the lower right corner. At 08:00 and at 19:00, we can clearly notice the impact of the small hill and that the shadow generated by the houses is more important than at 10:00 and 14:00.

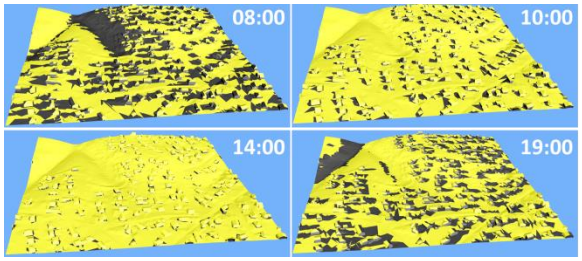


Figure 17 - Sunlight and shadow visualisation on a district of the city of Francheville on the 17/04/2017 at different times.

5. CONCLUSION & FUTURE WORKS

We have presented a method allowing sunlight and shadow impact computation on large city models. Our method allows not only to know which objects are sunlit and which are in the shadow at any time of the studied period but also which objects create the shadows. The genericity of our method allows considering all types of city objects and the use of standards permits to apply our method to datasets of various cities of the world. The sBVH structure presented in this paper allows to handle very large areas and to consider both close and far shadow impacts. Finally, the multiple possible outputs allow urban specialists to study the shadow impact of city objects and thus to understand today's city and better plan its future.

The accuracy of our results depends on the precision of the geometry and semantic of the input city model. In order to obtain more precise results, one can either provide improved input quality of the 3D geometries (through pre-processing) or add more semantic levels in the city model (as planned in CityGML 3.0). Computation time would be increased but the parallel nature of our method has the potential to drastically reduce the global computation time.

6. ACKNOWLEDGMENTS

This work was performed within the BQI program of Université Lyon 1. This work was also supported by the LABEX IMU (ANR-10-LABX-0088) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR). Special thanks to Clémence Dutel who designed most of the figures. We would also like to thank Eric Boix and Jeremy Gaillard for their feedbacks.

7. REFERENCES

[Ala12] Alam, N., Coors, V., Zlatanova, S. & Oosterom, P.V. Shadow effect on photovoltaic potentiality analysis using 3D city models. International Society for Photogrammetry and Remote Sensing (ISPRS), 2012.

[Ala16] Alam, N., Coors, V., Zlatanova, S. & Oosterom, P. J. M. Resolution in Photovoltaic Potential Computation. ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 89-96, 2016.

[Bil15] Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S. & Çöltekin, A. Applications of 3D city models: state of the art review. ISPRS International Journal of Geo-Information, 4(4), 2842-2889, 2015.

[Cha17] Chaturvedi, K., Smyth, C. S., Gesquière, G., Kutzner, T. & Kolbe, T. H. Managing versions and history within semantic 3D city models for the next generation of CityGML. In Advances in 3D Geoinformation (pp. 191-206), 2017.

[Dia11] Díaz-Dorado, E., Suárez-García, A., Carrillo, C. J. & Cidrás, J. Optimal distribution for photovoltaic solar trackers to minimize power losses caused by shadows. Renewable Energy, 36(6), 1826-1835, 2011.

[Fre15] Freitas, S., Catita, C., Redweik, P. & Brito, M. C. Modelling solar potential in the urban environment: State-of-the-art review. Renewable and Sustainable Energy Reviews, 41, 915-931, 2015.

[Hof12] Hofierka, J. & Zlocha, M. A New 3-D Solar Radiation Model for 3-D City Models. Transactions in GIS, 16(5), 681-690, 2012.

[Joh15] Johnson, M. S., Lathuillière, M. J., Tooke, T. R. & Coops, N. C. Attenuation of urban agricultural production potential and crop water footprint due to shading from buildings and trees. Environmental Research Letters, 10(6), 064007, 2015.

[Kol05] Kolbe, T. H., Gröger, G. & Plümer, L. CityGML: Interoperable access to 3D city models. In Geo-information for disaster management, (883-899), 2005.

[Lou07] Loutzenhiser, P.G., Manz, H., Felsmann, C., Strachan, P. A., Frank, T. & Maxwell, G.M. Empirical validation of models to compute solar irradiance on inclined surfaces for building energy simulation. Solar Energy, 81(2), 254-267, 2007.

[McG03] McGuire, M., Hughes, J. F., Egan, K., Kilgard, M. J. & Everitt, C. Fast, practical and robust shadows. Brown University Computer Science Tech Report CS-03-19, November 2003.

[Mic88] Michalsky, J.J. "The Astronomical Almanac's Algorithm for Approximate Solar Position (1950-2050)". Solar Energy. Vol. 40, No. 3, 1988; pp. 227-235, USA. 1988.

[Nou15] Nouvel, R., Brassel, K.H., Bruse, M., Duminil, E., Coors, V., Eicker, U. & Robinson, D. A New Workflow-driven Urban Energy Simulation Platform for CityGML City Models. CISBAT 2015, September 9-11, 2015.

[Ped17] Pedrinis, F. & Gesquière, G. Reconstructing 3D Building Models with the 2D Cadastre for Semantic Enhancement. In Advances in 3D Geoinformation (pp. 119-135), 2017.

[Red13] Redweik, P., Catita, C. & Brito, M. Solar energy potential on roofs and facades in an urban landscape. Solar Energy, 97, 332-341, 2013

[Str12] Strzalka, A., Alam, N., Duminil, E., Coors, V. & Eicker, U. Large scale integration of photovoltaics in cities. Applied Energy, 93, 413-421, 2012.

[Wat87] Watson, I. D. & Johnson, G. T. Graphical estimation of sky view-factors in urban environments. Journal of Climatology, 7(2), 193-197, 1987.

[Wie15] Wieland, M., Nichersu, A., Murshed, S.M. & Wendel, J. Computing solar radiation on CityGML building data. In 18th AGILE international conference on geographic information science, 2015.

A Primary Morphological Classifier for Skin Lesion Images

Jules Matthew A. Macatangay
De La Salle University
2410 Taft Ave., Malate
Philippines 1004 Manila,
Metro Manila
jules_macatangay@dlsu.edu.ph

Conrado R. Ruiz, Jr.
De La Salle University
2410 Taft Ave., Malate
Philippines 1004 Manila,
Metro Manila
conrado.ruiz@dlsu.edu.ph

Richard P. Usatine
University of Texas
Health Science Center
8529 Raintree Woods Dr.
USA 78015 Fair Oaks
Ranch, Texas
usatine@uthscsa.edu

ABSTRACT

Classifying skin lesions, abnormal changes in skin, into their morphologies is the first step in diagnosing skin diseases. In dermatology, morphology is a categorization of a skin lesion's structure and appearance. Rather than directly classifying skin diseases, this research aims to explore classifying skin lesion images into primary morphologies. For preprocessing, k-means clustering for image segmentation and illumination equalization were applied. Additionally, features utilized considered color, texture, and shape. For classification, k-Nearest Neighbors, Decision Trees, Multilayer Perceptron, and Support Vector Machines were used. To evaluate the prototype, 10-fold cross validation was applied over a dataset assembled from online resources. In experimentation, the morphologies considered were macule, nodule, papule, and plaque. Moreover, different feature subsets were tested through feature selection experiments. Experimental results on the 4-class and 3-class tests show that of the classifiers selected, Decision Trees were best, having a Cohen's kappa of 0.503 and 0.558 respectively.

Keywords

Skin lesion, classification, machine learning, computer vision.

1 INTRODUCTION

Skin lesions are abnormalities in the surface of one's skin. These differences include changes in color, texture, and consistency. Various groups have become interested in applying technologies such as computer vision and machine learning techniques into the domain of skin lesions. Given that skin cancer is one of the most common, dangerous, and prevalent types of cancer, it has attracted many researchers to the development of effective automated detection of malignancies in skin lesions. With the technology available, many have done research on the computer-assisted analysis and diagnosis of skin lesions. For instance, [Oku13a], [Met14a], and [Sol16a] used images of skin lesions in determining whether the skin lesion is malignant or benign.

As much of the research focus on processing images of skin lesions has been on detecting malignancy, less have focused on classification besides the malignancy of a skin lesion. For one, [Ari12a] presented an automated dermatological diagnostic system that can classify skin lesion images into acne, eczema, psoriasis, tinea corporis, scabies, or vitiligo. On the other hand, [Yas14a] proposed the use of computer vision-based techniques to distinguish a skin lesion image as either acne, eczema, psoriasis, tinea corporis, scabies, vitiligo, foot ulcer, leprosy, or pityriasis rosea. It can be noted that one of the major differences between the research done by [Ari12a] and [Yas14a] is that the latter has a larger set of skin diseases as compared to the former. As much of the focus has been directly classifying from images to specific skin diseases, there is a research opportunity in shifting the focus of the classification problem.

Because of the vast amount of skin diseases that exist, another higher level categorization scheme that is inclusive of most, if not all, skin lesions may prove to be a useful and novel approach to the problem. Moreover, [Jam11a] wrote that the identification of a skin lesion's morphology is the first step in diagnosis. These along with the fact that morphology is indicative of a skin lesions' structure and appearance, makes morphology a proper means of categorizing skin lesions.

Morphology can be divided into primary and secondary. Primary morphology refers to the characteristic appearance of a skin lesion while secondary morphology refers to the temporal changes and modifications that occur on the skin lesion [Bol12a]. Only the primary morphologies will be explored in this research, as by

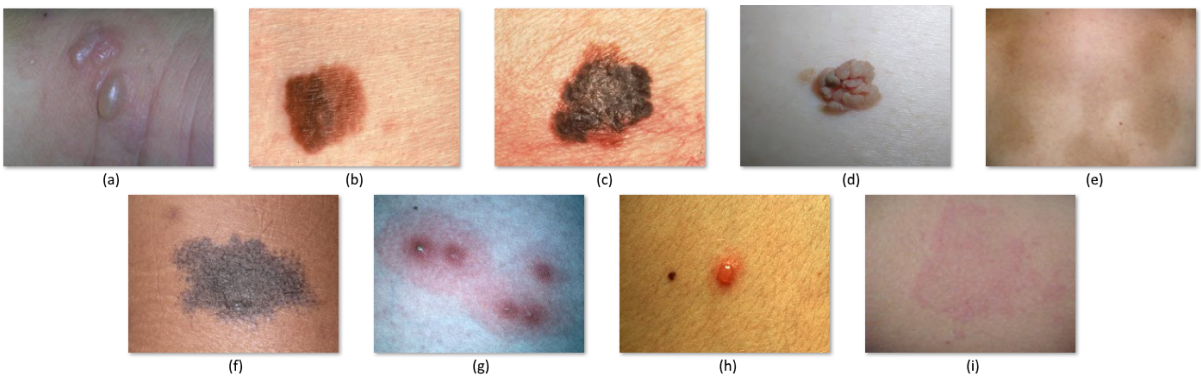


Figure 1: Examples of primary skin lesions: (a) Bulla (b) Macule (c) Nodule (d) Papule (e) Patch (f) Plaque, (g) Pustule, (h) Vesicle, and (i) Wheal.

definition they are more indicative of the skin lesion's core form and structure.

Thus, this research aims to explore classifying skin lesion images into primary morphologies. This is done by adapting techniques present in prior skin cancer malignancy and skin disease classification research to model a system that can classify images of skin lesions into the primary categorization of morphologies. As this research presents the novel research problem of classification by morphology, it sets the foundation for future work on further modeling of the diagnostic processes, or utilizing the higher level categorization to reduce complexity for classifying specific skin diseases.

2 RELATED WORK

Multiple research works have been done to create skin lesion classification systems. These works can be categorized, by their focus, into two groups. The first group, those focusing on skin malignancy classification, include studies determining whether a skin lesion is a malignant or benign, studies determining the severity of a skin cancer, and studies determining whether a skin cancer is a melanoma skin cancer or a non-melanoma skin cancer. The second group, those focusing on skin disease classification, include studies on classifying a skin lesion into a specific skin disease given a set of skin diseases. As of writing, there are no published studies focused on classifying skin lesions into their morphology.

2.1 Features and Feature Selection

There are a large variety of feature sets that have been defined in the literature. Most of the feature sets can be categorized into their focus. Primarily, features extracted from skin lesion images are those that describe the following elements of the skin lesion: color, texture, and shape.

Color is one common descriptor used in feature extraction for skin lesions. Commonly, an image can be expressed into a variety of color spaces. One example of this is to break down an image into the 3 color channels of the RGB color space. The resulting data can then be processed to extract features. In one study, [Yas14a] made use of the YCbCr color-encoding system in the extraction of color within a masked region.

Another crucial descriptor to skin lesions is their texture. Texture can be computationally expressed in a variety of ways [Mas13a]. One popular method is through the Grey Level Co-occurrence Matrix (GLCM). GLCM can be used to measure the spatial relationship between pixels, becoming a suitable means of extracting information on the texture of a skin lesion.

The shape of a skin lesion can be described in a variety of ways, such as area, size, and edge. [Yas14a] extracted features that quantify the area of a skin lesion and its apparent edges. [Yas14a] applied computations on the histogram and a masked region of the image for the area feature extraction, and applied the sobel operation on the masked region to extract the edges of the skin lesion.

In the study done by [Met14a], the Support Vector Machines Recursive Feature Elimination (SVMRFE) performed the best as compared to Information Gain and Correlation-based Feature Subset Selection. On the other hand, for [Mag09a], Correlation-based Feature Selection (CFS) performed the best as compared to using Principal Component Analysis (PCA) or Generalized Sequential Feature Selection (GSFS) for feature selection.

2.2 Classifiers

A lot of work has gone on the application of different classification methods in the context of skin lesions. A variety of methods can be applied to skin lesions: statistical classifiers, artificial neural networks, rule-based methods. [Yas14a] made use of a feed-forward back-propagation neural network in order to classify a skin lesion to one of 9 predefined skin diseases.

On the accuracy of such systems, many of the systems in the related works perform well in the detection and

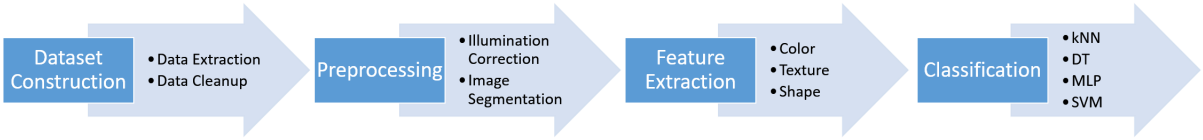


Figure 2: Process Flow

classification of skin cancer. For instance, [Yas14a] reports 91% - 97% accuracy for skin diseases that are low in elevation and 85% - 88% for skin diseases with high elevation. These studies served as the reference in selecting the various components of the classifier.

3 SKIN LESION MORPHOLOGIES

It is important to understand that morphology in this research is specifically in the context of dermatology. Morphology in dermatology is defined as the general appearance and structure of a particular skin lesion regardless of its function, etiology or pathophysiology [Bol12a]. Morphology can be further separated into primary and secondary morphology.

According to [Win86a], skin lesions can be grouped into two categories, primary and secondary morphologies. Primary morphologies differ in color or texture and are either acquired from birth, such as moles or birthmarks, or during a person's lifetime, as in the case of infectious diseases and allergic reactions. Secondary morphologies on the other hand are lesions that result from primary skin lesions, either as a natural progression or as a result of agitating the primary lesion. Because of this distinction, the categorization can be made to only consider primary morphologies.

Despite the vast amount of literature discussing skin lesions and morphology, different sources tend to list different sets of morphologies. For instance, the list presented by [Wol08a] is different from the one by [Pap04a], and by [Bic12a].

Regardless of these different listings, the description of each morphology is consistent amongst different references, so any of the references can be selected and utilized. For this study, a subset from the set of morphologies as listed by [Bic12a] and shown in Figure 1 is being used. This subset of morphologies was chosen to maximize the use of the data gathered for the research.

[Wel08a] provided the following brief description of the primary morphologies listed by [Bic12a]:

- Bulla - a fluid-filled circumscribed elevation of skin that is over 0.5 cm in diameter
- Macule - a small flat area with color or texture differing from surrounding skin
- Nodule - a solid mass in the skin that is palpated or elevated and is, in diameter of both width and depth, greater than 0.5 cm
- Papule - a solid elevation of skin that is less than 0.5 cm in diameter
- Patch - a large flat area with color or texture differing from surrounding skin
- Plaque - an elevated area of skin without substantial depth but is greater than 2 cm in diameter
- Pustule - an evident accumulation of pus in skin
- Vesicle - a fluid-filled circumscribed elevation of skin that is less than 0.5 cm in diameter
- Wheal - a white elevated compressible and faded area often surrounded by a red flare

As the descriptions for each morphology show, morphology in the dermatological sense is not only referring to shape but also referring to visual traits such as size, color, texture, and elevation. Furthermore, since no metric data will be utilized in this research, the system is limited in discriminating between morphologies. Another limitation is that this study focuses on skin lesions that are labelled as one morphology only, whereas cases wherein a particular skin lesion can be classified under multiple morphologies (e.g. a maculopapular rash belongs to both macule and papule) are not used in this research.

4 SKIN LESION CLASSIFIER

This section describes the concepts involved in the creation of a skin lesion classifier as outlined in Figure 2.

4.1 Preprocessing

Starting with an image, two major processes had to be made before proceeding with feature extraction: resolving nonuniform illumination and image segmentation.

Nonuniform illumination is the discrepancy between the images resulting from them being taken at different angles and lighting conditions, such as the example in Figure 3. For many instances, illumination normalization was enough to correct nonuniform illumination. However for other instances, Retinex [Zos13a] was used in tandem with segmentation. These methods simply refine the images and reduce nonuniform illumination rather than fix them completely.

Following the illumination refinements, the image will then be converted to the CIELab color space. After conversion, the k-means clustering algorithm was applied

on the a and b channels of the image in a CIELab color space to separate the image into two segments, as can be seen in Figure 4. Of the two segments, the one with the higher standard deviation was marked as the skin lesion, a step based on preliminary tests and observations. For a majority of cases, these steps were enough, whereas manual corrections had to be applied for the rest.

Once the two segments had been marked appropriately, the skin lesion segment was transformed into a mask. The mask was refined by filling holes, applying morphological opening, and removing small regions, a sample of which is at Figure 5. Lastly, this mask was then reapplied to the illumination refined image to get the properly divided skin lesion and healthy skin segments.

4.2 Features

After the image is properly segmented into normal skin and skin lesion, the segments undergo feature extraction. Many features extracted from skin lesion images covered in skin cancer and skin disease research revolve around three types: color, texture, or shape. Although many of these features were formulated for use in those specific research problems, there are features that could be adapted to classification by morphology.

4.2.1 Color

For color features, similar to [Sol16a], the common color features such as the mean and standard deviation of a value for each color channel in the RGB, HSV, and CIELab color spaces of the image were included. Additionally, by taking the mean color of the healthy skin, the difference between the mean color of the healthy skin and the mean color of the skin lesion can be computed and serve as a separate feature. These basic features, also used in this research, were previously utilized by works such as [Ari12a] and [Yas14a].

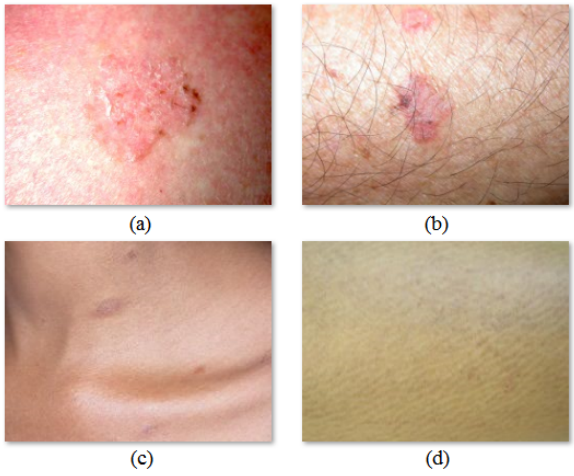


Figure 3: Issues in the dataset: (a) nonuniform illumination, (b) hair, (c) uneven skin surfaces, and (d) unclear skin lesion borders

Statistical measures such as uniformity or energy, and entropy applied to color channels also served as features, as done by [Met14a] for skin cancer classification. Both uniformity, computed through the equation

$$Uniformity = \sum_{k=0}^{M-1} p(k)^2 \quad (1)$$

and entropy, computed as

$$Entropy = \sum_{k=0}^{M-1} p(k)\log_2(p(k)) \quad (2)$$

define M as the number of bins of a distinct pixel value, and $p(k)$ as the probability associated with a specific color value k.

Moreover, the window-based color features of [Ari12a] were adapted to this research; in this case dropping the concentric circles scheme for a core, inner, and outer region division scheme.

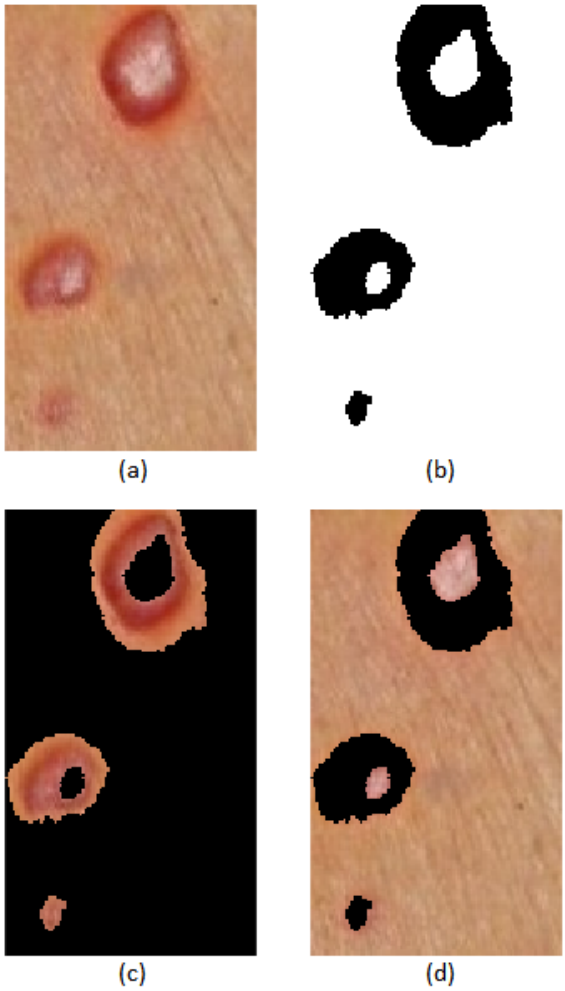


Figure 4: Segmentation steps: (a) Input, (b) K-means applied to a and b channels of image transformed to CIELab color space with k=2, (c) Skin Lesion, and (d) Healthy Skin

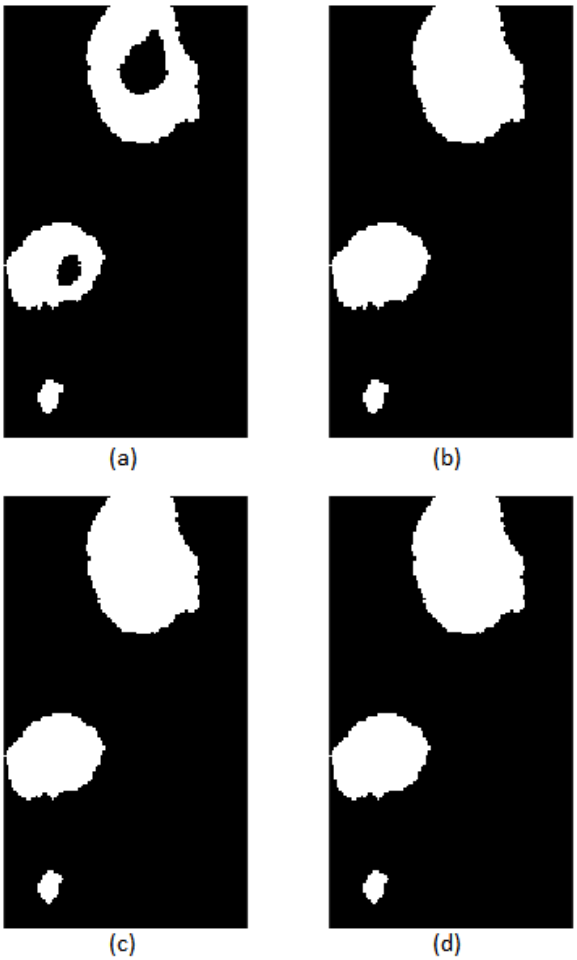


Figure 5: Mask refinements: (a) Input, (b) Filling holes, (c) Morphological opening, and (d) Removing small regions

Another feature, color asymmetry, was used by [Sma13a] for skin diseases through the chi-squared distance formula

$$D(h_1,h_2) = \sum_{i=1}^{N} \frac{(h_1-h_2)^2}{h_1+h_2} \tag{3}$$

wherein $D(h_1,h_2)$ is the chi-square distance of two histograms, N is the number of bins in histograms h_1 and h_2, and h_1 and h_2 are color histograms of opposite areas divided along a designated axis. This formula was applied over the two halves of an image as divided along either the minor or major axes; as was the feature designated for the relative distance of the intensity centroid and the mass centroid. In this research, the major and minor axes of a skin lesion is the same major and minor axes of an ellipse which has the same normalized second central moments as the skin lesion.

4.2.2 Texture

For texture features, many works have utilized various features available through the Gray-Level Co-occurence Matrix (GLCM). Generally, the common set which was selected for this research include energy, contrast, correlation and homogeneity. Energy is defined similar to how uniformity was treated in Equation 1, whereas the other three GLCM features are computed through the following equations

$$Contrast = \sum_{i=1,j=1}^{N} |i-j|^2 \, p(i,j) \tag{4}$$

$$Correlation = \sum_{i=1,j=1}^{N} \frac{(i-\mu i)(j-\mu j)p(i,j)}{\sigma_i \sigma_j} \tag{5}$$

$$Homogeneity = \sum_{i=1,j=1}^{N} \frac{p(i,j)}{1+|i-j|} \tag{6}$$

wherein i and j refer to the current column and row of the generated GLCM, μ is the mean of the current column or row, σ is the standard deviation of the current column or row, and $p(i,j)$ is the value found in the i-th column and j-th row of the GLCM.

Additionally, another set of features were derived from the Tamura texture features, which represent a visual description of texture based on human perception. As noted by [Cas02a], the first three features are enough for a metric on the general perception of texture. Moreover, as directionality is not visually emphasized in skin lesions and is more utilized in healthy skin, only contrast and coarseness were utilized for this research.

4.2.3 Shape

Shape features are not limited to the shape of the skin lesion, but also include how they are scattered and their number, as these are representative to how skin lesions present themselves. This means that the number of connected components serves as a feature as does distribution, as modified from [Ari12a] which is calculated through the formula

$$Distribution = \left(\frac{\sum_{i=1,j=1}^{N} dist_{ij}}{N} \right) \times \frac{1}{mmal} \tag{7}$$

wherein N is the number of regions, $dist_{ij}$ is the distance between the current region i and the current region j wherein both sets of regions refer to all skin lesions found, and *mmal* refers to the lowest measured minor axis length across all regions.

Moreover, [Ari12a] and [Paw14a] utilized features pertaining to the normalized area of the skin lesion, and the ratio of the min and max area, both of which were included for this research. The ratio of axis lengths and asymmetry along the major and minor axes as used by [Met14a] all served as features.

Data Source	Image Count
DermIS	6621
Global Skin Atlas	2551
DermAtlas	1114
Dermoscopy Atlas	394
Total	10680

Table 1: Raw dataset sources and image count

Morphology	Image Count
nodule	110
papule	108
plaque	93
macule	21
Total	332

Table 2: Final dataset size

For shape features focusing on the skin lesion border, features such as circularity, solidity, fractal dimension index and border solidity all served as varying means of measuring how defined, ragged, and uniform the borders appear to be [Met14a].

4.3 Feature Selection

Due to the large amount of features available, feature selection was also explored through two methods. Primarily, a reduced feature set was assembled based on preliminary tests and observations.

Alternatively, a genetic algorithm was also used to come up with a different reduced feature set. Genetic algorithms are based on the process of natural evolution. Generally, genetic algorithms use a heuristic to generate a solution to a specified problem.

4.4 Classification Methods

After feature extraction, different classifiers were then trained using a fraction of the generated dataset. Two of the classifiers chosen are standard algorithms utilized in machine learning, while the other two are based on what skin cancer and skin disease classification literature support.

4.4.1 K-Nearest Neighbors

The first classifier, K-Nearest Neighbor (kNN), is an algorithm that compares a given test sample described by a number of attributes to samples with similar attributes. Its name is derived from the fact that given a new sample, the algorithm searches the n-dimensional pattern space for k number of training samples similar to it based on a given distance metric. This algorithm is one of the simplest of the machine learning algorithms and yet still performs relatively well in particular classification problems.

4.4.2 Decision Trees

The second classifier, Decision Trees (DT), are inverted tree-like graphs that represents a prediction model for a target attribute given several input attributes. Each interior node of the tree corresponds to an input attribute, while the leaf nodes represents the predicted value of the target attribute upon traversal of the tree. Algorithms such as ID3 and C4.5 create decision trees based on attribute values of a dataset. In particular, these algorithms utilize information gain in order to designate the root node and recursively partition the values of the attributes into different nodes. It should also be noted that there is a chance that the algorithm may not be able to accommodate all data entries from the training data due to noise.

4.4.3 Multilayer Perceptron

According to [Ari12a], [Paw14a], and [Yas14a], Feed-Forward Back Propagation Artificial Neural Networks perform the best in classification problems on skin diseases. Artificial neural networks (ANN) consist of smaller processing units or neurons that are highly interconnected to form a computational model. Training involves having each neuron adjust their weights to accommodate the current input. Furthermore, ANN are popular in dermoscopic image analysis according to both [Mag09a] and [Mas13a]. A Multilayer Perceptron (MLP) is a type of artificial neural network that is an extension of perceptrons, and is said to be the simplest kind of feed-forward neural network.

4.4.4 Support Vector Machines

According to [Mag09a] and [Mas13a], Support Vector Machines (SVM) perform the best in classification problems concerning skin lesion malignancy. SVMs work based on statistical learning theory, essentially finding the optimal hyperplane between classes in a dataset through the resolution of an optimization problem.

5 EXPERIMENTS AND RESULTS

Given the flow outlined in Figure 2 and discussed in Section 3, specific experiments were designed and ran to evaluate the classifiers generated from the processes described.

5.1 Dataset and Experiment Setup

First, images and textual data were gathered from 4 data sources: DermIS [Hei03a], Global Skin Atlas [Aus05a], the Interactive Dermatology Atlas [Usa06a], and Dermoscopy Atlas [Aus07a]. After extraction, the data was combined based on the morphologies tagged and filtered further to images that were tagged with only one morphology belonging to the four utilized in this research. Furthermore, instances were also excluded from processing due to various issues such as uneven skin surfaces and unclear skin lesion borders,

Classifier	w/ Feat Select	accuracy	kappa
kNN	No	46.06%	0.215
	Yes	49.40%	0.264
DT	No	60.38%	0.427
	Yes	65.48%	0.501
MLP	No	67.17%	0.530
	Yes	65.14%	0.502

Table 3: Accuracy and Cohen's kappa for 4 Class dataset with or without genetic feature selection applied

Classifier	Feature Set	accuracy	kappa
KNN	Base	46.06%	0.215
	Reduced	56.10%	0.363
DT	Base	60.38%	0.427
	Reduced	65.63%	0.503
MLP	Base	67.17%	0.530
	Reduced	64.03%	0.486
SVM	Base	32.16%	-0.014
	Reduced	59.57%	0.408

Table 4: Accuracy and Cohen's kappa for 4 Class dataset with differing feature sets

such as those in Figure 3. Due to the aforementioned constraints, the number of instances in the dataset dwindled from 10,680 in Table 1 to 332 in Table 2.

For the configurations of the classifiers, preliminary tests were done to find the configurations that performed well for the classification problem. Firstly, the kNN classifier was set to use a weighted voting scheme with $k = 5$. For the DT, the criterion of accuracy was utilized and the classifier was set to have a maximal depth of 20, with pruning and pre-pruning active. The MLP utilized was set to have 10 maximum training cycles and run over 10 generations with 4 MLPs per ensemble. Lastly, A C-SVC type SVM with the rbf kernel was utilized.

5.2 Feature Selection Experiments

For this test, Table 3 shows the result of the genetic feature selection algorithm as trained using the SVM classifier. Alternatively, Table 4 shows the comparison between utilizing the full feature set or a manually reduced feature set. The reduction was done by removing features that correspond to any of the HSV and CIELab color space channels, leaving just color features based on intensity.

As can be observed from both Table 3 and 4, applying the genetic feature selection to the feature set did not affect the performance of the classifiers consistently. Moreover, the kNN classifier performed less as compared to both DT and MLP, which may be attributed to the still high dimensionality of the feature space even after many features had been filtered out.

On the contrary, utilizing the reduced feature set yielded similar if not better results than utilizing the whole feature set, with MLP being the exception. Furthermore, as the reduction on the feature set is high for this test, the improvements for the kNN and SVM classifiers is easily evident. As utilizing the reduced feature set allows for faster processing and utilizes a smaller feature set, it was the feature set utilized for the succeeding 4 class and 3 class tests.

An important insight is that although the reduced feature set provided better improvements than genetic feature selection, as Table 5 includes features relating to the removed HSV and CIELab colors pace channel features, these features not present in the reduced feature set may still hold considerable value.

5.3 4 Class

The 4 class test were done over the four specific classes, namely: macule, nodule, papule, and plaque.

As can be observed from Table 6, nodules and papules were the two classes that the system had the easiest time distinguishing. All of the classifiers seem to have difficulty with detecting skin lesions that are macules but did better with plaques. Given that the data is unbalanced in that there are fewer instances of macule than the other classes, this may have skewed the result against the class. For this test, DT and MLP performed better as compared to kNN and SVM.

5.4 3 Class

The 3 class tests were those that included only three classes: nodule, papule, and plaque. For this test, macule was removed as it contained a significantly fewer amount of cases as compared to the other three classes.

In Table 7, as compared to that of Table 6, the removal of the macule class showed improvements but certain trends remain. For instance, the classes of nodule and papule have higher f-measures than plaque which is still lagging behind. This may be attributed to the fact that papule and plaque are very similar. For instance, Figure 7 shows a set of plaque skin lesions tagged correctly and incorrectly which visually look to have the same texture and border definition. However, possibly due to the differences in their distribution and color, the samples on the left were misclassified.

Additionally, it must be noted that all three classes in the 3 class test are all categorically classified as raised

Distribution	Color(Inner - H)
Tamura Contrast	Color(Outer - I)
Contrast(Lesion)	Mean a(Lesion-Skin)
Uniformity b	Solidity
Circularity	Correlation(Lesion/Skin)
Mean b(Lesion-Skin)	Color(Inner - I)
Relative a Centroid	Color P. Asym.(Minor - I)

Table 5: Features in pruned decision tree based from genetic feature selection feature set

	macule	nodule	papule	plaque
kNN	0.00%	64.67%	69.54%	35.48%
DT	29.03%	71.77%	72.38%	55.11%
MLP	28.95%	72.22%	71.22%	52.91%
SVM	0.00%	64.74%	72.48%	43.83%

Table 6: F-measure for 4 Class test

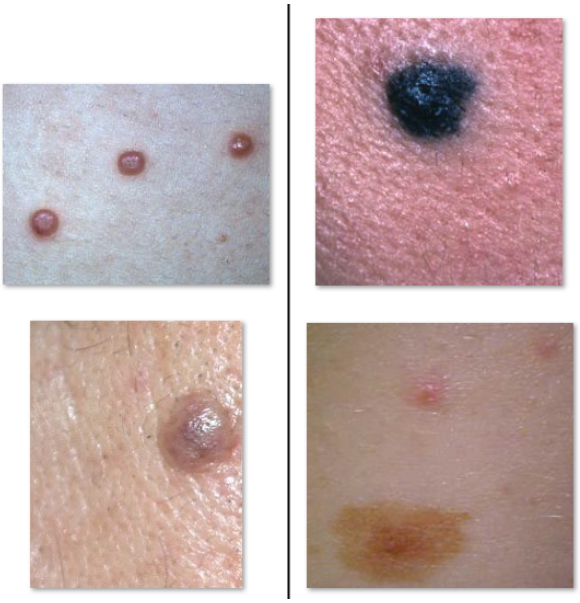


Figure 6: Sample images: (left) papules classified as nodules; (right) papules classified as papules

skin lesions (with macule being part of the flat morphologies) and so share many similarities with one another, making discrimination between them more difficult. For instance, Figures 6, 7, and 8 all show a series of nodules, papules, and plaques, all of which appear to be above the general elevation of skin. Based on the results shown in Table 7 for this test, DT performed the best, only slightly falling behind MLP in distinguishing nodules. Given Table 8, it can be inferred that based on both accuracy and Cohen's Kappa, the Decision Tree classifier is able to create the best classification model, a notion supported by all tests recorded.

	nodule	papule	plaque
kNN	68.58%	70.40%	43.12%
DT	77.80%	71.26%	60.50%
MLP	77.94%	70.87%	56.98%
SVM	69.48%	70.84%	42.27%

Table 7: F-measure for 3 Class test

	4 Classes		3 Classes	
	accuracy	kappa	accuracy	kappa
kNN	56.10%	0.363	62.23%	0.429
DT	65.63%	0.503	70.71%	0.558
MLP	64.03%	0.486	69.39%	0.539
SVM	59.57%	0.408	62.55%	0.434

Table 8: Accuracy and Cohen's kappa for 4 Class and 3 Class tests

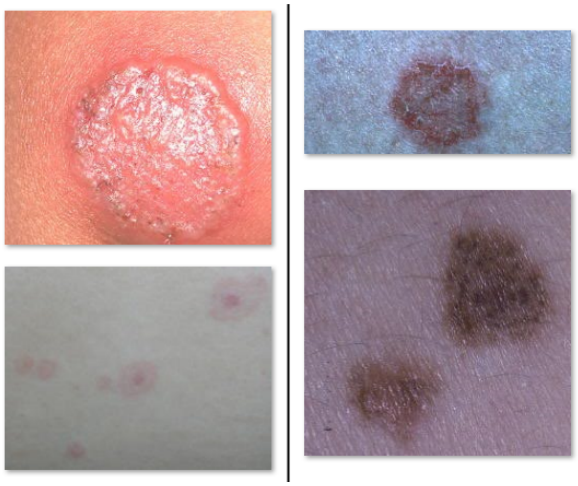


Figure 7: Sample images: (left) plaques classified as papules; (right) plaques classified as plaques

6 CONCLUSION

This research study was able to explore classifying skin lesion images into primary morphologies. Out of the four classifiers, Decision Trees performed best and thus is the recommended classifier both due to its resiliency and performance. For feature selection, genetic feature selection provides inconsistent results, and utilizing a reduced feature set showed an increase in performance in all except for MLP. In conclusion, the classification of skin lesion images by morphology is possible. However, more research is needed for significant use of a system based on the technology can be utilized, especially as the research area has not been thoroughly explored.

Further research can benefit from a larger and more consistent dataset, especially as the current dataset has many variations in lighting, scaling, and resolution. A new dataset built through strict guidelines may provide valuable insight into the deeper exploration of classification by morphology. Additionally, testing a new feature set, formulating new features specific to the research problem, and exploring features outside prior skin cancer and skin disease classification works may prove beneficial. Moreover, a multi-tier classification scheme may be possible given that morphologies can be grouped (e.g. papule, nodule, and plaque are all raised)

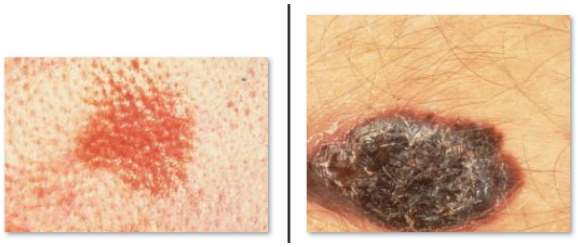


Figure 8: Sample images: (left) nodule classified as plaque; (right) nodule classified as nodule

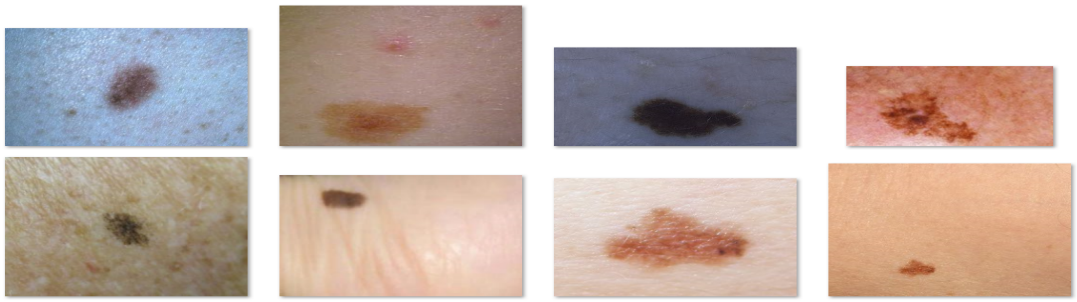


Figure 9: Sample macules correctly classified

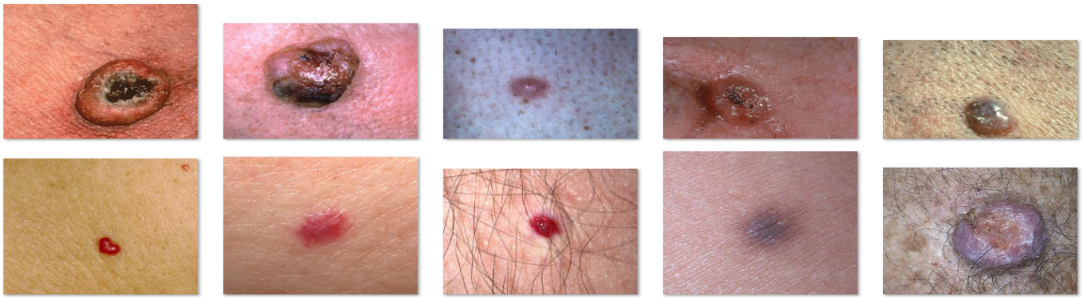


Figure 10: Sample nodules correctly classified

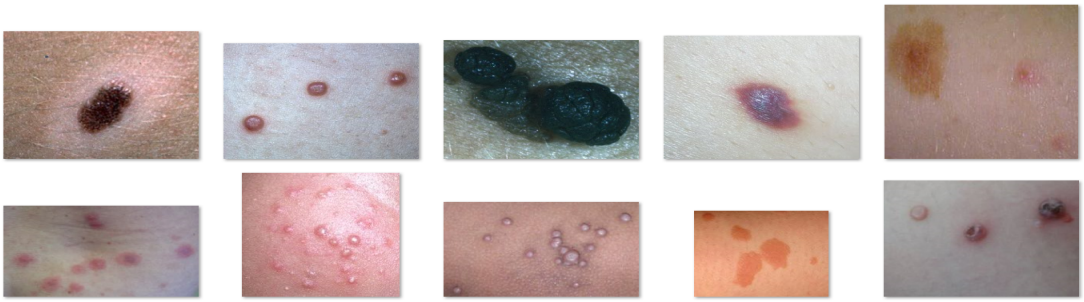


Figure 11: Sample papules correctly classified

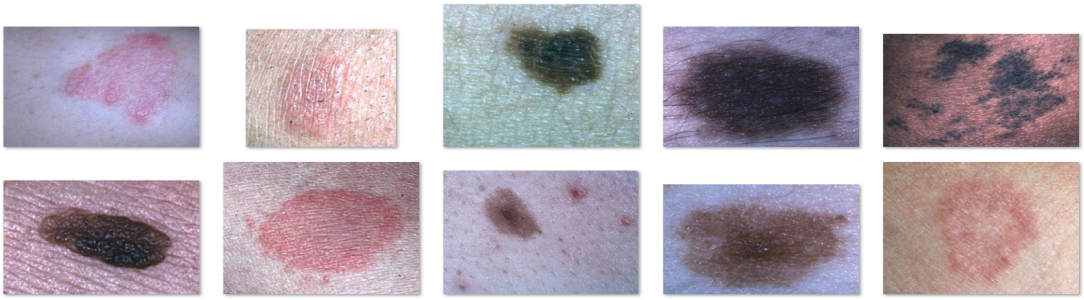


Figure 12: Sample plaques correctly classified

and that some pairs of morphologies are very similar to each other (e.g. papule and plaque are similar but differing in size). Also, further research can be made on multilabel classification to accommodate skin lesions that fall on multiple morphologies. Alternatively, the use of convolutional neural networks (CNNs) with regards to the research problem is also worth investigating. Lastly, further work can be on measuring the benefits of including the automatically tagged morphologies as a component in the classification of skin diseases.

7 ACKNOWLEDGMENTS

This research was made possible with funding from the University Research Coordination Office (URCO) of the De La Salle University. Many thanks to Dr. Arnulfo Azcarraga, Dr. Joel Ilao, Dr. Maria Franchesca Quinio and Dr. Erin Jane Tababa for their input, and to DermIS [Hei03a], Global Skin Atlas [Aus05a], the Interactive Dermatology Atlas [Usa06a], and Dermoscopy Atlas [Aus07a] for the dataset.

8 REFERENCES

[Ari12a] Arifin, M. S., Kibria, M. G., Firoze, A., Amini, M. A., & Yan, H. (2012). Dermatological disease diagnosis using color-skin images. In International Conference on Machine Learning and Cybernetics (ICMLC 2012), 5, 1675-1680.

[Aus05a] The Skin Cancer Society of Australia. (2005). Global Skin Atlas. Retrieved from http://www.globalskinatlas.com/index.cfm

[Aus07a] The Skin Cancer Society of Australia. (2007). Dermoscopy Atlas. Retrieved from http://www.dermoscopyatlas.com/index.cfm

[Bic12a] Bickley, L., & Szilagyi, P. G. (2012). Bates' guide to physical examination and history-taking. Philadelphia, PA: Lippincott Williams & Wilkins.

[Bol12a] Bolognia, J. L., Jorizzo, J. L., Schaffer, J. V., Cerroni, L., Heymann, W. R., & Callen, J. P. (2012). Dermatology (Vol. 2). New York, NY: Mosby.

[Cas02a] Castelli, V., & Bergman, L. D. (2002). Image Databases: Search and Retrieval of Digital Imagery. New York, NY: John Wiley & Sons.

[Hei03a] University of Heidelberg - Department of Clinical Social Medicine, & University of Erlangen - Department of Dermatology. (2003). DermIS - Dermatology Information System. Retrieved from http://www.dermis.net/

[Jam11a] James, W. D., Elston, D., & Berger, T. (2011). Andrew's diseases of the skin: clinical dermatology (11th ed.). London, UK: Saunders Elsevier .

[Mag09a] Maglogiannis, I., & Doukas, C. N. (2009). Overview of advanced computer vision systems for skin lesions characterization. IEEE transactions on information technology in biomedicine, 13(5), 721-733.

[Mas13a] Masood, A., & Ali Al-Jumaily, A. (2013). Computer aided diagnostic support system for skin cancer: a review of techniques and algorithms. International journal of biomedical imaging, 2013, 22.

[Met14a] Mete, M., & Sirakov, N. M. (2014). Optimal set of features for accurate skin cancer diagnosis. In 2014 IEEE International Conference on Image Processing (ICIP), 2256-2260.

[Oku13a] Okuboyejo, D., Olugbara, O., & Odunaike, S. (2013). Automating skin disease diagnosis using image classification. In Proceedings of the World Congress on Engineering and Computer Science, 2.

[Pap04a] Papier, A., Chalmers, R. J., Byrnes, J. A., & Goldsmith, L. A. (2004). Framework for improved communication: the Dermatology Lexicon Project. Journal of the American Academy of Dermatology, 50(4), 630-634.

[Paw14a] Pawar, M., Sharma, D. K., & Giri, R. N. (2014). Multiclass Skin Disease Classification Using Neural Network. International Journal of Computer Science and Information Technology Research. 2, 189-193.

[Sma13a] Smaoui, N., & Bessassi, S. (2013). A developed system for melanoma diagnosis. International Journal of Computer Vision and Signal Processing, 3(1), 10-17.

[Sol16a] Solomon, A. M., Murali, A., Sruthi, R. B., Sreekavya, M. K., Sasidharan, S., & Thomas, L. (2016). Identification of Skin Cancer based on Colour, Subregion and Texture. International Journal of Engineering Science, 8331

[Usa06a] Usatine, P., & Madden, B. (2006). DermAtlas - The Interactive Dermatology Atlas. Retrieved from http://www.dermatlas.net/index.cfm

[Wel08a] Weller, R., Hunter, J., Savin, J., & Dahl, M. (2008). Clinical Dermatology (4th ed.). Massachusetts, MA: Malden Publishing.

[Win86a] Winkelmann, R. K. (1986). Glossary of basic dermatology lesions. The International League of Dermatological Societies Committee on Nomenclature. Acta dermato-venereologica. Supplementum, 130, 1-16.

[Wol08a] Wolf, K., Goldsmith, L., Katz, S. I., Gilchrest, B., Paller, A. S., & Leffell, D. J. (2008). Fitzpatrick's Dermatology in General Medicine (7th ed.). USA: McGraw-Hill.

[Yas14a] Yasir, R., Rahman, M. A., & Ahmed, N. (2014). Dermatological disease detection using image processing and artificial neural network. In 2014 International Conference on Electrical and Computer Engineering (ICECE), 687-690.

[Zos13a] Zosso, D., Tran, G., & Osher, S. (2013). A unifying retinex model based on non-local differential operators. In International Society for Optics and Photonics IS&T/SPIE Electronic Imaging, 865702.

Predicting vehicle trajectories from surveillance video in a real scenario with Histogram of Oriented Gradient

Arthur Emidio T. Ferreira
University of Brasilia, Brazil
arthur.500@gmail.com

Bruno Luiggi M. Espinoza
University of Brasilia, Brazil
bruno@cic.unb.br

Flavio de Barros Vidal
University of Brasilia, Brazil
fbvidal@unb.br

Abstract

We propose a method capable to predict vehicle trajectories in a real scenario based on an unsupervised approach using Histogram of Oriented Gradients (HOG) features to construct an uniform path. The proposed algorithm extracts a sub-region of the input image defined as Field of View of the target vehicle, to output a possible trajectory that the given vehicle will follow through. We perform many experiments using the proposed technique, and based on qualitative/quantitative analyses, we conclude it is successfully able to predict reasonable trajectories.

Keywords

Histogram of Oriented Gradients, Path Prediction and Planning, Trajectory Forecasting.

1 INTRODUCTION

Consider the scene described in Figure 1. We, as humans, are able to predict the trajectory that the highlighted vehicle is likely to traverse in order to reach the goal indicated by the red circle. This ability is possible due to our capacity of using prior knowledge to forecast visual events [CBM12]. For instance, we may infer that in order to reach the destination, the vehicle won't collide with any other cars or pedestrians, nor have any contact with the sidewalk.

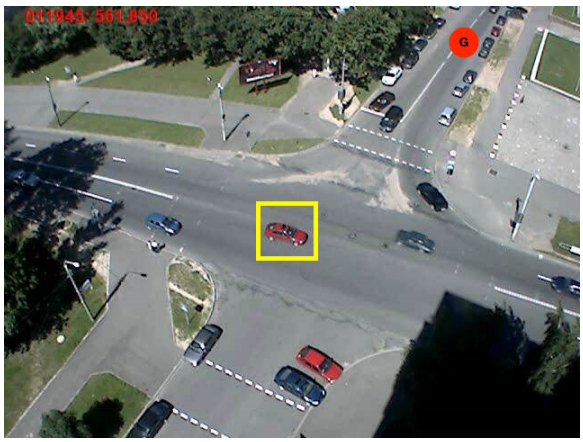


Figure 1: Given the highlighted vehicle and a goal point. What would be the trajectory traversed by the car before reaching the destination?

In computer vision, the topic of trajectory prediction has been explored in recent works, as presented in [WJM14, YY3, HSZ16], with the goal to forecast the trajectories of active elements in a static input image. A characteristic, that these works share in common, is the use of training data in order to give an output. In [HSZ16], for example, uses deep learning techniques to compute the most possible paths that an active agent is likely to traverse.

Trajectory forecasting is a very important topic in computer vision. For instance, predicting paths can improve the effectiveness of object tracking algorithms when dealing with significant occlusions [FBV16]. Moreover, prediction takes a very important role in scene understanding.

Is it possible to predict trajectories using a method that is not based on training data? In this work, we attempt to answer such question by proposing a framework that uses HOG features of the input image to compute future paths traversed by vehicles in a road.

The gradient of an image is an interesting approach to extract information about texture [ZZS14]. Based on previous knowledge, we know that vehicles tend to move on roads, a surface that is usually uniform in terms of texture. Therefore, we can compare HOG [DTB05] blocks to analyse different trajectories (sharing the same start and goal points) to determine which one is more suitable for a moving vehicle.

The previously mentioned works in trajectory forecasting propose generalized frameworks of prediction, meaning that it should be able to predict various types of active agents in different scenarios (e.g. pedestrians), based on what the training data consists of. In our proposed work, we restrict the domain of application because the characteristic of trajectories on uniform textured surfaces is inherent in vehicles moving on

a road. However, since our method is primarily based on HOG features, which are very robust to be computed, then the time and space efficiency can be greatly increased, meaning that our approach can provide prediction results significantly faster than other approaches, since ours doesn't rely on training.

This paper is organized as follows: Section 2 provides a brief background knowledge regarding event forecasting; Section 3 presents the proposed approach applied for path prediction; Section 4 presents the application and validation of the proposal in a specific vehicle path forecasting; and final remarks and future work envisioned are provided in Section 5.

2 RELATED WORKS

Prediction is an inherent human ability [SK10a], and has also been observed in several animal species [RW14], such as in Western scrub-jays [CSP07]. Event prediction is an important trait to comprehend and respond to the environment. Therefore, various recent works have been developed with the goal to bring such skill to the field of computer vision.

In computer vision, prediction has started to be explored in recent years for: i) tracking occluded objects [FHS10], ii) predicting missing frames or extrapolating future frames in a video [RMB14], iii) semantically forecasting the future contextual events that are likely to happen in unlabeled videos [VCP15], iv) predicting the future motion of individual pixels in a static image [MH16], v) anticipating human events so robots can better assist humans in daily activities [KS13], and vi) predicting the consequences of forces applied to objects in images [MR16].

We explore the topic of path prediction. Several techniques have been developed to predict the trajectories of active agents in a given scene. In the current literature, there are many works to predict: trajectories in egocentric videos [SKK16], the most likely trajectories of players in football games [LN16], predict trajectories performed by pedestrians [KKZ12, PS11], and more general approaches [WJM14, YY3, HSZ16].

The work done by [WJM14] consists of an unsupervised technique to predict the most possible trajectories that an active agent is likely to follow in a scene. The proposed method is based on the extraction of mid-level patches [SSG12] present in each frame. Then, it is created a transition matrix containing information about how each element can move or transition into another patch. It is also created a reward function for each element, which maps how likely a patch can move to any point in space. The problem of prediction is thus solved by modelling a graph, where each node is a state (i.e. a patch located in a 2-dimensional point in the image), and each edge corresponds to the transition from one state to another, weighted based on the information present in the transition matrix and the reward function. This turns into an maximization problem: find the sequence of states that maximizes the reward function, where the goal states are along the edges of the image. The author solves this problem using Dijkstra's algorithm [CTH2]. The work shows results of this technique using datasets of vehicles and pedestrians.

One limitation of the work of [WJM14] is that the proposed method does not take the movement of co-occurring elements into consideration. However, the work presented in [YY3] uses a Kanade-Lucas-Tomasi (KLT) trackers [SJ94] to obtain object trajectories, where each trajectory is converted into a quantized form. The work proposes an unsupervised Hierarchical Topic-Gaussian Mixture Model (HTGMM) that extracts semantical movement patterns (e.g. go straight, turn left) based on quantized trajectories. These patterns are divided into groups, such that all movement patterns inside a group may occur simultaneously. Based on this information, an energy potential map is created and iteratively updated, allowing to predict future trajectories considering the movement of other agents in scene.

Another recent work in the field of path prediction is presented in [HSZ16]. It proposes a framework that uses deep learning to predict future trajectories. The proposed method is based on two CNNs: a Spatial Matching Network and an Orientation Network. The Spatial Matching Network is responsible for generating a reward map of the scene, which is used to check if an agent is likely to reach a given region. The Orientation Network is responsible for estimating the agent orientation in order to predict the most possible direction that the agent is likely to pursue in the near future. Based on the information present in the reward map and in the estimated orientation, the technique of [HSZ16] uses a unified path planning scheme to predict future trajectories.

Our method is based on the technique of Histogram of Oriented Gradients (HOG) [DTB05]. The HOG technique has been commonly used with Support Vector Machines to perform human and object detection [BH2014]. HOG has also been used to assist on some previous prediction techniques. In the work of [WJM14], each extracted patch is considered to be a HOG cluster. The work of [KT10] uses HOG as one of the image representations to predict what could be observed if the camera changed its position. Here, we propose a technique that considers HOG as the main element of the path prediction process.

One similarity between the methods proposed in the works of [WJM14, YY3, HSZ16] is that they use a previously training stage in order to forecast future trajectories. One of the reasons for doing this is to make predictions more accurate in various types of scenar-

ios. Here, we propose a simpler framework to forecast vehicle trajectories. In this domain, we observe that vehicles tend to move in uniform regions (e.g. a vehicle usually maintains itself in a road area, which tends to be different in texture compared to a sidewalk, a pedestrian, or another vehicle). HOG can be a useful technique to find texture differences inside a small region. Therefore, we propose HOG to be used as a feature to detect regions that a vehicle is likely to move at. Additionally, the computational time of HOG features can be done in a very efficient manner, both in terms of time and space, therefore we propose a method that can be executed very close to real-time in today's hardware.

3 PROPOSED APPROACH

The objective behind our method is to predict the trajectory that a vehicle will follow to reach a given destination point using only the information from an image.

Therefore, we propose a framework (Figure 2) with the following input and output descriptions:

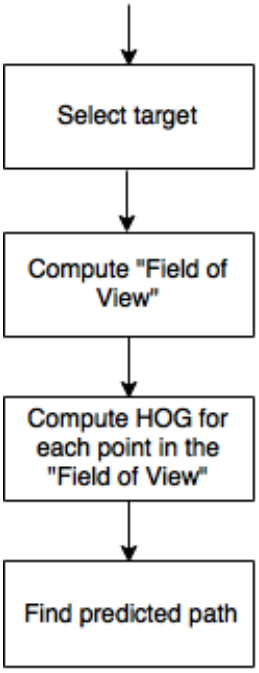


Figure 2: A flowchart describing the stages of the proposed approach.

Input: a small sequence of n sequential image frames $F = (f_1, f_2, ..., f_n)$ s.t. $n > 1$ taken by a stationary camera, the bounding rectangle R covering the vehicle that will have interest in its trajectory predicted in f_1, and a destination point D.

Output: a trajectory $T = (p_1, p_2, ..., p_m)$, where p_i is the ith two-dimensional point composing the path. T is the predicted trajectory for the input vehicle to reach the destination point D, based on frame f_1. The next four subsections explore each stage of the proposed algorithm and a visual representation of all these stages can be see in Figure 3.

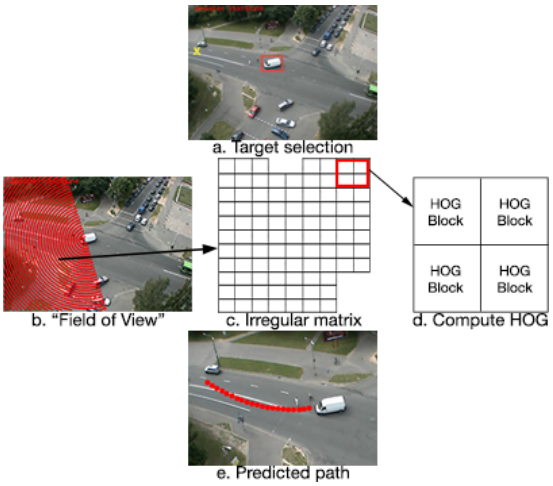


Figure 3: A visual representation of our proposed approach.

3.1 Target selection

The initial step of our approach is to provide the inputs from the image frame. First, we delimit a rectangle that covers the vehicle we are interested to have the trajectory predicted in frame f_1. Second, we mark the destination point D inside the boundaries of f_1.

3.2 Computing the "Field of View"

We denote the meaning of "Field of View" of a given vehicle as the set of all points that are located in the plane of 180^o created in terms of the orientation on which the vehicle is heading to. A visual representation of a field of view can be seen in Figure 3b.

We compute a "Field of View" to minimize the number of HOG cells to be computed in next step, with the goal of optimizing the average execution time of the algorithm.

The reason for using n input frames instead of only one is because we must infer in the selected vehicle's orientation. Therefore, we need a small number of n consecutive frames which should be enough to observe a noticeable motion of the vehicle between frames f_1 and f_n. Concurrently, n should not be too large, otherwise we could infer an incorrect vehicle's orientation.

Therefore, in order to find an orientation between frames f_1 and f_n, we must find the vehicle's location at frame f_n. The vehicle's location is achieved using template matching [FBV07] on frame f_2, having as a template image the *Region of Interest* (ROI) corresponding to the initial selection in the first step. Then, we update the template image, following the schema described in [FBV07], with the previous result

from template matching, and repeat the process until the vehicle is located at frame f_n.

Let M_i be the 2-dimensional point of the center of mass on the selected vehicle at frame f_i. Thus, we can compute the orientation of the vehicle between frames f_1 and f_n by calculating the slope m of the line connecting M_1 and M_n. Based on the slope, we can obtain the orientation angle θ.

From θ, we can compute 180 parallel lines starting at M_1 and reaching the edges of frame f_1, in terms of the vehicle's direction. All points composing these 180 lines are defined as the "Field of View" of the vehicle with center of mass M_1 in f_1.

3.3 Computing HOG for each cell in the "Field of View"

In order to use the field of view in our prediction scheme, we must discretize it as an irregular matrix, which each element in the matrix corresponds to a point in the field of view (see Figure 3c). We do this using Algorithm 1.

```
Input
     I: current frame.
     D: the distance between two neighboring
        nodes sharing the same line in the
        field of view.
    VP: the vehicle's initial position point.
EndInput.

Output
     M: the irregular matrix.
    DV: a matrix storing the HOG descriptor
        values for each element in M.
EndOutput.

Begin
    For each ith line L in the Field of View:
        CP := VP
        j := 1
        While CP is in frame:
            M[i][j] := CP
            DV[i][j] := compute HOG in a 32x32
                        ROI of I st. the central
                        point is M[i][j].
            CP := point P in L
                  s.t. dist(P, CP) = D
            j := j + 1
        EndWhile.
    EndFor.
End.
```

Algorithm 1: Algorithm to compute HOG for each cell in the field of view

The next step is to calculate the HOG features for each element in the irregular matrix (Figure 3d). We do this by extracting a ROI of size 32×32 pixels defining the central point of this region as the point stored in the matrix element. Then, we compute the HOG features of the ROI using a HOG block of the same size as the ROI. The resulting vector $DV_{i,j}$, containing the descriptor values for each element in the irregular matrix of row i and column j, must be saved for future use.

3.4 Predicting the path

The last step of our approach consists in transforming the irregular matrix into a directed graph, where each vertex corresponds to an element of the irregular matrix, being adjacent to all vertices that are in its 8-neighborhood region in the irregular matrix.

We assign a cost $c_{i,j}$ for every edge connecting the vertex v_i to v_j. In order to show how it is calculated, let H_x be the resulting HOG features vector for node v_x, which was obtained in the previous step. Consider the following equation:

$$|H_i - H_j| = \left| \begin{bmatrix} h_{i_1} \\ h_{i_2} \\ \vdots \\ h_{i_m} \end{bmatrix} - \begin{bmatrix} h_{j_1} \\ h_{j_2} \\ \vdots \\ h_{j_m} \end{bmatrix} \right| = \begin{bmatrix} |h_{i_1} - h_{j_1}| \\ |h_{i_2} - h_{j_2}| \\ \vdots \\ |h_{i_m} - h_{j_m}| \end{bmatrix} \quad (1)$$

Based on Equation 1, we define the cost $c_{i,j}$ in Equation 2.

$$c_{i,j} = 1^T |H_i - H_j| = |h_{i_1} - h_{j_1}| + \ldots + |h_{i_m} - h_{j_m}| \quad (2)$$

Finally, we use the $A*$ search algorithm to compute the predicted path of our proposed framework [PNR68]. However, we must first set the initial and goal vertices. Let the initial vertex be the one corresponding to the first element in the 90^{th} line of the field of view. Additionally, let the goal vertex be the one which is nearest to the destination point D which was selected as input in the first step of our approach.

The optimal path given by the $A*$ algorithm is the result of our prediction model, which can be seen in Figure 3e.

4 EXPERIMENTAL RESULTS

In order to demonstrate the effectiveness of our proposed methodology, we performed many experiments with several frames obtained from a image dataset, and compared the predicted trajectory with a ground truth achieved manually from the image sequence. We analysed the results in qualitative and quantitative manners. In the next subsections we describe the dataset, and present and discuss our results.

4.1 Dataset

For our experiments, we were interested in using a dataset that would contain several elements apart from vehicles and roads, such as pedestrians, sidewalks, trees, and different types of terrain (e.g. grass). We found all these elements of interest in the Minsk dataset [SNA14]. The Minsk dataset consists in a collection of video recordings from four different angles. In one of the angles, we are able to see a road intersection in aerial view. We used this subset as our primary experimental data presented in this work, given that crossroads present many possibilities of vehicle movements, and usually include more obstacles in scene.

4.2 Experiments

We present the result of two different input images from the Minsk dataset, which can be seen in Figures 4 to 9.

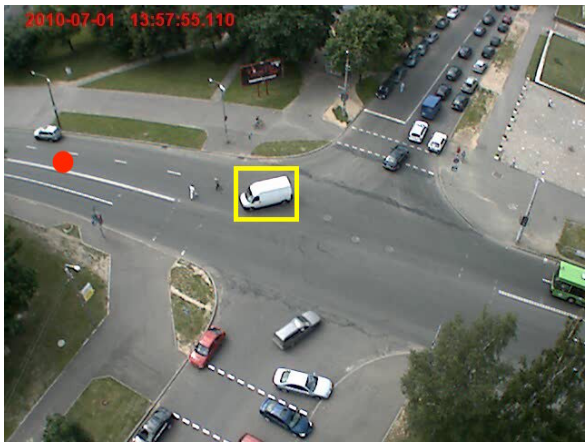


Figure 4: Exp. 1: The selected vehicle (in yellow) and its goal (in red).

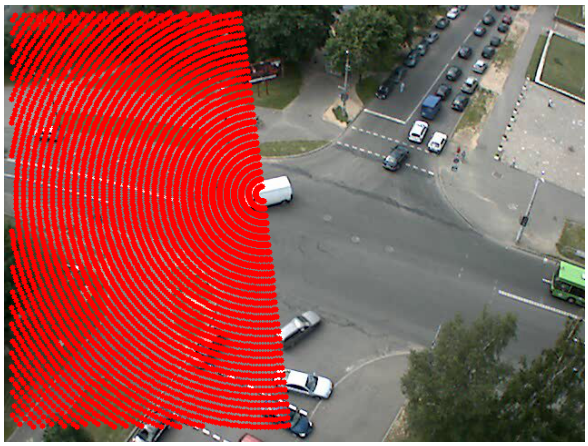


Figure 5: Exp. 1: The field of view of the selected vehicle.

For both experiments, we used the following parameters for the computation of HOG features:

- Block size: 16x16 pixels.

Figure 6: Exp. 1: The predicted trajectory of the selected vehicle.

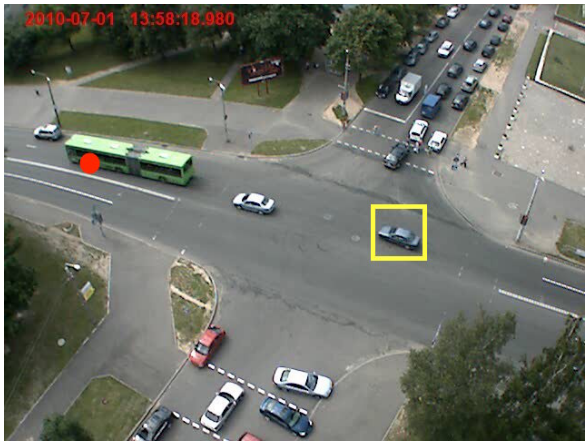


Figure 7: Exp. 2: The selected vehicle (in yellow) and its goal (in red).

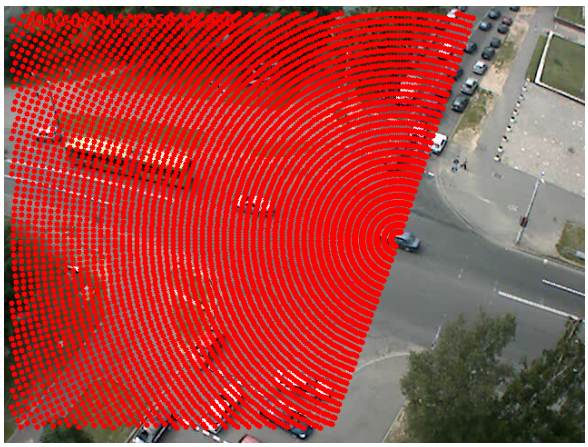


Figure 8: Exp. 2: The field of view of the selected vehicle.

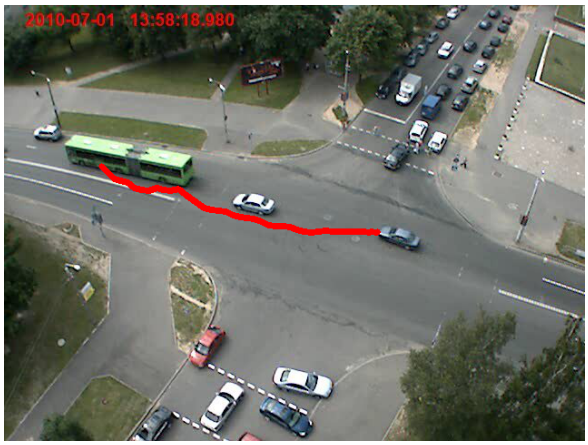


Figure 9: Exp. 2: The predicted trajectory of the selected vehicle.

- Cell size: 8x8 pixels.
- Number of bins: 9.
- Distance between blocks (in terms of its centroid): 8 pixels.

The Field of View from all experiments were obtained using a total of 5 (five) consecutive frames from the video sequence. Moreover, the goal from all experiments are the real vehicle's destination point, obtained from the ground truth. We choose two experiment scenarios, as described below.

4.3 Experiment 1

First, we can observe in Figure 5 that the field of view matches the direction that the vehicle is pointed towards to. Second, we can visualize the predicted trajectory in Figure 6. The path successfully avoids any contact with the pedestrians that are close to the selected vehicle. It is also observable that a considerable part of the trajectory remains in the road's center line, this can be explained due to the uniformity that this region tends to have, therefore, the absolute difference between HOG blocks in these parts of the frame are minimal.

4.4 Experiment 2

In this example, we select a goal that is farther away compared to the first experiment. Hence, the field of view, seen in Figure 8, is considerable larger, but still coinciding with the vehicle's direction. In the predicted trajectory, seen in Figure 6, we can observe that the vehicle is capable in avoiding a collision with the white car located in its front. Given that the goal was selected to be at where the bus is located, a collision is unavoidable. However, it can be seen that such collision only happens very close to the goal point.

After describing the experiments scenarios, for a quantitative analysis, we ran our technique for all frames that the selected vehicle is present in scene, and computed the average error between the predicted trajectory and the path obtained from the ground truth. We used the Equation 3 to compute the error for iteration i:

$$E_i = \frac{\sum_{j=1}^{n} minDist(B_j, GT)}{n}, \qquad (3)$$

where n is the number of points in the predicted trajectory, B_j is the jth point of the predicted path, GT is the set of points in the ground truth, and the function $minDist(x, Y)$ computes the euclidean distance from point x to the its nearest neighbor in the set of points Y.

We present the mean errors in Figures 10 and 11 using the selected vehicles from experiments 1 and 2.

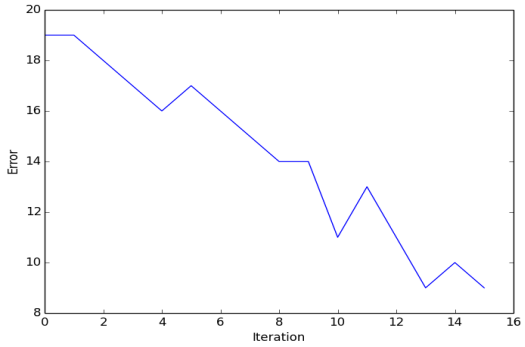


Figure 10: The errors calculated using the vehicle from Experiment 1.

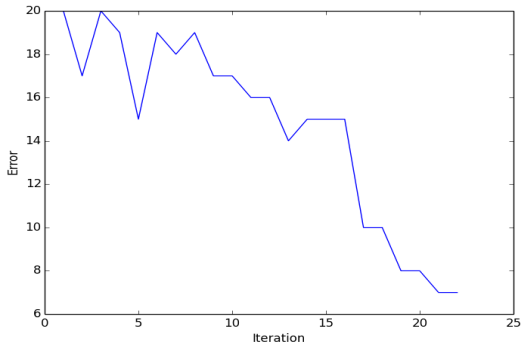


Figure 11: The average errors calculated using the vehicle from Experiment 2.

As we can observe in both plots, the mean error tends to decrease in each iteration. This can be explained due to the fact that the vehicle's field of view is smaller, and consequently the number of possible trajectories are decreased. Therefore, our method is expected to provide more accurate predictions when the vehicle is closer to its goal.

5 CONCLUSION

We have proposed a simple and efficient method to predict vehicle trajectories in a real scene using the Histogram of Oriented Gradients as the main feature and optimization algorithms. Based on the assumption that vehicles are likely to move in uniform regions (i.e. avoiding obstacles and different types of terrain), we have shown that the comparison between HOG descriptor vectors is an interesting approach to find differences in texture between small adjacent regions. We have demonstrated that our proposed framework provides satisfactory results with aerial view videos, being able to propose paths that don't intersect with additional vehicles, pedestrians, and other non-asphalt regions.

Additionally, given the time complexity $O(|E|)$ of the $A*$ search algorithm, and considering that the computation of HOG descriptors can be performed considerably fast in today's hardware, we have shown that the time/space efficiency of predicting vehicle trajectories is considerably positive and achievable for any computer architecture, including the mobile and embedded devices.

However, it is important to remark that our proposed method is an initial step in trajectory prediction in video. For future work, we plan to be able to predict trajectories of a vehicle by considering the existence of changes in the environment (e.g. movement of other vehicles and pedestrians). Additionally, we plan to use HOG as a feature in a trained model in order to observe whether we are able to forecast better trajectories. By using a trained model, we also plan to predict the vehicle's final position. Furthermore, since our goal is to predict paths executed by vehicles, it is very important to construct a framework that infers common traffic laws, such as recognizing whether a street only allows one-way traffic.

6 REFERENCES

[BH2014] Bristow, Hilton, and Lucey, Simon. 2014. *Why do linear SVMs trained on HOG features perform so well?* CoRR, abs/1406.2419.

[CBM12] Cheung, Olivia S, & Bar, Moshe. 2012. Visual prediction and perceptual expertise. *Int J Psychophysiol*, **83**(2), 156–63.

[CTH2] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. Dijkstra's algorithm. *Chap. 24, pages 595–599 of: Introduction to Algorithms 2nd edition*. MIT Press.

[CSP07] Correia, Sérgio P.C., Dickinson, Anthony, & Clayton, Nicola S. 2007. Western Scrub-Jays Anticipate Future Needs Independently of Their Current Motivational State. *Current Biology*, **17**(10), 856 – 861.

[DTB05] Dalal, N., & Triggs, B. 2005 (June). Histograms of oriented gradients for human detection. *Pages 886–893 vol. 1 of: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1.

[FBV16] de Barros Vidal, Flavio, Koike, Carla M. C. C., Cordoba, Diego A. L., & Zaghetto, Alexandre. 2014. Improving Visual Tracking Robustness in Cluttered and Occluded Environments using Particle Filter with Hybrid Resampling. *Pages 605–612 of: Proceedings of the 9th International Conference on Computer Vision Theory and Applications (VISIGRAPP 2014)*.

[FHS10] Fu, Hui-Xuan, Sun, Feng, & Liu, Sheng. 2010 (July). Anti-occlusion tracking algorithm based on LSSVM prediction and Kalman-MeanShift. *Pages 6031–6036 of: Intelligent Control and Automation (WCICA), 2010 8th World Congress on*.

[HSZ16] Huang, Siyu, Li, Xi, Zhang, Zhongfei (Mark), He, Zhouzhou, Wu, Fei, Liu, Wei, Tang, Jinhui, & Zhuang, Yueting. 2016. Deep Learning Driven Visual Path Prediction from a Single Image. *CoRR*, **abs/1601.07265**.

[KT10] Kaneva, B., Sivic, J., Torralba, A., Avidan, S., & Freeman, W. T. 2010. Matching and Predicting Street Level Images. *In: ECCV 2010 Workshop on Vision for Cognitive Tasks*.

[KKZ12] Kitani, Kris M., Ziebart, Brian D., Bagnell, James Andrew, & Hebert, Martial. 2012. *Activity Forecasting*. Berlin, Heidelberg: Springer Berlin Heidelberg. Pages 201–214.

[KS13] Koppula, H. S., & Saxena, A. 2013 (Nov). Anticipating human activities for reactive robotic response. *Pages 2071–2071 of: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[LN16] Lee, Namhoon, & Kitani, Kris M. 2016. Predicting wide receiver trajectories in American football. *Pages 1–9 of: 2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016, Lake Placid, NY, USA, March 7-10, 2016*.

[MR16] Mottaghi, Roozbeh, Rastegari, Mohammad, Gupta, Abhinav, & Farhadi, Ali. 2016. "What happens if..." Learning to Predict the Effect of Forces in Images. *CoRR*, **abs/1603.05600**.

[PNR68] P. E. Hart, N. J. Nilsson, & Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, **SSC-4**(2), 100–107.

[PS11] Pellegrini, Stefano, Ess, Andreas, & Van Gool, Luc. 2011. *Predicting Pedestrian Trajectories*. London: Springer London. Pages 473–491.

[RMB14] Ranzato, Marc'Aurelio, Szlam, Arthur, Bruna, Joan, Mathieu, Michaël, Collobert, Ronan, & Chopra, Sumit. 2014. Video (language) modeling: a baseline for generative models of natural videos. *CoRR*, **abs/1412.6604**.

[RW14] Roberts, William A. 2012. Evidence for future cognition in animals. *Learning and Motivation*, **43**(4), 169 – 180. Remembering the Future: The Influence of Past Experience on Future Behavior.

[SNA14] Saunier, Nicolas, Ardo, Hakan, Jodoin, Jean-Philippe, Nilsson, Aliaksei Laureshyn Mikael, Svensson, Åse, Miranda-Moreno, Luis, Bilodeau, Guillaume-Alexandre, & Astrom, Kalle. 2014. *A Public Video Dataset for Road Transportation Applications*.

[SJ94] Shi, Jianbo, & Tomasi, Carlo. 1994. Good Features to Track. *Pages 593 – 600 of: 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*.

[SKK16] Singh, Krishna Kumar, Fatahalian, Kayvon, & Efros, Alexei A. 2016. KrishnaCam: Using a longitudinal, single-person, egocentric dataset for scene understanding tasks. *Pages 1–9 of: 2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016, Lake Placid, NY, USA, March 7-10, 2016*.

[SSG12] Singh, Saurabh, Gupta, Abhinav, & Efros, Alexei A. 2012. Unsupervised Discovery of Mid-level Discriminative Patches. *In: European Conference on Computer Vision*.

[SK10a] Szpunar, Karl K. 2010. Episodic Future Thought: An Emerging Concept. *Perspectives on Psychological Science*, **5**(2), 142–162.

[FBV07] Vidal, F. B., & Alcalde, V. H. C. 2007 (June). Object Tracking by introducing Stochastic Filtering into Window-Matching Techniques. *Pages 31–36 of: 2007 International Symposium on Computational Intelligence in Robotics and Automation*.

[VCP15] Vondrick, Carl, Pirsiavash, Hamed, & Torralba, Antonio. 2015. Anticipating the future by watching unlabeled video. *CoRR*, **abs/1504.08023**.

[WJM14] Walker, Jacob, Gupta , Abhinav, & Hebert , Martial. 2014 (March). Patch to the Future: Unsupervised Visual Prediction. *In: Proc. Computer Vision and Pattern Recognition*.

[MH16] Walker, Jacob, Doersch, Carl, Gupta, Abhinav, & Hebert, Martial. 2016. An Uncertain Future: Forecasting from Static Images using Variational Autoencoders. *CoRR*, **abs/1606.07873**.

[YY3] Yoo, YoungJoon, Yun, Kimin, Yun, Sangdoo, Hong, JongHee, Jeong, Hawook, & Young Choi, Jin. 2016 (June). Visual Path Prediction in Complex Scenes With Crowded Moving Objects. *In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[ZZS14] Zuo, W., Zhang, L., Song, C., Zhang, D., & Gao, H. 2014. Gradient Histogram Estimation and Preservation for Texture Enhanced Image Denoising. *IEEE Transactions on Image Processing*, **23**(6), 2459–2472.

Speeding up the computation of uniform bicubic spline surfaces

Viliam Kačala
Institute of Computer Science,
Faculty of Science,
P. J. Šafárik University in Košice
Jesenná 5,
040 01 Košice, Slovakia
viliam.kacala@student.upjs.sk

Lukáš Miňo
Institute of Computer Science,
Faculty of Science,
P. J. Šafárik University in Košice
Jesenná 5,
040 01 Košice, Slovakia
lukas.mino@upjs.sk

ABSTRACT

Approximation of surfaces plays a key role in a wide variety of computer science fields such as graphics or CAD applications. Recently a new algorithm for evaluation of interpolating spline surfaces with C^2 continuity over uniform grids was proposed based on a special approximation property between biquartic and bicubic polynomials. The algorithm breaks down the classical de Boor's computational task to reduced tasks and simple remainder ones. The paper improves the reduced part's implementation, proposes an asymptotic equation to compute the theoretical speedup of the whole algorithm and provides results of computational experiments.
Both de Boor's and our reduced tasks involves tridiagonal linear systems. First of all, a memory-saving optimization is proposed for the solution of such equation systems. After setting the computational time complexity of arithmetic operations and clarifying the influence of modern microprocessors design on the algorithm's remainder tasks, a new expression is suggested for assessing theoretical speedup of the whole algorithm. Validity of the equation is then confirmed by measured speedup on various microprocessors.

Keywords

Spline interpolation, Bicubic spline, Hermite spline, Biquartic polynomial, Uniform grid, Tridiagonal systems, Speedup

1 INTRODUCTION

The paper is devoted to effective computation of tridiagonal systems. Since evaluation of such systems belongs to challenges of computer science and numerical mathematics, designing fast algorithms for their computation is a fundamental task. One of many applications of tridiagonal linear systems is construction of spline curves and surfaces that pass through the pre-set input points.

Our *reduced* algorithm is based on an interrelation between bicubic and biquartic polynomials that has been proved in [Sza16a], [Min16a] and its application was thoroughly described in [Min16a], [Min15a], [Min15b]. This interrelation was inspired by a similar property between cubic and quartic polynomials uncovered in [Tor14a]. A proof of this interrelation both in 2D and 3D is based on the IZA representation [Tor13a], [Sza13a] which incorporates both interpolation and approximation. The IZA representation was obtained using an r-point transformation that is a generalization of three point model introduced by Dikoussar [Dik97a]. A three point transformation was successfully applied to various approximation problems such as the assessment of unknown degrees in regression polynomials [Tor00a], [Mat05a] or a method for detecting piecewise cubic approximation segments for data with moderate errors [Dik06a].

An idea, based on which the IZA representation has been ultimately derived, appeared in [Rev07a]. The paper [Tor09a] showed how to properly use the IZA representation's reference points for segment connection and their relation to derivatives. Papers [Dik07a], [Sza13a] contain results on approximating 3D data utilizing the reference point approach. The basis for the quartic-cubic interrelation makes up a two-part model, which was first thoroughly studied in [Rev13a] and [Tor13a]. These works proved the validity of the two-part approximation model, which led first to approximation of a quartic polynomial by two cubic ones in [Tor14a] and then to approximation of a biquartic polynomial by two bicubic ones in [Min16a]. The reduced system approach to spline curve construction was proposed

in [TorTA] and afterwards it was generalized for case of spline surface construction in [Min15b]. The main goal of this work is both the theoretical and practical confirmation that the reduced algorithm for spline surfaces is faster than the de Boor's original algorithm which we refer to as *full algorithm*.

The structure of this article is as follows. Section 2 is devoted to a problem statement. To be self-contained, Section 3 briefly describes de Boor's algorithm and our recent algorithm based on reduced systems. Section 3.1 shows the standard way of solving tridiagonal linear systems and proposes a modified approach to solve such systems with lesser memory requirements. The next section analyses the architecture of microprocessors in search of a way to speedup the algorithms. The assessed speedup stated in Section 5 is confirmed by real-world measurements summarized in Section 6.

2 PROBLEM STATEMENT

This section defines inputs for the spline surface and requirements, based on which it can be constructed.

Consider a uniform grid

$$[u_0, u_1, \ldots, u_{I-1}] \times [v_0, v_1, \ldots, v_{J-1}], \tag{1}$$

where

$$u_i = u_0 + ih_x, \quad i = 1,2,\ldots,I-1, \quad I = 2m+1, m \in \mathbb{N},$$
$$v_j = v_0 + jh_y, \quad j = 1,2,\ldots,J-1, \quad J = 2n+1, n \in \mathbb{N}.$$

According to [Boo62a], a spline surface is defined by given values

$$z_{i,j}, \quad i = 0,1,\ldots,I-1, \quad j = 0,1,\ldots,J-1 \tag{2}$$

at equispaced grid-points, and given first directional derivatives

$$d^x_{i,j}, \quad i = 0,I-1, \quad j = 0,1,\ldots,J-1 \tag{3}$$

at boundary verticals,

$$d^y_{i,j}, \quad i = 0,1,\ldots,I-1, \quad j = 0,J-1 \tag{4}$$

at boundary horizontals and cross derivatives

$$d^{x,y}_{i,j}, \quad i = 0,I-1, \quad j = 0,J-1 \tag{5}$$

at four corners of the grid.

The task is to define a quadruple $[z_{i,j}, d^x_{i,j}, d^y_{i,j}, d^{x,y}_{i,j}]$ at every grid-point $[u_i, v_j]$, based on which a uniform bicubic clamped spline surface S of class C^2 can be constructed with properties

$$S(u_i,v_j) = z_{i,j}, \qquad \frac{\partial S(u_i,v_j)}{\partial y} = d^y_{i,j},$$
$$\frac{\partial S(u_i,v_j)}{\partial x} = d^x_{i,j}, \qquad \frac{\partial^2 S(u_i,v_j)}{\partial x \partial y} = d^{x,y}_{i,j},$$

where the adjacent spline segments are twice continuously differentiable.

3 FULL AND REDUCED ALGORITHMS

This section is devoted to the description of two algorithms for computing the unknown first derivatives of a C^2-class uniform spline surface's unknown first derivatives. The classic de Boor's algorithm is based on solving tridiagonal linear systems of equations that are further described in [Boo62a]. Henceforward we will refer to the de Boor's algorithm as the *full* algorithm. The recently proposed *reduced* algorithm based on the special approximation property between biquartic and bicubic polynomials breaks down the classical de Boor's computational task to reduced tasks and simple remainder ones as proposed in [Min15a], [Min15b]. The central part of the reduced algorithm comprises three new model equations and five new explicit formulas. Cross derivatives at the boundaries are computed using the classic de Boor's approach. Both algorithms are described in Appendix, thus the paper is self contained and the reader can count the number of mathematical operations for precise comparison.

3.1 Tridiagonal LU factorization

The standard way of solving tridiagonal linear systems

$$\underbrace{\begin{bmatrix} b_1 & 1 & 0 & & & \\ 1 & b_2 & 1 & & & \\ 0 & 1 & b_3 & & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & & & & b_K \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_K \end{bmatrix}}_{\boldsymbol{d}} = \underbrace{\begin{bmatrix} r_1 - d_0 \\ r_2 \\ r_3 \\ \vdots \\ r_K - d_{K+1} \end{bmatrix}}_{\boldsymbol{r}}$$

uses the LU factorization $\boldsymbol{A}\boldsymbol{d} = \boldsymbol{L}\underbrace{\boldsymbol{U}\boldsymbol{d}}_{\boldsymbol{y}} = \boldsymbol{r}$, where

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & & & \\ l_2 & 1 & & & \\ 0 & l_3 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots \\ & & & l_K & 1 \end{bmatrix},$$

$$\boldsymbol{U} = \begin{bmatrix} u_1 & 1 & 0 & & \\ 0 & u_1 & 1 & & \\ & & u_2 & & \\ & & \ddots & \ddots & \ddots \\ & & & & u_K \end{bmatrix},$$

the u_i and l_i elements are computed as, see [Bjo15a],

$$\boldsymbol{LU}: \quad u_1 = b, \{l_i = \frac{1}{u_{i-1}}, u_i = b - l_i\}, i = 2,...,K, \tag{6}$$

and the forward (Fw) and backward (Bw) steps of the solution are

$$\text{Forward:} \quad \boldsymbol{L}\boldsymbol{y} = \boldsymbol{r}, \tag{7}$$

where $y_1 = r_1$, $\{y_i = r_i - l_i y_{i-1}\}$, $i = 2,\ldots,K$;

$$\text{Backward:} \quad \boldsymbol{U}\boldsymbol{d} = \boldsymbol{y}, \tag{8}$$

where $d_K = \frac{y_K}{u_K}$, $\{d_i = \frac{1}{u_i}(y_i - d_{i+1})\}$, $i = K-1, \ldots, 1$.

```
1:  Input: b, r[1..K]
2:  Output: d[1..K]
3:  l[2..K]
4:  u[1..K]
5:  y[1..K]
6:  u[2] ← b
7:  y[1] ← r[1]
8:  for i from 2 to K do
9:      l[i] ← 1/u[i−1]
10:     u[i] ← b − l[i]
11:     y[i] ← r[i] − l[i]·y[i−1]
12: d[K] ← y[K]/u[K]
13: for i from K − 1 downto 1 do
14:     d[i] ← (y[i] − d[i+1])/u[i]
```

Algorithm 1: LU factorization

```
1:  Input: b, r[1..K]
2:  Output: r[1..K]
3:  p[1..K]
4:  m ← 1/b
5:  p[1] ← m
6:  r[1] ← m·r[1]
7:  for i from 2 to K − 1 do
8:      m ← 1/(b − p[i−1])
9:      p[i] ← m
10:     r[i] ← m·(r[i] − r[i−1])
11: m ← 1/(b − p[K])
12: p[K] ← m
13: r[K] ← m·(r[K] − r[K−1])
14: for i from K − 1 downto 1 do
15:     r[i] ← r[i] − p[i]·r[i+1]
```

Algorithm 2: LU factorization

Tridiagonal systems of equations for full and reduced algorithms are solved by LU factorization. All the systems of these algorithms are diagonally dominant with elements 1, 4, 1 or 1, −14, 1.

The process of computing a tridiagonal system is indicated in Algorithm 1. Since $b_1 = b_2 = \cdots = b_{T-1}$, where $T = K$ in case of the full algorithm or $T = K-1$ in case of the reduced algorithm, this method can been improved, see Algorithm 2, requiring less memory as stated in the lemma below.

Lemma 1 *Let K be the number of equations in a linear tridiagonal system with constant diagonals. Then Algorithms 1 and 2 require 5K and 2K of memory space, respectively.*

4 MICROPROCESSOR'S DESIGN INFLUENCE

In Section 3 we described the full and reduced algorithms for computing the unknown derivatives of spline surfaces. Their time complexity is $O(IJ)$. When determining the asymptotic time complexity it is common to ignore the speed of the algorithm's individual steps, arithmetic operations, etc. Since the asymptotic time complexity is equal for both aforementioned algorithms, it is vital to consider the influence of their individual steps.

One should also keep in mind that larger numbers of operations don't necessarily mean slower completion times as computation time also depends on the type of performed operations. In case of floating point operations it holds that additions and multiplications are similarly fast, but divisions are multiple times slower, see Table 1 and Table 2. In this section we briefly discuss some technical principles how modern CPUs work with data and how one can utilize this in implementation of algorithms. Results of computational experiments are presented in the last section.

Nowadays a performance increase cannot be achieved by just increasing the clock speed. The architectures of modern CPUs use other ways to improve performance, such as superscalar designs, pipelined instructions or thread parallelism.

4.1 Caching

One of the most important ways for a programmer to optimize the algorithm's implementation is the choice of proper data structures. To store input and output values of the two suggested algorithms we use matrices. A matrix can be represented as a jagged array or as a single continuous array where the element of the i-th row and j-th column has an index $n \cdot i + j$, where n is the number of columns.

The main system memory is slower than the CPU which has to wait tens or hundreds of machine cycles to load a value from the main memory. To address this latency issue, modern microprocessors are equipped with small and fast caches that preloads both data and machine instructions of programs from the main memory. Caching is an automated process controlled by the CPU [Pat15a].

Consider an $m \times n$ matrix represented by jagged arrays. When element $a_{0,0}$ is loaded from the matrix, the microprocessor might also cache elements $a_{0,1}$, $a_{0,2}$, …, but not element $a_{1,0}$. Therefore an evaluation of $a_{0,0} + a_{0,1}$ will be faster than the one of $a_{0,0} + a_{1,0}$.

In our case the jagged array representation proved to be more effective as most operations are evaluated on rows and so we do not need to cache the entire matrix which can be unfeasible considering large values of m and n.

4.2 Evaluation time of arithmetic operations

Microprocessor cores consist of several execution units specialized in different types of operations with varied instruction latencies. Values of latencies and throughputs can be found in CPU micro-architecture documentations. The x86 instruction set has many extensions and because it is not practical to include all instruction sets to this test, we chose the most commonly used instruction set extension, namely the Streaming SIMD Extensions 2 (SSE2) as all 64 bit x86 micro-architectures supports these. The sets got a more modern replacement in Advanced Vector Extensions (AVX) whose main advantage lies in the improved vector operations. However vector operations require independent calculations on each particular vector element [Pat15a] and this isn't the case of the considered full and reduced algorithms.

The speedup measurement of the reduced algorithm compared to the full one was conducted on five x86 CPUs covering the generations from AMD K10 to Intel Skylake. In the Table 1 we present four basic arithmetic instruction speeds on these four micro-architectures. The first column contains the name of the architecture and the year of its release to the market. The architectures are ordered alphabetically by the manufacturer and then by the year of release. Instruction **latency** is the number of CPU clocks it takes for an instruction to have its data available. Instruction **throughput** is the number of CPU clocks it takes for an instruction to execute. Some instructions have greater latency than throughput, meaning that the execution unit can process another instruction before the data from a current one are available for further processing. This is referred as pipelining which is one form of the instruction parallelism. The table confirms the expectation that addition and subtraction are equally fast. Therefore these operations will be jointly denoted as $\pm$. Hereafter when we mention the operation of addition we are meaning the subtraction as well. It is clear from the table that division is the slowest operation.

	Latency/Throughput			
Architecture (year)	+	−	×	÷
AMD K10.5 Llano (2011)	4/1	4/1	4/1	20/15
Intel Westmere (2010)	3/1	3/1	5/1	7-22/7-22
Intel Sandy Bridge (2011)	3/1	3/1	5/1	16-22/22
Intel Haswell (2013)	3/1	3/1	5/0,5	14-20/13
Intel Skylake (2015)	3/1	4/0,5	3/0,5	14/4

Table 1: Number of machine cycles for SSE2 double precision floating point arithmetic operations on different x86 generations by [Int16a], [Amd11a] and [Fog16a].

In Table 2 operations were measured in an array containing 512 random elements with the calculations repeated 500 000 times. The last two columns represent measured time ratio of multiplication to addition and ratio of division to addition. For the last two columns we define the following notations:

Definition 1

- *Value $\gamma^{\times}$ is the execution time ratio between multiplication and addition. It means the performance of one multiplication is equivalent to $\gamma^{\times}$ additions.*
- *Value $\gamma^{\div}$ is the execution time ratio between division and addition. It means the performance of one division is equivalent to $\gamma^{\div}$ additions.*

Operations were in the form of $a[i] = a[i] \circ a[i-1]$, where $\circ \in \{\pm, \times, \div\}$ to simulate the form of calculations in Algorithm 2.

A reason to perform measurements instead to rely on processor documentation is the fact, that given latencies and throughput for division of some microarchitectures depends on the input values which are not usually described in documentations. The rows of Table 2 corresponds to the rows of Table 1 but instead of architecture they indicate names of concrete microprocessors.

CPU	$\pm$	×	÷	$\gamma^{\times}$	$\gamma^{\div}$
A6-3420M	368	336	1747	0.91	5.20
Core i5 430M	253	347	544	1.37	2.15
Core i3 2350M	227	341	907	1.50	4.00
Core i7 4790	144	207	488	1.44	3.39
Core i7 6700K	135	136	422	1.01	3.13

Table 2: The speed of arithmetic operations on specific CPUs measured in milliseconds.

Comparing the second and fourth columns of Table 1 with the second and third columns of Table 2 we can say that addition and multiplication are similarly fast.

4.3 Parallelism of arithmetic operations

It remains to emphasize another property of the microprocessor's architecture called the *instruction level parallelism* (ILP). Modern processors are pipelined, superscalar and support vectorized computations as we briefly mentioned in the part 4.2. While the vectorization is not the concern for us due to form of Algorithm 2, the superscalar pipelined nature of modern CPU's is to be considered.

Consider the equations in Lemmas 3 and 4 in Section 9. The right-hand sides of the equation contain more than one arithmetic operation. Such expressions are broken automatically into more mutually independent subexpressions and evaluated automatically in parallel [Pat15a].

Table 3 shows how the increase in number of operations will extend the calculation time. For example from the second column in Table 3 it follows that evaluating

$a[i] = a[i] + a[i-1] + a[i-2]$ will be 1.61 times slower than $a[i] = a[i] + a[i-1]$. Operations were measured in an array containing 512 elements with the calculations repeated 500 000 times. Table 4 then shows similar values also for other CPU's, but for the sake of readability only for expressions containing ten numeric operators.

Num. of ops.	±	×	÷
2	1.61	1.61	2.00
3	2.15	2.16	3.00
4	2.59	2.6	4.01
5	3.03	3.04	5.01
6	3.43	3.45	6.02
7	3.85	3.86	7.02
8	4.25	4.27	8.03
9	4.68	4.69	9.03
10	5.09	5.11	10.04

Table 3: Evaluation times of arithmetical operation on Intel Core i7 6700K depending on the number of operations.

The results of this section will be used in Section 6.

CPU	±	×	÷	$\beta^{\pm}$	$\beta^{\times}$
A6-3420M	6.56	7.19	2.40	1.52	1.39
i5 430M	3.84	5.31	10.03	2.60	1.88
i3 2350M	5.10	5.93	9.99	1.96	1.67
i7 4790	6.43	6.17	10.00	1.56	1.62
i7 6700K	5.09	5.11	10.04	1.96	1.96

Table 4: Evaluation times multiples for mathematical expressions containing ten operations of said type compared to those expressions containing only a single operation.

Following the results of the Tables 3 and 4 we define the following notation:

Definition 2

- *Value $\beta^{\pm}$ denotes performance effect of instruction level parallelism on expressions containing more than one addition or subtraction. It means that an expression containing enough number of said operations will be evaluated in $\frac{1}{\beta^{\pm}}$ time compared to a CPU without such a feature.*
- *Analogous, the value $\beta^{\times}$ denotes performance effect of instruction level parallelism on expressions containing more than one multiplication.*

Remark 1 *Since both considered algorithms doesn't contain expressions with more than one floating point division, value $\beta^{\div}$ is not necessary.*

5 THEORETICAL SPEEDUP OF THE ALGORITHM

In this section we count the number of operations and their cost for full and reduced algorithms and provide our main result about the speedup of the latter. In Table 5 we have the cost of arithmetic operations for LU factorization of tridiagonal systems covering both de Boor's and the reduced algorithms. In summary we have the cost of operations for solving one tridiagonal system with K being the size of a matrix. The cost is defined as the sum of arithmetic operations where values for multiplications and divisions are multiplied by their execution time ratios of $\gamma^{\times}$ or $\gamma^{\div}$ respectively. The expressions containing more than one addition or multiplication operand were also multiplied by $\frac{1}{\beta^{\pm}}$ or $\frac{1}{\beta^{\times}}$ to accommodate the ILP effect on such expressions.

	Expression	±	×	÷
Full	LU (6) + Fw (7) + Bw (8)	$3K$	$2\gamma^{\times}K$	$\gamma^{\div}K$
	RHS (11), (12), (13), (14)	K	$\gamma^{\times}K$	0
	Summary full al.	$4K$	$3\gamma^{\times}K$	$\gamma^{\div}K$
Reduced	LU (6) + Fw (7) + Bw (8)	$3K$	$2\gamma^{\times}K$	$\gamma^{\div}K$
	RHS d^x, d^y (15), (17)	$\frac{3}{\beta^{\pm}}K$	$\frac{2}{\beta^{\times}}\gamma^{\times}K$	0
	Summary d^x, d^y	$3(1+\frac{1}{\beta^{\pm}})K$	$2(1+\frac{1}{\beta^{\times}})\gamma^{\times}K$	$\gamma^{\div}K$
	RHS $d^{x,y}$ (21)	$\frac{31}{\beta^{\pm}}K$	$\frac{17}{\beta^{\times}}\gamma^{\times}K$	0
	Summary $d^{x,y}$	$(3+\frac{31}{\beta^{\pm}})K$	$(2+\frac{17}{\beta^{\times}})\gamma^{\times}K$	$\gamma^{\div}K$

Table 5: Cost of operations for LU factorization of full and reduced algorithms in regards to the number of unknowns K.

Full	±	×	÷
d^x (11)	$4IJ$	$3\gamma^{\times}IJ$	$\gamma^{\div}IJ$
d^y (12)	$4IJ$	$3\gamma^{\times}IJ$	$\gamma^{\div}IJ$
$d^{x,y}$ (13)	$8I$	$6\gamma^{\times}I$	$2\gamma^{\div}I$
$d^{x,y}$ (14)	$4IJ$	$3\gamma^{\times}IJ$	$\gamma^{\div}IJ$
Summary	$12IJ$	$9\gamma^{\times}IJ$	$3\gamma^{\div}IJ$

Table 6: Cost of operations for the full algorithm.

Reduced	±	×	÷
d^x (15)	$\frac{3}{2}(1+\frac{1}{\beta^{\pm}})IJ$	$(1+\frac{1}{\beta^{\times}})\gamma^{\times}IJ$	$\frac{1}{2}\gamma^{\div}IJ$
d^x (16)	$\frac{3}{2\beta^{\pm}}IJ$	$\frac{1}{\beta^{\times}}\gamma^{\times}IJ$	0
d^y (17)	$\frac{3}{2}(1+\frac{1}{\beta^{\pm}})IJ$	$(1+\frac{1}{\beta^{\times}})\gamma^{\times}IJ$	$\frac{1}{2}\gamma^{\div}IJ$
d^y (18)	$\frac{3}{2\beta^{\pm}}IJ$	$\frac{1}{\beta^{\times}}\gamma^{\times}IJ$	0
$d^{x,y}$ (19), (20)	$8(I+J)$	$6\gamma^{\times}(I+J)$	$2\gamma^{\div}(I+J)$
$d^{x,y}$ (21)	$\frac{1}{4}(3+\frac{31}{\beta^{\pm}})IJ$	$\frac{1}{4}(2+\frac{17}{\beta^{\times}})\gamma^{\times}IJ$	$\frac{1}{4}\gamma^{\div}IJ$
$d^{x,y}$ (22), (23), (24)	$\frac{21}{4\beta^{\pm}}IJ$	$\frac{8}{4\beta^{\times}}\gamma^{\times}IJ$	0
Summary	$(\frac{15}{4}+\frac{19}{\beta^{\pm}})IJ$	$(\frac{5}{2}+\frac{41}{4\beta^{\times}})\gamma^{\times}IJ$	$\frac{5}{4}\gamma^{\div}IJ$

Table 7: Cost of operations for the reduced algorithm.

Remark 2 *Let us consider I being the number of gridpoints along the x-axis, J is the same along the y-axis. The cost of operations has the form $aIJ+bI+cJ+d$, but we provide the count for IJ only.*

Tables 6 and 7 show the cost of operations for individual steps and imply the following result.

Lemma 2 *Consider a uniform grid of size I and J. The total costs of operations are*

$$\left(12+9\gamma^{\times}+3\gamma^{\div}\right)IJ$$

for the full algorithm and

$$\left(\frac{15}{4}+\frac{19}{\beta^{\pm}}+\left(\frac{5}{2}+\frac{41}{4\beta^{\times}}\right)\gamma^{\times}+\frac{5}{4}\gamma^{\div}\right)IJ$$

for reduced algorithm.

We are ready to provide our main result about the speed increase achieved by the reduced algorithm in comparison with the classical full algorithm.

For measuring the expected speedup of the reduced algorithm with respect to the full one, the next theorem proposes an asymptotic expression.

Theorem 1 *Consider a uniform grid of size I and J. If $I,J \to \infty$, then the expected asymptotic speedup of the reduced algorithm is*

$$\frac{12+9\gamma^{\times}+3\gamma^{\div}}{\frac{15}{4}+\frac{19}{\beta^{\pm}}+\left(\frac{5}{2}+\frac{41}{4\beta^{\times}}\right)\gamma^{\times}+\frac{5}{4}\gamma^{\div}} \tag{9}$$

Proof. Consider a uniform grid of size I and J. From Lemma 2 we get the following costs of operations

- $(12+9\gamma^{\times}+3\gamma^{\div})IJ$ for the full algorithm,
- $\left(\frac{15}{4}+\frac{19}{\beta^{\pm}}+\left(\frac{5}{2}+\frac{41}{4\beta^{\times}}\right)\gamma^{\times}+\frac{5}{4}\gamma^{\div}\right)IJ$ for reduced algorithm.

The speedup is expressed as a cost ratio of both algorithms

$$\frac{(12+9\gamma^{\times}+3\gamma^{\div})IJ}{\left(\frac{15}{4}+\frac{19}{\beta^{\pm}}+\left(\frac{5}{2}+\frac{41}{4\beta^{\times}}\right)\gamma^{\times}+\frac{5}{4}\gamma^{\div}\right)IJ},$$

what completes the proof.

We underline that the asymptotic expression was derived in accordance with Remark 2.

6 MEASURED SPEEDUP

In the previous section, an asymptotic expression for the theoretical speedup has been derived. In this one we show the results of real measurements.

The tested data sets comprises of uniform grid $[u_0,u_1,\ldots,u_{2000}]\times[v_0,v_1,\ldots,v_{2000}]$ where $u_0=-20$, $u_{2000}=20$, $v_0=-20$, $v_{2000}=20$ and values $z_{i,j}$, $d^x_{i,j}$, $d^y_{i,j}$, $d^{x,y}_{i,j}$, see (2) – (5), are given from function $sin\sqrt{x^2+y^2}$ at equispaced grid-points. The speedup values were gained averaging 50 measurements of each algorithm.

The benchmark was implemented in C++14 and compiled with a 64 bit GCC 6.3 using *-Ofast* optimization level. Tests were conducted on five different computers with microprocessors from Tables 1 and 8, all equipped with 8 – 32 GB of RAM and Windows 10 operating system. The tests were conducted on freshly booted PCs after 10 minutes of idle time without running any non-system services or processes like browsers, database engines, etc.

The two $\gamma^{\times}$ and $\gamma^{\div}$ columns of Table 8 contains the execution time ratios of arithmetic operations with respect to addition taken from Table 2. For assessing the theoretical speedup from Theorem 1 we consider the instruction parallelism values $\beta^{\pm}$ and $\beta^{\times}$ from Table 4. The last column holds the values for the real measured speedup of the sequential reduced algorithm with respect to the full one.

	Ratios				Speed-up	
CPU	$\gamma^{\times}$	$\gamma^{\div}$	$\beta^{\pm}$	$\beta^{\times}$	Asses.	Meas.
A6-3420M	0.91	5.20	1.52	1.39	1.13	1.05
i5 430M	1.37	2.15	2.60	1.88	1.24	1.24
i3 2350M	1.39	3.43	1.96	1.67	1.15	1.15
i7 4790	1.44	3.39	1.56	1.62	1.07	1.10
i7 6700K	1.01	3.13	1.96	1.96	1.21	1.23

Table 8: Comparison of assessed and measured speedups on a 2001×2001 grid.

As we can see, the theoretical speedup is comparable with the measured one on the chosen CPU architectures.

7 DISCUSSION

Let us discuss the results from the numerical and experimental point of view. The reduced algorithm works with five types of tridiagonal system, see (15), (17), (19), (20) and (21), that differ from each other with the right hand sides similarly to the de Boor's full systems. Since three of these systems contains two times less equations then the corresponding full systems and their diagonal elements equal -14 instead of 4, from the theoretical view the reduced systems are diagonally dominant and therefore computationally stable [Bjo15a]. The second half of unknowns are computed from simple explicit formulas, see (16), (18), (22), (23), (24) and therefore do not present any issue from the computational view. The model equations of the reduced system were derived to fulfil the requirement of class C^2. The maximal error between the full and reduced system solutions is 10^{-12}, so we can conclude that the proposed reduced method yields numerically accurate results.

Although the reduced one contains twice as many cheap addition, subtraction and multiplication operations, it also contains less than half of expensive divisions giving to new approach a speed increase of factor 1.05 to 1.24 on the rest of tested CPU micro-architectures.

In the future we aim to improve the reduced algorithm for spline surfaces on a uniform grid with simpler equations and formulas for cross derivatives and so further increase the speedup.

8 CONCLUSION

We achieved performance increase of derivatives computation at uniform grid-points for spline surfaces and halved the memory space requirements.

The achieved speedup can be attributed to two interesting facts. Firstly, the reduced algorithm contains less than half the number of divisions. Depending on the CPU microarchitecture a floating-point division is several times slower than addition while floating point multiplication and addition are similarly fast.

Secondly, microprocessor cores are pipelined and superscalar. The reduced algorithm contains many expressions containing more than one arithmetic operation that can be and are evaluated in parallel on most modern x86 CPUs.

Although the reduced one contains twice as many cheap addition, subtraction and multiplication operations, it also contains less than half of expensive divisions giving to new approach a speed increase of factor 1.05 to 1.24 on the tested CPU micro-architectures.

9 APPENDIX

The full and reduced algorithms are given by two lemmas:

Lemma 3 (Full) *If the z values and d derivatives are given, see (2) – (5), then the values*

$$\begin{aligned} &d^x_{i,j}, && i=1,\dots,I-2, && j=0,\dots,J-1,\\ &d^y_{i,j}, && i=0,\dots,I-1, && j=1,\dots,J-2,\\ &d^{x,y}_{i,j}, && i=1,\dots,I-2, && j=0,\dots,J-1,\\ & && \text{and } i=0,\dots,I-1, && j=1,\dots,J-2 \end{aligned} \tag{10}$$

are uniquely determined by the following $2I+J+2$ linear systems of altogether $3IJ-2I-2J-4$ equations:
for $j=0,\dots,J-1$,

$$d^x_{i+1,j}+4d^x_{i,j}+d^x_{i-1,j}=\frac{3}{h_x}(z_{i+1,j}-z_{i-1,j}), \tag{11}$$

where $i=1,\dots,I-2$;
for $i=0,\dots,I-1$,

$$d^y_{i,j+1}+4d^y_{i,j}+d^y_{i,j-1}=\frac{3}{h_y}(z_{i,j+1}-z_{i,j-1}), \tag{12}$$

where $j=1,\dots,J-2$;
for $j=0,J-1$,

$$d^{x,y}_{i+1,j}+4d^{x,y}_{i,j}+d^{x,y}_{i-1,j}=\frac{3}{h_x}(d^y_{i+1,j}-d^y_{i-1,j}), \tag{13}$$

where $i=1,\dots,I-2$;
for $i=0,\dots,I-1$,

$$d^{x,y}_{i,j+1}+4d^{x,y}_{i,j}+d^{x,y}_{i,j-1}=\frac{3}{h_y}(d^x_{i,j+1}-d^x_{i,j-1}), \tag{14}$$

where $j=1,\dots,J-2$.

Lemma 4 (Reduced) *If the z values and d derivatives are given, see (2) – (5), then the values $d^x_{i,j}$, $d^y_{i,j}$, $d^{x,y}_{i,j}$ from (10) are uniquely determined by the following $\frac{3I+2J+5}{2}$ linear systems of altogether $\frac{5IJ-I-J-23}{4}$ equations and $\frac{7IJ-7I-7J+7}{4}$ formulas:*
for $j=0,1,\dots,J-1$,

$$\begin{aligned} &d^x_{i+2,j}-14d^x_{i,j}+d^x_{i-2,j}=\\ &=\frac{3}{h_x}(z_{i+2,j}-z_{i-2,j})-\frac{12}{h_x}(z_{i+1,j}-z_{i-1,j}), \end{aligned} \tag{15}$$

where $i=2,4,\dots,I-3$;

$$d^x_{i,j}=\frac{3}{4h_x}(z_{i+1,j}-z_{i-1,j})-\frac{1}{4}(d^x_{i+1,j}+d^x_{i-1,j}), \tag{16}$$

where $i=1,3,\dots,I-2$, $j=1,3,\dots,J-2$;
for $i=0,1,\dots,I-1$,

$$\begin{aligned} &d^y_{i,j+2}-14d^y_{i,j}+d^y_{i,j-2}=\\ &=\frac{3}{h_y}(z_{i,j+2}-z_{i,j-2})-\frac{12}{h_y}(z_{i,j+1}-z_{i,j-1}), \end{aligned} \tag{17}$$

where $j=2,4,\dots,J-3$;

$$d^y_{i,j}=\frac{3}{4h_y}(z_{i,j+1}-z_{i,j-1})-\frac{1}{4}(d^y_{i,j+1}+d^y_{i,j-1}), \tag{18}$$

where $i=1,3,\dots,I-2$, $j=1,3,\dots,J-2$;
for $j=0,J-1$,

$$d^{x,y}_{i+1,j}+4d^{x,y}_{i,j}+d^{x,y}_{i-1,j}=\frac{3}{h_x}(d^y_{i+1,j}-d^y_{i-1,j}), \tag{19}$$

where $i=1,\dots,J-2$;
for $i=0,I-1$,

$$d^{x,y}_{i,j+1}+4d^{x,y}_{i,j}+d^{x,y}_{i,j-1}=\frac{3}{h_y}(d^x_{i,j+1}-d^x_{i,j-1}), \tag{20}$$

where $j=1,2,\dots,J-2$;
for $i=2,4,6,\dots,I-3$,

$$\begin{aligned} &d^{x,y}_{i,j+2}-14d^{x,y}_{i,j}+d^{x,y}_{i,j-2}=\\ &=\frac{1}{7}(d^{x,y}_{i-2,j+2}+d^{x,y}_{i-2,j-2})-2d^{x,y}_{i-2,j}+\\ &+\frac{3}{7h_x}(d^y_{i-2,j+2}+d^y_{i-2,j-2})+\frac{3}{7h_y}(-d^x_{i-2,j+2}+d^x_{i-2,j-2})+\\ &+\frac{9}{7h_x}(d^y_{i,j+2}+d^y_{i,j-2})+\frac{9}{7h_xh_y}(-z_{i-2,j+2}+z_{i-2,j-2})+\\ &+\frac{12}{7h_x}(-d^y_{i-1,j+2}-d^y_{i-1,j-2})+\frac{12}{7h_y}(d^x_{i-2,j+1}-d^x_{i-2,j-1})+\\ &+\frac{3}{h_y}(d^x_{i,j+2}-d^x_{i,j-2})+\frac{27}{7h_xh_y}(-z_{i,j+2}+z_{i,j-2})+\\ &+\frac{36}{7h_xh_y}(z_{i-1,j+2}-z_{i-1,j-2}+z_{i-2,j+1}-z_{i-2,j-1})-\\ &-\frac{6}{h_x}d^y_{i-2,j}+\frac{12}{h_y}(d^x_{i,j+1}+d^x_{i,j-1})+\frac{108}{7h_xh_y}(z_{i,j+1}-z_{i,j-1})-\\ &-\frac{18}{h_x}d^y_{i,j}+\frac{144}{7h_xh_y}(-z_{i-1,j+1}+z_{i-1,j-1})+\frac{24}{h_x}d^y_{i-1,j}, \end{aligned} \tag{21}$$

where $j = 4,6,\ldots,J-5$;

$$d_{i,j}^{x,y} = \frac{1}{16}(d_{i+1,j+1}^{x,y} + d_{i+1,j-1}^{x,y} + d_{i-1,j+1}^{x,y} + d_{i-1,j-1}^{x,y}) - \\ - \frac{3}{16h_y}(d_{i+1,j+1}^{x} - d_{i+1,j-1}^{x} + d_{i-1,j+1}^{x} - d_{i-1,j-1}^{x}) - \\ - \frac{3}{16h_x}(d_{i+1,j+1}^{y} + d_{i+1,j-1}^{y} - d_{i-1,j+1}^{y} - d_{i-1,j-1}^{y}) + \\ + \frac{9}{16h_xh_y}(z_{i+1,j+1} - z_{i+1,j-1} - z_{i-1,j+1} + z_{i-1,j-1}), \quad (22)$$

where $i = 1,3,\ldots,I-2,\ j = 1,3,\ldots,J-2$;

$$d_{i,j}^{x,y} = \frac{3}{4h_y}(d_{i,j+1}^{x} - d_{i,j-1}^{x}) - \frac{1}{4}(d_{i,j+1}^{x,y} + d_{i,j-1}^{x,y}), \quad (23)$$

where $i = 1,3,\ldots,I-2,\ j = 2,4,\ldots,J-3$;

$$d_{i,j}^{x,y} = \frac{3}{4h_y}(d_{i,j+1}^{x} - d_{i,j-1}^{x}) - \frac{1}{4}(d_{i,j+1}^{x,y} + d_{i,j-1}^{x,y}), \quad (24)$$

where $i = 2,4,\ldots,I-3,\ j = 1,3,\ldots,J-2$.

10 ACKNOWLEDGEMENT

This work was partially supported by the research grants VEGA 1/0073/15 and VVGS-PF-2015-477.

11 REFERENCES

[Bjo15a] A. Björck: Numerical Methods in Matrix Computations, Springer, 2015.

[Boo62a] C. de Boor: Bicubic Spline Interpolation, Journal of Mathematics and Physics, 41(3), 1962, pp. 212–218.

[Dik97a] N. D. Dikoussar: Function parametrization by using 4-point transforms, Comput. Phys. Commun. 99 (1997), pp. 235–254.

[Dik06a] N. D. Dikoussar, Cs. Török: Automatic Knot Finding For Piecewise Cubic Approximation, Mat. Model., 2006, T-17, N.3.

[Dik07a] N. D. Dikoussar, Cs. Török: On one approach to local surface smoothing, Kybernetika 43 (4), N.4, pp. 533-546, 2007.

[Fog16a] A. Fog: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs, Technical University of Denmark, 1996 – 2016, Last updated 2016-12-01, http://www.agner.org/optimize/instruction_tables.pdf

[Int16a] Intel 64 and IA-32 Architectures Optimization Reference Manual, Intel Corp., 2016 http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf

[Mat05a] A. Matejčiková, Cs. Török: Noise Suppression in RDPT, Forum Statisticum Slovacum, 3/2005, Bratislava, ISSN 1336-7420, pp. 199–203.

[Min15a] L. Miňo: Efficient Computational Algorithm for Spline Surfaces, ITAT 2015, pp. 30–37.

[Min16a] L. Miňo, I. Szabó, Cs. Török: Bicubic Splines and Biquartic Polynomials, Open Computer Science, Volume 6, Issue 1, Pages 1–7, ISSN (Online) 2299–1093, February 2016.

[Min15b] L. Miňo, Cs. Török: Fast Algorithm for Spline Surfaces, Communication of the Joint Institute for Nuclear Research, Dubna, Russian Federation, E11-2015-77, (2015), pp. 1–19.

[Amd11a] Software Optimization Guide for AMD Family 10h and 12h Processors, 2011 http://support.amd.com/TechDocs/40546.pdf

[Pat15a] J. R. C. Patterson: Modern Microprocessors - A 90-Minute Guide, 2001-2015 http://www.lighterra.com/papers/modernmicroprocessors/

[Rev07a] M. Révayová, Cs. Török: Piecewise Approximation and Neural Networks, Kybernetika, Vol. 43 (4), No. 4, 2007, pp. 547–559.

[Rev13a] M. Révayová, Cs. Török: Reference Points Based Recursive Approximation, Kybernetika, Vol. 49, No. 1, 2013, pp. 60–72.

[Sza16a] I. Szabó: Approximation Algorithms for 3D Data Analysis, PhD Thesis, P. J. Šafárik University in Košice, Slovakia, 2016.

[Sza13a] I. Szabó, Cs. Török: Smoothing in 3D with Reference points and Polynomials, 29th Spring Conference on Computer Graphics SCCG 2013, Smolenice - Bratislava, Comenius University, ISBN 9788022333771, pp. 39–43.

[Tor00a] Cs. Török: 4-point transforms and approximation, Comput. Phys. Commun., 125 (2000) pp. 154–166.

[Tor14a] Cs. Török: On reduction of equations' number for cubic splines, Matematicheskoe modelirovanie, vol. 26 (2014), no. 11, ISSN 0234-0879, pp. 33–36.

[Tor09a] Cs. Török: Piecewise smoothing using shared parameters, Forum Statisticum Slovacum, 7/2009, pp. 188–193.

[Tor13a] Cs. Török: Reference Points Based Transformation and Approximation, 2013, Kybernetika, Vol. 49, No. 4, 2013, http://www.kybernetika.cz/content/2013/4/644/paper.pdf

[TorTA] Cs. Török: Speed-up of Interpolating Spline Construction, to appear.

Marine Snow Detection and Removal: Underwater Image Restoration using Background Modeling

Fahimeh Farhadifard
University of Rostock, Germany
fahimeh.farhadifard@igd-r.fraunhofer.de

Martin Radolko
University of Rostock, Germany
martin.radolko@igd-r.fraunhofer.de

Uwe Freiherr von Lukas
Fraunhofer IGD Rostock, University of Rostock, Germany
uwe.freiherr.von.lukas@igd-r.fraunhofer.de

ABSTRACT

It is a common problem that images captured underwater (UW) are corrupted by noise. This is due to the light absorption and scattering by the marine environment; therefore, the visibility distance is limited up to few meters. Despite blur, haze, low contrast, non-uniform lightening and color cast which occasionally are termed noise, additive noises, such as sensor noise, are the center of attention of denoising algorithms. However, visibility of UW scenes is distorted by another source termed marine snow. This signal not only distorts the scene visibility by its presence but also disturbs the performance of advanced image processing algorithms such as segmentation, classification or detection. In this article, we propose a new method that removes marine snow from successive frames of videos recorded UW. This method utilizes the characteristics of such a phenomenon and detects it in each frame. In the meanwhile, using a background modeling algorithm, a reference image is obtained. Employing this image as a training data, we learn some prior information of the scene and finally, using these priors together with an inpainting algorithm, marine snow is eliminated by restoring the scene behind the particles.

Keywords

Underwater Image Processing, Marine Snow, Background Model, Inpainting

1 INTRODUCTION

The growing interest in UW image processing lies in the poor performance of devices used to capture UW scenes. The major barrier is that light, unlike sound, is poorly propagated in water. This is explained by the propagation properties of light in water ([McG80, Wel69]). Light is exponentially attenuated while traveling in water. This is caused by two factors: light absorption and scattering, which leads to poor contrast, haze, blur and color cast.

- Light absorption reduces the light energy; therefore, colors drop one by one based on their wavelength (color cast). One can augment the visibility range by using artificial lightening; however,
- water reflects a significant fraction of light back to the camera before it even reaches the object in the scene. This so-called backscattering yields degraded contrast scene and a foggy appearance. Furthermore,
- a fraction of light reflects from the object to the camera with a small angle (forward scattering) which generally leads to a blurry image. Finally,
- organic and inorganic floating particles in water distort the scene visibility as an unwanted signal and are considered as noise, although, they belong to the scene.

As a result, visibility UW is limited at a distance of about twenty meters in clear water and five meters or less in turbid water [ABMK05b]. Naming distortions for UW imaging, the one which is not well-researched and mostly neglected from image processing algorithms, is the presence of floating particles. Although, in orders of magnitude these particles together with Backscatter have the greatest degradation factor [ABMK05b].

Marine snow is the term which is used for the macroscopic aggregates of detritus, dead material and dissolved organic matter floating in water. According to the properties of light propagation in water, smaller particles scatter the light more, thus, marine snow is one of the main sources of scattering (more specifically backscatter). Light reflection on marine snow creates white bright spots that lead to an inhomogeneous

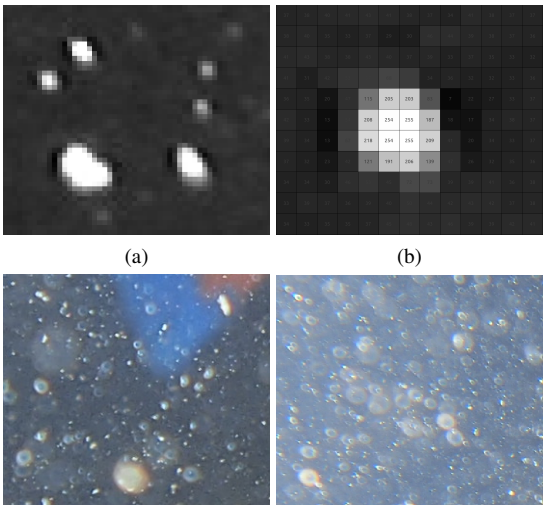


Figure 1: Illustration of physical characteristics of marine snow in acquired data. (a) particles with different sizes (3×3 to 20×20, (b) geometry of particles, (c) particles are present in different camera-scene depths (contrary to additive noise) and (d) strong light reflection of particles due to using artificial illumination.

medium [BG12]. Not only scattering and absorption are increased due to this phenomenon, but also it may appear dominant enough to reduce the scene perception (some examples are shown in Figure 1 (c) and (d)). Due to all the difficulties caused by marine snow, in this work, it is considered and treated as noise.

In this paper, we consider a novel approach to removing marine snow from frames of a video where the camera is assumed to be static. The information provided by the video sequence is used to eliminate marine snow from each individual frame. Our algorithms has three main steps, first, we employ our previously proposed background modeling algorithm [RG15] (which is based on the well-known Gaussian background modeling approach) and obtain an accurate model of the static components of the video. This model gives us the information about the background which is covered with the marine snow. Second, we detect the corrupted pixels based on our detection algorithm [FRvL17] and extract a mask which indicates the location of marine snow. Next, using the background model as a training data, some prior information about the scene are learned [RB05]. Finally, we employed the trained priors and the inpainting algorithm proposed by Roth and Black [RB05] together with the extracted mask, and eliminate marine snow by restoring the scene behind it with the most related prior information. Experiments show promising results where marine snow is almost completely removed and even small details are preserved.

The rest of this paper is structured as follows: in Section 2, we present a summary of the related works. Section 3 introduces marine snow and provides a short summary of its characteristics. Section 4.1 contains the explanation of background modeling method used to provide the training data [RG15]. In Section 4.2, we explain how to extract an accurate mask containing marine snow locations from a single frame. And at last, the inpainting algorithm in [RB05] is detailed in Section 4.3. Evaluation of the algorithm is provided in Section 5.

2 RELATED WORK

The popular approaches towards denoising consist of filtering [ABMK05a, LNHL15], wavelet decomposition and high-pass filtering [SZW11, PK10], a combination of curvelet and filtering [SSS13]. These methods assume that every kind of present noise including marine snow can be defined as one of the additive noises. Thus, salt & pepper, Gaussian and speckle noise are considered and with this assumption, authors provide a solution.

However, considering marine snow as an unwanted signal in UW images, these algorithms can not address eliminating of this phenomenon. This is due to their main assumptions (additive and single pixel noise) which do not match marine snow's characteristics. Marine snow is an object in the scene and has a structure of several pixels and covers the scene. Usually, these particles do not carry interesting information of the scene and therefore, are disturbing for image processing algorithms.

Banerjee et al. [BSG+14] proposed a probabilistic approach using median filtering to eliminate this phenomenon from single images. This approach checks the probability of the existence of marine snow in each patch. This is done by looking for high luminance pixels in a patch using a predefined threshold and calculating its probability as follows:

$$P(MS) = 1 - \frac{N_{HL}}{N} \tag{1}$$

where N_{HL} and N stand for the number of high luminance pixels and the total number of pixels in the current patch respectively. They consider a cross-checking to avoid misclassification of the true objects as marine snow. To this end, keeping the same center pixel, they increased the patch size by 2 (in both directions) and calculate the probability one more time. If the probability of having marine snow in the resized patch is still high (low number of high luminance pixels) then the center pixel is replaced by the median value of the local patch. The logic behind this is that they assume marine snow to have a structure of two or three pixels;

Figure 2: Comparison of different update schemes for the background modeling. In the top row are the first and 2000th frames of the Town Center Video. Second and the third rows correpond to the background models for the 2000th frame created with different updating mechanisms: the partial updating, a complete update, and GSM respectively.

therefore, if the probability of high luminance pixels increased it means that it is a bigger object which can not be marine snow.

However, this assumption does not hold always since usually marine snow, depending on the image resolution, have bigger structures (in our case it reaches to 20×20 pixels). Thus, considering it to have sizes bigger than three pixels, this criterion cannot differentiate between marine snow and other objects in the image. Increasing the patch size to take into account bigger sizes of marine snow may lead to a significantly blurred image. Moreover, this method does not use all the information provided in an image since it only considered gray scale image which could result in false detection of similar structures with different colors.

3 MARINE SNOW AND ITS CHARACTERISTICS

Decaying dead material and dissolved organic matter in water is referred to as marine snow since it is white and looks like snowflakes falling. These particles grow as they fall, some reaching several centimeters in diameter, this is due to aggregation of smaller particles. Thus, in an image, marine snow appears as white bright spots of different sizes and geometries randomly distributed in the image (Figure 1).

In view of Figure 1 (provided as an example), we could observe some physical characteristics of marine snow in captured images:

- it appears in different sizes depending on the image resolution. Usually between 3×3 to 20×20 pixels (Figure 1(a)).
- it can be roughly estimated as a Gaussian distribution in all directions, a high peak in the middle and lower intensities elsewhere proportional to the distance to the peak's location (Figure 1(b)).
- in contrary to additive noise, marine snow is present in all layers of a scene (considering the depth map of a scene consists of several layers) and can have a highly overlapped and non-uniform distribution over the image (Figure 1(c)).
- the most challenging fact about this phenomenon is that in the case of using an artificial light at the time of photography, it scatters the light to the camera and appears as circle shaped reflection (Figure 1(d)).

4 PROPOSED APPROACH

Having a video of reasonable length, we divide it into two parts. The first ~ 500 frames are used for training and the rest for testing. Although, the number of frames used for training can vary e.g. in the case of video in Figure 7 which is short (only 150 frames), we duplicate the training set by mirroring the order of training frames and conduct a bigger training set. The training frames are then used to learn a background model using Gaussian background modeling [RG15] (Section 4.1). Next, for each test frame, a mask containing marine snow locations is derived. The details of mask extraction are provided in Section 4.2. Once the background model of the scene and the mask corresponding to each test frame is available, the inpainting algorithm [RB05] is trained over the background model and recovers the scene behind marine snow using the corresponding mask (Section 4.3).

4.1 Gaussian Switch Model

It is a common practice to use a Gaussian distribution to model the color information of frame pixels in a video

sequence and extract one image which only contains the background. For this purpose, one can use the Mixture of Gaussian models [SG99, WBSP14], however, they are not ideal due to difficulty at unifying the different Gaussian distributions again. On the other hand, single Gaussian [WADP97] approaches lack accuracy. Thus, to keep the balance between accuracy and complexity, we use our Gaussian Switch Model (GSM) proposed in [RG15].

The idea behind this algorithm comes from the shortcoming of a single Gaussian approach which includes the information from foreground objects into the background model and corrupts it. Especially when there is a constant presence of many foreground objects. This can be solved by applying a *partial update*, this means that instead of updating the whole model, only the pixels that are classified as background are updated. Ideally, now only background information is included into the model which should lead to a more robust and precise model. For this, the segmentation of the current frame is computed by background subtraction before the model is updated with the information from this new frame. Then the segmentation can be used to update the background pixels and exclude foreground objects. In general, this improves the segmentation and stabilizes the model, but since the model is used to improve its updating process itself, a kind of self-fulfilling prophecy can occur.

An example of this is the presence of a foreground object in the initialization. This foreground object is a part of the model in the beginning and should slowly be overwritten with the background information during the updating process. However, when partial updating is applied, this usually does not happen because the actual background in that area will be marked as foreground; therefore, it does not get included into the model.

To still get the benefits from the partial updating without facing these problems, the GSM uses two Gaussians to model the background. The first Gaussian is partially updated and is taken as the background model and the second Gaussian is fully updated with every frame. The errors of the partially updated model can be discovered by a comparison between these two Gaussians since they always show the same characteristics:

- the means of the two Gaussians slowly diverge from each other as the Gaussian with the full update adapts to the new background and the other stays constant.
- for many successive frames a foreground object is detected at the same position.

If these characteristics are true for a specific pixel, the partial updated Gaussian for that pixel is overwritten with the values of the full updated Gaussian as it does not reflect the true background anymore. The represent version of the background model can then be simply extracted by taking the mean of the partially updated model for each pixel and color channel.

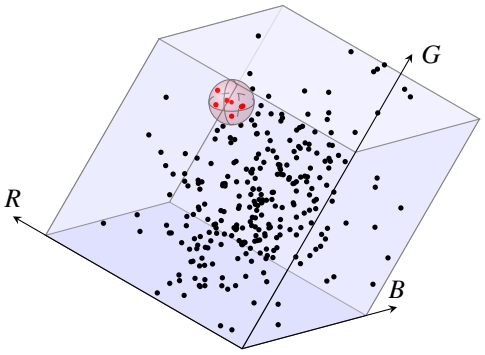


Figure 3: The pixels of current patch are visualized as points in RGB color space. The pink sphere demonstrates the search environment for the density calculation.

An example of the background modeling with the GSM compared to the partial and full update approaches can be seen in Figure 2 (the mean values of the Gaussians are displayed) where it is compared to the full and partial updating schemes on a video with many foreground objects. The parameters of the modeling are the same for all three methods and it can be seen that the complete update created a model which is very corrupted with the current foreground objects of the scene. The partial update eliminates this problem but many objects from the first frame can still be seen in the model there as they never get eliminated. The GSM can combine the advantages of both methods and can create an almost uncorrupted background model.

4.2 Mask Extraction

Knowing how marine snow appears in an image, we applied a detection algorithm to extract a mask indicating corrupted pixels in the image. This is done by looking for the pixels with the same characteristics as marine snow within the patches of an image. First, a rough detection of corrupted pixels is obtained and then, a voting algorithm conducts the final detection.

Marine snow is more visible and disturbing when the background is darker (lower intensity), although the particles are presented everywhere but they decrease the visibility of the scene especially when there is a higher contrast to the background. Thus, generally, the light reflection on the small particles are represented as bright spots in an image.

Thereby, the intensity of pixels within a patch is checked and a sudden high-intensity occurrence is marked as potential marine snow. A candidate pixel p has to satisfy the following inequality:

$$\|p-\mu(\Omega)\|_2^2 > W_1 \cdot \sigma(\Omega) \quad p \in \Omega, \tag{2}$$

here W_1 is an empirical weight, $\sigma(\Omega)$ is the standard deviation, and $\mu(\Omega)$ denotes the mean value of the local patch Ω. W_1 is defined heuristically and is affected by the image resolution, in our case where the data has a resolution about 850×478, $W_1 = 1.7$ is the best choice.

Next, we look for general outliers within the candidate pixels from the last step. This is done to distinguish between a high-intensity outlier and a high-intensity object's edge. For this, the idea of Gutzeit et al. [GOK+10] is employed. The RGB color space is considered as Euclidean space. The pixels within the current patch are then represented in this space and the density surrounding each candidate pixel is calculated. Figure 3 demonstrates this process. A sphere covering an area surrounding each high-intensity pixel defined in the last step is explored.

The number of pixels within this sphere

$$\#\{v \in \Omega \mid \quad \exists p \in \Omega : \quad \|p - v\|_2^2 < \sigma(\Omega)\}, \qquad (3)$$

together with the volume of the sphere and the overall number of pixels in the patch gives us the density. Here v and p are pixels in the local patch where $p \neq v$, and $\sigma(\Omega)$ is the standard deviation. The radius of the sphere is defined dynamically based on the weighted standard deviation of Ω to make the approach adaptive.

Another observation can be derived from marine snow characteristics: it mostly appears having high intensity and low saturation. Thus, by applying the following inequality the pixels with high saturation are discarded:

$$|p_c - p_l| < T \quad \forall c, l \in \{R, G, B\} \ \wedge \ c \neq l. \qquad (4)$$

Thereby, the candidate pixels are limited to have colors close to white by using a predefined threshold T (e.g. $T = 2$). All the pixel values that satisfy the aforementioned conditions are then discarded and the median value of the remaining pixel values within the local patch Ω is calculated. For now, all the eliminated values in this patch are replaced by this median value. The filtering is done in a copy version of the original image. This procedure, initial filtering, is repeated for the whole image.

The patches are extracted highly overlapped; this means each pixel can be in $n \times n$ possible patches except for the pixels at the border of the image with fewer possibilities ($n \times n$ is the patch size). Therefore, each pixel of the image could have been filtered in different patches accordingly, which results in having several filtered versions for a single pixel. The final decision about each pixel is then made by using a voting algorithm. If the majority of the filtered versions correspond to each pixel in the original frame indicate that it contains marine snow then, the location of that pixel is marked as noisy in a mask image. A mask image is a matrix, the same size as the original frame, whose pixel values are binary (one indicating marine snow and zero elsewhere).

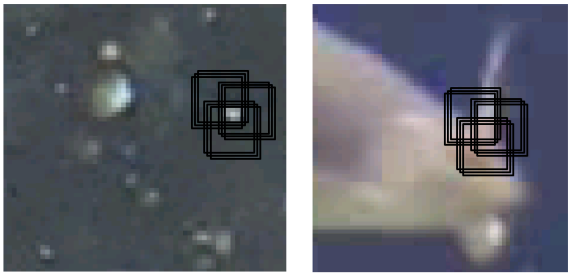


Figure 4: marine snow detection shows overlapping patches for marine snow (left) versus an object edge (right).

Figure 4 illustrates the condition with an example. This is applied on Laplacian pyramid of the image to detect marine snow with different sizes.

Once the final mask indicating marine snow locations is acquired, it is used to recover the denoised image using inpainting. At the end of this stage, we already can remove marine snow using filtering detected pixels. However, the results may suffer from smoothing the edges. In addition, in locations where the intensity of the image pixels varies (not only at the edges of the objects but also for example where color shades of a fish changes), the filtering can result in a wrong pixel value. An example of this situation is illustrated in the zoom-in presentation in Figure 6.

Thus, to improve the results, instead of filtering the image directly, we apply an inpainting algorithm which learns the most relevant priors to the test image by training over the background model of the same scene and restore the image accordingly.

4.3 Inpainting

Image inpainting is a useful application in several scenarios of image processing. It is used to fill in pixels which are missing in an image. Examples of inpainting in image manipulation include the removal of scratches on a photograph, unwanted occluding objects, superposed text, road-signs or publicity logos [ESQD05].

Generally, inpainting uses the information provided by the neighbors to fill in the missing pixels. However, whenever there is noise or any uncertainty, prior models of images such as depth maps, flow fields, *etc.* come into play. In our case where an object is considered as noise, prior information about the scene is advantageous. Therefore, we employ the inpainting algorithm proposed by Roth and Black [RB05]. This algorithm uses *Field of Experts* (FoE) to learn image priors from external data. We employ this algorithm rather than traditional inpainting algorithms so if the detection could not extract the particles precisely, the restoration will not be highly affected due to using direct neighborhoods.

FoE employs both sparse coding and Markov Random Field (MRF) to learn rich, generic prior models of any

class of images. Sparse coding provides an elegant and powerful way of learning prior distributions on small image patches. However, the result does not generalize to give a prior model for the whole image. This is where MRF provides not very rich but general prior information of the whole image.

The key idea behind FoE [RB05] is to extend MRF by modeling the local field potentials with learned bases. These bases capture important structural properties of images, respond to various edge and texture features. For this, they used the idea of the Product of Experts (PoE) framework [Hin02] and trained a model on a data set and develop a diffusion-like scheme that exploits the prior for approximate Bayesian inference.

For more insight, consider the pixels in an image be presented by nodes V in a graph $G = (V,E)$, where E stands for the edges connecting nodes. A rectangle region of $m \times m$ neighborhood connecting all nodes is defined, where every such neighborhood is centered on a node (pixel). The probability density of such a graphical model is defined as a Gibbs distribution:

$$p(x) = \frac{1}{Z} \exp^{(-\sum_k V_k(x_k))} \tag{5}$$

here x stands for an image and $V_k(x_k)$ is the potential function for clique x_k. They further assumed that the MRF is homogeneous which leads to translation invariance of MRF model. The potential Function V is then learned using training images. And as a result, the probability density of a full image is obtained as follows:

$$p(x) = \frac{1}{Z(\Theta)} \prod_k \prod_{i=1}^{N} \Phi_i(J_i^T x_k; \alpha_i), \tag{6}$$

where $Z(\Theta)$ is a normalizing function, J_i is a linear filter, Φ_i the experts and N stands for the number of experts. This model works for different image sizes, enjoys translation invariant property which is desirable for generic image priors and has few parameters which need to be learned. The parameters α_i and linear filters J_i are learned from the training images by maximizing its likelihood.

Once the parameters are learned, the inpainting algorithm propagates information using only FoE prior and refills the pixels iteratively by introducing an iteration index t and an update rate η as follows:

$$x^{(t+1)} = x^{(t)} + \eta M \sum_{i=1}^{N} J_i^- * \Psi_i(J_i * x^{(t)}) \tag{7}$$

let $\Psi_i(y) = \frac{\mathrm{d}}{\mathrm{d}y} \log \Phi_i(y; \alpha_i)$, J_i^- denotes the filter obtained by mirroring J_i around its center pixel [ZM97], and $*$ stands for convolution.

5 EXPERIMENTAL RESULTS AND DISCUSSIONS

We have performed our experiments on two different scenes. One is taken at Ozeaneum Stralsund (Figure 6), and the other one is courtesy of GEOMAR which is taken in the Black sea (Figure 7). For the first video (Figure 6), the first 500 frames are used to train the background model and the rest (50 frames) are added to the testing set for evaluation of the algorithm. Second video (Figure 7) is shorter; therefore, only 150 frames are available for training (and 5 frames for testing). For this video, we have duplicated the training frames to expand the training set for a better result. The background models of both scenes are obtained using GSM background modeling algorithm [RFvL16] (an example: Figure 5). Once the background model is available, we applied [RB05] to learn the FoE priors. For each frame in the testing set, we obtain a mask containing marine snow locations by applying the method explained in section 4.2. Finally, using the inpainting algorithm and mask together with the priors, the scene behind marine snow is recovered.

To quantitatively evaluate our algorithm, we need ground truth data which is not available in our case. Therefore, we have provided simulated frames. To this end, one simple approach would be to generate a salt and pepper noise on the image. However, as it was discussed before, this model does not take into account various physical parameters such as the effect of water absorption and scattering on the signal backscattered by the particles, the size, and shape of the particles or the defocus effect. Boffety and Galland [BG12] consider most of these properties and proposed a method to model this phenomenon by assuming that these particles behave like white Lambertian scatters. However, it still does not cover different geometries of marine snow and gives an artificial look to the image.

Thus, we have employed a different strategy. First, marine snow is extracted from the a test frame of each scene. This is done by human experts where the particles are manually extracted with pixel accuracy. Then, we have restored the scene behind marine snow with information of the neighborhood pixels and frames and conduct a ground-truth image. Once this image is available, we have placed the extracted marine snow randomly in the frame. This simulated image together with the ground truth image is then used to evaluate the performance of our proposed method. This way, we obtain a simulated data with a very natural look where marine snow has the most accurate model and is highly correlated to the real frames.

Our result is compared to the method in [BSG$^+$14], the result by directly filtering marine snow using the extracted mask (explained at the end of Section 4.2), our proposed algorithm when inpainting is trained on an arbitrary training set of the same class (UW images), and finally our proposed algorithm when inpainting is trained using the background model. Comparison is done via PSNR and MSE calculation (Table 1).

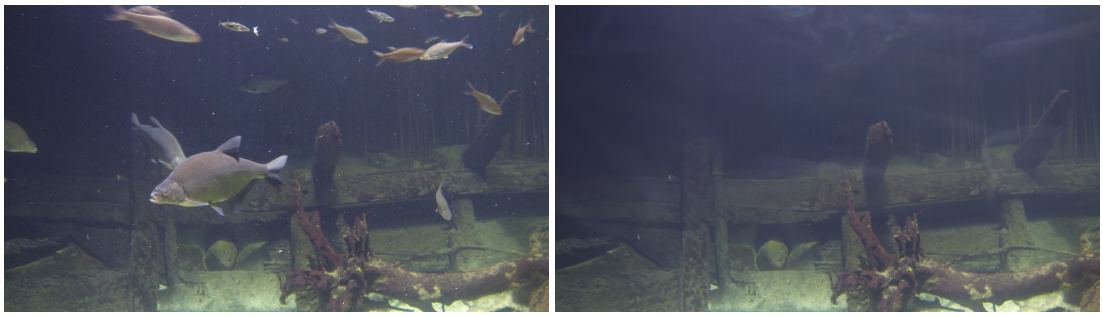


Figure 5: Results of the GSM background modeling on an UW video. On the left is the original frame of the video and on the right the background model is depicted.

In view of Table 1, one notices that filtering marine snow using the extracted mask has already succeeded by about 4 and 6.3 dB improvement on the results of [BSG$^+$14] for simulated images of scene 1 and 2 respectively. The advantage of using inpainting algorithm together with the extracted mask has been proved by achieving further improvement of 1.2 and 3.4 dB respectively. It can be seen that training data for the inpainting algorithm plays an effective role, where the result of the proposed algorithm differs when the training data changes. When the training data has a high correlation with the input image, the algorithm can achieve about 0.2 dB and 0.5 dB improvement, for the first and the second scene respectively, compared to the situation where a set of arbitrary images of the same class (UW images) are used.

Figures 6 and 7 illustrate the qualitative results corresponding to the table 1. The improvement over [BSG$^+$14] is clear, where the edges are smoothed and marine snow is not removed completely. Filtering the image using the extracted mask has already succeeded to remove marine snow effectively without smoothing the edges too much. However, a closer look demonstrates the shortcoming of this approach. It fails at removing marine snow correctly where it lies on the edges of objects or where there is no edge but the intensity values change (e.g. color shades of a fish). It happens due to the fact that median value of different candidate patches may not exactly match the true value. An example can be seen in the zoom-in presentation of Figure 6. In these situations, inpainting provides smoother results because it does not use the neighborhood pixels directly but the learned priors of the whole scene.

PSNR Values		
Approach	Scene 1	Scene 2
[BSG$^+$14]	43.6504	37.4235
Mask + Filtering Only	47.4818	43.7806
P. A. without BG model	48.6403	47.1723
P. A. with BG model	48.8275	47.6833
MSE Values		
[BSG$^+$14]	1.6750	3.4306
Mask + Filtering Only	1.0776	1.6501
P. A. without BG model	0.9430	1.1167
P. A. with BG model	0.9229	1.0529

Table 1: Evaluation using PSNR and MSE values.

6 CONCLUSION

In this paper, we have proposed an algorithm to eliminate marine snow from UW videos where the camera is static. The approach has three main steps; first, our background modeling algorithm [RG15] has been employed to extract an accurate model of the static components of a video. Second, we have detected marine particles in each individual test frame and a mask containing marine snow locations is derived. Finally, the background model is used as training data for an inpainting algorithm to extract the generic distribution of the scene which is then used together with the mask to restore the information behind the marine snow.

The results have illustrated the success of the proposed algorithm at eliminating marine snow. Simple filtering using the extracted mask has shown superior to the results of [BSG$^+$14] both quantitatively and qualitatively. In addition, employing inpainting has enhanced the results by restoring the image more accurately.

However, there is still space for improvement especially in extracting the mask. If the mask is not accurate enough to cover marine snow completely, inpaining may rebuild it back. Furthermore, when marine snow lies at the edge of two regions with low and high luminance, the algorithm may not be able to detect it. This is due to our first assumption that a sudden high luminance occurrence in a patch is a candidate to be marine snow.

7 ACKNOWLEDGMENTS

This research has been supported by the German Federal State of Mecklenburg-Western Pomerania and the European Social Fund under grant ESF/IV-BM-B35-0006/12.

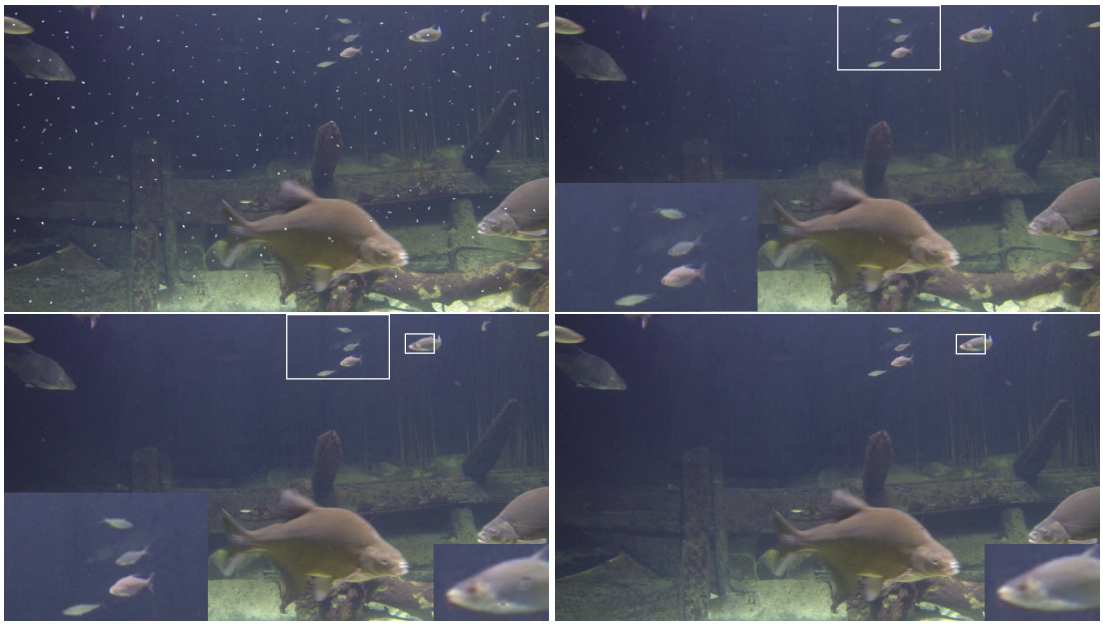


Figure 6: Results of marine snow removal for a frame of the first video. From top to bottom and left to right: input corrupted image, result of [BSG$^+$14], result of filtering using the proposed mask and final result of the proposed algorithm. Marked areas illustrate the improvement of the proposed method on previous work.

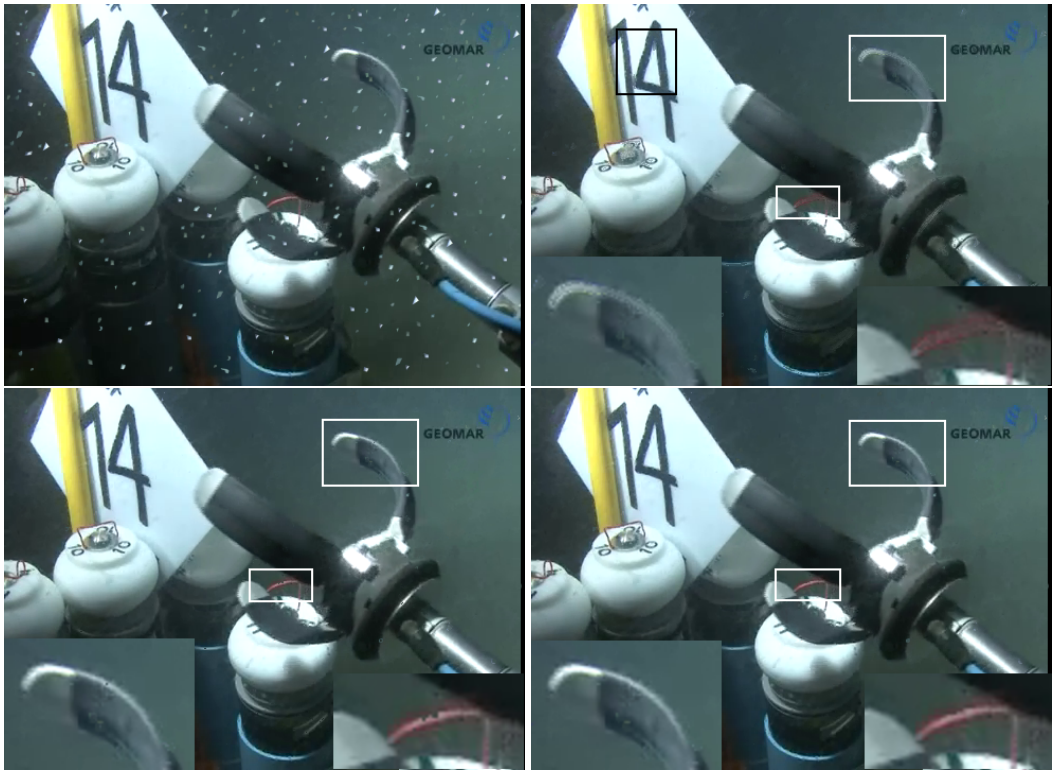


Figure 7: Results of marine snow removal for a simulated frame of the second video. From left to right and top to bottom: input corrupted image, result of [BSG$^+$14], result of filtering using the proposed mask and final result of the proposed algorithm. The marked areas show some examples of [BSG$^+$14] failure which are improved using the proposed method. Full video of this scene is courtesy of JAGO-Team, GEOMAR Kiel, Germany.

8 REFERENCES

[ABMK05a] A. Arnold-Bos, J. P. Malkasse, and G. Kervern. A preprocessing framework for automatic underwater images denoising. In *European Conference on Propagation and Systems*, 2005.

[ABMK05b] A. Arnold-Bos, J. P. Malkasse, and G. Kervern. Towards a model-free denoising of underwater optical images. In *Europe Oceans 2005*, pages 527–532, June 2005.

[BG12] M. Boffety and F. Galland. Phenomenological marine snow model for optical underwater image simulation: Applications to color restoration. In *OCEANS, 2012-Yeosu*, pages 1–6. IEEE, 2012.

[BSG+14] S. Banerjee, G. Sanyal, S. Ghosh, R. Ray, and S. N. Shome. Elimination of marine snow effect from underwater image - an adaptive probabilistic approach. In *Electrical, Electronics and Computer Science (SCEECS), 2014 IEEE Students' Conference on*, pages 1–4, 2014.

[ESQD05] 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*, pages 340–358, 2005.

[FRvL17] F. Farhadifard, M. Radolko, and U. F. von Lukas. Single image marine snow removal based on a supervised median filtering scheme. to appear in 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications *VISAPP 2017*, 2017.

[GOK+10] E. Gutzeit, S. Ohl, A. Kuijper, J. Voskamp, and B. Urban. Setting graph cut weights for automatic foreground extraction in wood log images. In *VISAPP 2010*, pages 60–67, 2010.

[Hin02] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[LNHL15] X Liu, R. Nian, B. He, and A. Lendasse. A rapid weighted median filter based on saliency region for underwater image denoising. In *OCEANS 2015 - MTS/IEEE Washington*, pages 1–4, Oct 2015.

[McG80] B. L. McGlamery. A computer model for underwater camera systems. volume 0208, pages 221–231, 1980.

[PK10] C. J. Prabhakar and P. P. Kumar. Underwater image denoising using adaptive wavelet subband thresholding. In *Signal and Image Processing (ICSIP), 2010 International Conference on*, pages 322–327. IEEE, 2010.

[RB05] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, pages 860–867, 2005.

[RFvL16] M. Radolko, F. Farhadifard, and U. F. von Lukas. Dataset on underwater change detection. 2016.

[RG15] M. Radolko and E. Gutzeit. Video segmentation via a gaussian switch background-model and higher order markov random fields. In *Proceedings of the 10th International Conference on Computer Vision Theory and Applications* VISAPP 2015, pages 537–544, 2015.

[SG99] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Vol. Two*, pages 246–252. IEEE Computer Society Press, June 1999.

[SSS13] M. Shanmugasundaram, S. Sukumaran, and N. Shanmugavadivu. Fusion based denoise-engine for underwater images using curvelet transform. In *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*, pages 941–946, 2013.

[SZW11] F. Sun, X. Zhang, and G. Wang. An approach for underwater image denoising via wavelet decomposition and high-pass filter. In *Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on*, pages 417–420. IEEE, 2011.

[WADP97] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785, 1997.

[WBSP14] R. Wang, F. Bunyak, G. Seetharaman, and K. Palaniappan. Static and moving object detection using flux tensor with split gaussian models. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 420–424, June 2014.

[Wel69] W. H. Wells. Loss of resolution in water as a result of multiple small-angle scattering. *JOSA*, 59(6):686–691, 1969.

[ZM97] S. C. Zhu and D. Mumford. Prior learning and gibbs reaction-diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1236–1250, 1997.

Head Movement Based Temporal Antialiasing for VR HMDs

Jung-Bum Kim Soo-Ryum Choi Joon-Hyun Choi Sang-Jun Ahn Chan-Min Park

Samsung Electronics
{jb83.kim, sooryum.choi, jh53.choi, sjun.ahn, chanmin.park}@samsung.com

ABSTRACT

Inherent properties of VR HMDs cause degradation of visual quality which disrupts immersive VR experience. We identify a new temporal aliasing problem caused by unintended tiny head movement of VR HMD users. The images that users see slightly change, even in the case that the users intend to hold and concentrate on a certain part of VR content. The slight change is more perceivable, because the images are magnified by lenses of VR HMDs. We propose the head movement based temporal antialiasing approach which blends colors that users see in the middle of head movement. In our approach, the way to determine locations and weights of colors to be blended is based on head movement and time stamp. Speed of head movement also determines proportions of colors in the past and at present in blending. The experimental result shows that our approach is effective to reduce the temporal aliasing caused by unintended head movement in real-time performance.

Keywords

Temporal antialiasing, head movement, virtual reality, head mounted displays

1 INTRODUCTION

VR, Virtual Reality, has been recently gaining enormous attention, since the advent of advanced VR HMD, Head Mounted Display, devices such as Oculus Rift [Ocu16a], Vive [Viv16a], and Gear VR [Gea16a]. The devices significantly enhance immersiveness of VR experience by displaying images full of users' field of view, which promptly reflect movement of users' head posture [Ear14a]. That is, the VR HMDs provide a part of VR content at which users look at a real-time frame rate. However, in terms of visual quality, improvement is required because of inherent properties of the recent VR HMDs. VR HMDs are equipped with optics systems which magnify display panels showing images of VR content. In order to manufacture lightweight and affordable hardware, the optics systems in the recent VR HMDs are relatively uncomplicated, which cause problems with visual quality including spherical and chromatic aberrations [Hen97a]. Although modern displays, in terms of resolution, are dense enough so that users are not able to recognize individual pixels in a panel, they are insufficient for VR HMDs which magnify displays using their lenses. As a result, screen-door effect, which is a problem of a grid of fine lines between pixels is observable, appears. Furthermore, insignificant visual artifacts such as aliasing become noticeable, although they are not serious defects in typical smartphone and desktop environment.

This paper concentrates on identifying and solving a visual quality degradation problem caused by inherent characteristics of VR HMDs. In general, users hold their heads when appreciating a certain part of VR content. However, it is unavoidable for users to make tiny movement which sensors in VR HMDs are able to detect. In response to the tiny head movement, VR HMDs slightly change the images that users see, even in the case that they intend to hold their heads. This slight change in the images is perceived as a temporal aliasing which disturbs comfortable VR experience. In this paper, we define the temporal aliasing which is caused by unintended tiny head movement as *head jittered aliasing*. Although many researches have investigated to resolve aliasing problems, constraints of VR HMDs have not been their concerns. Most of the previous researches are not suitable for eliminating head jittered aliasing. In addition, in VR environment, real-time performance is critical for immersive and long-lasting experience, since users feel motion sickness if high frame rate is not supported [Lav00a] . To preserve real time performance, an antialiasing technique for VR HMDs should be very fast as well as not burdensome. Therefore, a new antialiasing which takes into account VR HMDs is necessary.

In this paper, we propose a head movement based temporal antialiasing which blends the colors that a user sees in the middle of head movement. In terms of performance, the approach is executed at a real time frame

rate on a modern mobile VR HMD. Distinctive features of our approach in blending colors are 1) the location of the colors to be blended is determined by partially inverting head movement, 2) the way to derive the weight of the colors is based on speed of head movement and time stamp, 3) blending is localized based on the amount of temporal change of colors. To evaluate our approach, we define a measurement which computes the amount of temporal change of colors in the images. The measurement accounts for how effective an antialiasing is to reduce change in temporally consecutive images. As a value of the measurement lowers, it becomes more effective to reduce head jittered aliasing. The experimental result indicates that our approach outperforms other candidate approaches in reducing head jittered aliasing, and is accomplished at a real-time frame rate.

2 RELATED WORK

2.1 Spatial antialiasing techniques

Supersample Anti-Aliasing(SSAA) and Multisample Anti-Aliasing(MSAA) are basic antialiasing techniques. They have been used to reduce spatial aliasing as generic methods for the past years.
SSAA reduces aliasing artifacts by the following 3 steps; generating the image at a higher resolution, filtering multiple samples for each pixel and then downsampling to the final resolution. Since SSAA is performed for the whole pixel in the image, it has the highest-quality results. However, it is the most expensive method in terms of its processing and memory bandwidth requirements [Jim11a]. Recently most of graphical processors have stopped supporting SSAA to avoid performance degradation [Jia14a].
MSAA is a special case of SSAA that is only performed for pixels at the edges of polygons. By reducing number of samples, it becomes less expensive and faster than SSAA, but it could not improve aliasing artifacts inside geometries and textures. It is supported by all of the latest graphics processors and application programming interfaces(APIs) [Jim11a].
Morphological Anti-Aliasing(MLAA), developed by Intel Labs, blends colors around silhouettes which are detected with certain patterns [Res09a]. These patterns (Z-shapes, U-shapes and L-shapes) are used to search for color discontinuities and determine blending weights. It has advantages in terms of quality and implementation. MLAA provides the quality comparable to 4X SSAA. It also allows for better processor utilization, since it is independent from the rendering pipeline and parallel with rendering threads. However, MLAA might produce temporal artifacts between frames, because it uses only image data for reconstruction. In addition, it cannot identify pixel-size features which very small or thin geometries and unfiltered textures have. Thus, it could be resulted in moire pattern with these input.
Fast approXimate Anti-Aliasing(FXAA), developed by NVIDIA [Lot09a], reduces edge aliasing in a similar way to MLAA. However, it is simpler and faster. It detects edges by checking for a significant change in average luminance, and filters directions of sub-pixel on the edge perpendicularly. It can be easily implemented as one per-pixel filter. In addition, it is extremely fast, averaging just 0.11 ms per million pixels on NVIDIA GTX 480 [Jim11a]. It can handle edge alias even inside textures by processing with all the pixels on the screen. However, FXAA also cannot solve the temporal artifacts.
These spatial antialiasing techniques introduced in this section are not enough to solve temporal aliasing artifacts between consecutive frames, because they only uses a current frame image [Sch12a].

2.2 Temporal antialiasing techniques

Temporal aliasing is caused due to incoherence between continuous frames. This artifact is shown as flickering or crawling pixels temporally during camera and object motions. There are some approaches to reduce temporal aliasing.
A temporal antialiasing method for CryENGINE 3.0 has been popularly used in video games [Jim11a], because it is a simple method which is also known as Motion Blur. It is performed in real-time by using two images of previous and current frame and a velocity vector between them.
Amortized supersampling by Yang et al.[Yan09a] proposed an adaptive temporal antialiasing with supersampling, reusing shading samples from previous frames. It controls the tradeoff between blurring and aliasing with the smoothing factor calculated in a recursive temporal filter. However, it cannot properly handle temporal artifacts resulted in fast changes which cannot be predicted by reprojection.
Recently, Karis [Kar14a] presented high-quality temporal supersampling as a temporal antialiasing technique for Unreal Engine [Epi16a]. It generates super samples by jittering the camera projection, and then takes samples with a pattern such as Halton [Hal60a] Sequence. It accumulates the previous moving average of samples and uses it as the smoothing factor to reduce temporal alias.
Some approaches with supersampling, such as Yang's and Karis's methods, produce high quality results, but they have a limitation in terms of the performance with high-resolution images. In addition, temporal reprojection can cause ghosting artifacts, since it cannot accurately reproject when the images are significantly changed between consecutive frames.

3 HEAD JITTERED ALIASING

A majority of advanced VR HMDs consist of sensors, displays, and an optics system. Sensors in a VR HMD detect movement of users' head posture at a very high frequency. As users' head posture change, VR HMDs render images of VR content in the direction of users' field of view to the displays. The images in the displays are magnified by the optics system to fully occupy users' field of view. By instantly displaying images full of users' field of view in response to head movement, VR HMDs provide users with immersive experience. The magnification of the images influences density of pixels in the displays to be decreased, and makes it more vulnerable for insignificant visual artifacts to be noticed. We identify a visual quality problem that users with a VR HMD experience, when they intend to hold and concentrate on a certain part of VR content. It is unavoidable for users to make tiny head movement, even in the case they attempt to hold. The tiny head movement of users is detected by the sensors, then the images displayed to users are slightly changed. Figure 1 illustrates the slight change in the images in response to the tiny head movement. Since VR HMDs update images at real-time frame rate, the slight change is supposed to be noticed as temporal aliasing artifact. While it is not critically noticeable at typical devices such as smartphones, users with VR HMDs are able to easily perceive the temporal aliasing because of the magnification of the optics system. In Figure 1, it is difficult to observe the difference between the two images. However, in the magnified regions in the images, the change in colors of the images is more perceivable. Therefore, this is a problem that arises due to inherent properties of VR HMDs. The temporal aliasing problem occurs in various cases including computer generated-geometries, texts, and images.

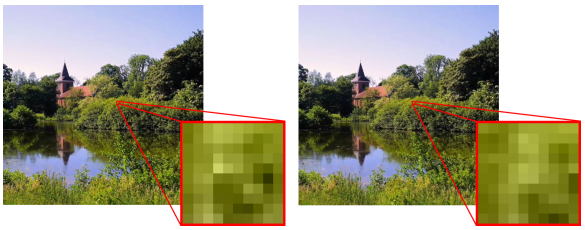


Figure 1: Slight change of temporally consecutive images in the case that users concentrate on a certain part of VR content.

In this paper, we define the temporal aliasing caused by unintended tiny head movement as *head jittered aliasing*, as it occurs because of head jittering of users. The previous antialiasing techniques, introduced in section 2, are not appropriate to remove this artifact. The spatial antialiasing techniques which aim to solve aliasing problems in a spatial manner are not effective for eliminating temporal aliasing. The temporal antialiasing techniques are not feasible for mobile VR HMDs. They are designed for desktop GPUs such as NVIDIA GeForce, which indicates that supporting real-time performance is not achievable. We propose a temporal antialiasing approach which is based on head movement to solve the problem. Our approach is suitable for mobile VR HMDs in terms of performance and effectiveness.

4 HEAD MOVEMENT BASED TEMPORAL ANTIALIASING

Head jittered aliasing is basically caused due to abrupt change of colors in images during tiny head movement. Blending colors with organized weights is a common technique to compensate abrupt change of colors. Basically, our temporal antialiasing approach blends colors that users see in the middle of head movement. Our approach is head movement based temporal antialiasing, which indicates selection of colors and deriving weights to be blended are based on head movement.

4.1 Interpolated reprojection

In order to select a color that a user sees during head movement, we partially invert head movement. In this paper, we assume that the type of head movement detected by VR HMDs is rotation. It is possible to extend our approach to 6 DoF head movement. To achieve a partial inverse of head movement, we introduce *interpolated reprojection* which transforms a sample at which a user is currently looking to the past locations in the middle of head movement. A location of a color that interpolated reprojection returns is a two-dimensional coordinate in an image space. We call a location of a color in an image space a sample. Interpolated reprojection is represented by the following function.

$$s = P \cdot slerp(V_{n-1}, V_n, d) \cdot V_n^{-1} \cdot v_p \qquad (1)$$

s is a past sample that is acquired after applying a partial inverse of head movement. d is a degree of inverting head movement, which is equivalent to closeness to the current head posture. d ranges from 0 to 1, and the value of 0 indicates a full inversion of head movement. v_p is a three-dimensional coordinate in $R^3 \in [-1,1]$, which is also known as a clip space. v_p represents a sample in the current image with a depth information and is obtained after applying a projection. V_n is a view matrix that denotes the current head posture. V_{n-1} is a view matrix that denotes the head posture in the previous frame. $slerp$ is a spherical linear interpolation function that calculates a matrix in the middle of the two view matrices V_n and V_{n-1}. d is used as a parameter that determines closeness to V_{n-1}. P is a projection matrix applying camera parameters. Interpolated reprojection is a process that finds a sample to be blended. By using multiple different values of d, it is possible to variously

control the number of samples to be blended. By substituting the *slerp* function to a function that returns an intermediate transform between two transforms, Interpolated reprojection is extended to support 6 DoF head movement.

4.2 Determination of blending weight

In our approach, determination of a weight of individual color to be blended is based on both time stamp and speed of head movement. Basically, we assign a greater or equal weight to a more recent sample. We compute c_{past}, accumulated colors of past samples, using the following equation.

$$c_{past} = \sum_{0}^{n-1} W(d_i) C_k(s_i) \quad (2)$$

n is the number of past samples. s_i is a past sample which is an output of Equation (1). d is a closeness to the current head posture. d_i is used to compute a sample s_i. That is, a past sample with a value of d closer to 1 is more recent one. C_k is a function that returns a color of a sample from an image displayed in the k^{th} frame. W is a monotone function that returns a weight of a sample based on value d. Given the number of samples n, the total sum of the values returned by the function W is 1. According to the function W, a more recent sample has a greater or equal weight.

Head jittered aliasing becomes serious when users attempt to hold their head movement. In addition, blending with past colors is possible to cause an excessive blur which deteriorates quality of images. Therefore, we decrease strength of blending, as speed of head movement gets faster. In order to find c_k - a color in the k^{th} frame, we blend the current color with the accumulated past color through the following equation.

$$c_k = (1 - A(h)) \sum_{0}^{n-1} W(d_i) C_{k-1}(s_i) + A(h) C_k(s_c) \quad (3)$$

The first term of the equation comes from Equation (2). As the function C_{k-1} is used in the first term, colors of past samples are obtained from the $k-1^{th}$ frame. Accordingly, s_c is a sample in the k^{th} frame. Since a color c_k is derived from colors in the k^{th} and $k-1^{th}$ frames, blending in our approaches recursively accumulates colors in a period of frames. A is a function that determines a weight of a color of s_c from h, speed of head movement, as an input. The function A should be monotonously increasing for speed of head movement, and has a curve similar to a ease-in and ease-out curve. Figure 2 illustrates a graph of the function A, which satisfies the conditions. The graph has different forms depending on a parameter specifying a minimum weight, w_{min} in Figure 2.

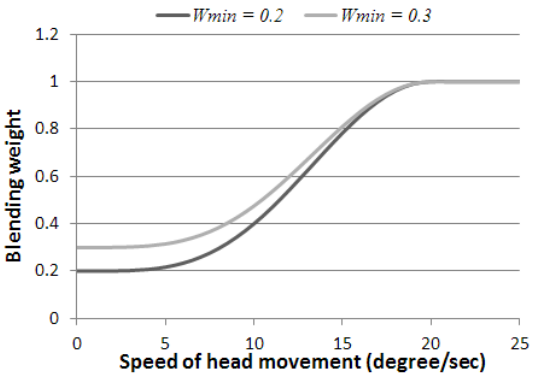


Figure 2: A graph representing the function A in Equation (3)

From experiments, we conclude that an appropriate function for A is as follows.

$$w = (1 - w_{min})\{-\frac{1}{2} * (\cos(\pi \frac{h}{h_{max}}) - 1)\}^2 + w_{min} \ , \quad w = 1 \quad if \quad h > h_{max} \quad (4)$$

w is a result weight value. h is speed of head movement, and h_{max} is a maximum value of speed of head movement. w_{min} is a minimum value of a weight. Using the function A, a weight of an accumulated past color gets smaller, as speed of head movement becomes faster.

4.3 Localization of blending weight

The blending function in Equation (3) assigns a constant weight to the entire colors in an image. However, the amount of change of each color in an image during head movement is diversified. And we observe that temporal aliasing is more noticeable in areas that have larger color change. In order to enhance effectiveness of our temporal antialiasing, we locally assign a weight depending on a temporal difference of individual colors. A sample with a larger color difference has a smaller weight. w_{min} in Equation(4) is substituted to w'_{min} to achieve localization of blending weight as follows.

$$w'_{min} = w_{min} - (w_{min} - w_{lb}) * c_{diff} \quad (5)$$

c_{diff} is a temporal difference of a color. w_{lb} is a lower bound of a minimum weight, which indicates that the largest value of c_{diff} has a weight value of w_{lb}.

Our localized weight determination is devised to be suitable for parallel processing on GPUs. VR HMDs normally produce images using GPU for better performance. Calculation of each color in images is parallelly executed on shaders of GPUs. However, computation of functions with high complexity such as Equation (4) for all the colors in a image is burdensome, which leads performance degradation. To secure high performance, our approach minimizes the amount of weight computation on shaders of GPUs, by separating a complex part of the computation - Equation (4) in this case

- that is globally applied to all the colors in an image. The complex part is operated on CPU, and the result of computation is delivered to shaders. As a result, in our approach, the relatively simple function - Equation (5) - is performed on GPUs for weight computation.

4.4 Compatibility with dynamic scenes

Blending of temporally consecutive images possibly causes a motion blur problem. To avoid this problem, one of common methods is to analyze a velocity of individual colors, and to selectively apply blending. However, in VR HMD environment, high performance is a top priority for users not to experience motion sickness. Therefore, complicated analysis requiring enormous computation is not feasible. Our approach is designed to be compatible with a map which is a form of a 2D image specifying dynamic areas. By referencing the map, we selectively apply blending to static regions in images. For a performance reason, we take advantage of scene information. Our approach rasterizes a region on which static objects are projected on the map, which is a process marking static regions on the map. Since rasterization of the map, with less cost, is accompanied with rendering of a scene with the help of a functionality of GPUs, producing the map in our approach is able to preserve performance.

5 EVALUATION

5.1 Mean Temporal Color Difference

To quantitatively evaluate effectiveness of temporal antialiasing approaches, we define a new measurement - MTCD, Mean Temporal Color Difference, which computes the average amount of change in colors from temporally consecutive images. Some researches employ PSNR, Peak signal-to-noise ratio, for evaluation of antialiasing. However, PSNR to an optimal image is not appropriate for measuring effectiveness of temporal antialiasing. It is even possible that a sequence of temporally consecutive images having considerable temporal aliasing is able to achieve low PSNR to corresponding optimal images. Suppose that all the images in a temporal sequence of images has a specific PSNR value ε to corresponding images in an optimal sequence of images. All the difference between pixels in k^{th} images is negative and that of $k-1^{th}$ images is positive. In this case, temporal change of colors in images is possibly regarded larger than PSNR indicates. Therefore, we need a new measurement that takes into account temporal coherence of image sequences. MTCD of an image sequence is defined as Equation (6).

$$MTCD(I,t) = \frac{1}{nm(t-1)} \sum_{k=1}^{t-1} \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} d_{ijk},$$
$$d_{ijk} = \sqrt{(r_{ij(k-1)} - r_{ijk})^2 + (g_{ij(k-1)} - g_{ijk})^2 + (b_{ij(k-1)} - b_{ijk})^2} \quad (6)$$

I is a sequence of temporally consecutive images in a period of t frames. A width and a height of an image in I are n and m. i and j denote x and y coordinates in an image, and k represents the k^{th} image in a sequence of images. r,g,b represent color components for red, green, and blue respectively. An optimal value of MTCD is definitely zero. As MTCD gets smaller, a sequence of images is more robust to temporal aliasing.

5.2 Experimental result

For evaluation of efficiency and effectiveness of temporal antialiasing, we build a platform that consists of a mobile VR HMD (Gear VR), a smartphone (Galaxy S7), and an image viewer. The resolution of the VR HMD in our experiments is 1024x1024 for each left and right eye. The device is equipped with Exynos 8890 processor which includes a 2.3GHz Quad-core and 1.6 GHz Quad-core CPU, and a Muli-T880 MP12 GPU. The dataset in the experiments includes 11 images. With the dataset, we perform experiments measuring performance using frame rate, and effectiveness of reducing temporal aliasing using MTCD.

5.2.1 Performance

Real-time performance is essential for immersive and long-lasting VR experience. Temporal antialiasing is an additional process that requires more execution time. Therefore, performance requirement is very intensive to preserve real-time performance of applied applications. In the experiments, we measure additional execution time after applying our approach. The additional execution time of our approach is approximately 2 *msec* in average. As antialiasing is applied for each left and right eye in VR environment, the measured execution time contains the amount of time for applying our approach twice for both eyes. For quality experience, we observe that the entire execution time for an application to render one frame should be less than 32 *msec*. Because the additional execution time of our approach is approximately less than 10% of 32 *msec*, we conclude that our approach performs at reasonable performance for immersive VR experience.

5.2.2 Effectiveness

We utilize MTCD measurement to quantitatively measure effectiveness of our approach for the purpose of reducing head jittered aliasing. To measure MTCD, we set up experiments simulating an image viewing application. In the experiment, participants are requested to hold and concentrate on a certain part of an input image. Then, we compute MTCD from a sequence of images displayed to users while the participants attempt to hold. A sequence of images in experiments contains 10 images.

For comparison, we choose MSAA [Jim11a] which is the most common antialiasing in mobile environments

because of its high performance. Other recent antialiasing techniques are possibly considered as candidates. The techniques, however, are not feasible for adopting mobile environment such as Gear VR for a performance reason, since they are intended to operate on desktops or consoles. In experiments, three variations - no antialiasing, MSAA, and our approach - are compared. The comparison result of the approaches is plotted in Figure 3. The parameter values used are as follows. d in Equation(3) is $2/3$. One past sample is used in the experiments, so n in Equation(3) is 1. h_{max} in Equation (4) is 25. For Equation (5), values of w_{min} and w_{lb} are 0.3 and 0.1 respectively. For measuring MTCD values, the number of images in a sequence - t - is 10.

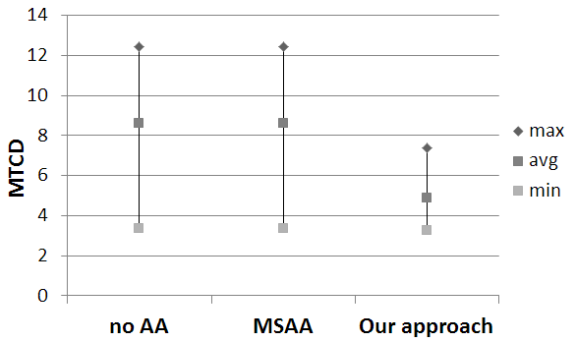


Figure 3: A comparison of MTCD results

The experimental result shows maximum, minimum, and average MTCD values of the approaches for the dataset. Our approach achieves the lowest MTCD in average, which implies it is the most effective to reduce temporal aliasing. The average MTCD of our approach is approximately 56% of MSAA. The results of no antialiasing and MSAA are almost identical, because antialiasing is applied to edges in case of MSAA and our dataset mostly consists of textures. Minimum values of all the approach are almost same, although a minimum MTCD value of our approach is slightly lower than other two approaches. One of the images in the dataset has almost same color in the entire image, and its spatial color change is insignificant. This image contributes to the result that minimum values of MTCDs are almost indistinguishable.

It is possible for our approach to be applied in combination with MSAA. Combined with MSAA, our approach is expected to perform most effectively.

Figure 4 illustrates a comparison of result images. In Figure 4(a), the result images of MSAA are represented. And $k-1^{th}$ (left) and k^{th} (center) images in a temporal sequence of images are depicted. The smaller images with red borders on the right side of each image magnify red rectangular regions on the corresponding images. The difference between the two images on left and center is shown on the rightmost side. The images in Figure 4(b) are the result of our approach. The difference images on the rightmost side describe the amount of change of colors in temporally consecutive images. Darker regions represent larger difference. The difference images indicate that the result of our approach is more effective for reducing temporal aliasing.

6 CONCLUSION

In this paper, we define head jittered aliasing which is a new temporal aliasing problem identified due to properties of VR HMDs. To alleviate head jittered aliasing, we propose head movement based temporal antialiasing which blends colors that users see in the middle of head movement. Our approach determines weights for blending based on head movement, time stamp, and speed of head movement. In addition, the derived weight is localized based on the amount of temporal color difference. For quantitative evaluation of effectiveness, we define a new metric - MTCD - which measures the average amount of change in colors from temporally consecutive images. In the experimental results, our approach has the lowest MTCD among other competitive antialiasing approaches, which implies that our approach is the most effective for reducing head jittered aliasing. In terms of performance, the additional execution time after applying our approach is 2.5 msec in average, which is reasonable for quality VR experience.

7 FUTURE WORK

Our approach takes advantage of a map specifying dynamic regions to be compatible with dynamic scenes. For an easier and a more portable application of our approach, we plan to develop a method identifying dynamic regions independent upon scenes at high performance.

Also, we expect that reducing head jittered aliasing is effective to alleviate visual fatigue which is one of the serious problems of VR HMDs. To validate the expectation, we plan to conduct qualitative analysis on effectiveness of our approach for relieving visual fatigue, and figure out correlation between MTCD and the qualitative measurement.

8 REFERENCES

[Ocu16a] Oculus Rift website: http://www.oculus.com/rift/, 2016.

[Viv16a] HTC Vive website: https://www.vive.com/, 2016.

[Gea16a] Samsung Gear VR website: http://www.samsung.com/global/galaxy/gear-vr/, 2016.

[Ear14a] Earnshaw, Rae A., ed. Virtual reality systems. Academic press, 2014.

[Hen97a] Hendee, William R., and Peter NT Wells. The perception of visual information. Springer Science & Business Media, 1997.

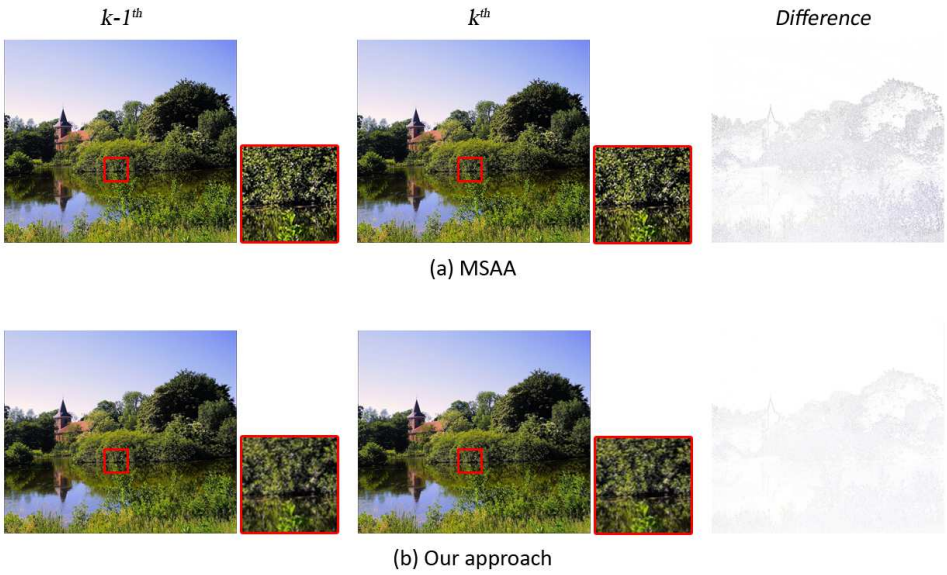


Figure 4: A comparison of result images. Two temporally consecutive images and their difference of the cases of no antialiasing (a) and our approach (b)

[Lav00a] LaViola Jr, Joseph J. A discussion of cybersickness in virtual environments. ACM SIGCHI Bulletin 32.1: 47-56, 2000.

[Jim11a] Jimenez, Jorge, et al. Filtering approaches for real-time anti-aliasing. ACM SIGGRAPH Courses 2.3: 4, 2011.

[Jia14a] Jiang, X. D., Sheng, B., Lin, W. Y., Lu, W., Ma, L. Z. Image anti-aliasing techniques for Internet visual media processing: a review. Journal of Zhejiang University SCIENCE C, 15(9), 717-728, 2014

[Res09a] Reshetov, Alexander. Morphological antialiasing. Proceedings of the Conference on High Performance Graphics, ACM, 2009.

[Lot09a] Lottes, T. FXAA-Whitepaper. Tech. rep., NVIDIA, 2011. 2, 2009.

[Sch12a] Scherzer, Daniel, et al. Temporal Coherence Methods in Real-Time Rendering. Computer Graphics Forum. Vol. 31. No. 8. Blackwell Publishing Ltd, 2012.

[Yan09a] Yang, Lei, et al. Amortized supersampling. ACM Transactions on Graphics (TOG). Vol. 28. No. 5. ACM, 2009.

[Kar14a] Karis, B. High-quality temporal supersampling. Advances in Real-Time Rendering in Games, SIGGRAPH Courses 1, 2014.

[Epi16a] EPIC GAMES: Unreal Engine 4. https://www.unrealengine.com/, 2016.

[Hal60a] Halton, J. H. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik, 2(1), 84-90, 1960.

Accelerating Radiosity on GPUs

Alexandr Shcherbakov
Lomonosov Moscow State University
GSP-1, Leninskie Gory
119991, Moscow, Russia
alex.shcherbakov@graphics.cs.msu.ru

Frolov Vladimir
Lomonosov Moscow State University
Keldysh Institute of Applied Mathematics
(Russian Academy of Sciences)
GSP-1, Leninskie Gory
119991, Moscow, Russia
vfrolov@graphics.cs.msu.ru

ABSTRACT

We propose a novel approach to implement radiosity on GPU with specific optimizations via form-factor matrix transformations. The proposed transformations enable to reduce the amount of computations for multiple-bounce global illumination and apply DXT compression (with subsequent hardware decompression when reading form-factors on GPU). Our implementation is 10 times faster running and requires 3 times less memory than the naive radiosity GPU implementation.

Keywords

Global illumination, radiosity, real-time applications.

1 INTRODUCTION

The main difficulty of real time global illumination involves accurate evaluation of reflected light. Precise value can be computed only through the lighting integral evaluation, which becomes much more complicated with each new reflection. Therefore, in practice different approximate methods are used. Today, there are a variety of popular techniques for global illumination evaluation. Each of them represents the evolution of some basic method of global illumination with several modifications, which make it more suitable for specific conditions and hardware.

2 INTERACTIVE GLOBAL ILLUMINATION METHODS

2.1 Instant Radiosity

Despite its name, instant radiosity [1] is not a variation of radiosity method [4]. The idea behind this method is to approximate indirect illumination with the direct light from a large number of "secondary" point light sources placed on the surface. The Reflective Shadow Maps (RSM) algorithm [3] is the most popular extension of instant radiosity method for GPU applications.

RSM creates a set of secondary light sources from mip-mapped texture of shadow map [9]. These lights are used further in the fragment shader just like any other lights. The shadows from indirect light sources are not included in this case. The RSM method is the fastest and requires the least amount of memory over all existing methods of real-time global illumination solutions. However, its main disadvantage is poor accuracy.

2.2 Light Propagation Volumes

The main idea of Light Propagation Volumes (LPV) is to represent the lighting in a scene sampled on the lattice or grid. The algorithm consists of the four main steps:

1. Generate a radiance point set by rendering the scene into the reflective shadow map;
2. Inject virtual light sources into the radiance field;
3. Propagate radiance by iteratively solving a differential scheme inside the grid;
4. Apply a result radiance volume to the scene.

The disadvantage of LPV is $O(n^3)$, where n is the size of grid, complexity and memory consumption. The significant performance disadvantage appears in the propagation light through large empty spaces. Through, the Cascaded LPV [2] algorithm amortizes some of these problems, the accuracy of this method is not enough for many cases (especially for architecture-related applications).

2.3 Voxel Cone Tracing

Voxel Cone Tracing (VCT) [5] allows for voxelized approximation of the scene and trace cones via ray marching through the different mip-map levels of 3D texture. The key idea of VCT is to pre-integrate the incoming light via mip-mapping: gather the nearby light from the detailed mip level and the far light from the coarse mip level which averaged the emitted light from many surfaces. Therefore, distant areas are used with less precision.

The main disadvantage of VCT is a computational cost, because it traces several cones per pixel. The algorithm also suffers from light leaks due to coarse approximation of geometry by voxels.

2.4 Spherical Harmonics

The method of Spherical Harmonics [7] is a popular global illumination method based on the approximation of complicated functions by decomposition into a sum of simple spherical functions. For each vertex in scene polygons (or grid positions called light probes [8]) its own representation of lighting function is computed and approximated by spherical harmonics. The next stage of evaluating the global illumination is relatively cheap. Thus, the large part of computation is executed on a precomputing stage.

It generates acceptable global illumination. On the edge of light and shadow, it can create some artifacts caused by rough approximation of illumination functions.

2.5 Radiosity

Despite the considerable effort of researchers in real-time global illumination, an original radiosity method [4] has some advantages over the previously discussed methods. The first advantage is the conservation of energy and sufficiently high accuracy of the solution. The second one is low computational cost for a small number of patches, because the main evaluation runs on the precompute stage. However, it is difficult to use radiosity directly due to the fact that in modern 3D-scenes there are millions of polygons.

One of the modern versions of radiosity is the "Enlighten" [11] graphics engine in which simplified geometry is used for global illumination computing. Simplification is provided manually by 3D artists using some tools in 3D content modeling programs. Their implementation uses CPU for computing and updates indirect illumination once per 5-10 frames.

3 PROPOSED SOLUTION

3.1 GPU Radiosity

There were several works related to the implementation of radiosity on GPU. In [6] the progressive refinement radiosity algorithm running completely on GPU is presented. The work is mostly focused on form-factors computation with adaptive subdivision and using fixed functionality of GPU (like Rasterizer). In [13] an extended GPU progressive radiosity is presented. Their solver integrates ideal diffuse as well as specular transmittance and reflection and is capable to handle multiple specular reflections with correct mirror-object-mirror occlusions. In [14] another adaptive subdivision implementation on GPU is presented.

Unlike those discussed above, the work [10] is focused on the efficient linear system equation solution and matrix operations. It deals with the performance of different floating point format for matrix of form-factors and investigates the differences between GPU and CPU performance on matrix operations; it also explores a hierarchical radiosity approach via multiresolution meshed atlas. The authors of [10] pack 4 sequential form factors to a single color of texture. However, they didn't explore hardware compression and didn't propose algorithmic optimizations (except hierarchical radiosity). The work [10] also suggests sub surface scattering computation possibility via radiosity.

We believe that all the GPU radiosity works discussed above can be significantly improved. In this paper, we propose a new multibounce method based on special modifications of the radiosity algorithm [4] for global illumination computing. Our extensions are aimed to accelerate radiosity, reduce the required memory for form-factors storing and increase the accuracy of computation in comparison to popular global illumination methods.

3.2 Automatic Geometry Simplification

It is impractical to use radiosity for scenes that consist of millions of polygons, due to the high computational complexity of this problem. Therefore, in practice, radiosity is applied to a simplified scene and the produced result is used for original scene. We used the geometry simplification method based on the voxel representation of original scene [12].

3.3 Key Terms and Definitions

Global illumination computing is performed according to the following scheme. For each patch, initial luminance is computed or set. These values form vector *emission*. Each element of this vector is a three-component vector, one component per color.

$$emission_i = (red, green, blue) \tag{1}$$

Vector *excident* consists of colors that are sent from patches. It can be computed using *emission*.

$$excident_i = emission_i * color_i \tag{2}$$

$color_i$ is the color of i-th patch. Then, form-factors matrix F is multiplied by $excident$. The result of this operation is the lighting received by patches from light sources $incident$.

$$incident_i = \sum_{j=0}^{n} F_{ij} \cdot excident_j \tag{3}$$

n is the number of patches.

$$excident_i^{(1)} = incident_i^{(0)} \cdot color_i \tag{4}$$

$excident$ is the lighting sent from patches after first reflection.

Then we repeat multiplication of form-factors matrix and $excident$ vector for computing lighting after first reflection. We can repeat these operations for an arbitrary number of reflections.

3.4 Optimizing radiosity for multi-bounce global illumination

First, we define color form-factors matrix F^c. Element of this matrix on row i and column j are defined in the following way:

$$\begin{aligned} F_{ij}^c &= F_{ij} \cdot color_j = \\ &= (red_j \cdot F_{ij}, green_j \cdot F_{ij}, blue_j \cdot F_{ij}) \end{aligned} \tag{5}$$

Then, we can change the computing of patches lighting after the first reflection using equations (3), (4) and (5).

$$incident^{(1)} = F^c \cdot emission \tag{6}$$

Furthermore, we can generalize this equation for computing lighting received by patches after arbitrary reflection.

$$incident^{(h)} = F^c \cdot incident^{(h-1)} = (F^c)^h \cdot emission \tag{7}$$

Total lighting of the patch for k reflections is summarized from values of lighting received after each reflection.

$$\begin{aligned} incident_{total} &= \sum_{h=0}^{k} incident^{(h)} = \\ &= \sum_{h=0}^{k} \left((F^c)^h \right) \cdot emission) = \\ &= \left(\sum_{h=0}^{k} (F^c)^h \right) \cdot emission = \\ &= F^{k-reflection} \cdot emission \\ incident_{total} &= F^{k-reflection} \cdot emission \end{aligned} \tag{8}$$

So, we can use matrix $F^{k-reflection}$ to perform multiplication only once for k reflections. However, this matrix needs 3 times more memory than the original one.

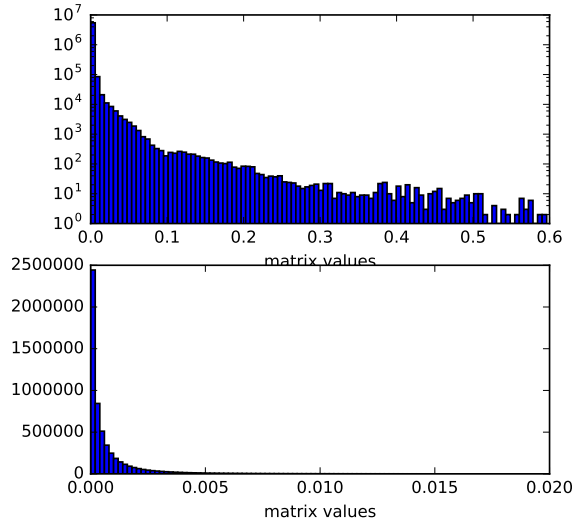


Figure 1: Distribution of form-factors values on the test scene (logarithm and linear scales).

3.5 Using DXT1 compression for form-factor matrix compression

For the form-factor matrix values less than 0.005 are prevailed (see Fig. 1). Since the contribution of these values in result is less than the others, they can be effectively compressed with losses.

For this reason, we split a form-factor matrix in two parts.

The first part contains 4% biggest numbers for each color channel for each row in the matrix. This value is based on the experimental results shown on figure 2. It provides a high quality of image.

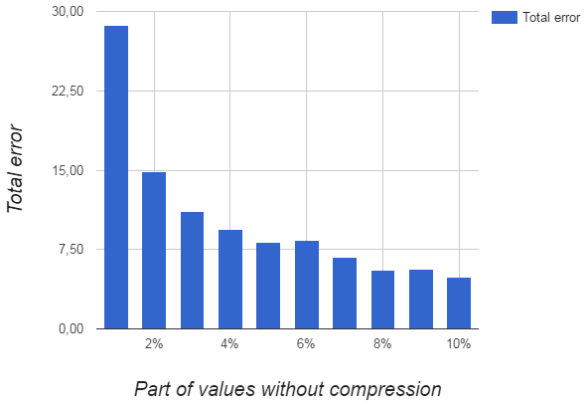


Figure 2: Total compression error for different parts of uncompressed values.

The second part has the same shape as the original matrix, but values from the first part are set to zero. Since these values fail to make significant contribution, accuracy for exponent is more important than accuracy for mantissa. Therefore, we apply some transformations to them.

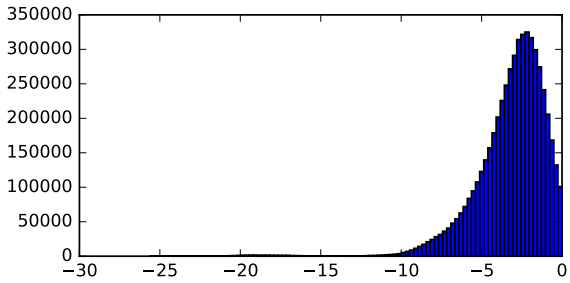


Figure 3: Distribution of lower form-factors values after normalization and logarithmization.

Firstly, values for each channel are divided on the maximum value for this channel. Secondly, the logarithm function is applied to the values (see Fig. 3).

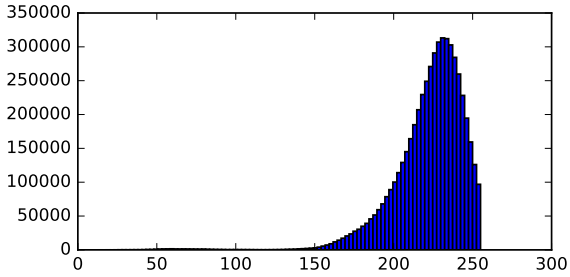


Figure 4: Result distribution of values in form-factors matrix.

Since the computed values are lie between -30 and 0 (see Fig. 3), we add a positive constant to them (in this case we add 25, because the values that remain negative are not important for computation). Then these values become to range from 0 to 255 (see Fig. 4).

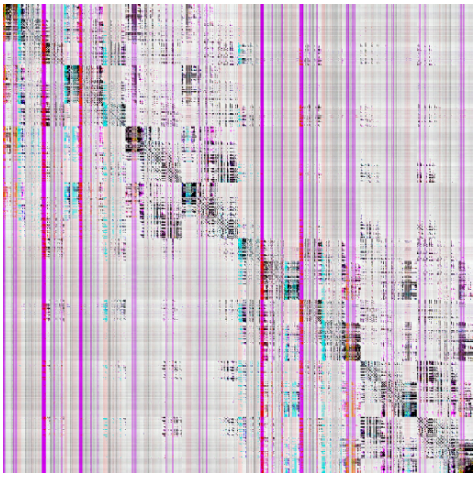


Figure 5: Form-factor matrix before reordering.

DTX1 compression is applied to a transformed form-factor matrix (see Fig. 5). To reduce the losses, we swap some rows of matrix. In this process, we also swap columns and patches matched these rows (see Fig. 6).

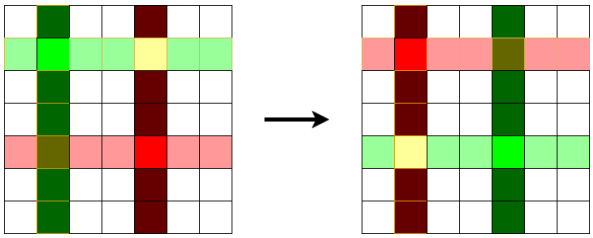


Figure 6: Scheme of columns/rows swapping.

Since we make matrix reorganization to reduce compress losses, we want to store similar values closer and hence we strive to place similar rows and columns next to each other. We use Euclidean distance as a measure of similarity.

In general case, if we represent columns or rows of matrix as vertices of full graph and set the measure of similarity as edges weights, then the problem of finding of the optimal order of rows and columns is the problem of finding of the shortest path in the graph, which includes all vertices. This problem is assigned to NP class.

Since the number of patches is the number of the order of several thousand, we use a heuristic approach. The first row always stays in the place. Most similar to the first row is put in the second place. Most similar to the second of the remaining row is put in the third place, etc. These matrix transformations decrease compression losses about 5 times (see Fig. 7).

	Without reordering	Reordering by rows	Reordering by rows & columns
Avg. error	1,28E-06	8,13E-07	5,88E-07
Max error	0,4622	0,4684	0,1524
Total error	47,12	18,86	9,40

Figure 7: Comparison of error for reordering.

Then, we apply second reordering, but measure is computed for columns.

After all reorderings, resulting texture is saved using DXT1 compression. Matrix after compression is presented in the figure 8. It is difficult to tell the difference, so the difference for the red component of the texture is shown in the figure 9. For green and blue channels, difference is similar in structure.

3.6 Implementation Details

All computations are executed on GPU using OpenGL framework. Form-factors matrix stored as a compressed texture. Other data stored in array buffers. The order of computation is the following:

1. Rendering of shadow maps for all light sources;
2. Running compute shader. This shader uses shadow maps for computing emission on the patches. Emission is square of illuminated part of patch;

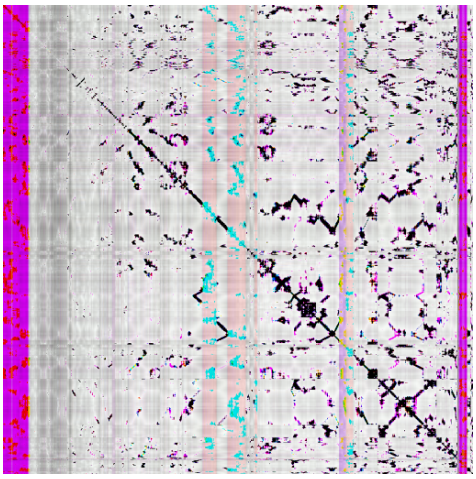


Figure 8: Compressed form-factor matrix after two reordering.

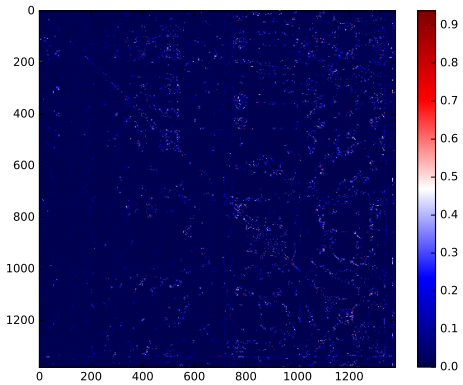


Figure 9: Form-factor matrices difference in red component.

3. Next, we run another compute shader for radiosity. On this step compressed form-factor matrix is used. The form-factor matrix is multiplying by emission vector;

4. And the last compute shader transfers indirect illumination from the simplified scene to the original;

5. The final step is the scene rendering.

4 EXPERIMENTAL RESULTS

In this paper, Naive Radiosity and Path Tracing are compared with our approach.

4.1 Comparison with naive radiosity implementation

4.1.1 Computation speed comparison

Our multibounce radiosity implementation is asymptotically faster than both naive radiosity and the algorithm described in [10]. Presented method can be used with other optimization techniques for radiosity (for example, hierarchical radiosity).

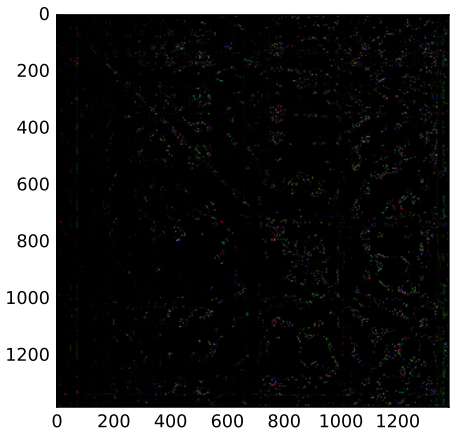


Figure 10: Form-factor matrices difference in all components.

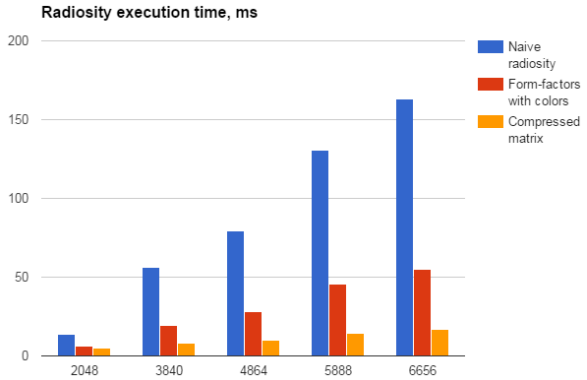


Figure 11: Speed comparison.

It can be seen in the figure 11, our method significantly superiors the naive implementation of radiosity. In addition, using the compressed by DXT1 form-factor matrix is faster due to decreased memory amount that we need to transfer between DRAM and GPU multiprocessor.

4.1.2 Memory Requirements Comparison

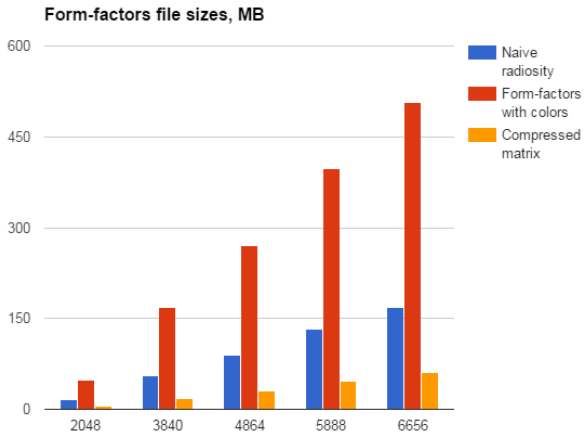


Figure 12: Memory comparison.

In figure 12 we show the sizes of files containing form-factor matrices. After compression, matrix requires significantly less memory. Thus, our modifications reduce the amount of data being stored and loaded into video memory.

4.1.3 Images Comparison

Our approach generates an image with similar quality as naive radiosity implementation, but we use less amount of memory and execute the radiosity algorithm faster.

4.2 Comparison with Light Propagation Volumes

Light Propagation Volumes generates image with visually perceptible inaccuracy as can be seen on figure 13. A column in selection (green rectangle) is pink, but on reference image this column painted with a gradient. Both images (our approach and LPV) rendered with 40 FPS. The size of voxel grid for LPV was chosen to reach the same fps with our implementation of radiosity.

4.3 Comparison with Path Tracing

In comparison with Path Tracing (see Fig. 13) we show that our method generates a similar image. Meanwhile, our image was generated less than 17ms. Path Tracing needs more than 5 minutes to generate a reference image on the same GTX670.

5 CONCLUSION

In this paper, we present a practical approach for real-time global illumination on GPU using radiosity. First, we reduced the multiple bounce computation cost by introducing color information to a form-factor matrix and powering it. Second, we apply rows/columns reordering and DXT compression to both save memory and increase speed when reading form-factors from DRAM on GPU (because radiosity is memory-bound). Our implementation runs 10 times faster than naive radiosity and requires 3 times less memory. The resulting image hardly differs from the path-traced reference and has comparable FPS to other real-time global illumination algorithms.

6 ACKNOWLEDGMENTS

This work is supported by RFBR 16-31-60048 mol_a_dk.

7 REFERENCES

[1] Alexander Keller. 1997. Instant radiosity. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 49-56. DOI=http://dx.doi.org/10.1145/258734.258769

[2] Anton Kaplanyan and Carsten Dachsbacher. 2010. Cascaded light propagation volumes for real-time indirect illumination. In Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games (I3D '10). ACM, New York, NY, USA, 99-107. DOI=http://dx.doi.org/10.1145/1730804.1730821

[3] Carsten Dachsbacher and Marc Stamminger. 2005. Reflective shadow maps. In Proceedings of the 2005 symposium on Interactive 3D graphics and games (I3D '05). ACM, New York, NY, USA, 203-231. DOI=http://dx.doi.org/10.1145/1053427.1053460

[4] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. 1984. Modeling the interaction of light between diffuse surfaces. In Proceedings of the 11th annual conference on Computer graphics and interactive techniques (SIGGRAPH '84), Hank Christiansen (Ed.). ACM, New York, NY, USA, 213-222. DOI=http://dx.doi.org/10.1145/800031.808601

[5] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. 2011. Interactive indirect illumination using voxel-based cone tracing: an insight. In ACM SIGGRAPH 2011 Talks (SIGGRAPH '11). ACM, New York, NY, USA, , Article 20 , 1 pages. DOI=http://dx.doi.org/10.1145/2037826.2037853

[6] Greg Coombe, Mark J. Harris, and Anselmo Lastra. 2004. Radiosity on graphics hardware. In Proceedings of Graphics Interface 2004 (GI '04). Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 161-168.

[7] Ian G. Lisle and S.-L. Tracy Huang. 2007. Algorithms for spherical harmonic lighting. In Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia (GRAPHITE '07). ACM, New York, NY, USA, 235-238. DOI=http://dx.doi.org/10.1145/1321261.1321303

[8] Jaroslav Krivanek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. 2008. Radiance caching for efficient global illumination computation. In ACM SIGGRAPH 2008 classes (SIGGRAPH '08). ACM, New York, NY, USA, , Article 75 , 19 pages. DOI: https://doi.org/10.1145/1401132.1401228

[9] Michael Wimmer, Daniel Scherzer, and Werner Purgathofer. 2004. Light space perspective shadow maps. In Proceedings of the Fifteenth Eurographics conference on Rendering Techniques (EGSR'04). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 143-151.

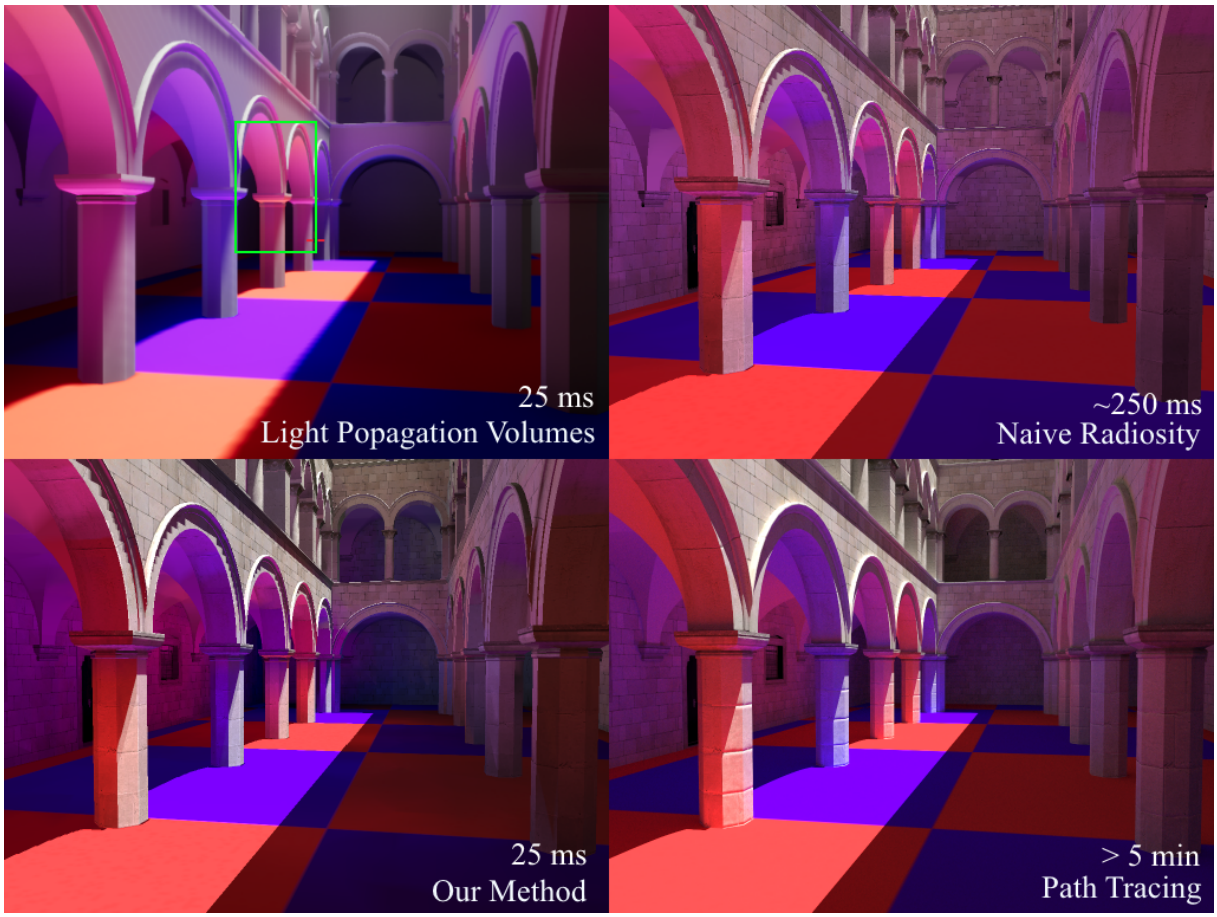


Figure 13: Our method (left-bottom), Light Propagation Volumes (Unreal Engine 4) (left-top), naive radiosity (right-top) and Path Tracing (right-bottom).

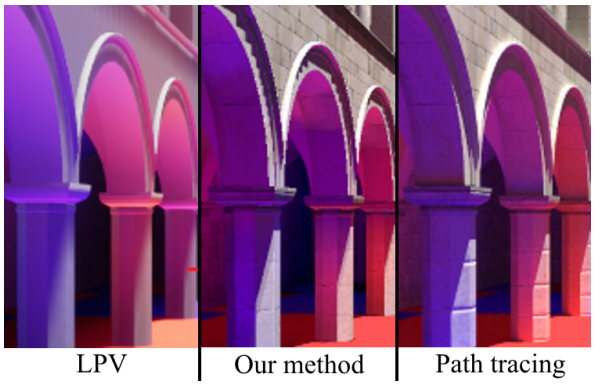


Figure 14: Comparison with Light Propagation Volumes and Path Tracing

DOI=http://dx.doi.org/10.2312/EGWR/EGSR04/143-151

[10] Nathan A. Carr, Jesse D. Hall, and John C. Hart. 2003. GPU algorithms for radiosity and subsurface scattering. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware (HWWS '03). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 51-59.

[11] SamMartin, Per Einarsson. A Real Time Radiosity Architecture for Video Games. Siggraph 2010. http://advances.realtimerendering.com/s2010/Martin-Einarsson-RadiosityArchitecture (SIGGRAPH%202010%20Advanced%20RealTime%20Rendering%20Course).pdf

[12] Shcherbakov, A., and Frolov, V. Automatic geometry simplification for computation of indirect lighting using radiosity. In Graphicon-2016 (2016), NNGASU, pp. 34-38

[13] Wallner, G. Vis Comput (2009) 25: 529. doi:10.1007/s00371-009-0347-z

[14] Wallner, G. GPU radiosity for triangular meshes with support of normal mapping and arbitrary light distributions. J. WSCG 16(1-3), 1-8 (2008)

Accurate Triangular Regular Network adjustment to Large Digital Elevation Models

José M. Santana
University of Las Palmas de G.C. Imaging Technology Center (CTIM) Spain (35017), Las Palmas de G.C.
josemiguel.santana@ulpgc.es

Agustín Trujillo
University of Las Palmas de G.C. Imaging Technology Center (CTIM) Spain (35017), Las Palmas de G.C.
agustin.trujillo@ulpgc.es

José P. Suárez
University of Las Palmas de G.C. Division of Mathematics, Graphics and Computation (MAGiC), IUMA Spain (35017), Las Palmas de G.C.
josepablo.suarez@ulpgc.es

Sebastián Ortega
University of Las Palmas de G.C. Division of Mathematics, Graphics and Computation (MAGiC), IUMA Spain (35017), Las Palmas de G.C.
sebastian.ortegatrujillo@gmail.com

ABSTRACT

Nowadays, large volumes of terrain data are available to use as *Digital Elevation Models*, from which coarser meshes can be progressively generated for visualization and other purposes. Previous studies compared different methods to adjust those meshes, concluding that no method performs the best for all kinds of terrain. In this work, a pipeline to accurately adjust TRNs to DEM is proposed. An initial approximation is calculated by solving a linear system from input data. Vertices with a major contribution to the global error of the mesh are then tuned using a local refinement algorithm. Experimentation shows that meshes adjusted using the proposed pipeline fit better the original DEM than ones generated using classic methods as linear interpolation for several benchmark elevation models.

Keywords

Triangular Regular Networks, adjustment, Digital Elevation Model, level of detail, terrain visualization.

1 INTRODUCTION

The growing capacity to adquire and store data has allowed that big volumes of terrain data are currently available for the general public, in which the earth is modelled with fine resolutions. These data are presented as *Digital Elevation Models* or DEM, which can be structured in several raster and vector formats. The visualization of large terrain models in applications like virtual globes [Tru12a] requires an intensive use of computation and memory, so it is usual to generate multi-resolution schemes [Lue02a] in which different, progressively coarser meshes represent points in the original model. In this regard, the number of triangles that a 3D scene is going to display is highly dependent on the implemented LoD test, which ideally limits the graphics requirements for any given point of view [Sua15a].

We are particularly interested in finding a method that maximizes the vertical accuracy of the generated mesh in relation with the model. In this regard, some studies have researched on how to generate surface points from discrete elevation models. These techniques are ultimately used to generate continuous terrain representations that try to fit the original model as well as possible. Comparatives between different interpolation methods have been introduced in [Agu05a] and [Cha06a], from which can be extracted that no method performs the best for all terrain scenarios, being more convenient to use different methods depending on the terrain to be represented. It is also worth considering that, for multiresolution schemes and visualization purposes, vertical accuracy between two inmediate levels of detail should also be maximized in order to avoid popping artifacts in dynamic multi-lod visualizations whenever possible.

In this study, a pipeline to adjust TRNs to fit large DEMs is proposed. An initial mesh is obtained by approximating the solution of a linear system from the input data. After that, a refinement method tunes vertices to optimize the global error of the mesh. Finally, exper-

imentation has been conducted to ensure the applicability of our proposed algorithm to very large DEMs. In this regard, a tiling strategy to speed up the process is analyzed. Tests are also conducted to check the behavior of the whole pipeline, compared with classic methods, in a benchmark elevation model.

1.1 Related work

There are several methods to generate meshes from terrain data. One of the most popular approaches is the use of *Triangular Irregular Networks* (TIN) [Peu78a], which minimize the amount of triangles to be rendered given an acceptable error threshold. Different authors, such as Hoppe [Hop98a], Puppo [Pup96a] or El-Sana and Varshney [Els99a], have worked with TIN applied to multi-resolution schemes and variable mesh resolutions. Other option is the use of regular grids and *Triangulated Regular Networks* (TRN) [Agu03a], which are easier to implement and consists of triangle meshes that represent a surface in which all vertices are placed on a regular grid. In this line, the studies of Lindstrom et al. [Lin96a] and Pajarola [Paj98a] can be highlighted. More approaches on TRN and TIN are introduced in a survey from Pajarola and Gobbetti [Paj07a].

An alternative approach, known as Right-Triangulated Irregular (RTIN) Networks, has also been discussed in the literature. This variation of the TIN structure places the vertices on a grid, but still uses fewer triangles where they are not needed. However, when presented as raster information to the GPU, this scheme is not more efficient in terms of memory, and saving triangle drawing implies a local lost of detail. The work of Suárez and Plaza [Sua03a] describes a pipeline to refine this kind of model.

At the present time, lines of work are more focused on taking advantage of the capacities of GPU for terrain visualization. Authors such as Yusov and Shevtsov [Yus11a] or Kang et al. [Kan15a] have proposed to perform the triangulation task in the tessellation stage of OpenGL, saving computation time and data transfers between CPU and GPU. However, on-the-fly tessellation of surfaces is an intensive process that is not available on many graphics pipelines, e.g. mobile platforms implementing OpenGL ES 2.0. Besides, the downscaling of these models normally relies on linear interpolations of the underlying DEM, which does not assure a solution with the best height accuracy.

2 DEM REPRESENTATION ERROR METRIC

The proposed system utilizes a linear system to approximate a terrain sector using TRNs of resolutions lower than the one of the provided DEM. Given the input data defining punctual values of the terrain elevation,

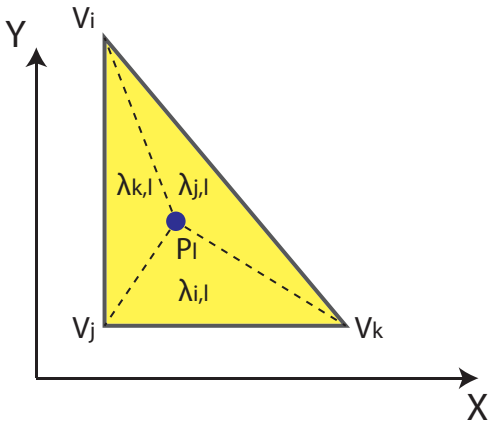


Figure 1: Barycentric coordinates of P_l

the proposed method adjusts the TRN by approximating the solution of a linear system. For any point P_l of the input elevation data, it is determined which triangle of the final triangular mesh is going to represent that particular point in cartographic space. As the triangles do not overlap on the latitude/longitude plane, this calculation is rapidly achieved by mapping the geographic coordinates of P_l to the barycentric coordinates of the tested triangle.

Thus, for all triangles present on the final TRN, the algorithm defines the projection of its vertices (V_i, V_j, V_k) in geographic space, by linearly interpolating their position within the TRN geographic sector. On this 2D triangle, the coordinates of P_l are transformed to its barycentric coordinates as depicted in Figure 1. If the three barycentric components are equal or greater than zero, the projection of P_l falls into the triangle area.

Barycentric coordinates are homogenous so they can be used to interpolate any point within the triangle. By using normalised barycentric coordinates, any inner point can be computed as a weighted sum of the triangle vertices. This coordinate system conversion can be efficiently achieved applying Cramer's rule.

Considering an input elevation data of n points and a desired TRN of t triangles, the limiting behavior of the algorithm that composes the linear system is of $O(n*t)$ asymptotically. This could lead to efficiency problems due to these magnitudes growing exponentially as both the DEM and the TRN increase their geographical space or spatial resolution.

In practice, however, the regularity of TRNs can be exploited to speed up the generation of the barycentric coordinates. First, all the DEM points must be expressed as x, y, z coordinates, orthogonal to the grid axis. Then, the normalized x, y coordinates, relative to their grid cell inner coordinates, can be expressed as:

$$\{x_C, y_C\} = \{\frac{((x - X_0) \bmod \Delta X)}{\Delta X}, \frac{((y - Y_0) \bmod \Delta Y)}{\Delta Y}\} \quad (1)$$

where $\{X_0, Y_0\}$ are the origin of the TRN, and $\{\Delta X, \Delta Y\}$ are the distances in X and Y between two consecutive vertices. The barycentric coordinates can then be computed in linear time, within a set of 2 two triangles. To determine in which triangle each point falls, the condition $x_C > y_C$ indicates a point belonging to the upper triangle.

For each point in the input data, its corresponding projection on the TRN can be calculated as an interpolation of the three vertices of the triangle weighted by the barycentric coordinates calculated on the previous step. On that assumption, the error function for a particular point P_{l} on the input set is given by Equation 2:

$$E_l = P_{l_z} - (\lambda_{i_l} V_{i_z} + \lambda_{j_l} V_{j_z} + \lambda_{k_l} V_{k_z}) \quad (2)$$

In this equation, i, j and k are, for the point P_{l}, the indices of the vertices of its enclosing triangle, and λ_{i_l} are the barycentric coordinates of $P_{\mathrm{l\,xy}}$ on the $V_{\mathrm{i\,xy}}$, $V_{\mathrm{j\,xy}}$, $V_{\mathrm{k\,xy}}$ projection of the triangle V_{i},V_{j},V_{k} with respect to the V_{i} vertex.

Given a similar expression for every point of the DEM, we can express the resulting linear system as a matrix equation of the form $A \cdot X = B$.

The A matrix is a large non-square sparse matrix which contains the coefficients of Equation 2. These coefficients are the above-mentioned barycentric coordinates computed in 2D. This way, for any given row l, it expresses the error function of the input datum P_{l}.

The B vector is conformed by $P_{\mathrm{l\,z}}$, which are the heights associated to the P_{l} coordinates based on the input data. Finally, the X vector represents the to-be-computed heights of all the vertices of the TRN.

This linear system is conformed by a coefficient matrix of size $n \times (q \cdot w)$, being n the number of input points on the DEM and $(q \cdot w)$ the resolution of the desired TRN. This final size can be extremely high considering the size of most available DEMs.

The solution of this system, in case of existing, is a set of heights from which a triangulation that perfectly matches the provided DEM can be generated. For most cases, the system is unsolvable and the goal is to find an X that optimises the vertical accuracy by minimising $\mu(\|A \cdot X - B\|)$. Ultimately, $\mu(A \cdot X - B)$ is the mean error produced by the method for a given TRN resolution.

3 METHOD PROPOSAL

There are many ways to approximate the solution of the above-mentioned linear system. The Least Square Method [Pai82a] is a regression analysis technique that, given the equations system, minimises the value $\|(A \cdot X - B)\|$ by exploring the space of solutions via gradient descent. This numerical approach is a suitable option for our case, as it does not require the matrix to be square, and is specially efficient for sparse matrices like A. Finally, the execution of the Least Square Method can be accelerated by providing a X_0 from which the space of solutions starts to be explored. In our case, we have calculated this first approximation to the solution by assigning to every TRN vertex an initial height equivalent to the closest point on the DEM.

In case many levels of detail are computed for the same DEM, a multi-grid strategy can be followed. This technique is broadly used to approximate solutions of linear systems that represent spatial characteristics at different resolutions [Qua16a]. In our case, the TRNs can be generated from the finest to the coarsest. Thus, an alternative value for X_0 could be provided by the immediately coarser model, improving the initial approximation and shortening the computation time.

After obtaining a first approximation to the solution, we propose to use an iterative stage to tune vertices with a strong contribution to the global error. Every iteration in the algorithm processes the input solution as follows:

1. A list of the contribution to the overall error of each one of the vertices of the TRN, Q, is obtained by the following expression : $Q = A^T \cdot E$, where A is the matrix presented in Section 2 and E is the result of the expression $A \cdot X - B$.

2. For all vertices in the TRN whose Q value is non-zero, a new value for the vertex is computed using the expression $V_{\mathrm{n}} := V_{\mathrm{n}} - \alpha \cdot Q_{\mathrm{n}}$. This will be the height of the vertex in the following iteration only if the change decreases the global error. Otherwise, the vertex remains at the same height. In the rule, α is a factor which ranges between 0 and 1. Experimentally, a value of 0.7 gives us good results.

3. The process stops when the error difference between two iteration is below a tolerance of 10^{-10} meters.

4 LARGE DEM SIMPLIFICATION PIPELINE

The main drawback of the iterative stage proposed in the previous section is its execution time for large TRNs. A higher number of iterations could be needed to find the solution, each of them modifying vertex heights and checking the error for a rising number of vertices in the TRN, thus, heavily increasing the execution time.

A straightforward technique we propose to reduce the complexity of the linear system is to subdivide the desired TRN into tiles that cover a maximum number of

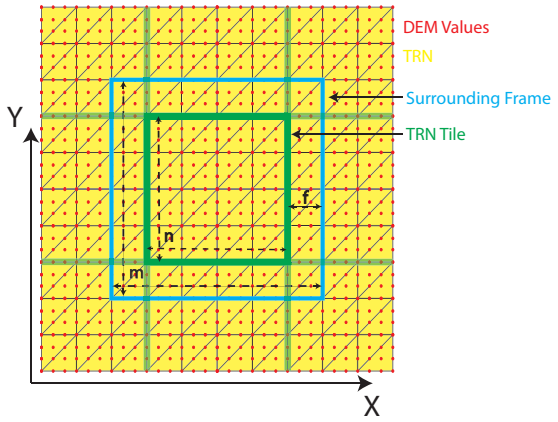


Figure 2: Division in tiles of a TRN

data points. Considering a regular axis-aligned DEM, a n-fold subdivision in each dimension of the DEM sector generates n^2 linear systems, reducing their coefficient matrix by a factor of $1/n^4$. Thus, the overall dimensionality of the systems to be solved is reduced by a factor of n^2 by tiling. Furthermore, the generated linear systems are independently solvable, hence it is easier for the available hardware to parallelise their resolution, taking advantage of multi-core and distributed architectures. However, the TRN-DEM adjustment is not independent between different zones, and tiling induces a penalty on the error results, since border vertices lack information about neighboring areas outside of the tile, leading in an inaccurate adjustment. To avoid undesired effects, a *frame area* should be added so the vertices of the tile do not present significant variations against the same vertices when the entire mesh is computed at once. The results for that frame are dismissed, using only the results for the tile itself. Such a configuration can be seen in Figure 2.

When the used frame area has enough width, the height values for the boundary of two adjacent tiles should not have appreciable differences and thus it is possible to treat tiles independently. However, as a safety mechanism, the average of the values of a boundary vertex in both tiles is taken to address any discontinuity that could happen during the process. Experimentation has been conducted to determine the ideal width of the referred frame and can be read in Section 5.2.

5 EXPERIMENTATION

During this work, experiments have been designed to test the impact of the proposed splitting strategy, find the best size for a tile and check the accuracy and goodness of the proposed solution. The following subsections present detailed descriptions of each experiment and its results.

5.1 Case of study and machine specs

Two different elevation models have been used during the experimentation work. The first elevation model covers the Puget Sound region (NW coast of the United States), is downloadable at the website [Pug00a] and has been used as a showcase of DEM analyses by other works such as [Kan15a],[Los04a] or [Liv09a]. The second elevation model covers the island of Tenerife. It was chosen due to its variable morphology and it can be downloaded from the NASA Worldwind services, specifying a bounding box ranging from 27.99°to 28.59°in latitude and -16.9322°to -16.0978°in longitude. Table 1 contains some information about these elevation models. There, *maximum cliff* refers to the maximum difference between a point and its neighbours.

	Tenerife	**Puget Sound**
Resolution	2089 x 2849	7169 x 6657
Minimum height	0	0
Maximum height	3689	2429.6
Mean	326.05	230.38
Standard deviation	604.72	355.31
Maximum cliff	504	484

Table 1: Resolution and characteristics of the chosen datasets. All height data are expressed in meters.

All experiments in this section have been executed using Matlab R2013a and run in a machine whose internal specifications include a CPU Intel Core i7-4790 at 3.60 GHz with a main memory size of 8 GB and uses Windows 8 as a operating system.

5.2 Determining whether splitting is possible for a large TRN

This first experiment checks the error penalty produced due to splitting a TRN in tiles, and particularly, how errors induced by splitting are distributed in the tile. We are checking if differences over a maximum allowed error are found near the edges of the tile after comparing the results obtained from calculating a tile and computing the whole TRN. For the sake of this experiment, that tolerance was set to 0.1 m. If this hypothesis can be confirmed, splitting in tiles can be considered and a frame margin of f vertices of width will be set and added to every tile area to be calculated so the error-prone border area can be discarded.

Let us consider *ratio* as the quotient between the length of the edge of a squared DEM and the length of the edge of a squared TRN to be generated from it. For different chosen ratios, different squared tiles of $n \times n$ vertices are selected, representing $N \times N$-point areas of the DEM. Using the pipeline, the tiles are adjusted independently so they fit their corresponding areas in the DEM. A TRN for the whole DEM is also adjusted at once. After the adjustment, differences between the tile mesh and the tile area in the global TRN have been measured. The relation between the error induced on a TRN vertex and its distance to the border of the tiles

Ratio	*n*	*N*	*f*
1:2	49	97	5
1:2	241	481	5
1:2	497	993	6
1:4	49	193	7
1:4	241	961	10
1:4	497	1985	8
1:8	49	385	7
1:8	241	1921	7
1:8	497	3969	9

Table 2: Frame margin f needed for different tile sizes and levels of detail.

has been analyzed, as it is shown in Figure 3. The pairs of selected n and N values, among with the results of the described analysis are presented in Table 2.

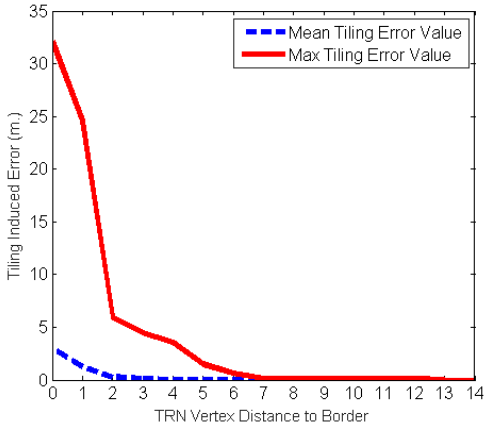


Figure 3: Distribution of differences between meshes for case $n = 241$, $N = 961$ (Ratio = 1 : 4).

As expected, the results show that all errors greater than the tolerance are present in the edges for all the experiments, with a maximum depth of 10 vertices. That leads us to infer it is possible to split the calculation of a TRN in tiles. Adding a frame of width $f = 10$ vertices guarantees that the obtained results will be similar to those achieved by processing the whole DEM at once.

5.3 Minimising execution times for large TRNs

The second test is aimed to determine the proper tile size t that minimises the time required to generate a full large TRN using the proposed pipeline.

For the sake of this experiment, we are trying to generate a TRN of size 3585×3329 from the Puget Sound DEM. For it, a set of different tile sizes are selected so the entire TRN can be split in an exact number of adjacent tiles. We are adding the frame margin f calculated in Section 5.2 to all tiles before processing them using the pipeline. Execution times have been calculated by averaging several execution samples on different DEM areas. Finally, the margin area is discarded, using only the inner area to compound the final TRN, from which a total execution time is also obtained. We will choose the t which minimizes this final execution time. The selected tile sizes and their average execution times can be seen in Table 3.

Tiles	Vertices per tile	Time per tile
8x8	449 x 417	3718
12x12	299 x 278	389.6
16x16	225 x 209	171.7
20x20	180 x 139	130.7
24x24	150 x 119	80.60
28x28	129 x 119	51.43
32x32	113 x 105	40.03
40x40	90 x 84	32.77
48x48	75 x 70	20.97
56x56	65 x 60	15.21
64x64	57 x 53	12.55
128x128	29 x 27	5.295

Table 3: Chosen tile sizes and execution times. All times are expressed in seconds.

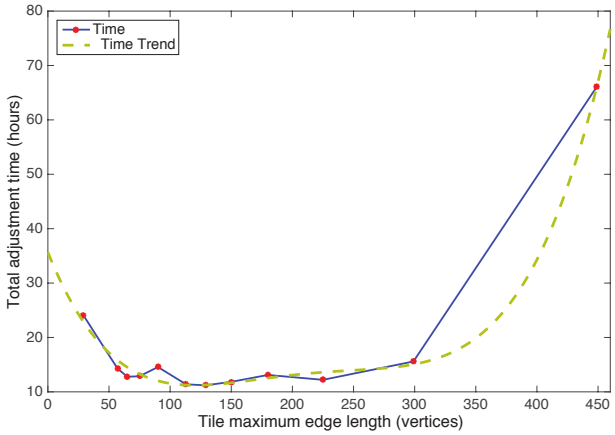


Figure 4: Evolution of estimated time to complete a TRN for the first LoD of the Puget Sound elevation model.

By analyzing the results it is observed that, for large resolutions, the cost of computing a tile is greater than the cost of splitting the tile in four equally-sized subtiles and computing all of them. This can be seen in Table 3: a tile with a resolution of 225×209 needs 171.7 seconds to be computed, whilst the computation of 4 tiles of resolution 57×53 lasts in average 50.2 seconds. On the other hand, very low resolutions have an overhead due to the redundant calculation of vertices which fall in frames of different tiles. The ideal size for the tile should have a cost similar to the one resulting of the computation of its four possible children.

As seen in Figure 4, the ideal t for this experiment is of 113×105 vertices, plus a frame of width f = 10.

5.4 Accuracy of the proposed method

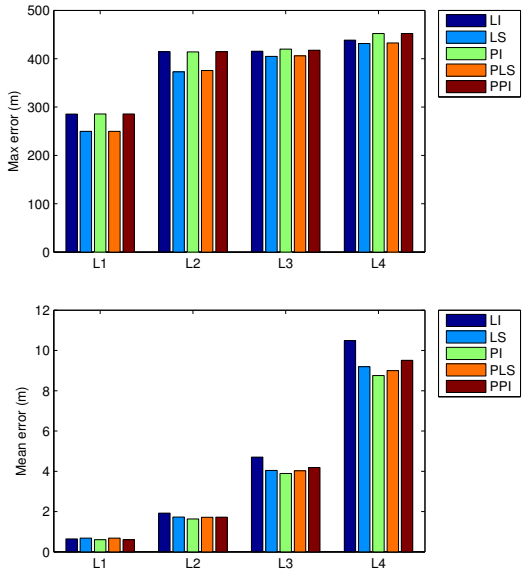


Figure 5: Maximum and mean errors between meshes for levels of detail L1 (most detailed) to L4 (coarsest) and the Puget Sound DEM, for all considered techniques. All units are expressed in meters.

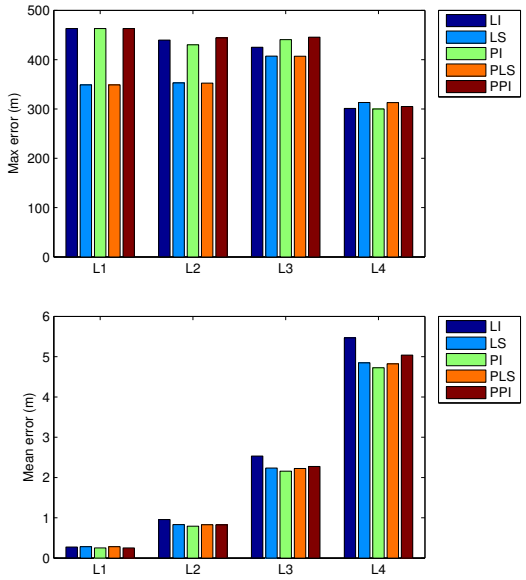


Figure 6: Maximum and mean errors between meshes for levels of detail L1 (most detailed) to L4 (coarsest) and the Tenerife DEM, for all considered techniques. All units are expressed in meters.

As this work describes a method to adjust TRN to a large DEM, comparing the results given by the proposed pipeline with classic techniques becomes of interest.

Using the two models presented in Section 5.1, two experiments were designed in which four levels of detail for both DEMs are calculated, each one with a resolution 2 times lower than the immediate most detailed level, for our proposed pipeline and other reference methods. The generated meshes have been compared to the DEM and also between them, calculating their vertical accuracy.

These new experiments differ in how subsequent levels of detail are calculated via the same adjustment methods. For the first one, all levels of detail are calculated using a downsampled version of the DEM as initial approximation. The different methods used in this experiment are the linear interpolation (LI), LSQR (LS) and the proposed pipeline (PI). In the second one, the result of the generation of the precedent level of detail has been used instead as initial approximation. The methods for this case are labeled as Progressive LSQR (PLS) and Progressive Pipeline (PPI).

Regarding the LI method, it is important to clarify that the value of an intermediate point is computed as the bilinear interpolation of the four neighbouring data points. In contrast, the LSQR-based method obtains the optimal values for each vertex by iteratively approximating the solution of a linear system of equations. The LSQR implementation of Matlab has been set to seek a tolerance of 10^{-6} meters in the achieved solution, within a maximum of 300 iterations. As the experimentation shows, however, that number of iterations is never met whereas the total error is several orders of magnitude greater than the tolerance, indicating that the LSQR method finds itself stalled at undesired local minimums.

A first analysis of the results was done and presented in Figures 5 and 6, focusing on the resemblance between the meshes and the DEM. It is observed that the use of PI or PPI gives, for all levels, TRN with lower mean errors related to the DEM than the ones generated with the classic methods. Some examples are shown in Figures 11 and 12. They have also maximum errors similar to the ones obtained using the linear interpolation. On the other hand, LS and PLS only outrun the linear interpolation, in both maximum and mean error, for the TRNs of the coarsest levels of detail (3-4).

It is also observed that, for both scenarios in this experiment and all the tested techniques, the maximum error of a given TRN versus the DEM is always over 250 meters, with some cases presenting errors of 463 meters. By carefully analyzing those errors, it was discovered that all tested methods find difficulties in modeling large cliffs, as it is seen in Figure 10. The larger space between points in a regular grid implies that some sea points adjacent to the actual cliff will have a great height value in the TRN, thus, large errors will be produced. Moreover, line-shaped artifacts have been ob-

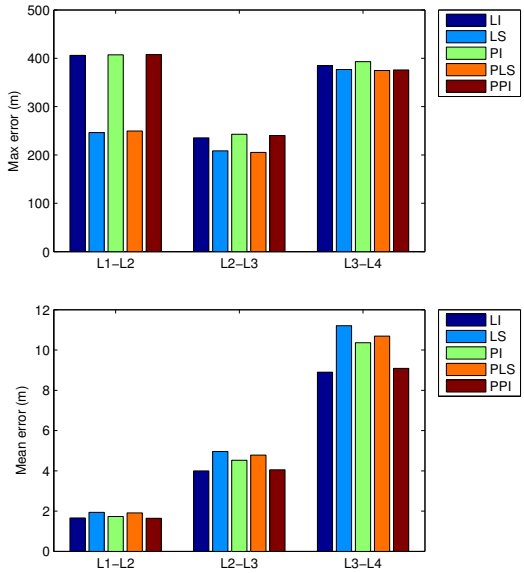


Figure 7: Maximum and mean errors between two consecutive levels of detail of the Puget Sound model, for all considered techniques. All units are expressed in meters.

served in some regions of the Puget Sound model, as it is presented in Figure 9. Variations on the height of the affected vertices can be of hundreds of meters compared with the surrounding regions, making hard the adjustment of TRNs from the model.

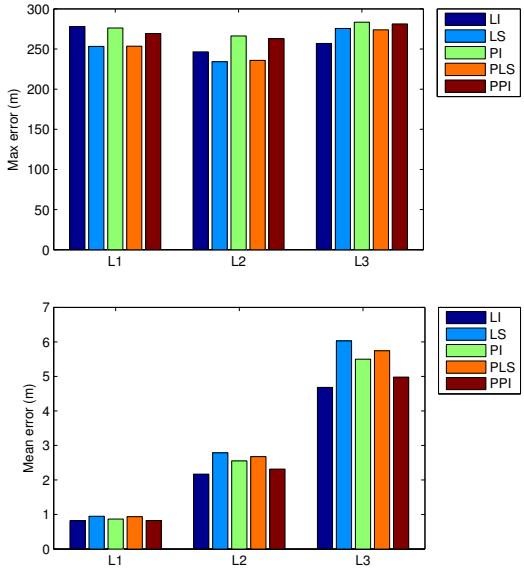
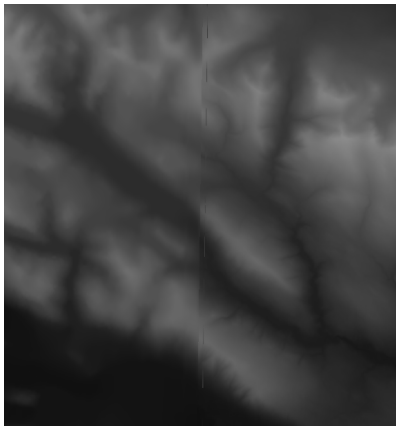
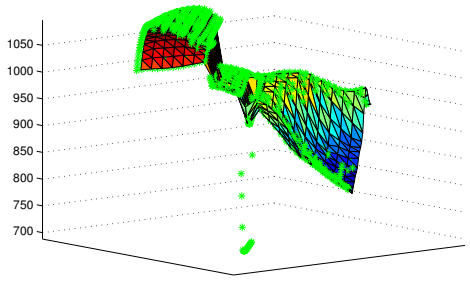


Figure 8: Maximum and mean errors between two consecutive levels of detail of the Tenerife model, for all considered techniques. All units are expressed in meters.



(a) Line-shaped artifacts in a DEM region.



(b) TRN compared with original DEM in an affected area.

Figure 9: An example of line-shaped artifacts in regions of the Puget Sound model.

The second analysis compares the resulting levels with their inmediate precedent and its results are given in Figures 7 and 8. In general, it is seen that the linear interpolation provides the best results in both models. Computing TRNs directly from the DEM using LS, PLS or PI becomes inadvisable for visualization pur-

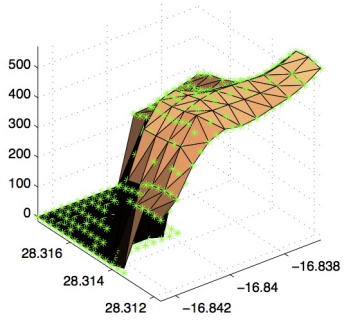


Figure 10: A cliff showing high errors in Tenerife DEM. Green markers represent DEM points to be confronted with TRN vertices.

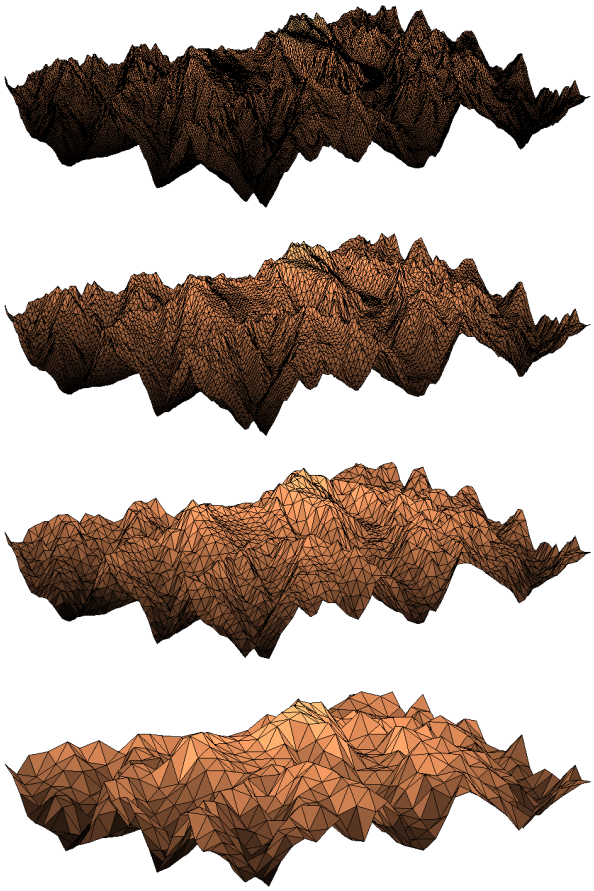


Figure 11: Area surrounding the highest point in the Puget Sound model, for all levels of detail generated using the proposed pipeline.

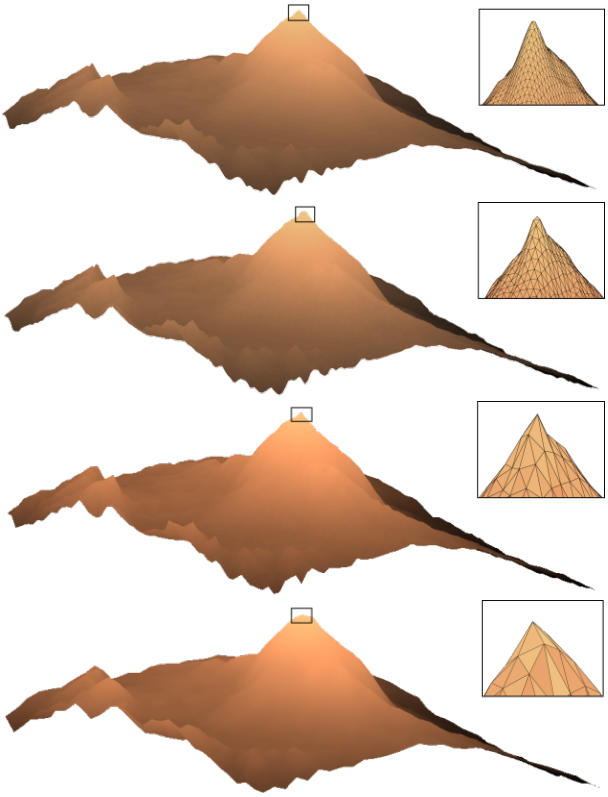


Figure 12: Results of the proposed pipeline for different levels of detail. In this case, the highest mountain of the Tenerife DEM (El Teide) is represented.

poses since it produces much higher errors between two consecutive levels of detail, unless a popping reduction technique such as geomorphing is applied to make smoother transitions. However, the use of PPI achieves better inter-level mean errors than the classic methods for detailed TRNs and only slightly worse mean errors for the coarsest levels. As this technique also allows to generate more accurate TRNs than the linear interpolation, as it was commented in the first analysis, the PPI method is also competitive for the adjustment of terrain models.

6 CONCLUSIONS

The present work has explored a way to adjust TRNs to better fit large DEMs. It consists in solving a linear system of equations that represent the height error of our representation. A technique to solve the linear system, such as LSQR, is used to approximate the system solution, and a later vertex refinement stage narrows down the mesh error.

The local refinement algorithm has an initial drawback due to large execution times when it is applied to large grids, but experimentation demonstrates that computational needs can be reduced by splitting the TRN in smaller windows without affecting the final result of the process.

The proposed pipeline was tested, generating meshes that fit the Puget Sound and Tenerife elevation models with promising results. Every mesh adjusted using our pipeline fits better the model than ones resulting of using classic methods as linear interpolation, without increasing heavily the difference between every level of detail. This fact suggests the pipeline can be used to generate meshes in order to visualize terrain in applications like virtual globes.

Future work will be focused on testing the pipeline versus other interpolation techniques in different scenarios and accelerating the local refinement stage. Moreover, a GPU implementation of the technique is also thought as future line of work.

7 ACKNOWLEDGEMENTS

The first author wants to thank Agencia Canaria de Investigación, Innovación y Sociedad de la Información, and the European Social Fund, for the grant “Formación del Personal Investigador-2012” that made possible this work.

The fourth author wants to thank Universidad de Las Palmas de Gran Canaria for its grant 'Programa de personal investigador predoctoral en formación 2015', which made possible this work.

8 REFERENCES

[Hop98a] Hoppe, H. Smooth View-dependent Level-of-detail Control and Its Application to Terrain Rendering, in Conf. proc VIS '98, Research Triangle Park, North Carolina, USA, IEEE Computer Society Press, pp. 35-42, 1998

[Els99a] El-Sana, S., and Varshney, A. Generalized View-Dependent Simplification. Computer Graphics Fourm, 1999.

[Pup96a] Puppo, E. Variable Resolution Terrain Surfaces, 1996.

[Lin96a] Lindstrom, P., and Koller, D., and Ribarsky, W. , and Hodges, L.F., and Faust, N., and Turner, G.A. Real-time, Continuous Level of Detail Rendering of Height Fields, in Conf. proc SIGGRAPH '96, ACM Press, pp. 109-118, 1996.

[Paj98a] Pajarola, R. Large Scale Terrain Visualization Using the Restricted Quadtree Triangulation,in Conf. proc VIS '98, Research Triangle Park, North Carolina, USA, IEEE Computer Society Press, pp. 19-26, 1998

[Paj07a] Pajarola, R., and Gobbetti, E. Survey of Semi-regular Multiresolution Models for Interactive Terrain Rendering. Vis. Comput., vol. 23, pp. 583-605. Springer-Verlag New York, Inc. , July 2007.

[Yus11a] Yusov, E., and Shevtsov, M., High-Performance Terrain Rendering Using Hardware Tessellation., Journal of WSCG, vol. 19, pp. 85-92, 2011

[Kan15a] Kang, H., and Jang, H. , and Cho, C.S., and Han, J. Multi-resolution Terrain Rendering with GPU Tessellation. Vis. Comput., vol. 31, pp. 455-469, Springer-Verlag New York, Inc. , April 2015.

[Agu05a] Aguilar, F.J., and Aguera, F., and Aguilar, M.A., and Carvajal, F. Effects of terrain morphology, sampling density, and interpolation methods on grid DEM accuracy. Photogrammetric Engineering and Remote Sensing, vol. 71, pag 805-816, American Society for Photogrammetry and Remote Sensing, 2005.

[Cha06a] Chaplot, V., and Darboux, F., and Bourennane, H., and Leguédois, S., and Silvera, N.,and Phachomphon, K. Accuracy of interpolation techniques for the derivation of digital elevation models in relation to landform types and data density. Geomorphology, vol. 77, pp. 126-141, 2006.

[Qua16a] Quaglino, A., and Krause, R. Towards a multigrid method for the minimum-cost flow problem. arXiv preprint arXiv:1612.00201, 2016.

[Pai82a] Paige, C.C., and Saunders, M.A. LSQR: An algorithm for sparse linear equations and sparse least squares. ACM transaction on mathematical software, vol. 8, pp. 43-71, 1982.

[Peu78a] Peucker, T.K., et al. The triangulated irregular network. In Amer. Soc. Photogrammetry Proc. Digital Terrain Models Symposium. 1978. p. 532.

[Lue02a] Luebke, D., and Watson, B., and Cohen, J.D., and Reddy, M., and Varshney, A. Level of Detail for 3D Graphics. Elsevier Science Inc., 2002.

[Agu03a] Aguero, J. C., and Feuer, A., and Goodwin, G. C. Terrain Modelling via Triangular Regular Networks. MODSIM 2003, vol. 33, 2003.

[Pug00a] Finlayson, D., and Haugerud, R., and Greenberg, H., and Logsdon, M. : Combined Bathymetry and Topography DEM of Western Washington State. School of Oceanography of the University of Washington. October 2000. Web resource: http://www.ocean.washington.edu/data/pugetsound (Last accessed: 2017.02.17)

[Los04a] Losasso, F., and Hoppe, H. Geometry Clipmaps: Terrain rendering using Nested Regular Grids. ACM Transactions on Graphics, vol. 23, n. 3, pp. 769-776. August 2004.

[Liv09a] Livny, Y., and Kogan, Z., and El-Sana, J. Seamless patches for GPU-based terrain rendering. The Visual Computer, vol. 25, no 3, p. 197-208. 2009

[Tru12a] Trujillo, A., Suárez, J.P., De La Calle, M., Gómez-Deck, D., Santana, J.M. An open source virtual globe framework for iOS, Android and WebGL compliant browser. In Conf.Proc. COM.Geo'12, New York, NY, USA; ACM Press; 212, p. 22:1 - 22:10

[Sua03a] Suárez, J.P., Plaza, A. Refinement and hierarchical coarsening schemes for triangulated surfaces. Journal of WSCG., vol. 11, no. 1-3, 2003.

[Sua15a] Suárez, J.P., Trujillo, A., Santana, J.M.,De La Calle, M., Gómez-Deck, D. An efficient terrain Level of Detail implementation for mobile devices and performance study. Computers, Environment and Urban Systems, vol. 52, pp. 21-33. 2015.

iDotter – an interactive dot plot viewer

Daniel Gerighausen[1,2]
daniel@informatik.uni-leipzig.de

Alrik Hausdorf[1]
hausdorf@informatik.uni-leipzig.de

Sebastian Zänker[1]
sebastianz541@gmail.com

Dirk Zeckzer[1]
zeckzer@informatik.uni-leipzig.de

[1]Image and Signal Processing Group, Leipzig University

[2]Bioinformatics, Leipzig University

ABSTRACT

Bioinformaticians judge the likelihood of the overall RNA secondary structure based on comparing its base pair probabilities. These probabilities can be calculated by various tools and are frequently displayed using dot plots for further analysis. However, most tools produce only static dot plot images which restricts possible interactions to the capabilities of the respective viewers (mostly PostScript-viewers). Moreover, this approach does not scale well with larger RNAs since most PostScript viewers are not designed to show a huge number of elements and have only legacy support for PostScript. Therefore, we developed iDotter, an interactive tool for analyzing RNA secondary structures. iDotter overcomes the previously described limitations providing multiple interaction mechanisms facilitating the interactive analysis of the displayed data. According to the biologists and bioinformaticians that regularly use out interactive dot plot viewer, iDotter is superior to all previous approaches with respect to facilitating dot plot based analysis of RNA secondary structures.

Keywords

Bioinformatics Visualization, Tabular Data, User Interfaces, Dot Plots

1 INTRODUCTION

In bioinformatics, one frequent task is judging the likelihood of the overall RNA secondary structure. This judgment is based on comparing the base pair probabilities of RNA secondary structures. Therefore, the probabilities for two nucleotides of an RNA sequence forming such base pairs are calculated. Dot plots are used for displaying probabilities or similarity measures between a row and a column of a matrix. Hence, dot plots are frequently used for RNA secondary analysis displaying the probability of a row and a column nucleotide forming a base pair.

Most currently available tools produce dot plots in postscript (ps) format (e.g., [6,8]). These ps-images are then viewed using suitable postscript viewers. However, postscript itself is no longer actively developed and was replaced by the portable document format (pdf). Moreover, the images are static and possible interactions are restricted to standard *viewing* interactions like zooming and panning the image. Further, the scalability of this approach is low as during zoom-in the nucleotide sequence that is displayed at the border of the image might not be visible any more.

Therefore, we developed iDotter, an interactive tool for analyzing RNA secondary structures that overcomes these limitations. Concretely, the contributions of this paper are:

- Sophisticated zooming and panning methods
- Presenting details for each dot on demand
- Highlighting of elements in the dot plot
- Recoloring of the dot plot
- Export of parts or the whole dot plot for further analysis
- A powerful API for using iDotter within analysis pipelines
- A sharing function for collaborative analyses

iDotter provides an interactive web interface that is implemented using the current state of the art web-programming languages HTML5, PHP, and JavaScript.

It proved superior to all previous approaches, and is already regularly used by biologists and bioinformaticians.

2 BACKGROUND AND RELATED WORK

Dot plots were introduced by Gibbs and Mcintyre [5]. Originally, dot plots were used to visualize alignments of two nucleotide sequences or proteins. A dot plot is a two dimensional matrix where the sequences 'A' and 'B' that are compared are visualized on the x- and y-axis, respectively. A dot in a cell means that Sequence 'A' is similar to Sequence 'B' at this nucleotide/amino acid (position). Both color and size of a dot represent the similarity of the sequences calculated using application dependent measurements. With the aid of dot plots, identifying highly similar regions between two sequences is easily possible. These regions are the diagonal lines in the matrix. An example for an interactive dot plot viewer for alignments was introduced by Sonnhammer and Durbin [12]. We, however, focus on RNA folding structures that can not be handled by their program. Moreover, we allow additional interactions like highlighting, semantic zoom, and export of (sub-)sequences not provided by their tool.

While the nucleotide sequence (RNA primary structure) is important for the analysis of RNA sequences, the folded structure of the RNA (RNA secondary structure) provides additional vital information. With the emergence of RNA folding tools [8–10], visualizing RNA secondary structure became more and more important to foster its analysis. Tools like Varna [4] or the NAVIEW algorithm [2] generate graph-based, node-link visualizations of RNA secondary structure showing *one* possible folding of the RNA, only. Further, dot plots were adapted to visualize the predicted base pair probabilities within a single RNA sequence. Thus, they support analyzing the changes of an RNA sequence between different species. Usually, the size of a dot describes the probability of a base pair between the corresponding nucleotides.

Static dot plots can be calculated with R using the R package R-CHIE [3]. The ViennaRNA package [8] can generate one dot plot in postscript format for each RNA secondary structure prediction (an example being shown in Figure 1a). Moreover, the ViennaRNA Web Services [6] provide the functionality of the ViennaRNA package without the necessity to compile the package. Therefore, it can be used platform independently. We use the ViennaRNA package for generating the initial dot plots, importing the data from them, and providing additional interactive visualizations for analyzing them. While the original dot plots of Gibbs and Mcintyre [5] for alignments show the same information in the upper and the lower triangle, dot plots generated by RNA folding software contain *two different* folding predictions, e.g., the energetically best solution and all possible base pair probabilities in the upper and lower triangles, respectively. As iDotter is based on the latter, it supports comparing *two different* folding probabilities predicted by the respective folding algorithms.

An alternative for visualizing RNA secondary predictions is the arc diagram introduced by Wattenberg [14] and later implemented as arc plot in R [7]. The RNA sequence is plotted as a linear sequence and an arc between two nucleotides describes a base pair while the color of an arc might encode the probability of the pair. Besides the fact, that this approach has limited scalability, the arcs produce a lot of clutter and it is hard to determine the corresponding base pair.

Arc diagrams and dot plots can be used for character sequence comparison (alignment) in general. For arc diagrams this was already introduced in the original paper [14]. Abdul-Rahman et al. [1] use dot plots to visualize text alignments between different documents.

3 PROBLEM, SOLUTION, AND METHOD

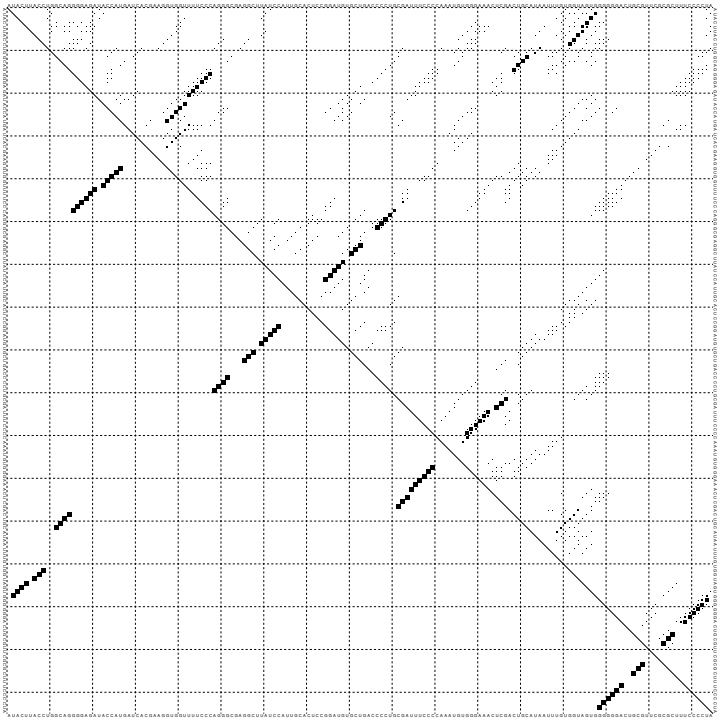
3.1 Problem Statement and Proposed Solution

Current state and Issues A dot plot fulfills the standard design goals taken from the information visualization literature [13]. Dot plots

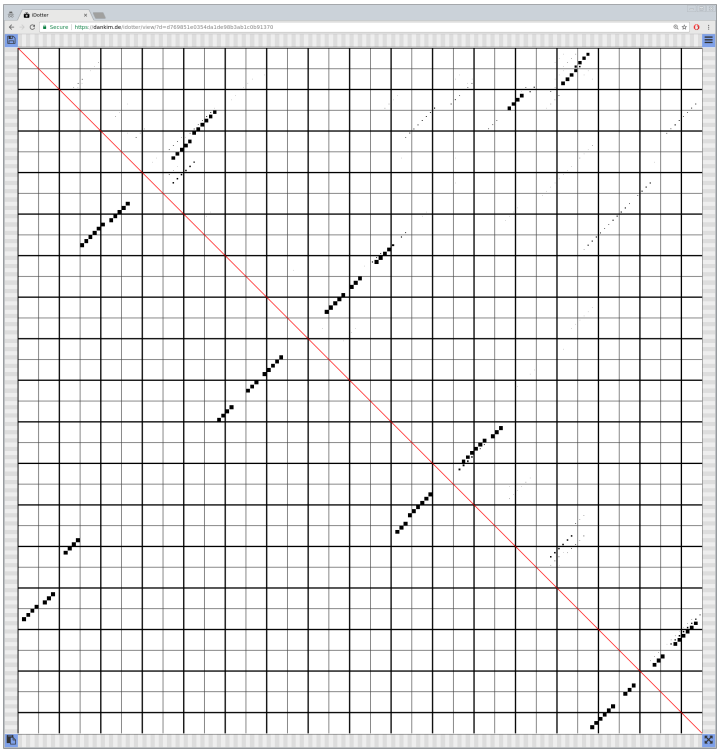
1. are flexible (can be used for different tasks and application areas)
2. are space efficient
3. provide a good overview of the data
4. ease the identification of pattern in the data
5. are fast to create

However, dot plots are static images without any interaction provided. Figure 1 shows a relatively small RNA having a length 165nt. As can be seen in the ps version (Figure 1a), the nucleotide names are no longer readable. Moreover, it is difficult to impossible to spot small base pair probabilities. Zooming into a part of the ps view is possible (Figure 1a). Then, all dots become larger and small base pair probabilities are more easily spotted. However, due to the limitations of the ps-viewers, the nucleotide sequence related to the zoomed-in area might no longer be visible.

Solution To overcome these limitations of existing dot plot generators, we propose iDotter, a fully interactive web-interface that supports experts in analyzing RNA secondary structure. iDotter is based on the dot plots generated by existing folding tools and provides

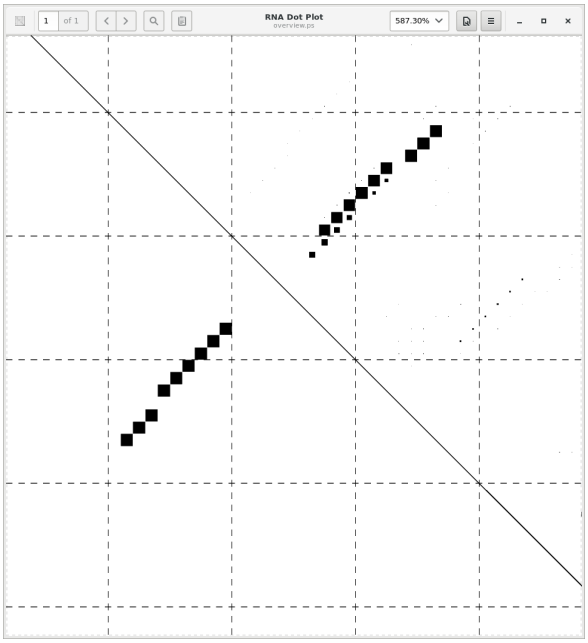


(a) Overview of the postscript dot plot generated by ViennaRNA [8]. The nucleotide sequence is always shown at the borders.

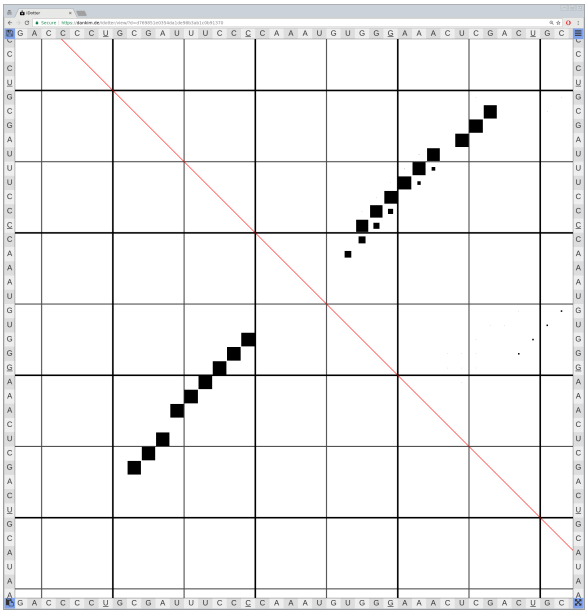


(b) Overview of the iDotter interface showing the same dot plot as Figure 1a. The nucleotide sequence is only shown at the borders, if the nucleotides are readable in the current zoom level.

Figure 1: Complete dot plot showing base pair probabilities of an RNA. Black squares are used for showing the possibility of two nucleotides forming a base pair. The probability for forming a base pair is encoded in the size of these squares. Large squares imply a high probability, while small squares imply a low probability. The diagonal is used as a landmark only. It divides the upper-right triangle showing the centroid probabilities from the lower-left triangle showing the probabilities according to the energetically best solution.



(a) PostScript-View, zoom-in of Figure 1a



(b) iDotter view, zoom-in of Figure 1b

Figure 2: Comparison of the dot plot interfaces after zooming into a sub-sequence. In the postscript view, the nucleotide sequence is no longer visible, while in iDotter it stays visible at all borders easing the analysis of sub-sequences.

additional interactions. After importing the data (Section 3.2.1), the dot plot is shown in the web browser (Section 3.2.2). Then, the expert can zoom in and out as well as pan the view (Section 3.2.3). Moreover, the expert can mark rectangular regions of dots in the dot plot as well as single columns and single rows (Section 3.2.3). Finally, the highlighted part of the dot plot or the complete dot plot can be exported in postscript-format (Section 3.2.4). A web-based API provides a connection with dot plot generating services (Section 3.2.5).

3.2 Methods

3.2.1 Data Import

After starting iDotter, the original ps-file generated by the ViennaRNA package [8] is transformed into a JSON file by iDotter, if the JSON file does not already exist. To do so, the RNA sequence, as well as the ubox and lbox containers are extracted from the ps-file and stored in a JSON array representing the box plot. Each ubox and lbox container comprises an x- and a y-coordinate designating the cell in the dot plot matrix, the size of the dot, and the color of the dot. The color information is optional. By transforming the input file into a generic JSON file iDotter can easily be extended for other input types by implementing a corresponding import routine.

3.2.2 Dot Plot View

The JSON file is imported by iDotter and the complete dot plot (zoom out) is displayed in the browser (Figure 1b). This follows the Shneiderman Mantra, presenting an “overview first” [11]. On each border, the nucleotide sequence is displayed. For convenience, the diagonal showing the same nucleotide on both the x- and the y-axis is shown in red. At the same time, this diagonal separates the upper from the lower triangle of the matrix. In the upper and lower triangle, either the same or two different base pair probabilities (encoded as size in the input file) are shown. The size of each dot is relatively encoded depending on the zoom level so that the expert can compare the probabilities easily on each zooming level. As default, we show the centroid probabilities in the upper and the energetically best solution probabilities in the lower triangle of the matrix, respectively. The probability of a dot is mapped to its size. The color can be used to represent, e.g., the conservation of the sequence between species. An adaptive background grid is displayed to enable an easy counting of the base pairs. Matching the zoom-level of the dot plot, the different grid levels can be shown or faded out. This view corresponds to the zoomed out standard dot plots, except that the nucleotide sequence is not shown, if the text becomes unreadable.

3.2.3 *Dot Plot Interaction*

The second step in Shneiderman's Mantra is "zoom and filter" [11]. The expert can use the semantic zoom to more closely analyze a sub-sequence (Figure 2b). The expert benefits from the sequence labels staying visible at all borders of the dot plot all the time. This is an improvement over the state of the art (Figure 2a) where the nucleotide sequence might disappear during zoom. This improves the scalability with respect to the size of the data that can be analyzed conveniently. Moreover, the individual nucleotides of the nucleotide sequence are only shown, if the zoom level allows displaying them in a readable manner. Otherwise, they are hidden (Figure 1b). The semantic zoom is triggered by mouse wheel motion. Moreover, the expert can pan the viewport by holding the left mouse button and moving the mouse.

Filtering is not provided for the original data. It would not be useful in this context. However, parts of the dot plot can be selected and this selection can then be exported (see below). This corresponds to a filtering step while its primary use is for reporting and collaborating.

The third step in Shneiderman's Mantra is "details on demand" [11]. While working with the dot plot, information about individual dots can be displayed on demand as a tool tip by mouse over. All available information is shown (Figure 3). Thus, the user can get exact information about the nucleotides (names and positions) involved in a base pair even though the respective names are not longer visible at the corners because they would be too small to read. Moreover, the values for the size (here: representing the base pair probability) and the color are shown.

Following the taxonomy of Yi et al. [15], selecting dots is provided by iDotter. The expert can mark a dot by left clicking on it (Figure 4a). Then, the selected dot is highlighted with the dot marker color. Moreover, the expert can select multiple dots by left clicking into the viewing area and dragging the mouse while holding the 'Shift' key pressed. This creates a rectangular region. Within this region, all columns and rows that contain dots are highlighted with the dot marker color (Figure 6). Additionally, the selected dots are highlighted in a different color (currently yellow) in both cases. Deselecting a region of dots is achieved by pressing the 'Ctrl' key while using the mouse. Besides marking dots, the expert can mark single columns by left clicking on them (Figure 4b). Then, the selected column is highlighted with the line marker color (see Figure 6). In the same way, the expert left clicks on a row to select it (Figure 4b). Both—marking dots as well as marking columns and rows—can be combined (Figure 5) to mark those parts of the dot plot that are of interest to the expert. Finally, the highlighting can be reset by pressing the 'Remove Marker' button in the settings view

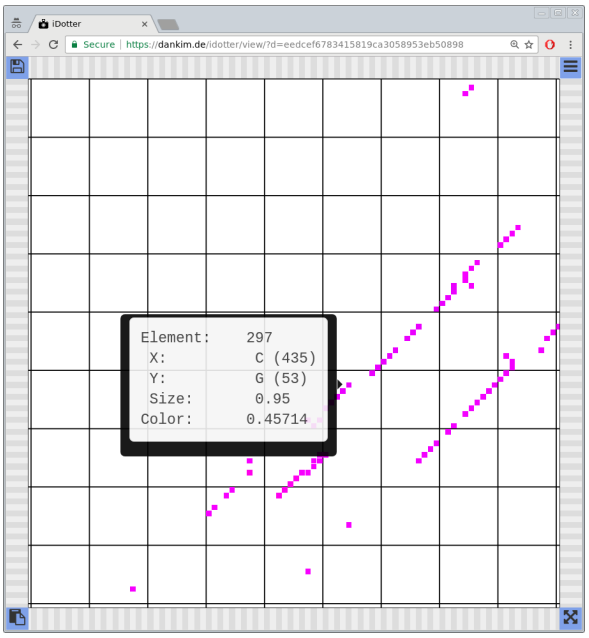


Figure 3: Each dot provides details on demand by mouse-over showing a tool tip: element ID, X shows the nucleotide of the column and its position, and Y shows the nucleotide of the row and its position. The (biological) attributes mapped onto 'Size' and 'Color' are application dependent.

which is invoked by left clicking on the 'three horizontal bars' icon in the upper right corner of the dot plot.

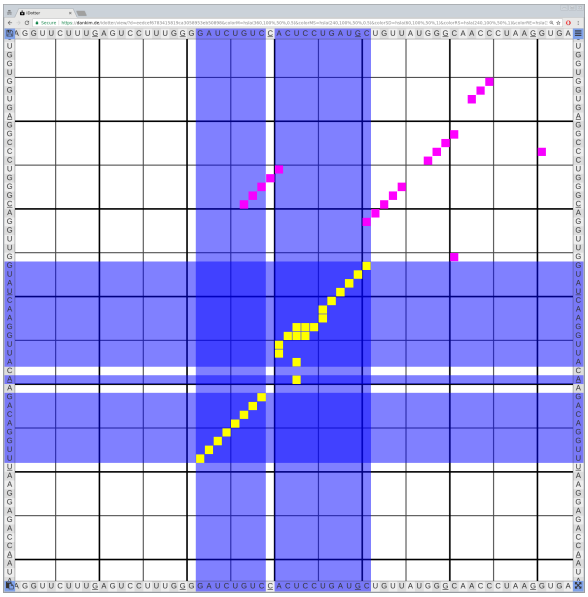
The color coding (corresponding to 'encode' [15]) can be adapted using the settings view. The two colors, the color gradient is generated from (Figure 6) can be changed. This directly influences the colors of the dots. Moreover, the colors of the marked dots (Dotmarker Color) and of the marked lines (Linemarker Color) can be chosen.

3.2.4 *Data Export*

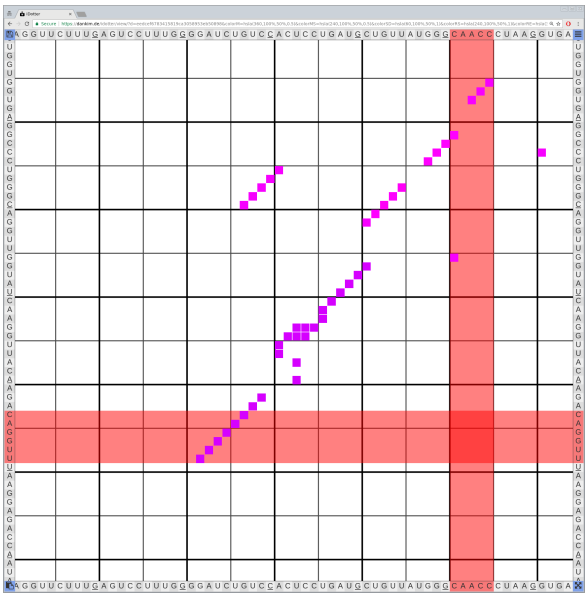
After working with the data, the expert can export the *highlighted parts* of the dot plot into a new ps-file for publication or other purposes by pressing the disc icon in the *upper left* corner and selecting the menu item 'export only selection'. Further, the expert can export the *complete* dot plot into a new ps-file by pressing the disc icon and selecting the menu item 'export all'. This is useful, if the API functionality of iDotter is used.

3.2.5 *API*

We designed a web-based API that provides a connection with dot plot generating services like the ViennaRNA Web Services [6]. This API supports direct import of ps-files into the view, pre-selecting highlighted regions, and exporting the highlighted regions for automatic workflows. The API is controlled by URL parameters. This type of control provides iDotter with additional possibilities for collaboration between users. The



(a) The 'mark dot' interaction allows selecting single dots by clicking on them. In this case, the selected dot is highlighted with the dot marker color (see Figure 6). Moreover, multiple dots can be selected by marking a rectangular region. (Clicking into the viewing area and dragging the mouse while holding the 'Shift' key pressed. For deselection, the 'Ctrl' key should be pressed instead.) All columns and rows that contain dots in the selected region are highlighted with the dot marker color (Figure 6). All selected dots are highlighted in a different color (currently yellow).



(b) The 'mark row' interaction allows selecting single rows by clicking on them. In this case, the selected row will be highlighted with the line marker color (see Figure 6). In the same way, columns can be selected.

Figure 4: Highlighting dots (a), and rows and columns (b).

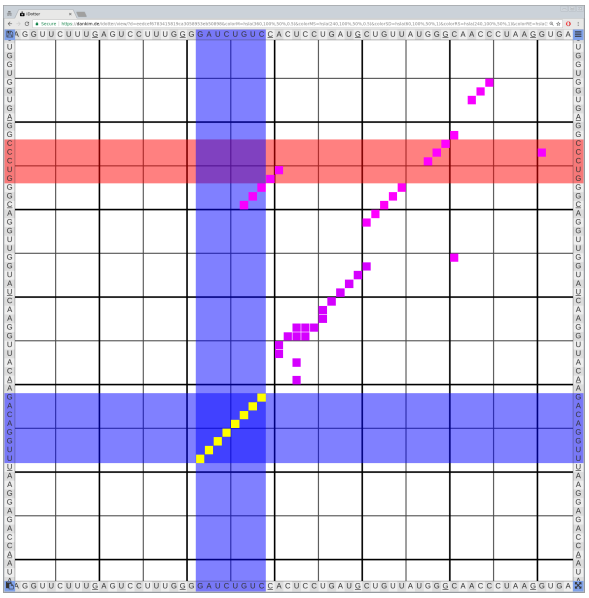


Figure 5: Marking dots and regions of dots (Figure 4a) and marking rows and columns (Figure 4b) can be combined.

expert can export her current settings, like zoom level, position, and color settings, and share these with her collaborators or save them for documentation purposes. The URL export is triggered by pressing the clipboard icon in the *lower left* corner. The URL contains all necessary parameters and is copied into the clipboard of the operating system. The expert can copy it afterwards to any application.

3.3 Interaction Properties

iDotter supports all interaction mechanisms required by the scientists for the analysis of dot plots. All useful steps of Shneiderman's Mantra [11] are supported. Moreover, the interactions 'select' and 'encode' proposed by Yi et al. [15] are supported.

4 EVALUATION

Dot plots are one of the default visualizations for the analysis of secondary RNA structure predictions. Therefore, the requirement was to enhance and extend this visualization for state of the art interaction techniques. In our case study our biological collaborator used iDotter for analyzing the evolution of so called long non coding RNAs (lncRNA). Since these RNAs are longer than 200nt, it is challenging to analyze the generated dot plots in ps-format due the lack of interactivity. Furthermore, it is hard to compare specific regions between different dot plots. For that reason, the expert used the interactivity features for selecting regions of interests. By exporting these regions with the API from all investigated RNA samples, it was possible to detect evolutionary changes between several species.

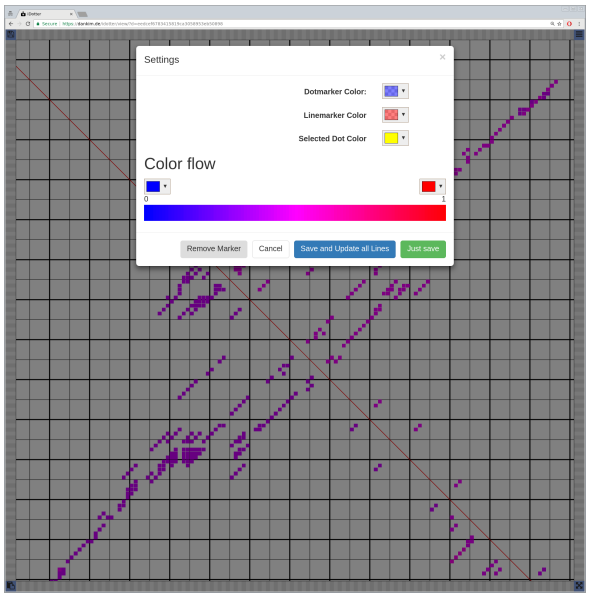


Figure 6: In contrast to the postscript visualization, iDotter provides choosing the color gradient. Additionally, choosing the highlighting colors for dots (Dotmarker Color, Figure 4a) and columns/rows (Linemarker Color, Figure 4b) is possible. Moreover, it is possible to reset highlighting in the dot plot by pressing the 'Remove Marker' button.

5 DISSEMINATION AND FUTURE WORK

The iDotter project is available under the GNU GPL v3 on `https://git.gurkware.de/biovis/idotter.git`. In the future, a close integration with the ViennaRNA Web Services [6] using the already existing API (Section 3.2.5) will be provided.

With respect to interaction, it is planned to add a small inset that provides an overview which part of the RNA sequence is currently zoomed-in. All interactions of the Shneiderman Mantra [11] and the interactions "select", "encode", "abstract/elaborate" (details on demand) and "filter" from the taxonomy proposed by Yi et al. [15] are already provided. Regarding the remaining three interactions from latter taxonomy, "explore" requires a closer integration with the folding tools using the already existing API. The "reconfigure" and "connect" interactions, however, are beyond the scope of the analysis task.

Further, it is planned to extend iDotter for analyzing data from other application areas similar to, e.g., alignments or text similarity [1]. Adapting iDotter for the different input file formats is straight forward. For this, a new data wrapper has to be created in iDotter that transforms the input data into a valid JSON input file.

6 CONCLUSION

We introduced iDotter, an interactive dot plot viewer for RNA secondary predictions. According to the biologists and bioinformaticians that regularly use out interactive dot plot viewer, iDotter is outperforms previous approaches with respect to facilitating dot plot based analysis of RNA secondary structures. By using the different interaction methods the experts were able to generate new insights and new hypotheses for their further work. The API enables the automated usage of iDotter in analysis pipelines or common RNA folding web services. The collaboration functionality allows the expert sharing her focus with her collaborators and documenting her insights.

7 ACKNOWLEDGMENTS

We thank all our colleagues from the BSV and Bioinformatics research groups for fruitful discussions on earlier versions of the project. This work was partially funded by the German Federal Ministry of Education and Research (BMBF) within the project Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig (BMBF grant 01IS14014B).

8 REFERENCES

[1] A. Abdul-Rahman, G. Roe, M. Olsen, C. Gladstone, R. Whaling, N. Cronk, R. Morrissey, and M. Chen. Constructive Visual Analytics for Text Similarity Detection. *Computer Graphics Forum*, 36(1):237–248, 2016. doi: 10.1111/cgf.12798

[2] R. E. Bruccoleri and G. Heinrich. An improved algorithm for nucleic acid secondary structure display. *Computer applications in the biosciences: CABIOS*, 4(1):167–173, 1988.

[3] D. Charif and J. Lobry. SeqinR 1.0-2: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis. In U. Bastolla, M. Porto, H. Roman, and M. Vendruscolo, eds., *Structural approaches to sequence evolution: Molecules, networks, populations*, Biological and Medical Physics, Biomedical Engineering, pp. 207–232. Springer Verlag, New York, 2007.

[4] K. Darty, A. Denise, and Y. Ponty. VARNA: Interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, 25(15):1974–5, 2009.

[5] A. J. Gibbs and G. A. Mcintyre. The Diagram, a Method for Comparing Sequences. *European Journal of Biochemistry*, 16(1):1–11, 1970. doi: 10.1111/j.1432-1033.1970.tb01046.x

[6] A. R. Gruber, R. Lorenz, S. H. Bernhart, R. Neuböck, and I. L. Hofacker. The vienna RNA websuite. *Nucleic acids research*, 36(suppl 2):W70–W74, 2008.

[7] D. Lai, J. R. Proctor, J. Y. A. Zhu, and I. M. Meyer. R-CHIE: a web server and R package for visualizing RNA secondary structures. *Nucleic acids research*, p. gks241, 2012.

[8] R. Lorenz, S. H. Bernhart, C. H. Z. Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. ViennaRNA Package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

[9] N. R. Markham and M. Zuker. UNAFold. *Bioinformatics: Structure, Function and Applications*, pp. 3–31, 2008.

[10] J. S. Reuter and D. H. Mathews. RNAstructure: software for RNA secondary structure prediction and analysis. *BMC bioinformatics*, 11(1):129, 2010.

[11] B. Shneiderman. *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations*. VL '96. IEEE Computer Society, Washington, DC, USA, 1996.

[12] E. L. Sonnhammer and R. Durbin. A dot-matrix program with dynamic threshold control suited for genomic DNA and protein sequence analysis. *Gene*, 167(1):GC1–GC10, 1995.

[13] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 ed., 2004.

[14] M. Wattenberg. Arc diagrams: visualizing structure in strings. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pp. 110–116, 2002. doi: 10.1109/INFVIS.2002.1173155

[15] J. S. Yi, Y. ah Kang, J. Stasko, and J. Jacko. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, Nov 2007. doi: 10.1109/TVCG.2007.70515

Fast and Effective Dynamic Mesh Completion

Gerasimos Arvanitis Aris S. Lalos Konstantinos Moustakas Nikos Fakotakis
Dept. of Electrical and Computer Engineering
University of Patras, Rio, Patras, Greece
{arvanitis, aris.lalos, moustakas}@ece.upatras.gr, fakotaki@upatras.gr

ABSTRACT

We introduce a novel approach to support fast and efficient completion of arbitrary animation sequences, ideally suited for real-time scenarios, such as immersive tele-presence systems and gaming. In most of these applications, the reconstruction of 3D animations is based on dynamic meshes which are highly incomplete, stressing the need of completion approaches with low computational requirements. In this paper, we present a new online approach for fast and effective completion of 3D animated models that estimates the position of the unknown vertices of the current frame by exploiting the connectivity information and the current motion vectors of the known vertices. Extensive evaluation studies carried out using a collection of different incomplete animated models, verify that the proposed technique achieves plausible reconstruction output despite the constraints posed by arbitrarily complex and motion scenarios.

Keywords

3D animated meshes, missing vertices, weighed iterative function

1 INTRODUCTION

Recently, there has been increasing interest in real-time 3D capture enabling the acquisition of dynamically deforming shapes at sustained ”video” rates. Although resolution and accuracy of 3D scanners are constantly improving, they are still unable to capture the full surface at once. Even in scenarios where multiple sensors are placed around the subject, most scanned shapes are likely to exhibit large holes, noise and outliers due to occlusions [BTSAL14], limited sensor range capabilities, high light absorption and low surface albedo.

Although a large number of prior works [SGP03] has investigated the problem of completion in static geometries, resulting in excellent filled static meshes, their direct application to every frame separately usually causes incorrect topologies and temporally incoherent surfaces. A fast and efficient approach for reconstructing surfaces from a set of known points have been proposed in [SC04] where the authors reconstruct meshes with a prescribed connectivity that approximate a set of control points in a least-squares sense.

Building on this direction, in this work we introduce a novel technique for reconstruction of highly incomplete dynamic meshes[1], by exploiting a known connectivity and set of motion vectors corresponding to the known points in an online setting. In this setting the animation sequence in not known a priori and at each time our method exploits information that has been presented so far. An extensive analysis has demonstrated that the high-frequency details of the animated model can be adequately recovered from a highly incomplete geometry dataset at very fast execution times.

2 RELATED WORKS & CONTRIBUTIONS

In recent years, a lot of research has been carried out into the field of 3D mesh reconstruction, having presented excellent results applied to incomplete static meshes. However, little attention has been given to the reconstruction of animated meshes. Traditional methods usually cause temporally incoherent surfaces when they are directly applied to each frame individually. These methods do not take advantage of previous frames knowledge, as a result they basically deal with n individual meshes instead of a sequence of temporally coherent meshes. A common approach to produce a temporal consistent dynamic mesh is to use a template prior [ZA04], however, this approach is not ideal for real-time applications because the entire captured animated mesh is required before the execution of the process.

The authors in [ACSTD07] focus on reconstructing watertight surfaces from unoriented point sets using a

[1] A dynamic mesh is a common term defined as a series of static, mainly triangular, meshes representing a 3D animation.

Voronoi-based variational approach, while the method in [SLS07] tries to handle the missing points by trying to infer topological structures in the original surface at the potential expense of retaining geometric fidelity. The researchers in [DGQ12] perform reconstruction only on the available information, effectively preserving the boundaries from the scan. Recently, a new signal processing technique known as matrix completion (MC) [CAN12] has been successfully applied to several computer vision problems, including the recovery of occluded faces/dynamic meshes [DDZ11], [VLMB07], [SWG08] and the face image alignment [PGWXM12]. It has been also used for the fusion of point clouds from multiview images of the same object [DDZW12]. In [RPMR13], it is applied on RGB-D data for the simultaneous tracking and reconstruction of 3D objects.

The common limitation of all the aforementioned approaches is the high computational complexity that significantly affects the execution time and renders them inappropriate for real time applications. These limitations motivated us to search for a fast and effective approach that can satisfy the reconstruction efficiency supporting at the same time real time applications. In summary, the main contributions of our work are:

- A general out-of-core approach to dynamic mesh completion ideally suited for fast and accurate filling (in spatial or temporal space) of incomplete arbitrary mesh sequences.
- An extensive experimental evaluation under different configurations and mesh animations showing that our approach achieves the highest reconstruction quality offering at the same time faster execution times as compared to previous methods.

The rest of this paper is organized as follows: Section 2 includes a detailed summary of prior art. Section 3 presents an overview of our method. Section 4 presents our experimental results and discusses the advantages and limitations of the proposal method. Section 5 draws conclusions and identifies future directions.

3 OVERVIEW OF OUR METHOD

Initial Assumptions and Preliminaries

In this section we present the basic assumptions and preliminaries related to animated meshes. Firstly, a dynamic mesh is defined $A = [M_1;\ M_2;\ \dots\ M_n]$ as a sequence of n static meshes consisting of k vertices. Each one of these meshes can be represented by two different sets $M = (\mathscr{V}, \mathscr{F})$ corresponding to the vertices $(\mathscr{V})$ and the indexed faces $(\mathscr{F})$ of the mesh. Each vertex can be represented as a point in the Euclidean space. Let us define with $\mathbf{v} = [\ \mathbf{x},\ \mathbf{y},\ \mathbf{z}\ ]$ a vector of vertices in a 3D coordinate space denoted as $\mathbf{x},\mathbf{y},\mathbf{z} \in \Re^{k\times 1}$, $\mathbf{v} = [v_1,\ v_2,\ \dots\ v_k] \in \Re^{k\times 3}$ and $V = \{v_1,\ v_2,\ \dots\ v_k\}$ is the corresponding set of vertices. Additionally, each face is represented as a set of 3 connected vertices $f_i = [v_{i1},\ v_{i2},\ v_{i3}]\ \forall\ i = 1,m$ where $m > k$ and the corresponding set of faces is denoted by $F = \{f_1,\ f_2,\ \dots\ f_m\}$. The set of edges $\mathscr{E}$ can be directly derived from $\mathscr{V}$ and $\mathscr{F}$, corresponds to the connectivity information.

Let us assume that A' is a highly incomplete dynamic mesh. In other words, each mesh of the animation has only a subset of known vertices while the rest have been removed. The incomplete animated model is represented by a sequence of incomplete meshes $A' = [M_1;\ M'_2;\ \dots\ M'_n]$ where $M'_i \subset M_i\ \forall\ i = 2,n$. Each incomplete dynamic mesh is described by a matrix of dimension $3k \times n$:

$$A' = \begin{bmatrix} M_1 \\ M'_2 \\ M'_3 \\ M'_4 \\ \vdots \\ M'_n \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & \dots & v_{1k-1} & v_{1k} \\ 0 & v_{22} & 0 & 0 & \dots & 0 & 0 \\ v_{31} & 0 & 0 & v_{34} & \dots & 0 & 0 \\ 0 & 0 & v_{43} & 0 & \dots & v_{4k-1} & v_{4k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & v_{n2} & 0 & 0 & \dots & v_{nk-1} & 0 \end{bmatrix}$$

The incomplete meshes are created by randomly removing points from the original ones. Fig. 1 depicts some indicative frames (meshes) assuming different densities of known points.

Adjacency and Laplacian Matrix

To estimate the coordinates of the missing points, we initially use a prescribed connectivity information (i.e., adjacency matrix), constructed from the faces of the mesh. Despite the fact that the position of vertices is changing, the adjacency matrix remains fixed over time, since we assume that every mesh has the same connectivity [WJHB07]. This observation allows us to estimate the adjacency matrix only once and use it repeatedly for any subsequent mesh of the same model. Moreover, we assume that we have full knowledge of the first mesh M_1 of the sequence. We define as $\mathbf{R} \in \Re_{k\times k}$ the adjacency matrix which is estimated as described below:

$$\mathbf{R}_{ij} = \begin{cases} 1 & if\ i,j \in E \\ 0 & otherwise \end{cases} \quad (1)$$

The matrix $\mathbf{R}$ is binary and it is used for the creation of the Laplacian matrix defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{R} \quad (2)$$

$\mathbf{D} = diag\{d_1,\dots,d_k\}$ is a diagonal matrix with $d_i = \sum_{j=1}^{k} r_{ij}$, being the degree of its node.

Spatial Classification of Each Frame Vertices

We create k cells of nodes $c_i\ \forall\ i = 1,k$ using the knowledge of the adjacency matrix $\mathbf{R}$. Each cell c_i rep-

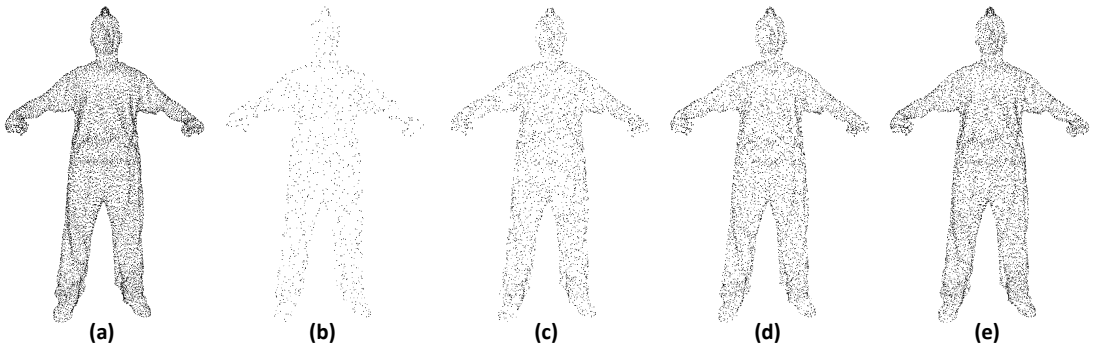


Figure 1: Indicative incomplete frames of the animated sequence: (a) original mesh (10002 points), (b) 10% of original points, (c) 30% of original points, (d) 50% of original points, (e) 70% of original points

resents the first ring area of each vertex i. We define as $\mathscr{C} = [\mathbf{c}_1\ \mathbf{c}_2\ \ldots\ \mathbf{c}_k]$ the set of k cells where $\mathbf{c}_i = [\dot{c}_{i1}\ \dot{c}_{i2}\ \ldots\ \dot{c}_{ij}]\ \forall\ i = 1,k$. However, it is worth mentioning here that each cell has different connectivity valence denoted by j. The element $\dot{c}_{ip}$ represents the index of p-th connected vertex with the vertex i. The set $\mathbf{C}$ remains fixed for every frame (mesh) and it is used for recovering the missing points. Fig. 2 illustrates an example of two connected cells $\mathbf{c}_1, \mathbf{c}_2$ with their related connections, cell c_1 has $j = 6$ connected neighbors (valence), while c_2 has a valence $j = 5$.

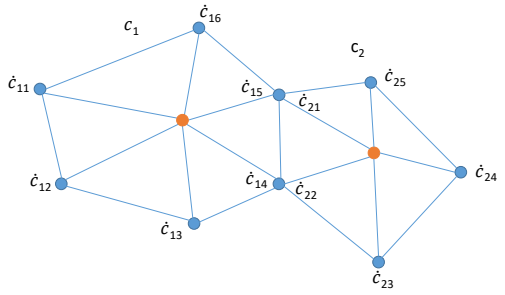


Figure 2: Representation of two connected cells

After the definition of cells we focus on the classification of vertices for each frame $M'_i\ \forall\ i = 2,n$. This process is executed in a sequential manner for each mesh starting from the second mesh until the end of the animation sequence. We assume three different classes for each vertex:

- **Anchor vertices** are the known vertices of the mesh.
- **Satellites vertices** are those belonging to a cell of an already known vertex. If a vertex belong to a cell of an anchor vertex then is defined as first generation satellite otherwise is defined as second generation satellite and so on.
- **Unknown vertices** are the vertices that do not belong to a cell of an already known vertex. Those vertices will be recovered and placed in the mesh at a future iteration.

The classification procedure is an iterative process that tries to eliminate any remaining unknown vertex so that the mesh will be composed of only anchor or satellite vertices. This means that the coordinates of an unknown vertex are evaluated in a following iteration. This method has been proved to be robust and all the missing vertices are always recovered. The method starts with the assumption that if the position of a vertex is known then it is called anchor point, $i\ \forall\ i = 1, k'\ k' < k$. All the vertices that belong to the cell $\mathbf{c}_i$ are classified as satellite vertices (first generation satellite). The rest vertices are classified as unknown. At the following iteration the position of the satellite points are taken into account and their cells are used for identifying new satellites (second generation satellites).

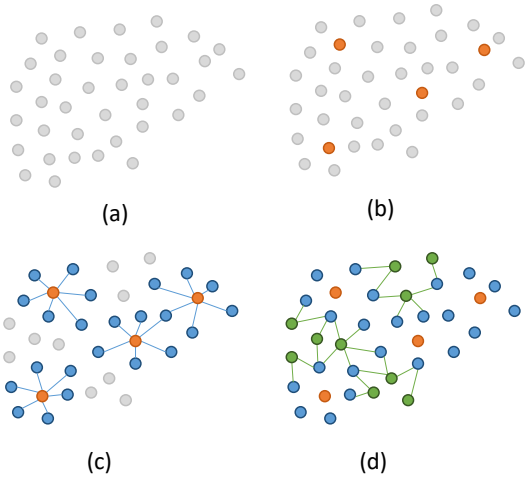


Figure 3: (a) Part of mesh with unknown points, (b) Anchors are indicated with red (Iteration 1), (c) First generation satellites identified based on anchors cells (Iteration 2), (d) Second generation satellites are set based on first generation satellites cells (Iteration 3).

Fig. 3 illustrates the aforementioned classification procedure. Red points represent the known vertices (anchor points). Blue points represent the satellites (first generation) which are connected with the anchor points, and correspondingly green point represent the satellites

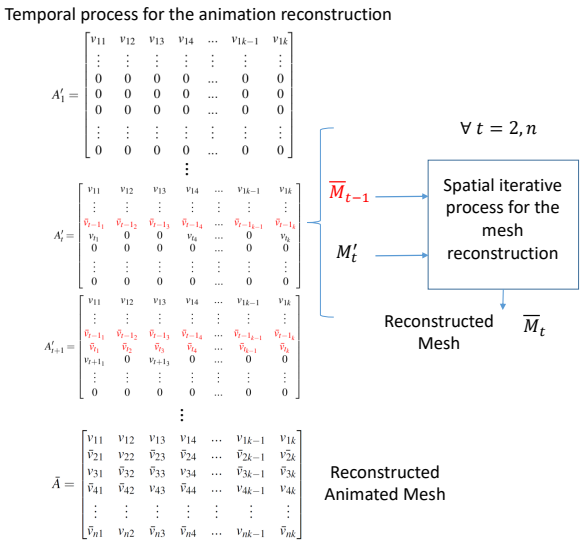


Figure 4: The animation is reconstructed frame by frame (temporal process) taking into account the previous reconstructed mesh and the current incomplete (spatial iterative process).

(second generation) that are connected with the first generation satellites. It is important to mention here, that a satellite point can be also a satellite point of more than one anchor point while, an anchor point can be a satellite point for another neighbor anchor point.

A cell c_i can indicate the existence of satellite vertices in a first ring area of vertex i, nevertheless it does not specify their real position. The classification procedure is used for assigning weights and then estimating the unknown vertex position using a weighted filtering approach based on their previous position and the motion vector of the anchor points. A detailed description of the method is provided in the following section.

Geometry Completion Based on Topological Characteristics

In this section we present the main steps of our interpolation procedure which is divided in two stages. In the first stage, a spatial iterative process is executed for reconstructing the mesh using only the known vertices of the current mesh. In the second stage, a temporal process tries to reconstruct the entire animated mesh using knowledge of the previous frame. Fig. 4 presents the process while the reconstruction takes place gradually, starting from the second mesh and continues until the end of the animation sequence rendering the method appropriate for online setups.

As mentioned earlier, a cell **c** represents the first ring area where points (satellite vertices) are connected with an anchor vertex. Therefore we decided to build upon the assumption that the satellite vertices are expected to move towards the direction of an anchor keeping their

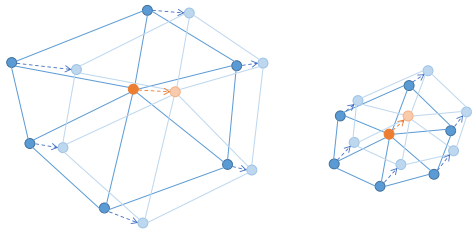


Figure 5: Example of large and small variations in cell movements.

common topological characteristics (e.g., distances between each other) unchanged. However, some satellites are connected with more than one anchors, meaning that their new position will be affected by the motion vectors of every connected anchor.

For making the estimation of coordinates more accurate we suggest a weighted reconstruction function which is defined by exploiting the following observations. As it was mentioned earlier, when a satellite is connected with more than one anchors then its new position is affected by the motion vectors of all the anchors. However, each anchor contributes with a different weight that is related to the relative distance between anchors and satellites. Smaller cells are more rigid so that their satellite points are expected to follow the motion vectors of the closest anchor, as shown in Fig. 5.

The second rule that we apply, is based on the fact that some points are more trustworthy than others. In other words we give more emphasis on the anchor points instead of the satellites due to their known position. Subsequently, we give more emphasis in the first generation satellites rather than the next generation because they are connected directly with the anchor points so that their estimated position is expected to be more accurate. According to the aforementioned observations, we distinguish two different weighted factors:

(a) the weighting factor s_{ij} that represents the inverse distance between points j (new discovered satellite) and i (already known point) such as $s_{ij} = 1/||\mathbf{v}_i - \mathbf{v}_j||_2^2$.

(b) the weighting factor w_i that represents the prioritization weight of vertex i. More specifically, anchors prioritization weights have the highest value while the last generation satellites have the smallest one.

For each vertex $v_j \forall\ j = 1, k-k'$ we define a weighted matrix $\mathbf{S}_j = [s_{1j}\ s_{2j}\ \cdots\ s_{nj}]$ that consists of n' elements that represent the distance between the current vertex and the known vertices that are connected with the vertex v_j. The matrix $\mathbf{W} = [w_1\ w_2\ \cdots\ w_{k'}]$ is universal and can be used by every vertex v_j. However, in each iteration the values of their elements increase by one, while in every new element we assign a unit weight. For each satellite j we estimate its new coordinates in the $(p+1)$

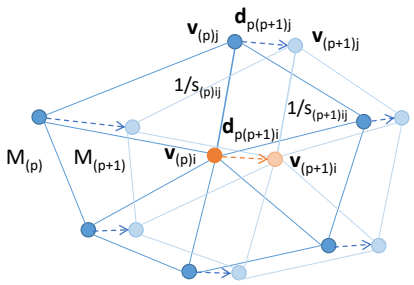


Figure 6: Anchor and satellites movement in a cell of two sequential meshes.

mesh, by updating its previous coordinates in (p) mesh, based on the following equation:

$$v_{(p+1)j} = v_{(p)j} + \mathbf{d}_{p(p+1)j} \tag{3}$$

where

$$\mathbf{d}_{p(p+1)j} = \frac{\sum_{i=1}^{n} s_{ij} w_i \mathbf{d}_{p(p+1)i}}{\sum_{i=1}^{n} |s_{ij} w_i|} \tag{4}$$

The $\mathbf{d}_{p(p+1)j}$ represents the motion vector of vertex v_i from (p)-th mesh to $(p+1)$-th mesh (see Fig. 6), i represents the known vertex v_i (anchors and satellites) and $\mathbf{d} = [d_x = |x_{(p+1)} - x_{(p)}| \;\; d_y = |y_{(p+1)} - y_{(p)}| \;\; d_z = |z_{(p+1)} - z_{(p)}|]$ is a distance vector. An overview of the proposed method is briefly presented in the following Algorithm 1.

Algorithm 1: Reconstruction of 3D animated model

Function: 3D mesh reconstruction based on previous complete mesh
Input : Animated 3D model A' with missing data.
Output : A reconstructed animated model $\bar{A}$.

```
Find the connectivity R of M_1;
for i ≤ n do
    while Number of known vertices < k do
        Search for satellate points using the connectivity of C;
        Estimate the wighted distance via Eq. (4);
        Update vertex based on its previous frame coordiantes via Eq. (3) ;
    end
end
return Reconstructed animated model Ā;
```

4 RESULTS

In this section, we present an experimental analysis of the proposed completion approach on different dynamic meshes. The evaluation of both the execution time and the reconstruction quality shows the effectiveness of our method even in complex motion scenario that include rapid changes between sequential frames or in cased with a small percentage of known points.

Experimental Setup

In all the experiments we have used a PC Intel core i7-4710HQ CPU @ 2.50GHz 2.50GHz, 8 GB RAM. The algorithms have been implemented using the Julia scientific language.

Metrics

The quality of the reconstructed results are evaluated using the metrics that are briefly presented below:

NMSVE. Normalized mean square visual error is used in order to evaluate the reconstruction quality of results, by capturing the average distortion between the original and the approximated frame [CG04]:

$$NMSVE = \frac{1}{2k} \sum_{j=1}^{k} \left(\left\| {}_{v_i - \tilde{v}_i} \right\|_2 + \left\| {}_{GL(v_i) - GL(\tilde{v}_i)} \right\|_2 \right) \tag{5}$$

$$GL(v_i) = v_i - \frac{\sum_{j \in \mathbf{N}_i} d_{ij}^{-1} v_j}{\sum_{j \in \mathbf{N}_i} d_{ij}^{-1}} \tag{6}$$

d_{ij} denotes the Euclidean distance between i and j.

Heatmap. To efficiently highlight the visual difference between reconstructed and original mesh we use heatmap visualization of $|M_i - \bar{M}_i| \; \forall \; i = 1, n$.

Dataset

Two types of 3D animated models were used in our experiments. These models represent different case studies because of their inherent properties and the fact that they target on different applications (e.g., immersive tele-presence systems, gaming). Specifically, (a) Handstand has many smooth areas, while there are no abrupt temporal changes (175 frames, 10002 vertices and 20000 triangles). (b) Ocean on the other hand is full of repetitive abrupt changes (1500 frames 2500 vertices and 4802 triangles).

Comparison Methods

For comparison purposes, we have also employed conventional techniques for the reconstruction of the animation models, namely the least-square meshes (LSM) algorithm [SC04] and the Laplacian interpolation approach (LIA) [OOH89]. LSM is described as the solution of an extended system of equations $[L^T I^T_{n \times k'}]^T x_p = [0^T, x_{k',k}]$, for $p = 1, \cdots, n$, for k' known (anchor) vertices in the p-th frame. Laplacian interpolation is described as a fast and effective method with a lot of similarities with LSM. According to [OOH89] a way to interpolate a triangulated mesh is by putting constraints on the Laplacian Δf of the function and trying to minimize its Euclidean norm.

Experimental Results

The processing time is related to the number of initial known vertices. Specifically, the execution time increases linearly with the number of unknown vertices. In Fig. 7 we present the processing time for the reconstruction of each mesh of the animated sequence using different initialization schemes. The required number of iterations for a complete mesh reconstruction depends on the percentage of known vertices. Additionally, we can observe that the two models have a similar behavior.

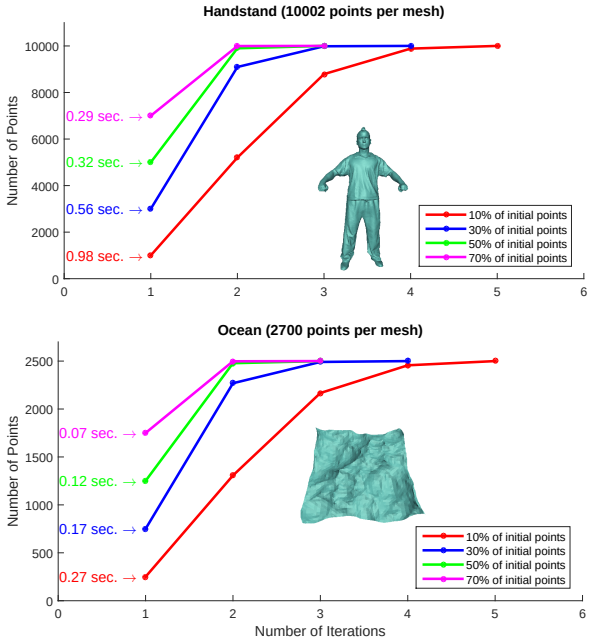


Figure 7: Number of iteration and processing time for a full mesh reconstruction.

In Fig. 8 the missing vertices are visualized with red color and the known vertices with blue (1st and 3rd row). A heatmap visualization is also offered presenting the squared difference between original and reconstructed mesh for different numbers of known vertices (2nd and 4th row). The compared results of our approach and LIA are presented in Fig. 9 showing that our method outperforms LIA in both reconstruction quality and execution time. A major disadvantage of LIA is the smoothed results even in cases with a high percentage of remaining points. In terms of execution time, our method becomes faster when more remaining points are used, because of the less iterations that are required, contrary to LIA where the execution time increases because of the larger matrix operations. Fig. 10 illustrates some indicative reconstructed frames of the Handstand model after using the aforementioned approaches. Our method seems to outperform the others while LIA and LSM have similar performance. Fig. 11 presents some indicative reconstructed frames of different animated

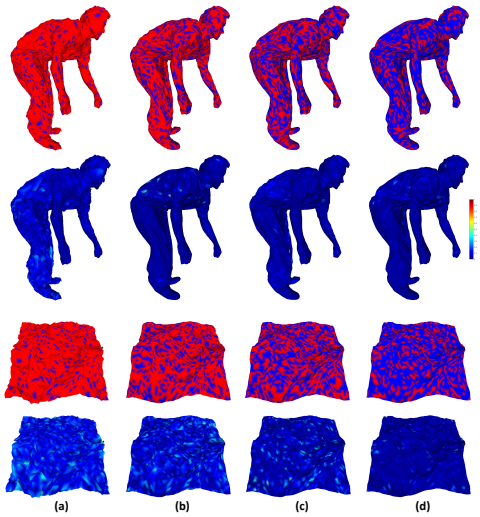


Figure 8: Visualized missing data and heatmap visualization for different density of points (a) 10% of original points, (b) 30% of original points, (c) 50% of original points, (d) 70% of original points. (Handstand frame 110, Ocean frame 1500).

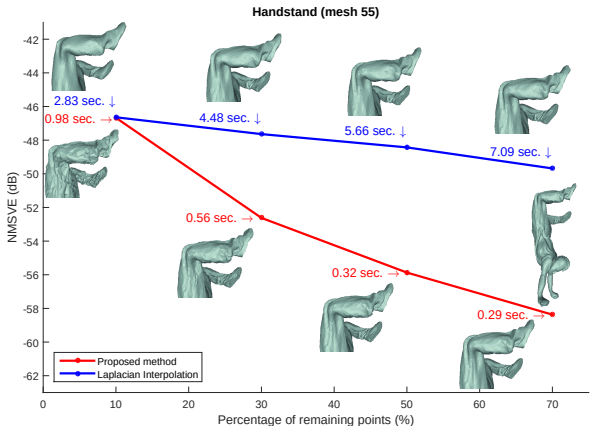


Figure 9: NMSVE and processing time results for the two compared methods.

models. For the sake of completeness, the NMSVE values are also illustrated under each reconstructed mesh. Finally, it should be noted that despite the high motion variance of animated trajectories, the perceptual quality of the reconstructed dynamic meshes when the density of the known point is higher than $> 30\%$ is considerably high. This totally satisfies the main goal of this work which is the design and implementation of fast and effective dynamic mesh reconstruction approaches.

5 CONCLUSIONS AND FUTURES EXTENDS

In this work we introduced a fast and effective method for reconstructing animated 3D models with missing data. The proposed method takes advantage of the adjacency matrix information in order to identify the coordinated of the missing vertices. After that a weighted

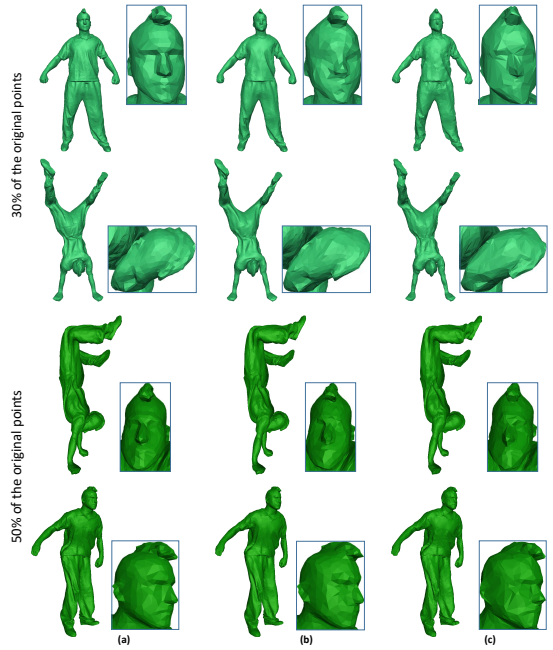


Figure 10: Handstand with 30% of original points (Frames 150 & 80) and with 50% of the original points (Frames 55 & 120) (a) our method, (b) LIA, (c) LSM.

iterative procedure estimates the position of missing vertices based on their previous position and the motion vectors of the connected anchors. An extensive evaluation study using a collection of different 3D animation models verified that the proposed technique achieve plausible reconstruction output and fast execution times.

6 REFERENCES

[BTSAL14] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, et al.. State of the Art in Surface Reconstruction from Point Clouds. Eurographics 2014 - State of the Art Reports, Apr 2014, Strasbourg, France. 1 (1), pp.161-185, 2014, EUROGRAPHICS.

[SGP03] P. Liepa. 2003. Filling holes in meshes. In Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP '03). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 200-205

[OOH89] Thom F Oostendorp, Adriaan van Oosterom and Geertjan Huiskamp, Interpolation on a triangulated 3D surface. Journal of Computational Physics, 80: 331-343, 1989.

[CG04] Z. Karni and C. Gotsman, Compression of soft-body animation sequences, Computers Graphics, vol. 28, pp. 25-34, 2004.

[WJHB07] M. Wand, P. Jenke, Q. Huang, M. Bokeloh, L. Guibas, and A. Schilling, Reconstruction of Deforming Geometry from Time-Varying Point Clouds, EUROGRAPHICS, 2007.

[DDZ11] Y. Deng, Q. Dai, and Z. Zhang, Graph laplace for occluded face completion and recognition, IEEE Transactions on Image Processing, vol. 20, no. 8, pp. 2329-2338, Aug 2011.

[PGWXM12] Y. Peng, A. Ganesh, J. Wright,W. Xu, and Y. Ma, Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 11, pp. 2233-2246, Nov 2012.

[CAN12] Candes, Emmanuel, and Benjamin Recht. "Exact matrix completion via convex optimization." Communications of the ACM 55.6 (2012): 111-119.

[DDZW12] Y. Deng, Y. Liu, Q. Dai, Z. Zhang, and Y.Wang, Noisy depth maps fusion for multiview stereo via matrix completion, IEEE Journal of Selected Topics in Signal Processing, vol. 6, no. 5, pp. 566-582, Sept 2012.

[SWG08] J. Süßmuth, M. Winter, G. Greiner, Reconstructing animated meshes from time-varying point clouds, Eurographics Association, vol. 27, no. 5, pp. 1469-1476, 2008.

[RPMR13] C. Ren, V. Prisacariu, D. Murray and I. Reid, STAR3D: Simultaneous Tracking And Reconstruction of 3D Objects Using RGB-D Data, IEEE International Conference on Computer Vision, 2013.

[ACSTD07] P. Alliez, D. Cohen-Steiner, Y. Tong, M. Desbrun, Voronoi-based variational reconstruction of unoriented point sets. In Computer Graphics Forum (Proc. of the Symposium on Geometry Processing), 2007.

[SLS07] A. Sharf, T. Lewiner, G. Shklarski, S. Toledo and D. Cohen-Or, Interactive topology-aware surface reconstruction. ACM Trans. Graph. (Proc. SIGGRAPH), 2007.

[DGQ12] T. K. Dey, X. Ge, Q. Que, I. Safa, L. Wang, Y. Wang, Feature-preserving reconstruction of singular surfaces, In Computer Graphics Forum, 2012.

[SC04] Least-Squares Meshes. In Proceedings of the Shape Modeling International 2004 (SMI '04). IEEE Computer Society, Washington, DC, USA, 191-199.

[VLMB07] E. Vlachos, A. Lalos, K. Moustakas, K. Berberidis, Efficient graph-based matrix completion on incomplete animated models, IEEE International Conference on Multimedia and EXPO (ICME) 2017, At HONG KONG.

[ZA04] Li Zhang et. al., 'Spacetime faces: High resolution capture for modeling and animation,' ACM Trans. Graph., vol. 23, no. 3, pp. 548-558, Aug. 2004.

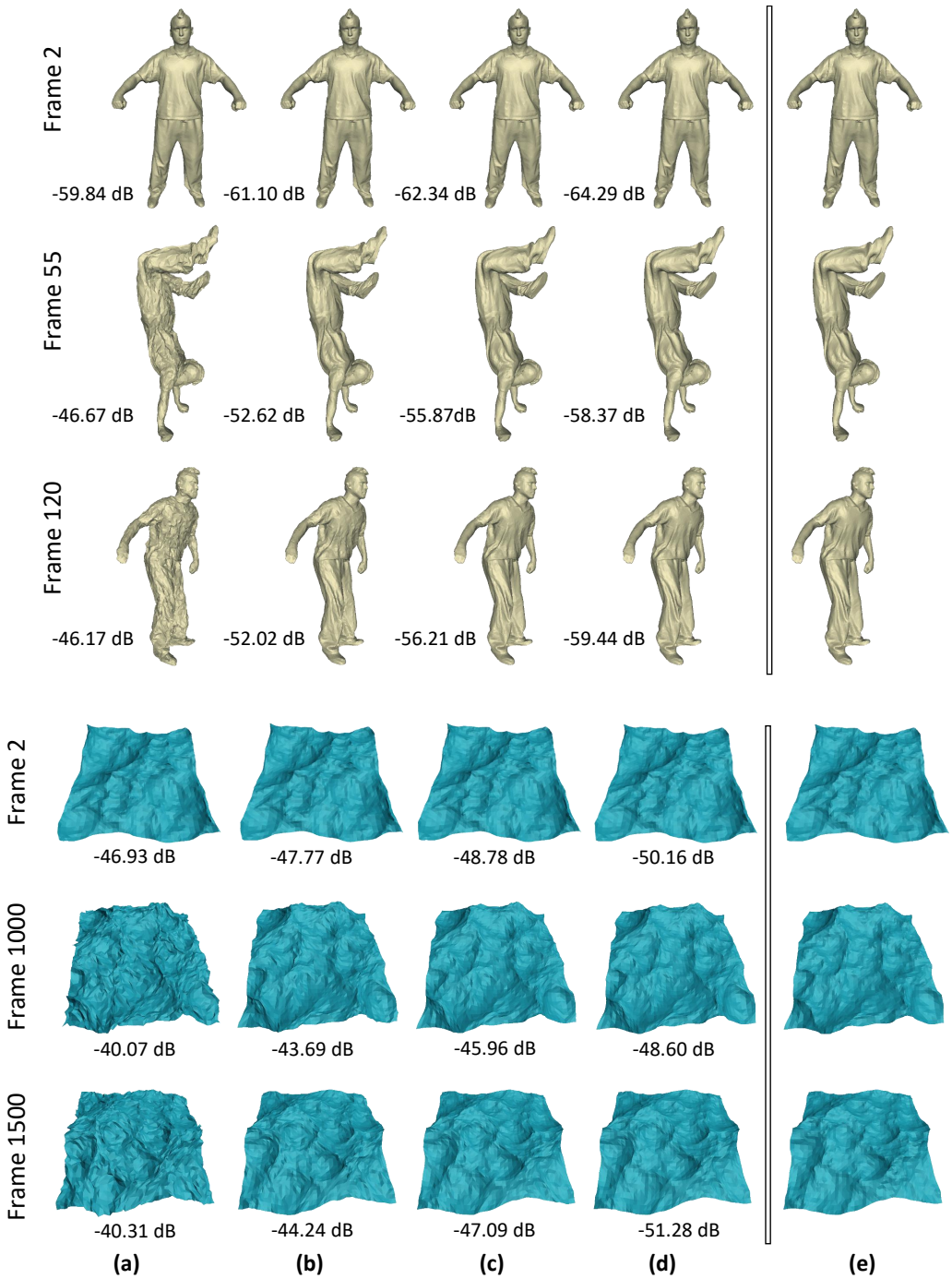


Figure 11: Reconstructed meshes for different density of remaining points (a) 10% of original points, (b) 30% of original points, (c) 50% of original points, (d) 70% of original points, (e) original mesh. (Handstand & Ocean)

StreetGAN: Towards Road Network Synthesis with Generative Adversarial Networks

Stefan Hartmann Michael Weinmann Raoul Wessel Reinhard Klein
University of Bonn
Institute for Computer Science II
Friedrich-Ebert-Allee 144
Germany, 53113 Bonn
{hartmans,mw,wesselr,rk}@cs.uni-bonn.de

ABSTRACT

We propose a novel example-based approach for road network synthesis relying on Generative Adversarial Networks (GANs), a recently introduced deep learning technique. In a pre-processing step, we first convert a given representation of a road network patch into a binary image where pixel intensities encode the presence or absence of streets. We then train a GAN that is able to automatically synthesize a multitude of arbitrary sized street networks that faithfully reproduce the style of the original patch. In a post-processing step, we extract a graph-based representation from the generated images. In contrast to other methods, our approach does neither require domain-specific expert knowledge, nor is it restricted to a limited number of street network templates. We demonstrate the general feasibility of our approach by synthesizing street networks of largely varying style and evaluate the results in terms of visual similarity as well as statistical similarity based on road network similarity measures.

Keywords

deep learning, generative modeling, generative adversarial networks (GANs), road network generation.

1 INTRODUCTION

High-quality productions such as video games, simulations or movies rely on high-quality content including detailed virtual environments, buildings and realistic road networks. However, the manual design and modeling of such content from scratch is a time-consuming and tedious task. Therefore, the automation of high-quality content production has become an active line of research in recent years. Automatic content generation has been addressed using various approaches that follow the concepts of procedural modeling, inverse-procedural modeling or example-based modeling. While procedural approaches rely on manually designed grammar snippets and rule sets to derive geometric representations of buildings, plants or road networks, inverse procedural approaches try to infer the production rules from a given set of existing examples. In contrast, example-based approaches inspect small real-world examples and decompose them into a set of building blocks in an offline step. Novel content is then generated by custom tailored algorithms that reshuffle, recombine, and bend the content in order to statistically and perceptually match the style present in the examples. Furthermore, the potential of deep learning techniques for procedural content generation [23, 15, 32] has been investigated. These techniques based on convolutional neural networks (CNNs) have already been established as promising workhorses in other areas of computer graphics like texture synthesis [11, 20].

In this work, we propose a novel example-based approach for road network generation that leverages the potential of modern deep learning techniques. The input for our method is a road network patch extracted from *OpenStreetMap* (OSM). As the data is publicly available and maintained by a large community no further domain-specific expert knowledge for data preparation and/or annotation is required. Our method comprises three major components. The first component prepares and converts an input road network into a binary image, where the pixel intensities encode the presence or absence of roads. The second component trains a generative adversarial network (GAN) [14] on image patches extracted from the prepared road network image. The third step utilizes the GAN to synthesize arbitrary sized images that contain a rastered road network. In order to use the produced road network encoded in the image in GIS applications such as CityEngine[8], we extract the road graph and post-process it in a final step. The results shown in Section 5 illustrate that our approach is able to synthesize road networks that are visually similar when compared to

the original road networks. Moreover, they also faithfully reproduce road network properties like city block area, compactness and block aspect ratio (see Section 5.2). The statistical evaluation furthermore shows that the major characteristics and the style of the road network present in the original networks can also be found in the synthesized results.

To the best of our knowledge, no other method for road network generation using a GAN approach has been published so far. However, we believe that this technique is particularly well-suited for the envisioned task, as due to their nature GANs try to learn the distributions that underlie a set of training images or patches. This might overcome the need for manual definition of rule sets and parameter tuning for procedural algorithms which can be tedious for non-expert users. In addition, such a technique might boost the expressiveness of custom-tailored example-based synthesis algorithms that is typically limited by the variation found within the input template.

2 RELATED WORK

The first part of this section reviews procedural and example-based approaches with a strong focus on road network generation. In the second part, we briefly review content generation approaches that leverage the power of deep learning techniques. As no approach for road network generation that leverages deep learning techniques has been presented so far, we instead review approaches that combine CNNs with procedural and data-driven content generation algorithms.

Procedural approaches: In a comprehensive survey on procedural and inverse procedural modeling, Smelik et al. [29] discuss different approaches and applications. In general, procedural methods rely on the use of manually defined or automatically determined rule sets for content generation. Such approaches have e.g. been followed by Parish and Müller [25], where *Open L-Systems* are used to procedurally grow road networks from an initial seed point. Galin et al. [10] generate procedural roads between two end points by computing the anisotropic shortest path incorporating the underlying terrain and user defined environmental cost functions. In Galin et al. [9], the focus is on generating road networks for inter-city connection. Benes et al. [2] grow street networks from multiple seed points. Each seed points represents an individual city, that is expanded by guiding the growth process with a virtual traffic simulation. The controllability of procedural road networks was improved by Chen et al. [4] using tensor and direction fields to guide the road network generator. Emilien et al. [6] focus on procedural generation of villages on arbitrary terrain. Their road network generator is custom tailored to courses of streets found in small villages.

Example-based approaches: In contrast to procedural approaches, example-based methods do not require an underlying rule set to generate content. Instead, they rely on analyzing the data such as the road network or a city layout in a pre-processing step to extract templates and/or statistical information. In Aliaga et al. [1], intersections are enriched with attributes such as intersection degree, street level, etc. A novel network is generated by employing a random walk using the attributes stored at junctions as guidance. Yang et al. [31] focus on the synthesis of suburban areas and districts. Starting from an empty domain they apply a recursive splitting technique based on either template matching followed by deformation, or streamline-based splitting using a crossfield. Emilien et al. [7] learn distributions from small patches of generated or manually modeled 3D content. The learned distributions are applied in a brushed-based modeling metaphor in order to steer the underlying procedural content generators that produce roads, foliage, trees or buildings. Nishida et al. [22] extract road patches from real road networks. From an initial seed point a road network is grown by attaching road patches to connector streets taking the terrain into account. In cases where no example-based growth is possible, statistical growing similar to [1] is employed.

Learning-based approaches: Emerging deep learning techniques have been used for procedural and data-driven content generation. In Yumer et al. [32], a low-dimensional generative model from high-dimensional procedural models incorporating shape features is learned. Novel models can then be generated by interpolating between points in a low-dimensional latent space enabling faster and more intuitive modeling of shape variations. Huang et al. [15] present a sketch-modeling system using CNNs. CNNs are trained on synthetic line drawings produced by procedural models with varying parameters. Novel shapes can then be generated by regressing the parameters according to a user provided sketch depicting the desired output. A similar approach was proposed by Nishida et al. [23] focusing on interactive modeling of buildings. The authors train CNNs for classifying the type of a rule snippet as well as for regressing their parameter sets from synthetic line renderings of the rule snippets. The user iteratively sketches building mass, roof, etc., and the CNNs are used to classify the resulting shapes and to infer their parameters. Ritchie et al. [27] focus on controlling procedural algorithms using neural networks. In particular, the neural network manipulates the next steps of the algorithm based on the content generated so far. Apart from the approaches that require the existence of procedural models/algorithms, pure image-based algorithms have been investigated for controlled content generation. Isola et al. [16] investigate GANs

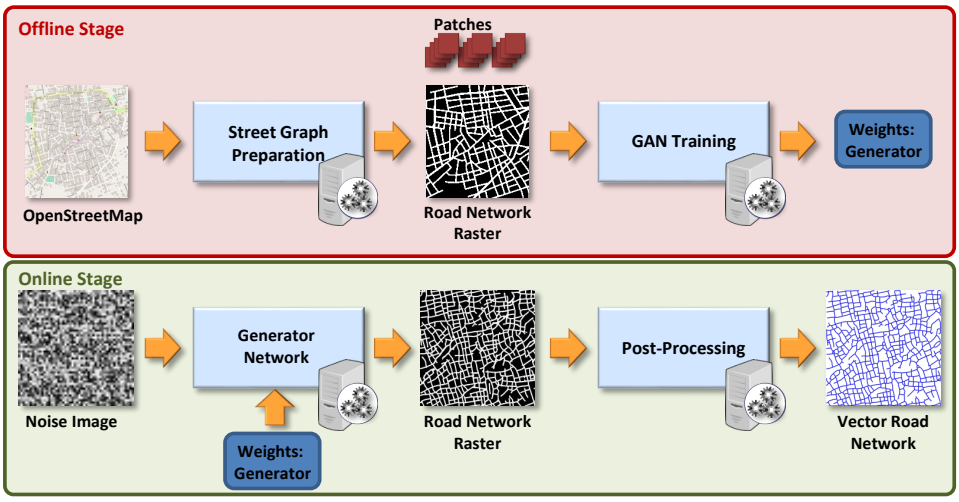


Figure 1: Our system is composed of two components. In an offline step, a road network patch taken from a real-world city is rastered into an image. The rastered road network is used to train a GAN and the generator weights are stored. In an online step, the trained model, i.e. the generator weights, are used to synthesize road network variations from images containing uniformly sampled noise. A clean graph is extracted from the produced image ready to use in GIS applications.

for transfer learning, i.e. they learn a mapping from one object representation into another such as *urban map to aerial image* or *sketch to image*. The authors show that GANs can be used to learn such a mapping without custom feature engineering. More recently, a texture synthesis approach utilizing GANs has been proposed by Jetchev et al. [17]. With their framework, they are able to synthesize textures of arbitrary size from spatial noise. This technique called *Spatial GAN (SGAN)*, a specialized GAN technique, serves as basis for our approach.

3 REVIEW OF GENERATIVE ADVERSARIAL NETWORKS

Before outlining our approach in Section 4, we provide a brief overview about generative adversarial networks (GANs) that we apply to generate road networks. GANs are a technique to learn a generative model based on concepts from game theory. The key ingredients of GANs are given by two players, a generator G and a discriminator D. The generator is a function $G_{\theta^G}(z) : \mathbb{R}^d \rightarrow \mathbb{R}^{w \times h \times c}$ that takes as input a vector $z \in \mathbb{R}^d$ sampled from a d-dimensional prior distribution $p_z(z)$ such as a uniform distribution and uses the parameters $\theta^{(G)}$ to transform it into a sample image x'. The fabricated sample $x' = G(z)$ is an image $x' \in \mathbb{R}^{h \times w \times c}$, where w and h denote its width and its height and c denotes its channels. In contrast, the discriminator D is a function $D_{\theta^D}(x) : \mathbb{R}^{w \times h \times c} \rightarrow \mathbb{R}$ that takes as input either an image patch x from the training set or a fabricated image x', and uses its parameters $\theta^{(D)}$ to produce a scalar that represents the probability that the investigated sample is a example x from training set, or a fabrication x' produced by G. The discriminator cost is accordingly given by

$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x \sim p_{data}(x)} \log(D(x)) - \frac{1}{2}\mathbb{E}_{z \sim p_z(z)} \log(1 - D(x'))$$

which is the standard cross-entropy for a binary classification problem. The discriminator tries to minimize $J^{(D)}(\theta^{(D)}, \theta^{(G)})$ while it controls only $\theta^{(D)}$, however, it also depends on the parameters $\theta^{(G)}$ of the generator. The term, $\mathbb{E}_{x \sim p_{data}(x)}[\log(D(x)]$, measures the skill of D to distinguish fabricated samples x' from real ones x that are produced by the data-generating distribution p_{data}. In contrast, $\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(x')]$ measures the skill of G to fabricate examples, that are misclassified by D and thus considered as real examples. In the previous terms, $\mathbb{E}$ represents the expectation value of the *log*-probability, which in practice boils down to the arithmetic mean of the *log*-probabilities computed using the samples of the current training iteration. The cost function of G is given by $J^{(G)}(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}(\theta^{(D)}, \theta^{(G)})$ and its goal is to maximize D's error on the fabricated examples x'. As both cost functions follow different goals and compete against each other, the underlying problem is described as a game between the two players [13, 12]. One strategy to solve the described problem is in terms of a zero-sum game also called *minimax* game. The game is accordingly described by the objective

$$\arg\min_{\theta^{(G)}} \max_{\theta^{(D)}} -J^{(D)}(\theta^{(D)}, \theta^{(G)})$$

where $-J^{(D)}(\theta^{(D)}, \theta^{(G)})$ represents the discriminator's pay-off. The overall goal of such a game is to

minimize the possible loss for a worst case maximum loss. In particular, this is realized by performing a minimization in an outer loop, while performing a maximization in an inner loop. We refer the reader to a recent tutorial by Goodfellow [12] for additional details.

In practice, G and D are represented as neural networks and training them is similar to finding the *Nash* equilibrium of the described *minimax* game played by G and D. A *Nash* equilibrium in such a context can be described as a parameter state $(\theta^{(D)}, \theta^{(G)})$ that is a local minimum of $J^{(D)}$ and a local minimum of $J^{(G)}$. In order to keep the problem tractable, G and D are trained in an alternating fashion instead of using the nested loops as described above. Furthermore, G's cost function $J^{(G)}(\theta^{(D)}, \theta^{(G)})$ is changed to $-\frac{1}{2}\mathbb{E}_{z\sim p_z(z)} \log(D(x'))$. The term in the original cost function $-\frac{1}{2}\mathbb{E}_{z\sim p_z(z)} \log(1-D(x'))$ would lead to vanishing gradients during the training, when D successfully rejects examples fabricated by G. Instead of previously minimizing the *log*-probability that the sample x' is classified as fabricated, the new goal of the generator G is now to maximize the *log*-probability that D performs a wrong classification. As noted in [12], that change enables both G and D to produce strong gradients during the final training.

For the modified game and its training this particularly means that in one iteration D is trained, while in the next iteration G is trained. As we search a local minimum for D and G, the parameters of current component are updated in each iteration using stochastic gradient descent. When G is trained, its parameters are tuned towards the production of samples x' that are indistinguishable from the real training data and thus to fool the discriminator D. In contrast, when D is trained its parameters are tuned to improve D's skill to discriminate the fabricated samples x' from the real samples x. For additional details about the theoretic background we refer the interested reader to [12, 17].

So far, when the GAN is trained using neural networks, no well-founded theory about the determination of the success of training procedure of a GAN can be found in literature. Therefore, it is necessary to visually check generated samples and to capture the weights θ^G of G that fabricate visually pleasing outputs. Note, there is no need to capture θ^D because after the training D can be omitted and only the generator G is necessary to produce new samples [13, 12, 17].

4 ROAD NETWORK SYNTHESIS USING GAN

In this Section, we outline our street network generation approach by providing an brief description of its major components.

4.1 Pipeline Overview

In order to successfully apply a GAN model to street network data in vector representation as provided by *OpenStreetMap* (OSM), we developed three components to approach this task (see Figure 1). In an offline step, a sample map from OSM is used to create an image that contains a raster representation of the street network (see Section 4.2). The produced image is used to extract a set of randomly chosen image patches of size $n \times n$. These patches contain subparts of the initial road network and are used to train the GAN (see Section 4.3). Afterwards, novel road network patches can be generated from a d-dimensional z with samples drawn from $p_z(z)$. z serve as input for the generator network G that maps them to a grayscale image $x \in \mathbb{R}^{w\times h\times 1}$ representing a rastered road network. The resulting road network is encoded by pixel intensities (cf. Section 4.4) and the discrete representation is transformed into a graph-based one in a post-processing step.

4.2 Road Network Preparation

We use publicly available community mapping data from OpenStreetMap (OSM) [24] datasets in which road networks are represented as piecewise-linear polylines, that are attached a *highway* label in order to distinguish them from other structures like buildings, rivers, and parks. Among other polylines in an OSM dataset roads can be identified by the label *highway* attached them. Each road is assigned a specific *highway* type representing its category. For all our examples, we extract roads from the OSM dataset that have one of the following *highway*-categories: **motorway, primary, secondary, tertiary, residential, living_street, pedestrian**. The raw road network extracted from OSM is represented as vector data in geo-coordinates. As well-established CNN pipelines require images as input, we transform the road network into a raster representation. First, we project the geo-coordinates to WGS84, which is a well-established coordinate projection that transforms geo-coordinates given in *Latitude/Longitude* to meters. Next, we scale the road network that each pixel in the rastered representation represents an area of 3×3 meters. Finally, we raster the road segments as lines with a width of 15 pixels using the line rasterization routine of OpenCV [3] to produce a binary image, in which white pixels now represent the presence of roads while black pixels represent the absence of roads. Please note that we inverted the colors in the Figures shown in the paper.

4.3 Training Procedure

We train the GAN on images patches with fixed size of $n \times n$ pixels extracted from the image containing the

rastered road network. In order to provide a suitable large training set and to enable the network to capture the local statistics well, we perform the training on images patches. A well-established approach for training GANs is an iterative and alternating training procedure. This means that G and D are trained in an alternating fashion every second iteration. In the step when G is trained, it takes as input a set $Z = \{z_0, \ldots, z_k\}$. As described in Jetchev et al. [17], that serves as basis for our training, each z_i is a tensor $z_i \in \mathbb{R}^{m \times n \times d}$ where at each spatial location a d-dimensional vector drawn from p_z it used. The z_i and is then transformed by G into a grayscale image $x'_i = G(z_i)$ of size $x'_i \in \mathbb{R}^{n \times n \times 1}$. When D is trained, a set $X = \{x_0, \ldots, x_k\}$ of k image patches $x_i \in \mathbb{R}^{n \times n \times 1}$ and the set $X' = \{x'_0, \ldots, x'_k\}$ generated by G in the previous step serve as input. The samples $x \in X$ are extracted from random locations inside the training image. We refer the reader to the work by Jetchev et al. [17] for additional details about the training procedure. Please note that we use only a single image, that provides patches for the training procedure, but using multiple different images would be possible and would increase the examples in the training set.

4.4 Road Network Post-Processing

In order to use the resulting road network in GIS applications or in a road network analysis task, we need to transform grayscale image intensities to a road network graph. For this purpose, we apply a post-processing to the synthesized images.

Image post-processing: The grayscale images produced by the generator network contain pixel intensities in the range $[0, 255]$ (see Figure 4). In a first step, we threshold the gray values at 127 in order to produce a binary image where pixels set to *true* represent the presence and non-set pixels represent the absence of road. Applying the threshold might produce undesirable isolated pixels and also small cusps along road regions. In order to get rid of these artifacts, we apply a morphological erosion operation. However, the produced result might still contain small holes or road regions that do not touch within a radius of up to five pixels. In order to close such small gaps, we apply five steps of morphological dilation. For all morphological operations, we use a 3×3 structuring element with the shape of a cross. The obtained initial road network, however, contains road regions that are too thick to extract an initial road graph. Therefore, we thin out the result to extract a skeleton from the cleaned binary image using the algorithm from Zhang et al. [33].

Road graph construction: We utilize the pixel-skeleton from the previous step to construct an initial graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ representation of the synthesized road network, where $\mathscr{V}$ are its vertices and $\mathscr{E}$ are its edges. In order to construct $\mathscr{G}$, we add a node V_i to $\mathscr{V}$ for each of the skeleton pixels. Next, we examine the 8-neighborhood of each V_i in the image. For each skeleton pixel V_j inside the 8-neighborhood of V_i, we add an edge $E_{ij} = (V_i, V_j)$.

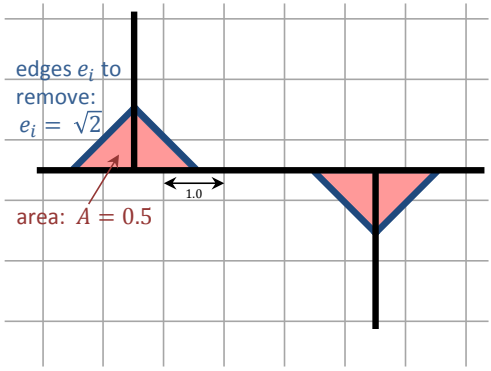


Figure 2: Block artifacts resulting from graph construction

Cityblock cleanup: The graph construction from the pixel skeleton produces regions within the road network that have a very small area of 0.5 square pixels (see Figure 2), which are removed in a first step. The regions within a road network graph are typically called city blocks. Strictly speaking, a city block is a region within the graph $\mathscr{G}$, that is enclosed by a set of road segments and might contain dead-end street segments. In order to identify these small regions, we first compute all the city blocks of the graph. As the graph is a directed graph and embedded in $\mathbb{R}^2$, the city blocks can be computed by determining the minimal cycles of the graph by computing loops of edges. Next, we filter out blocks with an area of 0.5 pixels. These artifact blocks can be removed by identifying and removing their longest edge, which has a length of $\sqrt{2}$.

Road courses smoothing: Another artifact produced by constructing $\mathscr{G}$ from the image raster are jagged edges. In order to smooth these in the final graph, we extract a set of street chains $S = \{S_i\}$ from the graph. Each $S_i = \{V_0, \ldots, V_n\}$ consists of n nodes, while the degrees of V_0 and V_n are constrained by $deg(V_0) \neq 2$ and $deg(V_n) \neq 2$. From each of the S_i, a polyline $P_i = \{p_0, \ldots, p_n\}$ with n positions is built. A smoothed version of the positions can be obtained by applying 5 steps of local averaging $\hat{p}_i = \frac{1}{4}p_{i-1} + \frac{1}{2}p_i + \frac{1}{4}p_{i+1}$ to the p_i's with $i \in [1, n-1]$, and replacing the original p_i's with their smoothed version $\hat{p}_i$. Finally, we additionally straighten the road courses, by removing superfluous nodes using the Douglas Peucker simplification [5]. We allow to remove nodes that deviate up to 3 meters from a straight line. The last step removes short dead-end street chains with an total length of less than 25 meters.

5 CASE STUDY: ROAD NETWORK TYPES

In order to showcase the versatility of our road network synthesis approach, we evaluate our approach on a set of challenging test cases. We composed a collection of real-world as well as synthetic road network examples (see Figure 3 a)-d)). The real-world examples were taken from OSM, while the synthetic ones were taken from [21]. For all the examples shown in here, we used a patch size of 321×321 (cf. Figure 9) pixel during the training procedure, because that size captures the local structures found in the our test road networks. Furthermore, we used only a single road network image from which patches were extracted. We synthesized two examples for each road network shown in Figure 3 a)-d). In our evaluation, we investigate the visual appearance of the generated results and analyse the similarity in terms of road networks measures such as area, compactness and aspect ratio of the city blocks by comparing the resulting distributions.

5.1 Visual Evaluation

Irregular: Synthetic As a first test case, we considered a synthetic road network (see Figure 3a). The major characteristics of this road network are blocks of different sizes with and without dead-ends, and similar sized blocks, that form small groups. Nearly all blocks have a rectangular shape except for a few exceptions. Figure 4 (a) and (b) show road networks generated by GAN model after passing our post-processing pipeline. It can be noticed, that the generated results contain blocks similar in shape and size when compared to the original network. Notice, that the results even contain the small groups of nearly square shaped blocks that are present in the original network. Larger road courses are present in the examples, although they have a curvature different to that in the original network.

Irregular: San Marco Next, we evaluated a street network patch from a village in Italy (see Figure 3b). A major characteristic of that network is its large amount of small city blocks in comparison to only a few larger ones. Generated samples of this network type are depicted in Figure 4c and 4d. Both samples contain a significant number of small blocks when compared to the number of medium sized and large city blocks. It is also noticeable that smaller blocks are located next to each other. Furthermore, the result contains large scale structure such as connected road courses that separate groups of smaller blocks. Another produced sample visualized using CityEngine can be seen in Figure 12.

Irregular: Berlin In contrast to the previous example, the next network shown in Figure 3c is composed of a significant amount of larger, mainly square or rectangular shaped blocks. Only a few blocks are irregularly shaped and contain dead-ends. The generated samples shown in Figure 4e and 4f contain a significant amount of nearly square shaped blocks and rectangular shaped blocks. It can be recognized that the generated networks also contain irregularly shaped blocks and even L-shaped blocks not being present in the examples.

Suburban: Synthetic Next we show results generated from a synthetic network of a suburban region with structures mainly found in suburban regions of the US (see Figure 3d). A major property of such network types is the presence of curved road courses. Our produced results shown in Figure 4g and 4h contain these typical curved roads shapes.

5.2 Statistical Evaluation

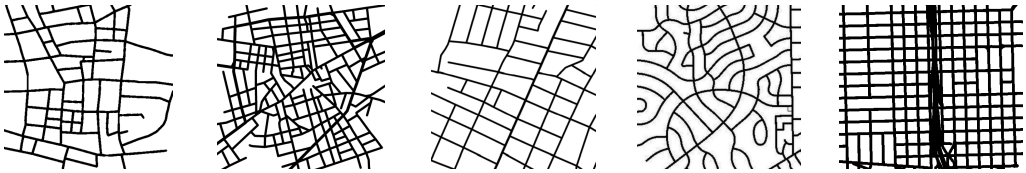
Apart from the visual comparison of the results, we performed an evaluation of graph measures computed on the synthesized road networks and the original road networks. These considered measures include the *cityblock area*, the *compactness*, i.e. the ratio between block area and its minimal bounding box, and the *city block aspect ratio*, i.e. the ratio between the shorter and the longer side of the minimal bounding box.

Irregular: Synthetic Figure 5 compares the graph measures between the synthetic irregular network shown in Figure 3a with the ones obtained from our synthesized results. While the distributions of the block area and the compactness have a similar shape, the aspect ratio distribution varies as the generated result contains much more variation of rectangular shaped blocks than the original road network.

Irregular: San Marco In this result (see Figure 3b), the distributions of block area, aspect ratio and compactness are similar (see Figure 6). The resulting network mostly consists of small city blocks as illustrated by the block area distribution. Both the original and the generated road network contain a significant amount of nearly rectangular blocks (see compactness). As the aspect ratios within the generated network are also similar, thus, learned model has captured the properties of the original network.

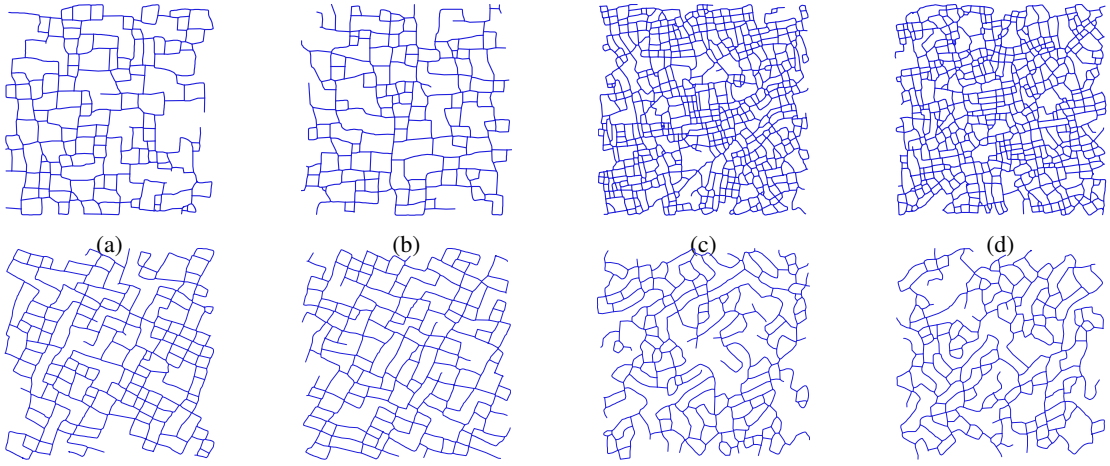
Irregular: Berlin In Figure 7, we illustrate the distributions for the Berlin example shown in Figure 3c. While the block area and the aspect ratio of the blocks found in the generated example tend to be similar, the compactness varies more than in the previous examples. As the streets in the produced network are not perfectly straight anymore, the compactness of the blocks deviates from being nearly 1.0.

Suburban: Synthetic For suburban networks such as the one shown in Figure 3d, the distributions of block area and aspect ratio differ, while especially the aspect ratios within the generated network have a few



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

Figure 3: Overview of the different road network styles used in our case study: (a) Synthetic irregular, (b) Cellino San Marco irregular, (c) Berlin irregular, (d) Synthetic suburban, (e) Portland with highway ramps



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

(e) (f) (g) (h)

Figure 4: Different samples fabricated by the generator learned from the synthetic irregular example shown in Figure 3. Synthetic irregular (a) and (b), Cellino San Marco (c) and (d), Berlin irregular (e) and (f), Synthetic suburban (g) and (h). The results are discussed in detail in Section 5.

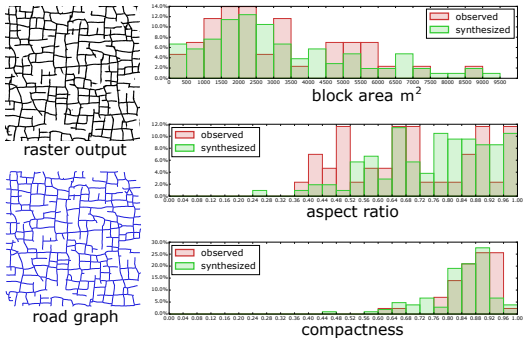


Figure 5: Statistical evaluation of Synthetic irregular

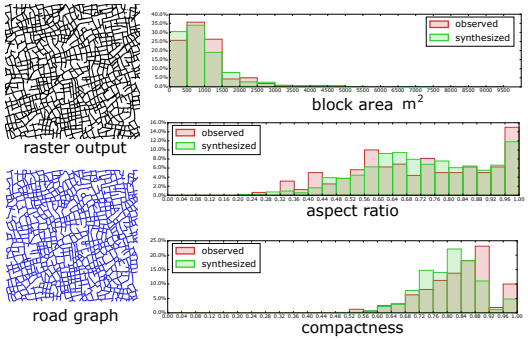


Figure 6: Statistical evaluation of Cellino San Marco

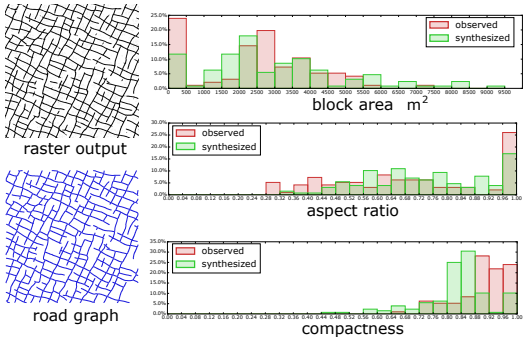


Figure 7: Statistical evaluation of Berlin irregular

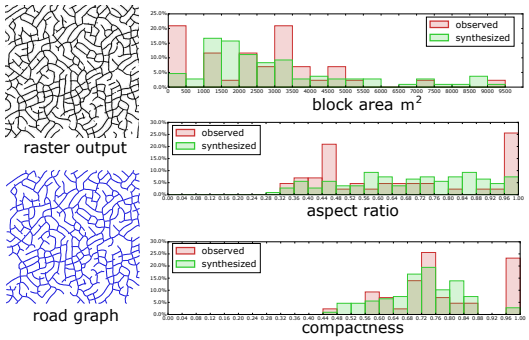


Figure 8: Statistical evaluation of synthetic suburban

spikes (cf. Figure 8). However, at a larger scale the overall shape of the distribution is similar. As this road network type contains large-scale structures such as curved roads that pass through the whole network the chosen context size cannot capture these, thus, the generated network will suffer from these missing global properties. This leads to a structurally different generated road network which is reflected by the distributions of the different graph measures.

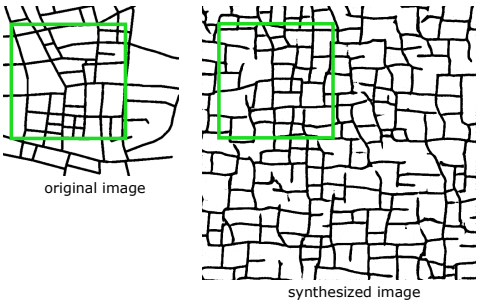


Figure 9: Illustration of the context size using during the training stage. Left: Original image with overlaid extent of the training image. Right: Generated sample with an overlay of the training image size.

5.3 Limitations

Large-scale structures and ramps. We noticed that our approach cannot successfully handle road network patches that contain highway ramps and networks that contain street lanes that are located very close to each other, as illustrated in the road network example taken from Portland (see Figure 3e). When the road network is rastered nearby lanes will be merged with and form even thicker lanes. If highway ramps are present, additional pixel blobs are introduced as illustrated in the synthesized example shown in Figure 10. It can be noticed that the grid-like road pattern is faithfully reproduced. However, due to the thick lanes and the limited context size (see Figure 9) the highway structures present in the training data cannot be recovered successfully. Instead, thick road structures occur on the left border (cf. green arrows) and blob shaped artifacts are scattered over the synthesized example (cf. region surrounded by green ellipses). When the post-processing is applied, these artifacts will be alleviated, however, irregularly shaped blocks will be present in the final road network.

Deadend roads. All the synthesized examples contain much more dead-ends when compared with the number of dead-ends present in their corresponding original road network. This might be due to the patch-based training procedure. Each patch that is used for training typically contains virtual dead-end street segments that abruptly end at the patch boundary.

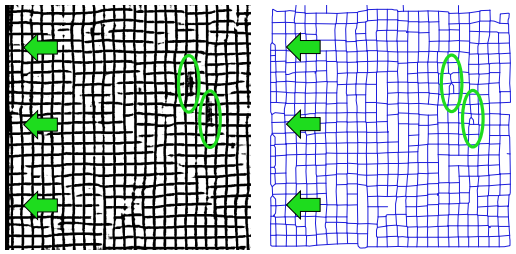


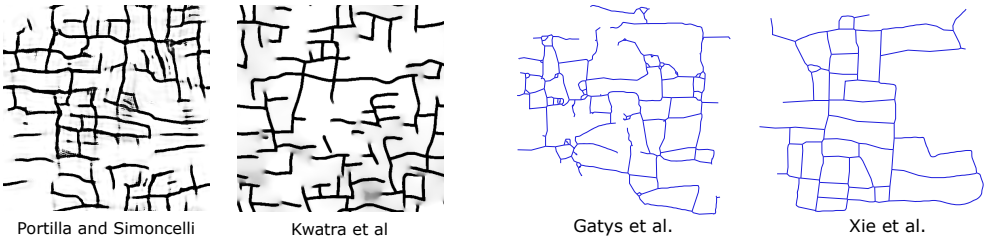
Figure 10: In case of nearby located highway lanes and highway ramps, the GAN fails to capture these properties. This leads to blob-like artifacts in the generated samples.

5.4 Street Network Generation with Texture Synthesis Techniques

We complete our case study with a brief evaluation of texture synthesis algorithms for street network generation. In particular, we synthesize road networks with patch-based texture synthesis algorithms such as method of Portilla and Simoncelli [26] and Kwatra et al. [19]. Furthermore, we evaluate recent CNN based texture synthesis algorithms, specifically the method of Gatys et al. [11] and the Generative ConvNet technique proposed by Xie et al. [30] for road network generation. Figure 11a illustrates results from Portilla and Simoncelli on the left-hand side and results from the method of Kwatra on the right-hand side. For both algorithms a variation of the irregular road network shown in Figure 3a was synthesized. As it can be clearly noticed, these methods are not able to produce large scale structures such as city blocks. Furthermore, both algorithms have problems in consistently producing connected road courses. CNN-based algorithms are able to produce large scale structure as illustrated in Figure 11b. The road network produced by Gatys et al. [11], however, lacks visual similarity to the original network. In contrast, the approach by Xie et al. [30] is able to produce a visual similar road network and captures that properties found in the original road network.

6 IMPLEMENTATION DETAILS

Our algorithms are implemented in Python and we used the GAN implementation of [17] as a basis for learning the different road network models. However, we changed the original implementation in order to consistently support single channel images. The GAN model for the different road networks is trained on a single NVidia TitanX (Pascal). Each epoch takes 100 iterations with a batch size of 64 and takes about 90 seconds to compute. We trained all the models for at least 100 epochs and decided from a visual examination of samples taken from various epochs which model to choose. The overall training is done in an offline step that takes up to 3 hours. The single



(a) Patch-based texture synthesis methods (b) CNN based texture synthesis methods

Figure 11: Evaluation of texture synthesis algorithms using the irregular road network illustrated in Figure 3a. In 11a the result of patch-based synthesis algorithms i.e. the approach of Portilla and Simoncelli and the method of Kwatra et al. are illustrated. These methods suffer from producing larger scale structures such as city blocks and connected road structures. In contrast 11b illustrates result from modern CNN based methods, i.e. the algorithm Gatys et al. and the method Xie et al. are able to reproduce connected structure and even city blocks, however, they are only able to produce images of fixed that need additional resizing.

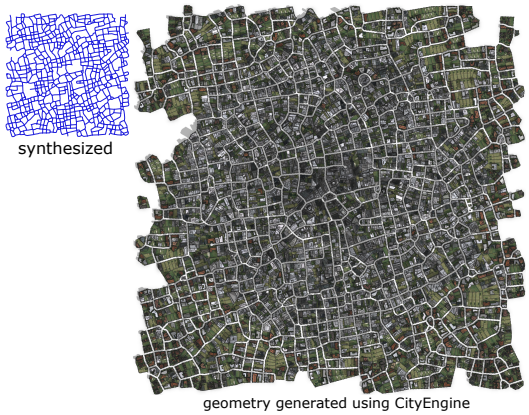


Figure 12: The resulting road network is directly usable in urban planning tools such as CityEngine.

steps of the online synthesis steps takes up to a few seconds. In more detail, the generation of a sample of size 769×769 pixels produced from a tensor $z \in \mathbb{R}^{25\times25\times100}$ sampled from $p_z(z)$, takes on average 0.08 seconds on a single NVidia TitanX (Pascal). The post-processing steps are performed on a Intel Core-i7 5820K, with only a single core in use. Each step takes: for *graph construction*: 1.5s, for *block computation*: 1.6s, for *simplification*: 2.0s and for *deadend removal*: 0.02s in average.

7 CONCLUSION

We have investigated the suitability of GANs for road network synthesis. In order to make it possible to train GAN on road network data we developed a preprocessing step. A post-processing step enabled us to extract a graph-based representation. Our results have demonstrated that GANs are able to produce novel road network patches, that are structurally sound and visually similar, when compared to the input network. Furthermore, we substantiated our results by a statistical evaluation of different road network measures such as *city block area*, *city block compactness*, and *city block aspect ratio*.

During the evaluation of our pipeline, we identified several limitations. First, structures like roundabouts, highway ramps and also roads that are very close to each other are not sufficiently captured during the training. This means that roundabouts or highway ramps cannot be successfully synthesized with our approach. Second, currently we consider all *highway* categories as part of the same street level. We did not succeed in learning models for different street levels, thus, we decided to perform the experiments using only a single street level (cf. Section 4.2). Typically, a road network naturally splits into multiple street levels such as *major* and *minor* roads. Thus, it is necessary to perform an in-depth evaluation of multiple street levels in future work. Furthermore, large-scale road courses are typically present in every road network. Although, these structures are rudimentary present in the synthesized examples shown in Section 5, our post-processing step lacks an additional step to consistently enforce such large-scale structures. One possibility to address this issue, would be fitting curves to road individual courses and enforcing global constraints such as parallelism. Another limitation is that we have only limited control over the output of the generator. In real road networks the road courses are specifically planned to fulfil specific requirements regarding landuse or terrain. Furthermore, the urban planner might also incorporate existing objects into its road design decisions. As our approach is a very first step towards using GANs for road network generation, we did not incorporate such external constraints. However, such constrains are necessary to steer the output of the generator G and leave this for future work as it would exceed the scope of the paper.

There are several interesting directions for future work. First, we would like to add attribute layers e.g. density maps, landuse maps, terrain maps etc.

in order to condition the learning process. This would make it possible to improve controllability of the generator network. Second, we would like to investigate further steps in order to train a GAN model that is able to synthesize multiple street levels. The post-processing needs to be extended to reproduce large-scale structures so that a fair comparison to existing example-based or procedural algorithms for road network generation can be given. Third, we would like to extend our approach and investigate the suitability of GANs to generate building footprints given a predefined city block shape. Finally, we would like to extend the road network generation in terms of a growing based road generation system. Apart from using GANs for urban structures we might also investigate their use for feature map generation for texture synthesis algorithms such as [28, 18].

8 REFERENCES

[1] Daniel G. Aliaga, Carlos A. Vanegas, and Bedřich Beneš. Interactive example-based urban layout synthesis. In *ACM TOG SIGGRAPH Asia*, pages 160:1–160:10, 2008.

[2] Jan Beneš, Alexander Wilkie, and Jaroslav Křivánek. Procedural modelling of urban road networks. *Computer Graphics Forum*, 2014.

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

[4] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. Interactive procedural street modeling. In *ACM transactions on graphics (TOG)*, volume 27, page 103. ACM, 2008.

[5] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[6] Arnaud Emilien, Adrien Bernhardt, Adrien Peytavie, Marie-Paule Cani, and Eric Galin. Procedural generation of villages on arbitrary terrains. *The Visual Computer*, 28(6-8):809–818, 2012.

[7] Arnaud Emilien, Ulysse Vimont, Marie-Paule Cani, Pierre Poulin, and Bedrich Benes. Worldbrush: Interactive example-based synthesis of procedural virtual worlds. *ACM Transactions on Graphics (TOG)*, 34(4):106, 2015.

[8] Esri Inc. Cityengine, 2017.

[9] Eric Galin, Adrien Peytavie, Eric Guerin, and Bedrich Benes. Authoring Hierarchical Road Networks. *Computer Graphics Forum*, 30(7), 2011.

[10] Eric Galin, Adrien Peytavie, Nicolas Marechal, and Eric Guerin. Procedural Generation of Roads. *Computer Graphics Forum (Proc. of Eurographics)*, 29(2):429–438, 2010.

[11] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, pages 262–270, 2015.

[12] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. pages 2672–2680. Curran Associates, Inc., 2014.

[15] Haibin Huang, Evangelos Kalogerakis, ME Yumer, and Radomir Mech. Shape synthesis from sketches via procedural models and convolutional networks. *IEEE Transactions on Visualization and Computer Graphics*, 2016.

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016.

[17] Nikolay Jetchev and Roland Bergmann Urs, Vollgraf. Texture synthesis with spatial generative adversarial networks. pages 2672–2680. Curran Associates, Inc., 2016.

[18] Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. Self tuning texture optimization. In *Computer Graphics Forum*, volume 34, pages 349–359. Wiley Online Library, 2015.

[19] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics (ToG)*, 24(3):795–802, 2005.

[20] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

[21] TANVI MISRA. X-ray your city's street network, 2017.

[22] G Nishida, I Garcia-Dorado, and DG Aliaga. Example-driven procedural urban roads. In *Computer Graphics Forum*. Wiley Online Library, 2015.

[23] Gen Nishida, Ignacio Garcia-Dorado, Daniel G Aliaga, Bedrich Benes, and Adrien Bousseau. Interactive sketching of urban procedural models. *ACM Transactions on Graphics (TOG)*, 35(4):130, 2016.

[24] Foundation OpenStreetMap. Openstreetmap, 2017.

[25] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. *ACM TOG (Proceedings of SIGGRAPH)*, 19, 2001.

[26] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70, 2000.

[27] Daniel Ritchie, Anna Thomas, Pat Hanrahan, and Noah D Goodman. Neurally-guided procedural models: learning to guide procedural models with deep neural networks. *arXiv preprint arXiv: 1603.06143*, 2016.

[28] Roland Ruiters, Ruwen Schnabel, and Reinhard Klein. Patch-based texture interpolation. *Computer Graphics Forum (Proc. of EGSR)*, 29(4):1421–1429, June 2010.

[29] Ruben M. Smelik, Tim Tuenel, Rafael Bidarra, and Bedrich Benes. A survey on procedural modelling of virtual worlds. *Computer Graphics Forum*, pages n/a–n/a, 2014.

[30] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative convnet. *arXiv preprint arXiv:1602.03264*, 2016.

[31] Yong-Liang Yang, Jun Wang, Etienne Vouga, and Peter Wonka. Urban pattern: Layout design by hierarchical domain splitting. *ACM TOG (Proceedings of SIGGRAPH Asia)*, 32, 2013.

[32] M. E. Yumer, P. Asente, Mech R., and L. B. Kara. Procedural modeling using autoencoder networks. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages –. ACM, 2015.

[33] TY Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.

Pushpins for Edit Propagation

Mylo, Marlon
Fraunhofer FKIE
KOM Department
Fraunhoferstraße 20
53343, Wachtberg,
Germany
and
University of Bonn
Institute of Computer
Science II
Regina-Pacis-Weg 3
53113, Bonn, Germany
mylo@cs.uni-bonn.de

Klein, Reinhard
University of Bonn
Institute of Computer
Science II
Regina-Pacis-Weg 3
53113, Bonn, Germany
rk@cs.uni-bonn.de

ABSTRACT

In this paper we present an approach for stroke-input based foreground estimation of measured materials with a near regular structure. To enable extraction of high-quality editing masks even from difficult materials, we combine a state of the art lattice-detection algorithm with a novel frequency convolution scheme, which we call *pushpins*. Despite being highly specialized, we consider this use-case as important for material design. A comparison with other state of the art editing and material recognition approaches will give proof of the robustness and ability of our algorithm.

Keywords

SVBRDF, Near Regular Texture, edit propagation.

1 INTRODUCTION

Measured materials are used to render 3D-scenes into images which evoke the impression of photorealism. While being able to edit those digital material representations is highly desirable for many applications, e.g. in film and advertising, manipulations are still a challenging tasks. Solving this problem may spare acquisition costs and admits to construct imaginary materials which appear as if they were real.

Editing measured materials is indivisibly tied to the process of isolating the geometrical or the radiometric regions which shall be manipulated.

While many brilliant algorithms have been published to master this classification problem, the productive use of those algorithms has to meet high demands. Small misclassifications lead to ugly artefacts in the resulting renderings and have to be corrected in tedious handcraft. The increasing quality in image segmentation and image matting is mostly based on a subtle exploitation of colour spaces and spatial continuity constraints. While those approaches do also apply for segmentation of materials, the results are often not good enough because different material components can very often not be distinguished by colour. But most digital material representations provide more than one diffuse colour channel. And many materials bear a *near regular structure* (*NRS*). In this paper we want to make use of those two facts to generate editing masks for measured materials in a quality which makes handcrafted optical debugging steps unnecessary. Our approach consists of a separated lattice detection step and and a classification by a *support vector machine* (*SVM*). The SVM-classification allows to use complex, high-dimensional descriptors whereas the lattice detection enables to construct only one model tile-mask and to propagate this mask via the detected lattice.

The technical contribution of this paper is 2-fold:

1. We provide a workchain to robustly solve the foreground estimation problem for measured materials with a NRS.
2. We introduce a convolutional technique to tag texels which have a similar environment like a given seed texel of the same material patch. This similarity recognition step alone is not reliable enough for stable lattice detection but it delivers a global similarity map which may be used to guide the indeterministic lattice detection step.

The structure of the paper is as follows: after a short section on the relevant related work (section 2), we will give an overview (section 3), which provides notations,

the problem statement and a walk-through. The algorithm is presented in section 4 and followed by the evaluation in section 5. The conclusion (section 6) closes with considerations about the possibilities to parallelize our system.

2 RELATED WORK

Editing measured opaque materials is an intensively studied field and there have been by far too many publications to give an exhaustive catalogue in this context. According to [7], interpolated reflectance data may directly be used for rendering materials. But those representations are expensive to store, lack explanatory power and are difficult to edit so there have been many approaches to fit measured reflectance data to analytical reflectance models, like [8,16,19,26]. Editing those analytical representations is still not easy. Some approaches operate directly on the radiometric data like the retargeting approach of An et al. [1] or the manifold based on aging simulation by Wang et al. [25]. Others try to estimate a propagation map, first, to isolate the texels to edit. Pellacini and Lawrence suggested, to use an k-nearest neighbour graph to construct a sparse adjacency matrix [21]. An and Pellacini made another step in this direction with AppProp [2], which has been extended to tabulated reflectance data by Xu et al. [27]. A recent state of the art report by Schmidt et al. [23] gives an extensive overview.

3 OVERVIEW

The overview provides the notations, the problem statement and a short walk through.

3.1 Notations and definitions

In this section we want to clarify our use of language.

Material By *material* we mean the digital representation of an existing or imaginary material-surface together with a description of the light exchange in every point of the surface.

BRDF A *BRDF* maps an incoming and an outgoing light direction onto a wavelength-dependent reflectance probability. Being reflectance distributions, BRDFs are limited to the upper directional hemisphere. In this paper we concentrate on *analytical, measured* BRDFs, meaning, that an analytical reflectance model has been optimized to fit a given set of reflectance measurements. We tested our algorithms also on tabulated reflectance representations. But to describe those reflectance tables in order to make them applicable for usage with a classifier it is necessary to bring them into a comparable format like for example Rusinkiewicz-parametrization [22] which makes a resampling-step necessary and to collect at least some elementary statistics. Investigations of this kind are beyond the scope of this paper.

SVBRDF A *spatially varying BRDF* (*SVBRDF*) is a material where the light exchange is described by a BRDF.

Ashikhmin Shirley reflectance model The measured reflectance distributions are modelled in the way suggested by Peter Ashikhmin and Michael Shirley in 2000 [3]. This is a Phong-like model which additionally controls the eccentricity of the specular lobe and is given by:

$$\begin{aligned}
&\rho(\omega_{\mathbf{in}}, \omega_{\mathbf{out}}) \\
&= \frac{\sqrt{(e_x+1)(e_y+1)}}{8\pi} \frac{\langle \mathbf{n}, \mathbf{h} \rangle^{e_x \cos^2\phi + e_y \sin^2\phi}}{\langle \omega_{in}, \mathbf{h} \rangle \max(\langle \omega_{in}, \mathbf{h} \rangle, \langle \omega_{out}, \mathbf{h} \rangle)} \\
&\cdot (R_s + (1-R_s)(1-\langle \omega_{in}, \mathbf{h} \rangle)^5) \\
&+ R_d (1-R_s) \frac{28}{23\pi} \\
&\cdot \left(1 - \left(1 - \frac{\langle \omega_{in}, \mathbf{n} \rangle}{2}\right)^5\right) \left(1 - \left(1 - \frac{\langle \omega_{out}, \mathbf{n} \rangle}{2}\right)^5\right)
\end{aligned} \tag{1}$$

for the incoming and outgoing directions $\omega_{\mathbf{in}}$ and $\omega_{\mathbf{out}}$. The vector $\mathbf{n}$ is the surface normal, $\mathbf{h} = (\omega_{\mathbf{in}} + \omega_{\mathbf{out}})/||\omega_{\mathbf{in}} + \omega_{\mathbf{out}}||$ and ϕ is the azimuth of $\mathbf{h}$.

This model has four reflectance parameters: the wavelength dependent diffuse and specular reflectance shares R_d and R_s and the surface roughness along the x-axis e_x and the surface roughness along the y-axis e_y. In the following, we will refer to R_d and R_s as the *diffuse colour* and the *specular colour*. We will assume that those colours are RGB colours and the term *lightness* will refer to the HSL description of the RGB-space. We assume that the parameters are stored in rectangular maps.

Being based on the Phong model, the Ashikhmin-Shirley model is neither normalized nor is the distribution function for the lobe physically founded. An up-to-date comparison between anisotropic analytical BRDF-models and a suggestion for a model without the mentioned flaws has been published by Murat et al. [14].

3.2 Problem statement

Given an SVBRDF, where at least some of the parameter channels bear a *roughly* periodic pattern in the following sense: there exists a periodic pattern which may be warped into those channel maps alongside of a small continuous flow field. Here we mean by periodic pattern an image which may be generated from a model tile and a concatenation of translations and rotations according to an appropriate wallpaper group. A specification of the term *small* is difficult and depends not only on the settings of the algorithm but also on the texturizing of the SVBRDF, itself.

Further we assume, that a user has marked a *foreground* component $\mathcal{F}$ of the SVBRDF and a *background* component $\mathcal{B}$ by the use of a stroke input $\mathcal{S}_\mathcal{F}$ for the foreground and a stroke input $\mathcal{S}_\mathcal{B}$ for the background stroke. Than we want to propagate this stroke input in a way that the periodic pattern is respected and a texel with the index i and the average reflectance distribution ρ_i obtains a value α_i which decomposes ρ_i into a convex combination of a foreground BRDF ϕ and a background BRDF ψ

$$\rho_i = \alpha_i \phi_i + (1 - \alpha_i)\psi_i$$

For the classical matting problem, the parameter α is described as opacity or transparency. For our application, this interpretation is not good, as transparency leads to complicated reflectance properties. α should be merely seen as area share of the foreground reflectance distribution. We will define the foreground $\mathcal{F} = \{t_i | \alpha_i = 1\}$, the background $\mathcal{B} = \{t_i | \alpha_i = 0\}$ and the boundary $\partial = \{t_i | 0 < \alpha_i < 1\}$.

3.3 Walk through

In figure 1 you can see an overview of our new algorithm. As input we take a SVBRDF together with a stroke input. Then we apply in parallel a segmentation via a support vector machine (paragraph 4.1.2) on the descriptors described in paragraph 4.1.1 and estimate a lattice on the diffuse colour (paragraph 4.2). Based on the detected lattice we extract a model tile (paragraph 4.3.1), calculate an optical flow between this model tile (paragraph 4.3.2) and all other tiles and warp the tiled SVM-classification results into the model tile. This set of warped masks is used to compose an average tile-mask which is then warped into the original tile positions (paragraph 4.3.3).

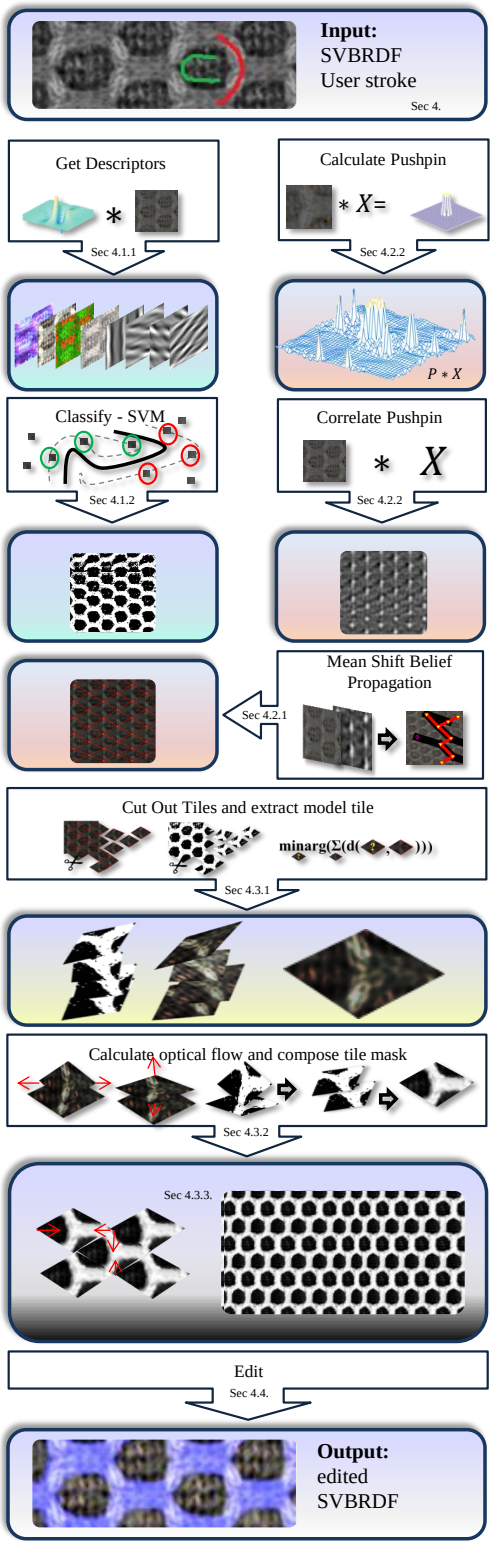


Figure 1: Overview: the arrows contain the processing steps and the boxes show the resulting data. In the top of the arrows we give the numbers of the paragraphs where the processing step is described in detail.

4 THE ALGORITHM IN DETAIL

In this section we want to describe the algorithm in detail.

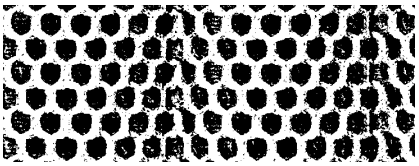


Figure 2: The mask resulting from the SVM classification step.

4.1 Classification

Based on the stroke input, we classify in this step all texels of the material probe, without reference to the NRS, into foreground and background texels. We tried several different descriptors and several different classifiers:

4.1.1 Descriptors

Additionally to the 8 reflectance parameters and the 2 parameters of the surface normal provided by the Ashikhmin-Shirley model (equation 1), we add the filter responses of Gabor filters. We use 8 different orientations and a wavelength of 3 texels. Gabor filters are applied to the volume-channel of the diffuse color. This strengthens the influence of line features on the classification result. We compare every texel on a patch with size 5x5 texel. So the dimension of our descriptor is altogether (8 + 2 + 8) x 5 x 5 = 450.

4.1.2 Classifier

We tried different state of the art classifiers: Support Vector Machines [6], Deep Belief Networks [10] and Convolutional Neural Networks [15]. The latter have been implemented in Theano for Python, for the SVM we used the implementation by Chang [5].

Though we made good experiences with neural networks in the past, they failed in the current scenario. According to a rule of thumb given in [18], the number of samples should be equal or more than the number of weights of the neural network. As stated in paragraph 4.1.1, the descriptor of a texel has the dimension of 450 which makes, dependent on the concrete topology, about 50,000 weights in a three layer neural network, whereas a stroke input provides between 100 and 500 samples. So the networks have simply not enough data for training. SVMs, in contrast, can be trained with a small amount of data and are easy to apply and quickly trained.

The trick of the SVM is that it estimates a decision boundary in an infinite dimensional space which makes it possible to have non-linear boundaries between clusters. By maximizing the margin between the decision boundary and the training-samples, the SVM reaches even in the linear setting better generalization than other linear classifiers. For the optimization, it is not necessary to map the data into the infinite dimensional feature space, but it is enough to calculate the inner product (so called *Kernel Trick*). We use radial basis functions as inner product kernels and parameter estimation is done by grid-search and 5-fold cross-validation.

In figure 2 you can see that the result of the svm classification step is already a good segmentation. Still there are some noticeable misclassifications.

4.2 Lattice detection

Our algorithm gains its strength from the combination of lattice detection and pattern-recognition. In our tests, the most successful approach to detecting lattices was the *mean shift belief propagation* (*MSBP*), published by Park et al. [20].

4.2.1 Mean Shift Belief Propagation

MSBP makes the assumption that a repeating structure in an image is a slightly deformed periodic pattern. As such it is possible to find an ideal pattern element and two linearly independent lattice base vectors to reconstruct this periodic pattern by operating via the corresponding wallpaper group [9, 17]. By clustering points of interest, MSBP estimates the base vectors for the periodic pattern and a seed point, and the algorithm extracts a characteristic tile around this seed point. The lattice base vectors define symmetry-mappings, so the seed point and all symmetry-images of this seed point may be mapped to further symmetry-images by translation along the base vectors. Those images are the vertices of the constructed lattice. As the lattice is deformed by assumption, the exact symmetry mapping has to be found by searching for a good fit for the characteristic tile in the area of the estimated new vertex position. This search is done for all new lattice-vertex candidates simultaneously, meaning that the search for two neighbouring vertices is constrained by an energy term which punishes deviation from the according base translation. Mean shift belief propagation has proven to be an extremely powerful algorithm. Still we had to struggle with two problems:

1. The results are not deterministic.
2. Regions of big distortions like the fold in the grey mesh material often stop the expansion of the lattice.

Both difficulties are illustrated in figure 3. The result of MSBP, reflected by the red lattice in the left image was successful: the algorithm found the smallest possible tile and the lattice covers the whole material patch. On the right image, we have an example for an abortive run of MSBP: you can see that the algorithm was not able to cross the fold in the material and the base vectors are the sum and the difference of the base vectors found in the right image.

We clear this problem by the use of a cross-correlation based approach we call *pushpins*.

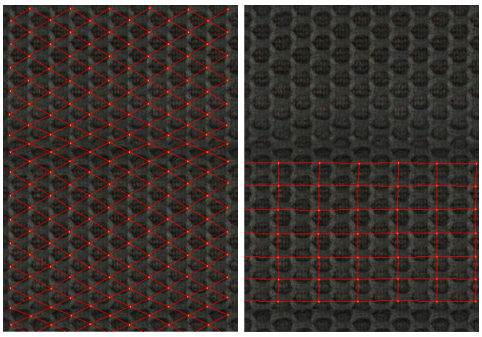


Figure 3: Two different runs of mean shift belief propagation on the grey mesh material.

4.2.2 *Pushpins*

To make the results of MSBP more stable and more predictable, we guide the lattice detection step by a weaker but therefore global repetition detector. The main idea is to mask the frequency spectrum of a given material $\mathcal{T}$ in such a way that a specific quadratic region $\mathcal{P} \subset \mathcal{T}$ in the spatial domain and therefore all similar regions in the spatial domain show a peak. This may be done by cross correlating $\mathcal{T}$ with $\mathcal{P}$, but simple cross correlation does not bring the desired results. Instead, we construct a patch which generates a peak when convolved with $\mathcal{P}$.

Masking the frequency domain in order to isolate particular features is a common technique in signal processing but we did not find our approach in the computer graphics literature so we will briefly introduce it.

Lets first assume that we are looking for a texture $\mathcal{X}$ with the same size as $\mathcal{P}$ so that:

$$\mathcal{P} * \mathcal{X} = \delta$$

where δ models a spike in form of the dirac distribution. By convolution and application of the convolution theorem, we get:

$$\mathbf{F}\mathcal{P}\mathbf{F}\mathcal{X} = \mathcal{C}$$

for a constant texture $\mathcal{C}$. $\mathbf{F}$ is the fourier-transform and $\mathbf{F}^{-1}$ its inverse. Thus, a candidate for $\mathcal{X}$ is:

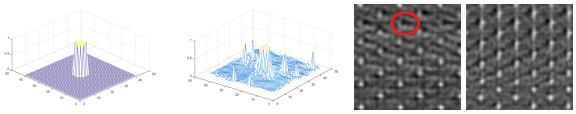
$$\mathbf{F}^{-1}\left(\frac{\mathcal{C}}{\mathbf{F}\mathcal{P}}\right)$$

Here we presumed correct scaling and frequency sampling and point wise multiplication.

For numerical reasons it is advisable to suppress high frequencies. Thus we substitute C by a gaussian filter $\mathcal{G}$ and get:

$$\mathcal{P} * \mathbf{F}^{-1}\left(\frac{\mathcal{G}_{-1/\sigma\pi^2}}{\mathbf{F}\mathcal{P}}\right) = \mathcal{G}_{-\sigma}$$

for the variance σ. Note that equation 4.2.2 becomes wrong, when $\mathbf{F}P$ is not continued by zeros, but by the surrounding pixels in the material. This can be circumvented by calculating $\mathcal{X}$ not by convolution but by deconvolution as the solution of



(a) Mesh($\mathcal{P} * \mathcal{X}$) (b) Mesh($\mathcal{P} * \mathcal{X}_\Delta$) (c) $\mathcal{T} * \mathcal{X}$ (d) $\mathcal{T} * \mathcal{X}_\Delta$

Figure 4: The effect of regularization to push pins. From left: the response of the original tile ($\mathcal{P}$) to an unregularized pushpin ($\mathcal{X}$), the reponse to a regularized push pin ($\mathcal{X}_\Delta$), the response of a distorted material-patch ($\mathcal{T}$) to the unregularized pushpin and the reponse of the same material-patch to the regularized filter. In image (c) you can see that the unregularized pushpin fails to produce some spikes (see red circle).

$$\sum_{i=0,j=0}^{n-1} \mathcal{X}(i,j)\mathcal{T}(i_0-i, j_0-j) = \mathcal{G}_\sigma(i_0, j_0) \qquad (2)$$

$\forall i_0, j_0 \in \text{supp}(\mathcal{P})$. n is the edge length of $\mathcal{P}$. As n is also the edge length of $\mathcal{X}$, we have the same number of variables and equations.

We will call the solution $\mathcal{X}$ of equation 2 a *pushpin* and $\mathcal{P}(\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor)$ the *puncture* of the pushpin. By *nailhead* we mean the support of $\mathcal{P}$.

A pushpin, constructed in this way, does respond a bit stiff: tiles have to be very similar to the original tile to generate a detectable spike. This may be relaxed massively by using a regularization: instead of solving the equation system 2, we constrain this equation system by a spatial smoothing term namely by the minimization of the discrete laplace operator (Δ). This leads to a minimization problem:

$$\mathcal{X} = \text{argmin}_{\mathcal{W}}\Bigg(\sum_{i_0=0,j_0=0}^{n-1} || \sum_{i=0,j=0}^{n-1} \mathcal{W}(i,j)\mathcal{T}(i_0-i, j_0-j) - \mathcal{G}_\sigma(i_0, j_0)|| + || \sum_{i=0,j=0}^{n-1} \Delta_{(i,j)(k,l)}\mathcal{W}|| \Bigg) \qquad (3)$$

The effectiveness of this regularization step is illustrated in figure 4. Pushpins can be made tolerant against noise or small distortions by adding energyterms to equationsystem 3. And the other way round it is possible to concentrate on certain regions of the pushpin by adding weights to the corresponding equations.

In figure 5 we visualize the influence of push pins to the lattice detection process. We have made several test runs some of which had one or two nodes missing, but we obtained always the same lattices covering the whole material patch.

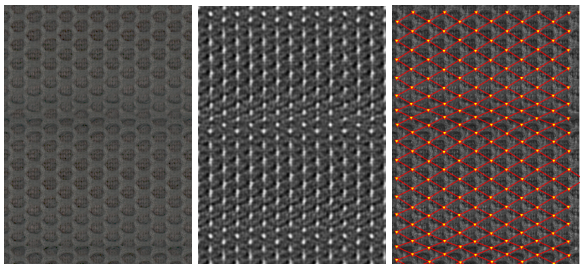


Figure 5: The left image shows the grey mesh material, the image in the middle depicts the filter response of a push pin applied to the volume channel of the diffuse color of the grey mesh material and the third image shows the result of MSBP on a combined map of the filter response and the diffuse channel. Now the lattice detection is extremely stable.

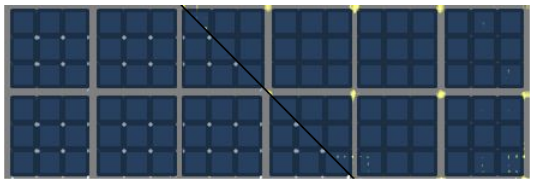


Figure 6: On the left side you can see in light blue the filter response of a pushpin with nailhead radius approximately equal to the size of half a small square on the right side we used a pushpin with a nailhead radius approximately equal to half a big square.

To conctruct a pushpin, it is necessary to determine a centerpoint and a radius. We have chosen the mean of the positions of the stroke inputs as center point and the size of the nailhead was chosen so as to cover the whole stroke.

Nested symmetry groups

Pushpins generate automatically a region of dominance. This shall be demonstrated on a simple example. In figure 6 you may see a simple texture consisting of small squares arranged in groups to bigger squares. On the left side, you can see the response of a pushpin with a nailhead diameter in the size of a small square, on the right side we used a pushpin with a nailhead diameter in the size of a big square. The clipped filter response of the smaller pushpins are held blue the filter response of the bigger pushpins is shown in yellow. You can see that the pushpin on the left side detected the crossings between the small squares whereas the pushpins on the right side detected *exclusively* the crossings between the big squares. This means that pushpins can distinguish between nested symmetry groups. That is an improvement against plain mean shift belief propagation because MSBP simply uses the symmetry group it gets first.

Though pushpins are not limited to a certain number of channels, particularly not to 1, we confine their use to the lightness channel of R_s or R_d. Note that the use of more channels does also lead to more noise in the filter response.

4.3 Generation of a mask

In the next step we combine the results of the classification and of the lattice detection to obtain a model mask tile and a warping field to plaster the whole material patch with this model mask patch. After cutting the mask and the material into a set of tiles which we interpret as distorted version of the same model tile, we extract a model tile, we calculate an optical flow between the model tile and all other tiles and we compose a mask for the whole material probe.

4.3.1 Finding a model tile

To generate a reliable segmentation of a single tile we first choose one tile which is every bodies friend. We assume that changes in the size of tiles are due to perspective distortion. Thus the best fit for an average tile should be a tile with maximum edge-length. So in the first step we resize all tiles to the maximum edge-length. The comparison is made on base of the $L2$-norm applied to the difference of the diffuse channel of two tiles. As the number of tiles is small, we simply apply a brute force approach and compare all tiles pairwise. This procedure is quadratic in the number of tiles, so for big numbers of tiles, the time requirement may be optimized by using a dynamic programming approach. Note that generating a mean tile instead of searching the tile with the most friends is not advised as we want to calculate the optical flow between this model tile and all other tiles. This is more difficult with a mean tile because the algorithm has to find features.

4.3.2 Optical flow

For the estimation of the optical flow between the principal tile and the test tile, we use the algorithm suggested by Sun et al. [24]. For warping we use thin-plate splines [4].

4.3.3 Reconstruction

An arithmetical mean mask is calculated from the warped masks. This mean mask is warped back into the position of the original tiles.

4.4 Applying the edits

Our algorithm assigns an alpha value to every texel. This value will scarcely be exactly one or zero. So we will do a segmentation by thresholding. Aside from distortions the alpha-values may be seen as voting for the background or the foreground, so 0.5 is a good threshold. The segmentation mask is of course not suitable for editing as it will obviously lead to strong artefacts. So we will substitute all texels, which have at least one

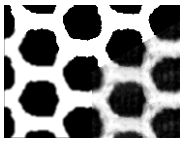


Figure 7: On the top the binary mask, on the right the original superposed mask and on the left the mixed mask.

corner-neighbour from the opposite component, by its alpha-value, so that the intuitive use of the word *boundary* and the definition given in section 3.2 coincide.

On the foreground component, editing can of course be done as e.g. described in the literature cited in section 2, but on the boundary it has to be taken under consideration, that for many edits it is necessary to know the exact decomposition $\rho_i = \alpha_i\phi_i + (1-\alpha_i)\psi_i$, which to find is an ill-posed problem.

5 EVALUATION

In the evaluation section we will show that our algorithm is capable of dealing with materials, which do not show the strong colour-contrasts, which are mostly necessary for matting and foreground-segmentation purposes.

5.1 Test set-up

To describe our test set-up we will start with a short description of the input data. Next, we will give a detailed overview over the competing algorithms to convey an idea where those algorithms run into problems. Of course the test set-up is strongly biased into the direction of our algorithm as both algorithms, AppProp and RepSnapping are by far more general. But we did not find a more fitting approach in literature.

5.1.1 Input data

The materials we use in this paper have been acquired with an enhanced version of the *linear light source reflectometer* (*LLSR*), introduced by Gardner et al. [8]. This new system has been developed by Meseth et al. [19] and is capable of measuring anisotropic reflectance distributions.

Additionally to the reflectance properties (equation 1), LLSR has to estimate a surface normal **n**. All values have been stored as 16 bit integer values. One texel represents a surface of roughly $1/4$ mm^2.

We use two different materials for the comparison: the grey mesh material which we have used to demonstrate the single steps of the algorithm and a structured steel material (see figure 8).

The grey mesh material is nearly uni coloured. It is particularly difficult to derive a near regular structure because it contains a strong bulge and the material normals do not convey much information.

While it is really simple, to derive the regular structure from the structured steel material, the only visible difference between foreground and background is a slightly less isotropic distribution of the noise. The metal material does not have a diffuse colour channel so we have to use R_s, instead.

5.1.2 Comparison with other algorihtms

Our algorithm combines techniques from the field of material manipulation with techniques from the field of repetition finding in images. Thus for comparison we have chosen one outstanding algorithm from each of those field. For the task of segmenting repetitions in images we decided for the RepSnapping algorithm [12], published in 2011 by Huang et al. And to cover the field of SVBRDF-editing we will compare against AppProp [2], published by An and Pellacini in 2008. Moreover, we compare those results with the segmentation of the SVM from step 4.1.

AppProp

The authors use a low rank approximation of the full appearance adjacency matrix and minimize the following functional:

$$\sum_{i,k} w_k z_{ik}(e_i - g_k)^2 + \lambda \sum_{i,j} z_{ij}(e_i - e_j)^2$$

with

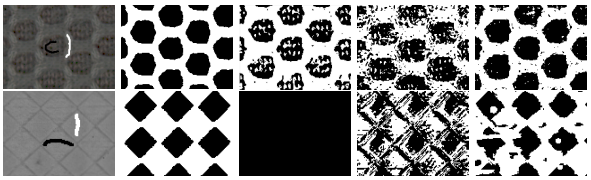
$$z_{ij} := \exp(-||f_i - f_j||^2/\sigma_a)\exp(-||x_i - x_j||^2/\sigma_s).$$

Where i and j go over all texel in the texture, k goes over all texels in the stroke input, w are weights, e is the edit and therefore the solution of the optimization problem, g is the stroke-input and therefore the right hand of the optimization problem, x is the position of the texel, λ the weight of the smoothing term and f is a texel-dependent appearance term. The resulting equation system is roughly solved by a low-rank approximation. The appearance comparison of AppProp is not limited to three dimensions or a single texel, so we can apply it to our descriptor (section 4.1.1).

The spatial parameter σ_s is not interesting in our setting, but to find a reasonable value for σ_a is difficult for our high dimensional descriptor and has to be done in a preprocessing step for every material separately. This is not surprising because the term

$$\exp(-||x_i - x_j||^2/\sigma_s) = \prod_k \frac{1}{e^{(x_i^k - x_j^k)^2/\sigma_s}}$$

consists of 450 factors in our case and has therefore the inclination to explode or to collapse beyond numerical accuracy. λ controls the consistency of the edit and had not much influence. We set λ and w_k to one. Thresholding has been done manually, in order, to get the best possible segmentation.



(a) Input (b) PushPin (c) RepSnap (d) AppProp (e) SVM

Figure 8: On the left a patch from the original image with the stroke input the second image shows the mask generated by RepSnapping. The third mask is the result of AppProp and the last mask is our result. The first row shows the grey mesh material the second row shows a metal.

RepSnapping

RepSnapping has been published by Huang et al. in 2011 [12], and is based on the idea of co-segmentation [11]. It is specialized to cutting out repeated elements in natural images. The algorithm solves the energy functional:

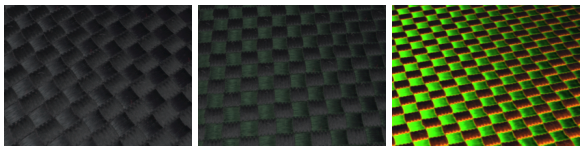
$$E(e) := \sum_i D_i(e_i) + \sum_{i<j} V_{i,j}(e_i,e_j) + \sum_{i,j\in H} U_{i,j\in Nbh}(e_i,e_j)$$

by the use of graph cuts [13]. Here D_i describes the probability that $e_i \in \mathcal{F}$ and is given as a normalized set distance to a clustering (H) of the foreground. $D_i(e_i=1) = \frac{\min_{k\in\mathcal{H}(F)}||f_i-f_k||}{\min_{k\in\mathcal{H}(F)}||f_i-f_k||+\min_{k\in\mathcal{H}(B)}||f_i-f_k||}$ with the appearance function f and $D_i(e_i=0) = 1-D_i(e_i=1)$. $V_{i,j} = \lambda|e_i-e_j|\exp(-\beta||f_i-f_j||^2)$ is a smoothing term and goes over all adjacent pixel pairs. $U_{j,j} = \mu|e_i-e_j|\exp(-\beta\gamma(i,j)^2)$ assures that pixels with similar appearance are treated similar. The main idea is that the neighbourhood graph is extended by the neighbourhood-system Nbh which contains edges between the pixels i and j iff $\gamma(i,j) < \varepsilon$, where γ is a correlation based similarity measure, described in [11].

We applied RepSnapping with the parameters given in [12], namely: $\mu = 10$, $\beta = 0.1$, $\lambda = 2$ and $\varepsilon = 4$. RepSnapping might easily be extended to the high-dimensional descriptor used in our algorithm but it would suffer from the same stability issues as AppProp.

5.2 The results

In figure 8 we present the comparison of the image segmentation step. You can see that our algorithm delivers artefact-free masks for both materials (8.b). The other three algorithms are more successful on the grey mesh material than on the metal material. An interesting result is, that the raw SVM delivers the second best results. We see the main reason in the descriptors: AppProp is numerically overcharged with the big number of descriptors, which results in this big amount of noise, and RepSnapping uses a correlation based approach to describe texel neighbourhoods. Autocorrelating the R_s-channel of the metal material reveals that the surface does not have enough structure to provide significant correlation results. Together with the fact that R_s is unicoloured, this explains, why RepSnapping fails completely.



(a) Original (b) Changed R_d (c) Changed R_s

Figure 9: On the left the original material in the middle a rather subtle edit of R_d, on the right a more noticeable manipulation of R_s.

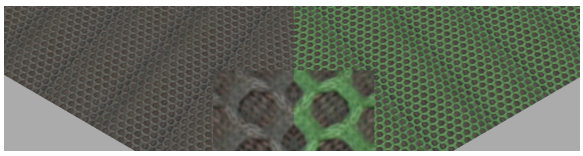


Figure 10: In the close-up of the edit of the grey mesh material one may see that the editing boundary coincides exactly with the perceived boundary of the foreground material.

Figure 11: A shiny material. The left image shows the unedited material. In the right image the background has been changed: R_s has been changed from yellow to green.

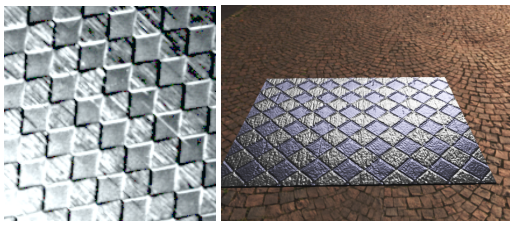


Figure 12: On the left the metal material, on the right a rendering of the edited material.

5.3 Editing examples

In this section we want to present the resulting edits on four different materials (figure 9 - 12).

5.4 Time Requirement

The bottleneck of the algorithm was to calculate the optical flow on all tiles (paragraph 4.3.2). For the grey-mesh material we had about 180 tiles. Calculating the optical flow on one tile (~80x40 texel) took about 2.2 s, which sums up to about 7 min. Depending on the

number of training samples, the SVM classification step took between 10 s and 3 min (paragraph 4.1.2). MSBP (paragraph 4.2.1) ran for about 45 s. Warping a tile with tps took about 0.03 s. Finding a principal tile took less than a second. So the overall processing time lay between 8 and 12 minutes.

For comparison: RepSnapping took 3 s, SVM took 10 s and AppProp took 40 s.

5.5 System

Computations have been done on an i5-2500 with a clock rate of 3.3 G/s and 8 GB RAM.

6 CONCLUSIONS AND FUTURE WORK

In this paper we demonstrated an algorithm to solve the task of extracting a repeating foreground pattern from a high dimensional reflectance representation map in a way, which is robust and reliable enough, to make additional optical debugging steps unnecessary. While the task is relatively simple on suitable materials, we could show, that the competing state of the art algorithms failed for difficult material probes. Our algorithm permits high quality segmentation and editing on complex materials.

Yet our algorithm is too slow for productive and industrial use. But many steps of the algorithm may be parallelized, particularly with respect to computations on the tiling, so that efficiency and responsiveness may be improved drastically.

7 REFERENCES

[1] Xiaobo An, Xin Tong, Jonathan D. Denning, and Fabio Pellacini. Appwarp: retargeting measured materials by appearance-space warping. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, pages 147:1–147:10, New York, NY, USA, 2011. ACM.

[2] Pellacini F. An X. Appprop: all-pairs appearance-space edit propagation. *ACM Transactions on Graphics*, 27(3):1–9, 2008.

[3] Michael Ashikhmin and Peter Shirley. An anisotropic phong brdf model. *Journal of graphics tools*, 5(2):25–32, 2000.

[4] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585, 1989.

[5] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[7] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 151–157, 1997.

[8] Andrew Gardner, Chris Tchou, Tim Hawkins, and Paul Debevec. Linear light source reflectometry. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 749–758. ACM, 2003. No. 3.

[9] Branko Grünbaum and Geoffrey Colin Shephard. *Tilings and patterns*. Freeman, 1987.

[10] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[11] Dorit S Hochbaum and Vikas Singh. An efficient algorithm for co-segmentation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 269–276. IEEE, 2009.

[12] Hua Huang, Lei Zhang, and Hong-Chao Zhang. Repsnapping: efficient image cutout for repeated scene elements. In *Computer Graphics Forum*, volume 30, pages 2059–2066. Wiley Online Library, 2011. No. 7.

[13] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *IEEE transactions on pattern analysis and machine intelligence*, 26(2):147–159, 2004.

[14] Murat Kurt, László Szirmay-Kalos, and Jaroslav Křivánek. An anisotropic brdf model for fitting and monte carlo rendering. *ACM SIGGRAPH Computer Graphics*, 44(1):3, 2010.

[15] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[16] Hendrik PA Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatially varying materials. In *Rendering Techniques 2001*, pages 103–114. Springer, 2001.

[17] Yanxi Liu, Robert Collins, and Yanghai Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):354 – 371, March 2004.

[18] Timothy Masters. *Practical neural network recipes in C++*. Morgan Kaufmann, 1993.

[19] Jan Meseth, Shawn Hempel, Andrea Weidlich,

Lynn Fyffe, Graham Fyffe, Craig Miller, Paul Carroll, and Paul Debevec. Improved linear light source material reflectance scanning. In *ACM SIGGRAPH 2012 Posters*, page 42. ACM, 2012.

[20] Minwoo Park, Kyle Brocklehurst, Robert T Collins, and Yanxi Liu. Deformed lattice detection in real-world images using mean-shift belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1804–1816, 2009.

[21] Fabio Pellacini and Jason Lawrence. Appwand: editing measured materials using appearance-driven optimization. In *ACM Transactions on Graphics (TOG)*, volume 26, page 54. ACM, 2007. No. 3.

[22] Szymon M Rusinkiewicz. A new change of variables for efficient brdf representation. In *Rendering techniques 98*, pages 11–22. Springer, 1998.

[23] Thorsten-Walther Schmidt, Fabio Pellacini, Derek Nowrouzezahrai, Wojciech Jarosz, and Carsten Dachsbacher. State of the art in artistic editing of appearance, lighting and material. In *Computer Graphics Forum*. Wiley Online Library, 2015.

[24] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE, 2010.

[25] Jiaping Wang, Xin Tong, Stephen Lin, Minghao Pan, Chao Wang, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Appearance manifolds for modeling time-variant appearance of materials. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 754–761, New York, NY, USA, 2006. ACM.

[26] Hongzhi Wu, Julie Dorsey, and Holly Rushmeier. A sparse parametric mixture model for btf compression, editing and rendering. In *Computer Graphics Forum*, volume 30, pages 465–473. Wiley Online Library, 2011. No. 2.

[27] Kun Xu, Jiaping Wang, Xin Tong, Shi-Min Hu, and Baining Guo. Edit propagation on bidirectional texture functions. In *Computer Graphics Forum*, volume 28, pages 1871–1877. Wiley Online Library, 2009. No. 7.

Improvement of Some Interpolation Methods for Terrain Reconstruction from Scattered Data

Róbert Bohdal
Comenius University in
Bratislava
Faculty of Mathematics,
Physics and Informatics
Mlynská dolina
Slovakia, 842 48
Bratislava
robert.bohdal@fmph.uniba.sk

ABSTRACT

Using GPS modules it is easy to obtain 3D data for areas that have not been digitized yet. Such terrain data are usually not arranged in a grid, and therefore we have to use scattered data interpolation methods. The aim of the paper is to create a digital terrain model from 3D data using modifications of known methods. Sibson interpolation method is often used when we need to interpolate large data sets. This method has low memory requirements, it is sufficiently fast, but creates undesired surface artefacts. Our aim is to have the resulting interpolation surface as similar as possible to the original surface. We have decided to replace the heights at specified points by local functions. We use biquadratic and bicubic polynomials, Hardy's multiquadrics and thin plate spline as local functions. In the paper, we have evaluated the time requirements and the accuracy with which the interpolated area matches the actual 3D data on 2 terrain samples (the Little Carpathians and a small part of the Little Carpathians).

Keywords

Digital terrain model, Radial basis functions, Inverse distant weights, Thin plate spline, Natural neighbours, Sibson interpolation

1 INTRODUCTION

Digital terrain model (DTM) can be used in many areas and applications. It is commonly used in urban planning, hydrology, geosciences, for investigating soil erosion, modelling of movement of avalanches, army applications, graphics information systems, creation of topographic maps and similarly. DTM can be understood as a 3D representation of a part of the Earth surface in digital format. It is commonly created from a large amount of 3D points in the form of surfaces, which are created using interpolation and approximation functions. These 3D points are obtained, for example, using *stereophotogrammetry*, *laser scanning* and *radargrametry*.

Creating of digital terrain from scattered data is still an interesting area of research. This is suggested by several comparative studies that we can find currently in literature.

In our research, we have decided to use local interpolants, which replace the specified height points that are used in creating the terrain model. We want to improve the accuracy of methods using the weighted average and remove shape artefacts that these methods produce in the final models. We want to take advantage of their algorithmic simplicity and their speed of calculation. In conclusion, we show that the use of local functions is meaningful, even at the cost of slightly increased computational time. We consider *radial basis functions* (RBF) to be the best local functions, which are currently used in many areas of research.

2 RELATED WORK

There are several methods that can be used to create a DTM from scattered data. In most cases interpolation methods are used. It is then possible to calculate the height value at any point of the considered area.

The most commonly used methods are based on triangulation entry points, the weighted average methods, methods using radial basis functions and others. Less-known techniques use, for example, dividing the input

data into areas using local interpolants or neural networks. There are also approaches that use a combination of several methods.

For the methods using *triangular irregular networks* (TIN), one first constructs a suitable triangulation from the input data, for example, *Delaunay triangulation*. Methods which use a piecewise continuous interpolant can be used afterwards, for example, Bezier patches in case of *Clough-Tocher* [Hug04] or *Powell-Sabin* [PS77] method. Further, one can use a triangular network method called *triangle based blending* (TBB) [GS13, DG02, Ami02], which uses the weighted average of three functions from a preselected class of local functions interpolating the corresponding vertex of the triangle and its nearest neighbours. Triangular network is also used by a method described earlier [HZDS01], in which one first constructs a uniform triangular network, and local approximation polynomials (Bezier triangular patches) are calculated using the method of least squares. The resulting spline function is then created using a combination of these polynomials with the use of Bernstein-Bezier smoothness conditions. Among the methods using triangular networks, we can also use a method of the *natural neighbours* (Sibson's interpolant) [DG02], which uses Voronoi diagram, which is the dual graph to the Delaunay triangulation. The function value (height at the searched point) is given by the weighted average of local height values of the relevant vertices.

Among the methods using only the weighted average, *Shepard's method* also known as the *Inverse Distance Weighting* (IDW) is the most famous [AAAC05, GG13, GS13, Hu13, Hug04]. Since Shepard's method does not give good results, its modification is used more often. The modification uses local interpolant calculated by the method of least squares [GS13].

To create a model of the terrain, RBF methods are probably the ones used the most often [AAAC05, CL12, CL13, GG13, GS13, Hu13, Hug04, MS16, PGTG04, SS09]. *Thin plate splines* (TPS) and *Hardy's multiquarics* (HMQ) are the most famous from this class of functions. However, the disadvantage of these methods is that for their calculation it is necessary to solve a system of equations. If the number of input points is big, we use methods that produce a final interpolation surface using a local interpolant. In [PGTG04], there is a procedure which in the first step recursively splits the input area into an overlapping sub-regions using k-d trees. The second step is calculating the functional value as a weighted average of two functions recursively enumerated in the respective sub-regions. For a large number of data points, we can use a RBF approximation given in [MS16]. The method uses a determination of significantly fewer so-called referent points, which together with the given points create an overdetermined system of equations. This system of equations is then solved by the method of least squares to obtain the unknown coefficients of the resulting interpolation function.

For the needs of creating a model of the terrain, we often use a geostatistics method called *Kriging* [GS13, Hug04]. It is based on predicting the value of a function at a given point using the weighted average of points in the neighbourhood of the calculated point.

From less-known methods for the terrain construction, it is necessary to mention also neural networks of type MLP, *Support Vector Machine Regression* and *Neural Networks* in [ON15] or genetic algoritms in [BSS14].

3 METHODS

The creation of a digital terrain model from scattered data points can be easily solved using suitable interpolation methods. In our case, we have focused on the modification of known methods, using local interpolants, which are used instead of the height values. As local interpolants, we choose thin plate splines and Hardy's multiquadrics [Isk03] and also well-known cubic and quadratic polynomials of two variables. As a further option, we choose the replacement of height values by planes, while their normal vector is calculated as a gradient of the local interpolation function.

Let us have a set $\mathscr{P}$ of N mutually different input points $\mathscr{P} = \{\mathbf{p}_1[p_1^x, p_1^y], \ldots, \mathbf{p}_N[p_N^x, p_N^y] \mid \mathbf{p}_i \in \mathbb{R}^2\}$ with height values $h_i \in \mathbb{R}$, for $i = 1, \ldots, N$. We search for such function $f : \mathbb{R}^2 \to \mathbb{R}$, for which the interpolation condition is true:

$$f(\mathbf{p}_i) = h_i,\ i = 1, \ldots, N. \tag{1}$$

3.1 Inverse Distance Weighted (IDW)

The simplest form of IDW interpolation is called Shepard's method. Shepard defined his interpolating function $f(\mathbf{x})$ with argument $\mathbf{x} \in \mathbb{R}^2$ to be the weighted average of the heights h_i [HL93]:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \omega_i(\mathbf{x}) h_i. \tag{2}$$

Weight functions $\omega_i(\mathbf{x})$ from formula (2) can be expressed as:

$$\omega_i(\mathbf{x}) = \frac{\sigma_i(\mathbf{x})}{\sum_{j=1}^{N} \sigma_j(\mathbf{x})},$$

where $\sigma_i(\mathbf{x}) = ||\mathbf{x} - \mathbf{p}_i||^{-\mu_i}$, for $\mu_i > 0$. The parameter μ_i allows to control the shape of the final surface in the neighbourhood of the interpolated points. The standard value for this parameter is $\mu_i = 2$.

The global character of this method can be made local by multiplying the weighted function $\omega_i(\mathbf{x})$ by the mollifying function [HL93]:

$$\lambda_i(\mathbf{x}) = \left(1 - \frac{\sigma_i(\mathbf{x})}{R_i}\right)_+^{\mu_i}, \text{ where } R_i > 0.$$

For example, we can set the radius R_i to $\frac{D}{2}\sqrt{\frac{N_w}{N}}$, where D is the maximum distance between arbitrary two points of the set $\mathscr{P}$ and $N_w = 19$ [TH10].

3.2 Radial Basis Functions (RBF)

Radial basis functions have gained immense popularity in the multi-dimensional interpolation of scattered data. They are simple to implement, and they generate an interpolation surface with a sufficient smoothness.

We can write the interpolation function $f(\mathbf{x})$ in the following form [HL93]:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i R(\|\mathbf{x}-\mathbf{p}_i\|) + \sum_{k=1}^{l} c_k \Phi_k(\mathbf{x}), \qquad (3)$$

where $\Phi_k(\mathbf{x}) \in \pi_m^2$, $l = \dim(\pi_m^2) = \binom{m-1+2}{2}$. Symbol π_m^d denotes a linear space containing all polynomials over the field $\mathbb{R}$ with d variables and a degree at most $m-1$. Functions $R(\|\mathbf{x}-\mathbf{x}_i\|)$ are radial basis functions with an argument expressing the euclidean distance between points $\mathbf{x}$ and $\mathbf{x}_i$.

Unknown coefficients $\boldsymbol{\lambda} = (\lambda_1,\ldots,\lambda_N)^\mathsf{T}$ and $\mathbf{c} = (c_1,\ldots,c_l)^\mathsf{T}$ in relation (3) are given by solving a system of equations:

$$\begin{pmatrix} \mathbf{A} & \mathbf{P} \\ \mathbf{P}^\mathsf{T} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \mathbf{c} \end{pmatrix} = \begin{pmatrix} \mathbf{h} \\ \mathbf{0} \end{pmatrix}, \qquad (4)$$

where $\mathbf{A}_{i,j} = R(\|\mathbf{p}_i - \mathbf{p}_j\|)$, $\mathbf{P}_{i,k} = \Phi_k(\mathbf{p}_i)$ and $\mathbf{h} = (h_1,\ldots,h_N)$, for $i,j = 1,\ldots,N$ and $k = 1,\ldots,l$. Due to the fact that both RBFs described below are conditional positive definite [Fas07], the system of equations (4) has a solution if the points $\mathbf{x}_i$ are non-collinear.

3.2.1 Thin Plate Splines (TPS)

Thin plate splines belong to the class of polyharmonic splines:

$$R_{d,m}(\|\mathbf{x}\|) = R_{d,m}(r) = \begin{cases} r^{2m-d} & \text{if } d \text{ is odd} \\ r^{2m-d}\log(r) & \text{if } d \text{ is even} \end{cases}$$

The name is derived from a relation, in which we search for the minimum of an integral describing the distribution of so-called bending energy on an infinitely thin elastic plate. According to [Isk03], it is possible to write the interpolation function in the form:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i R_{d,m}(\|\mathbf{x}-\mathbf{p}_i\|) + \sum_{|\boldsymbol{\alpha}|<m} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}, \qquad (5)$$

where $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_d)$ is so-called *multi-index* and $\mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1}\cdots x_d^{\alpha_d}$, $|\boldsymbol{\alpha}| = \alpha_1 + \ldots + \alpha_d$, $\alpha_k \in \mathbb{N}_0^d$. After substituting $d = m = 2$ (dimension of space $\mathbb{R}^2$) we get a standardly used interpolation function:

$$f(\mathbf{x}) = f(x,y) = \sum_{i=1}^{N} \lambda_i r_i^2 \log(r_i) + c_1 + c_2 x + c_3 y, \quad (6)$$

where $r_i = \|\mathbf{x}-\mathbf{p}_i\| = \sqrt{(x-p_i^x)^2 + (y-p_i^y)^2}$.

3.2.2 Hardy's Multiquadrics (HMQ)

This method is very similar to the previous method, but it uses different RBFs, and for $d = 2$ it does not have a polynomial term. For our interpolation problem, we get the following interpolation function:

$$f(\mathbf{x}) = f(x,y) = \sum_{i=1}^{N} \lambda_i \sqrt{r_i^2 + c^2}. \qquad (7)$$

Value c changes the shape of the resulting interpolation surface. In general, a smaller value of the parameter c creates so-called "sharp extremes" in the graph of the function, while its greater value "smoothes" the function. In literature, there are several ways of how to suitably choose it [HL93]:

- $c = 0.815d$, where d is the average distance between the points $\mathbf{p}_i$ of set $\mathscr{P}$ to their closest neighbours,
- $c = 1.25\frac{D}{n}$, where D is the average of the smallest circle, which contains all points of the set $\mathscr{P}$,
- $c = \sqrt{\frac{1}{10}\max_{i,j}\|\mathbf{p}_i - \mathbf{p}_j\|}$,
- $c = \sqrt{\frac{3}{5}\min_{i,j}\|\mathbf{p}_i - \mathbf{p}_j\|}$.

3.3 Triangle Based Blending (TBB)

This method belongs to a group of methods that use triangular irregular network $\mathscr{T}$ created from given points $\mathbf{p}_i$. Delaunay's triangulation is the most common method because it maximizes the minimum angle of triangles. There is a large number of optimal algorithms that construct this triangular net with $\mathscr{O}(n\log n)$ complexity. We can find one of these approaches in [BDH96]. First, we calculate for each point $\mathbf{p}_i$ from set $\mathscr{P}$ a biquadratic polynomial interpolating this point and its five nearest neighbours:

$$\begin{aligned} f_i(x,y) =& a_1(x-p_i^x)^2 + a_2(x-p_i^x)(y-p_i^y) + \\ & a_3(y-p_i^y)^2 + a_4(x-p_i^x) + a_5(y-p_i^y) + h_i. \end{aligned}$$

The unknown coefficients $a_1,\ldots,a_5$ are calculated from the condition that interpolates all 6 points. If we need to calculate the value of the height h for the point $\mathbf{x}[x,y]$ we have to find in which triangle Δijk of the triangulation $\mathscr{T}$ this point lies. Then the height h is calculated as the weighted average of three values of the corresponding local functions [Ami02]:

$$f(\mathbf{x}) = f(x,y) = w_i f_i(x,y) + w_j f_j(x,y) + w_k f_k(x,y), \qquad (8)$$

where i,j,k are indices of vertices of triangle Δijk with vertices $\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k$ (see Figure 1). In Figure 1, black stars denote vertices, from which local interpolant $f_i(x,y)$ corresponding to the vertex $\mathbf{p}_i$ is calculated. Similarly, blue circles denote vertices giving interpolant $f_j(x,y)$

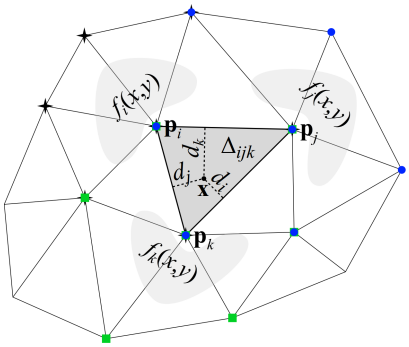


Figure 1: Local function is calculated by one of the triangle vertices and its adjacent vertices.

and green squares denote vertices creating local interpolant $f_k(x,y)$.

Smooth continuous transition between two triangles can be guaranted by calculating weights using relation:

$$w_i = d_i^r/(d_i^r + d_j^r + d_k^r),$$

where appropriate value for r is $r = 2$ or $r = 3$ and lengths d_i, d_j, d_k can be determined by using barycentric coordinates of the point $\mathbf{x}[x,y]$ of the triangle Δijk.

3.4 Natural Neighbours (NN)

This interpolation method belongs to the weighted average methods. To calculate the unknown height h at point $\mathbf{x}$, it uses a Voronoi diagram which can be constructed very effectively from Delaunay's triangulation by an algorithm with complexity $\mathcal{O}(n)$ [LH10]. We can simply say that a Voronoi diagram is the union of all Voronoi cells defined by the description:

$$\mathcal{V}(\mathbf{p}_i) = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x} - \mathbf{p}_i\| < \|\mathbf{x} - \mathbf{p}_j\| \; \forall j \neq i\},$$

where $\mathbf{p}_i \in \mathscr{P}$.

Let us call *natural neighbours* of a point $\mathbf{p}_i$ such points $\mathbf{p}_j$, whose Voronoi cells $\mathcal{V}(\mathbf{p}_j)$ have a common edge with Voronoi cell $\mathcal{V}(\mathbf{p}_i)$. It is also possible to extend the previous definition for an arbitrary point $\mathbf{x} \in \mathbb{R}^2$ by including the point $\mathbf{x}$ in the set of given points $\mathscr{P}$, and then we create a new Voronoi diagram. Natural neighbours of the point $\mathbf{x}$ are all its natural neighbours in the newly created Voronoi diagram (see Figure 2, the grey polygon represents a cell in the newly created Voronoi diagram).

As in the TBB method, the unknown height value h is calculated using the weighted average:

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{n} a_i h_i}{\sum_{i=1}^{n} a_i}, \tag{9}$$

where h_i are heights of n natural neighbours of the point $\mathbf{x}$ and weights a_i are areas of the polygon that are taken from the area of the original Voronoi cell $\mathcal{V}(\mathbf{p}_i)$ after including the point $\mathbf{x}$ into the set $\mathscr{P}$. Detailed description of the algorithm that calculates these weights without creating the new Voronoi diagram can be found in [LH10].

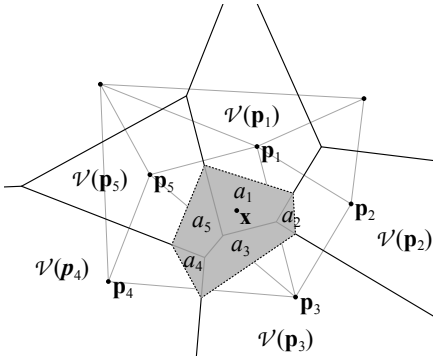


Figure 2: The original Voronoi diagram and the new Voronoi diagram which was created by including the point $\mathbf{x}$.

3.5 Least Square Methods (LSM)

Methods of least squares create approximation surfaces that do not meet the interpolation condition (1). For our purposes, we use it to determine the gradient at the points $[\mathbf{p}_i, h_i] \in \mathbb{R}^3$. Bivariate polynomials are used the most often for terrain modelling using LSM:

$$f(\mathbf{x}) = f(x,y) = \sum_{k=0}^{m} \sum_{r+s=k} a_{rs} x^r y^s, \tag{10}$$

where m is chosen polynomial degree ($m = 2$ or $m = 3$) and a_{rs} are unknown coefficients. To determine them, we need at least $(m+1)!$ given points $\mathbf{p}_i$. Unlike interpolation functions, which usually lead to solving a system of equations with a number of columns equal to the number of unknowns, in the method of least squares, we solve a system of equations with more equations than the number of unknowns. Consequently, the resulting surface cannot pass through the given entry points.

Let us calculate the bivariate polynomial for each point of $\mathbf{p}_i[p_i^x, p_i^y]$ and error e_i. Than we can create a system of equations:

$$h_i \approx \sum_{k=0}^{m} \sum_{r+s=k} a_{rs} (p_i^x)^r (p_i^y)^s + e_i, \; i = 1, \ldots, N, \tag{11}$$

where $N \gg m$.

We search for such values of coefficients a_{rs} so that the following holds:

$$\sum_{i=1}^{N} e_i^2 \to 0$$

After rewriting the system of equations (11) into matrix form, we obtain:

$$\mathbf{h} \approx \mathbf{P}\mathbf{a} + \mathbf{e}. \tag{12}$$

It is true that the sum of the squared errors $\sum_{i=1}^{N} e_i^2$ has a minimum for such vector of coefficients $\mathbf{a}$ that we can calculate using a system of normal equations [GR70]:

$$\mathbf{P}^\mathsf{T}\mathbf{h} = \mathbf{P}^\mathsf{T}\mathbf{P}\mathbf{a}$$

Vector of unknown coefficients $\mathbf{a}$ can be calculated from relation:

$$\mathbf{a} = (\mathbf{P}^\mathsf{T}\mathbf{P})^{-}\mathbf{P}^\mathsf{T}\mathbf{h},$$

where $(\mathbf{P}^\mathsf{T}\mathbf{P})^{-}$ is a pseudoinverse matrix to matrix $(\mathbf{P}^\mathsf{T}\mathbf{P})$. It is numerically convenient to use *singular value decomposition* (SVD) method while solving the system (12). We can find the SVD method, for example, in [GR70].

4 OUR APPROACH

Each of these interpolation methods has some drawbacks. RBF methods require solving systems of equations, which for a large number of input points results in high memory costs, very long calculation time and problems with numerical stability of the calculations. IDW methods, in addition to long calculation time, create unwanted artefacts (see Figure 7) in the shape of the resulting interpolation surface, which are present also in the TBB and NN methods.

Using local interpolation functions in this context is new. We have not found any study which uses local functions in interpolation methods for digital terrain model creation. In our comparison, we try to find such an interpolation method that has sufficient visual smoothness and does not suffer from shape artefacts. It should also have sufficient accuracy and a short time of calculation.

In this section, we give a procedure for finding close neighbours to a given point $\mathbf{p}_i$, which are necessary for constructing local interpolants. We will also introduce alterations to the methods using the weighted average, while we replace given height values h_i of points $\mathbf{p}_i$ by local functions $f_i(\mathbf{x})$.

4.1 Nearest Neighbours

Close neighbours of a point $\mathbf{p}_i$ can be found using Delaunay triangulation $\mathscr{T}$ created in-advance, because each vertex contains a pointer to all adjacent vertices when the triangulation is created. To determine local interpolants, we need to specify the minimum number of close neighbours of point $\mathbf{p}_i$. Without this condition, it is not possible to calculate all the unknown coefficients of local functions.

The procedure of finding these neighbours is shown in Figure 3. First, we find all adjacent vertices in the triangulation $\mathscr{T}$ to the vertex (point) $\mathbf{p}_i$, and we add them to the list of close neighbours. In Figure 3, they are marked by circles, and their index in the upper left corner has value 1 (level of the depth). If necessary, we add also the neighbouring vertices of these vertices to the list. They are marked by a star in the picture, and their index in the upper left corner shows depth value 2. We continue to the chosen level in this way. If the number of close neighbours does not achieve the necessary value, we find other vertices using the euclidean distance from vertex $\mathbf{p}_i$. In Figure 3, such vertices are labeled by green rectangles. To speed up the search by distance, we use a hash table in which all entry points $\mathbf{p}_i$ are assigned in advance.

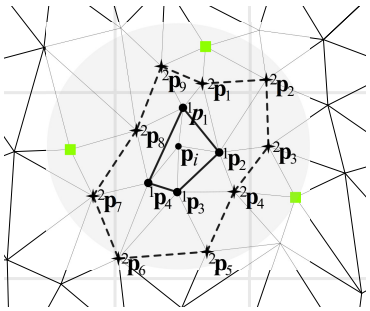


Figure 3: Selecting close neighbours of vertex $\mathbf{p}_i$ based on the neighbourliness and euclidean distance.

4.2 Modification of the Methods Using the Weighted Average

Replacing given height values by local interpolation functions allows the methods using the weighted average to make the resulting interpolation surface much more similar to the ideal (real) surface of the terrain.

Let us rewrite the expressions for interpolation functions $f(\mathbf{x})$ such that we use a local function $f_i(\mathbf{x})$ instead of the height h_i:

- We get the following expression for the IDW method (see relation (2)):

$$f(\mathbf{x}) = \sum_{i=1}^{N} \omega_i(\mathbf{x}) f_i(\mathbf{x}).$$

- The expression remains the same in the TBB methods (see relation (8)), but the original biquadratic polynomial is replaced by a general local function.

- For the method NN (relation (9)), we get the following expression:

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{n} a_i f_i(\mathbf{x})}{\sum_{i=1}^{n} a_i},$$

where n is number of natural neighbours of point $\mathbf{x}$.

4.3 Local Interpolants

At first, we choose thin plate splines (paragraph 3.2.1) and Hardy's multiquadrics (paragraph 3.2.2) as the local interpolants in our tests. Secondly, we use bivariate polynomial:

$$f_i(x,y) = \sum_{k=0}^{m} \sum_{r+s=k} a_{rs}(x - p_i^x)^r (y - p_i^y)^s + h_i$$

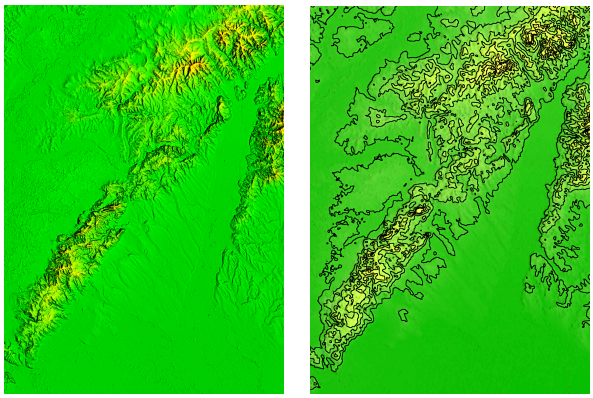


Figure 4: Original height map of the Little Carpathians is on the left, and the model created using TPS is on the right.

with $m = 2$ (biquadratic polynomial) and $m = 3$ (bicubic polynomial). Unknown coefficients a_{rs} are calculated from interpolation conditions $f_i(p_l^x, p_l^y) = h_l$, where $\mathbf{p}_l[p_l^x, p_l^y]$ are nearest neighbours of point $\mathbf{p}_i$.

Value of the height h_i at point $\mathbf{p}_i$ can be replaced by a relatively simple function of plane going through the point $\mathbf{p}_i[p_i^x, p_i^y, h_i]$ with expression:

$$f_i(x,y) = \frac{n_x}{n_z}(x - p_i^x) + \frac{n_y}{n_z}(y - p_i^y) + h_i,$$

while the normal vector $\mathbf{n}(n_x, n_y, n_z)$ is calculated from the gradient:

$$\mathbf{n}(n_x, n_y, n_z) = \left(\frac{\partial f(p_i^x, p_i^y)}{\partial x}, \frac{\partial f(p_i^x, p_i^y)}{\partial y}, -1 \right),$$

where $f(\mathbf{x})$ is any of the previously mentioned local functions, from which we can easily calculate the gradient.

5 TEST OF METHODS AND RESULTS

To test our modifications and compare different interpolation methods for creating digital terrain model, we have used a dataset of height points of the Little Carpathians obtained from the United States Geological Survey in SRTM format with a resolution of 1 arc second (30 meters). From this height map, we have created two files. The first contains the area region: $48.00°\text{N}, 17.00°\text{E} - 49.00°\text{N}, 18.00°\text{E}$, in figures and tables it is labelled with the name *Karpaty* (see Figure 4). The second file contains the area region: $48.15°\text{N}, 17.05°\text{E} - 48.20°\text{N}, 17.10°\text{E}$ labelled as *KarpatyCrop*.

For both files, we have created samples with $N = 2000$, 4000, 7000 and 10000 randomly selected points that were used to create the model of the terrain. We have also created a sample of $M = 20000$ test points to verify the accuracy of the model. We could not use a larger number of points for comparing interpolation methods because the TPS and HMQ methods require using matrices with a large number of nonzero elements. In this article, we present results only for a sample of $N = 10000$ points because of the limited space.

To create a digital terrain model, we have used not only all previously described methods, but also Powell-Sabin [PS77] and Clough-Tocher [Ami02] methods. However, we do not include these two methods in the results because we have not obtained for them an interpolation surface without unwanted artefacts, even though we have used the optimal normal vectors calculated from the gradient of the local TPS interpolation.

While evaluating the precision with which the model approximates the real terrain surface, we have used two statistical metrics:

$$\text{RMSE} = \sqrt{\frac{\sum_{j=1}^{M}(f(x_j,y_j) - h_j)^2}{m}}$$

and

$$\text{Max Absolute Error} = \max_{j=1,\ldots,M}\{|f(x_j,y_j) - h_j|\}.$$

In addition to these metrics, we have been interested also in the visual smoothness, calculation time, memory demand and suitability for creating contours, which are used in topographic maps. Our results are shown in Table 1. Value *Accuracy rank* in the third column indicates the average rank of the given method, or group of methods (lower value is better). For a group of methods, we have always chosen the best candidate for the current sample of test points. Suitability sign "+/○/−" of accuracy is based on the accuracy rank. Similarly, suitability of the calculation time is based on the elapsed time in Table 2 and 3, memory demand is based on if large matrices are used in the algorithm and visual smoothness is decided visually using the obtained images (see Figure 7), depending on whether surfaces contain artefacts.

In the graphs, tables and pictures, we use the following abbreviations: lHMQ, lTPS, lQLS, lCLS denote using Hardy's multiquadrics, thin plate spline, biquadratic and bicubic polynomials as the local interpolant. Abbreviations gHMQ, gTPS, gQLS, gCLS denote using the relevant local functions while calculating the gradient of the tangent plane.

In the right part of Figure 4, we can see the terrain model of the Little Carpathians which has been calculated using the TPS method using 10000 points. This figure also demonstrates the suitability of this method for creating topographic maps with contour lines.

In Figure 7, we can see how using local functions in methods IDW, TBB and NN improves the shape of the resulting surface of the model of the terrain, and removes existing shape artefacts. In the top row, we can

see at the same time the impact of improperly selected shape parameter in Hardy's multiquadrics, which leads to undesirable sharp points.

In Table 2 and Figure 5, we can see the evaluation of the accuracy of the resulting interpolation surfaces for data file *Karpaty*, and in Table 3 and Figure 6 for the data file *KarpatyCrop*. In the last column, we give in seconds the time necessary to calculate the height values for a grid 5463×8192 points. The evaluation time does not contain time for creating the hash table and the initial triangulation.

All methods have been tested on a desktop PC with Intel(R) Core(TM) i5-4670K CPU @3.40GHz processor with 8GB RAM.

Method	Accuracy	Accuracy rank	Calculation time	Memory demand	Visual smoothness
HMQ	+/−[1]	5.5	−	−	+/−[1]
TPS	+	5.3	−	−	+
NN	−	17.4	○/+	+	○
NN + plane	+	7.9	○/+	+	+
NN + local	+	4.1	○	+	+
IDW	−	22.3	−	+	−
IDW + plane	○	10.3	−	+	−
IDW + local	+	3.3	−	+	+
TBB	−	23.9	+	+	−
TBB + plane	○	12.5	+	+	−
TBB + local	+	4.9	+	+	○

Table 1: Suitability of using interpolation method. Symbol "+" represents suitability, "−" unsuitability and "○" average suitability of using a method.

Method	Max Absolute Error	Mean Absolute Error	RMSE	Elapsed Time (s)
HMQ	184.857	10.726	20.199	6715
TPS	236.593	10.839	20.787	20930
NN - gQLS	185.654	11.221	21.090	95
NN - gCLS	204.537	11.570	21.838	107
NN - gHMQ	247.088	11.754	22.616	67
NN - gTPS	279.048	12.416	24.074	84
NN - lQLS	188.975	11.114	20.900	87
NN - lCLS	468.349	11.256	21.660	84
NN - lHMQ	205.663	10.781	20.332	86
NN - lTPS	210.575	10.732	20.495	171
IDW - gQLS	195.854	11.582	21.817	2541
IDW - gCLS	283.569	11.664	22.367	2532
IDW - gHMQ	259.551	11.762	22.775	2536
IDW - gTPS	298.943	12.436	24.255	3347
IDW - lQLS	203.549	11.669	21.666	2552
IDW - lCLS	500.426	20.895	39.873	2547
IDW - lHMQ	202.336	10.816	20.400	2616
IDW - lTPS	199.480	10.855	20.771	3603
TBB - gQLS	244.792	11.755	22.490	8
TBB - gCLS	271.387	11.918	22.991	7
TBB - gHMQ	257.001	11.992	23.461	8
TBB - gTPS	300.422	12.764	25.434	8
TBB - lQLS	245.993	11.565	22.098	8
TBB - lCLS	316.560	11.542	22.305	9
TBB - lHMQ	193.657	10.799	20.338	18
TBB - lTPS	201.006	10.780	20.562	21

Table 2: Accuracy and time of calculation for the tested methods for data file *Karpaty*.

6 CONCLUSION

We have shown that using local functions to the known methods (IDW, TBB and NN) for creating a digital terrain model significantly improves the visual smoothness of the resulting spline surface. It also increases the accuracy with which this surface approximates the actual surface of the terrain, and it suppresses undesired shape artifacts. With a suitable local function, we can even achieve results comparable with RBF methods, which have great memory and calculation requirements. At the same time, we have also pointed out that a wrong choice of the shape parameter in the HMQ method leads to a problematic surface shape.

The most appropriate method for creating the digital model, taking into account the computation time, accuracy, memory requirements and visual smoothness, is the method of Natural Neighbor with a local thin plate spline interpolant. As the second in order, we could use Triangle Based Blending method again with the local TPS interpolant, which is faster to calculate, but has worse visual smoothness.

An interesting finding is also the fact that the number of near vertices in LSM methods relates to the complexity of the terrain. For a rugged terrain, we have achieved better results when we used more points, and for a less rugged terrain when we used fewer points.

7 FUTURE WORKS

In future work, we would like to focus on other ways to estimate parameter c, which occurs in some RBFs. In addition, we would like to verify the effect of selecting different neighbours on the accuracy of the interpolation surface and use compactly supported RBFs for calculation of a digital terrain model. Our testing algorithm has not used any accelerating techniques such as using parallelization or GPU, but it would be interesting to investigate the acceleration obtained using these techniques.

[1] Depends on the shape parameter c.

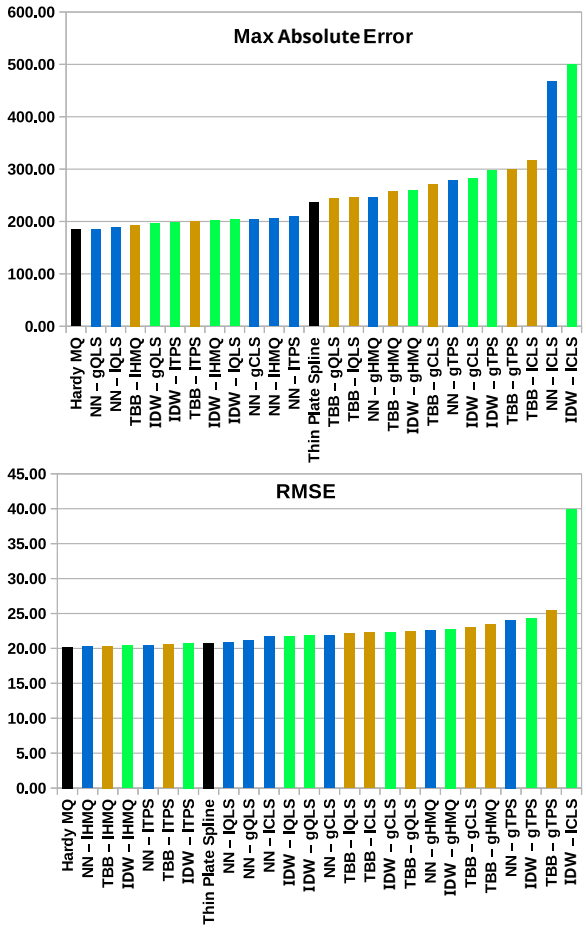


Figure 5: A plot showing the ranking of accuracy of individual methods for data file *Karpaty*.

Method	Max Absolute Error	Mean Absolute Error	RMSE	Elapsed Time (s)
HMQ	10.535	0.672	1.151	6603
TPS	8.817	0.583	0.968	26203
NN - gQLS	10.267	0.723	1.216	168
NN - gCLS	11.345	0.742	1.241	138
NN - gHMQ	13.767	0.650	1.094	76
NN - gTPS	14.911	0.665	1.128	79
NN - lQLS	10.171	0.742	1.255	171
NN - lCLS	10.163	0.677	1.142	281
NN - lHMQ	11.131	0.744	1.275	100
NN - lTPS	8.925	0.650	1.109	164
IDW - gQLS	12.914	0.790	1.311	2433
IDW - gCLS	11.689	0.757	1.242	2473
IDW - gHMQ	11.791	0.684	1.122	2441
IDW - gTPS	11.638	0.690	1.139	3131
IDW - lQLS	11.627	0.789	1.315	2443
IDW - lCLS	9.967	0.706	1.169	2479
IDW - lHMQ	10.552	0.763	1.283	2481
IDW - lTPS	8.666	0.654	1.093	3399
TBB - gQLS	11.778	0.754	1.274	8
TBB - gCLS	11.242	0.739	1.242	9
TBB - gHMQ	10.396	0.652	1.093	9
TBB - gTPS	11.587	0.675	1.128	8
TBB - lQLS	10.779	0.769	1.306	12
TBB - lCLS	9.515	0.702	1.185	11
TBB - lHMQ	10.557	0.747	1.277	20
TBB - lTPS	9.117	0.669	1.122	50

Table 3: Accuracy and time of calculation for the tested methods for data file *KarpatyCrop*.

8 REFERENCES

[AAAC05] Aguilar, F., Agüera, F., Aguilar, M., and Carvajal, F. Effects of terrain morphology, sampling density, and interpolation methods on grid dem accuracy. Photogrammetric Engineering & Remote Sensing, 71(7), pp.805-816, 2005.

[Ami02] Amidror, I. Scattered data interpolation methods for electronic imaging systems: a survey. Journal of electronic imaging, 11(2), pp.157-176, 2002.

[BDH96] Barber, B., Dobkin, D., and Huhdanpaa, H. The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software (TOMS), 22(4), pp.469-483, 1996.

[BSS14] Bagheri, H., Sadjadi, Y., and Sadeghian, S. Exploring the Role of Genetic Algorithms and Artificial Neural Networks for Interpolation of Elevation in Geoinformation Models, pages 107-121. Springer International Publishing, 2014.

[CL12] Chen, C., and Li, Y. A robust method of thin plate spline and its application to DEM construction. Computers & Geosciences, 48, pp.9-16, 2012.

[CL13] Chen, C., and Li, Y. A robust multiquadric method for digital elevation model construction. Mathematical Geosciences, 45(3), pp.297-319, 2013.

[DG02] Dakowicz, M., and Gold, C. Visualizing terrain models from contours - plausible ridge, valley and slope estimation. In Proceedings of the International Workshop on Visualization and Animation of Landscape, 2002.

[Fas07] Fasshauer, G. Meshfree approximation methods with MATLAB. vol. 6, World Scientific, 2007.

[GG13] Garnero, G., and Godone, D. Comparisons between different interpolation techniques. ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 1(3), pp.139-144, 2013.

[GR70] Golub, G., and Reinsch, C. Singular value decomposition and least squares solutions, volume 14, pages 403-420. Springer, 1970.

[GS13] Gumus, K., and Sen, A. Comparison of spatial interpolation methods and multi-layer neural networks for different point distributions on a digital elevation model. Geodetski vestnik, 57(3), pp.523-543, 2013.

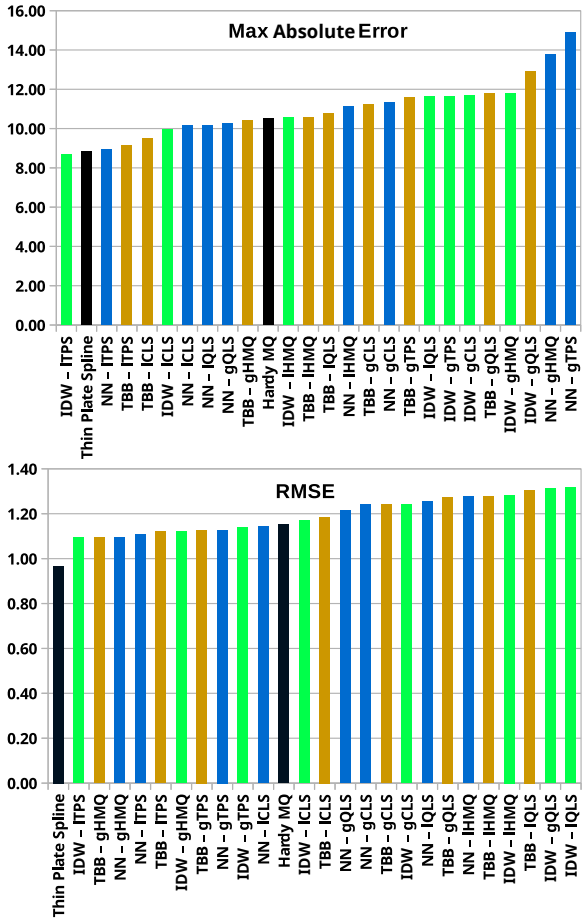


Figure 6: A plot showing the ranking of accuracy of individual methods for data file *KarpatyCrop*.

[HL93] Hoschek, J., and Lasser, D. Fundamentals of Computer Aided Geometric Design, pages 388-421. A K Peters, Wellesley, MA, 1993.

[Hu13] Hu, C. Comparison of the interpolation methods on digital terrain models. Master's thesis, Politecnico di Milano, 2013.

[Hug04] Hugentobler, M. Terrain Modelling with Triangle Based Free-Form Surfaces. PhD thesis, Mathematisch-naturwissenschaftlichen Fakultät der Universität Zürich, 2004.

[HZDS01] Haber, J., Zeilfelder, F., Davydov, O., and Seidel, H. Smooth approximation and rendering of large scattered data sets. In Proceedings of the Conference on Visualization '01, VIS '01, pp.341-348. IEEE Computer Society, 2001.

[Isk03] Iske, A. Radial basis functions: basics, advanced topics and meshfree methods for transport problems. Rendiconti del Seminario Matematico, Polytechnic University of Turin, 61(3), pp.247-285, 2003.

[LH10] Liang, L., and Hale, D. A stable and fast implementation of natural neighbor interpolation, 2010.

[MS16] Majdisova, Z., and Skala, V. A radial basis function approximation for large datasets. In Proceedings of SIGRAD 2016, May 23rd and 24th, Visby, Sweden, number 127, pages 9-14. Linköping University Electronic Press, 2016.

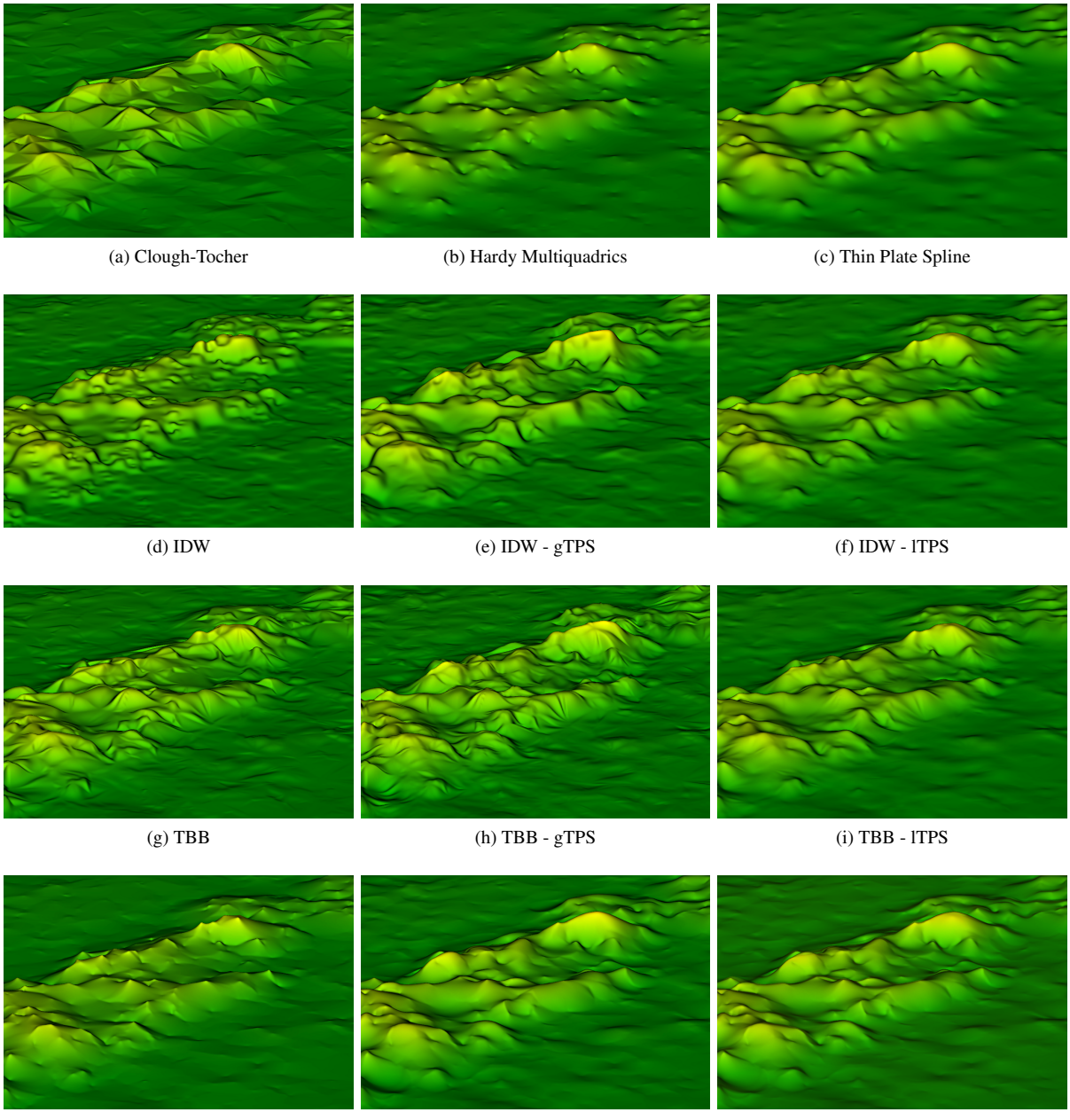
[ON15] Okwuashi, O., and Ndehedehe, C. Digital terrain model height estimation using support vector machine regression. South African Journal of Science, 111(9-10), pp.148-152, 2015.

[PGTG04] Pouderoux, J., Gonzato, J.-C., Tobor, I., and Guitton, P. Adaptive hierarchical rbf interpolation for creating smooth digital elevation models. In Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems, GIS '04, pages 232-240, New York, USA, 2004. ACM.

[PS77] Powell, M., and Sabin, M. Piecewise quadratic approximations on triangles. ACM Transactions on Mathematical Software (TOMS), 3(4), pp.316-325, 1977.

[SS09] Soycan, A., and Soycan, M. Digital elevation model production from scanned topographic contour maps via thin plate spline interpolation. Arabian Journal for Science and Engineering, 34(1), pp.121-134, 2009.

[TH10] Thacker, W., et al. Algorithm 905: SHEPPACK: Modified Shepard algorithm for interpolation of scattered multivariate data. ACM Transactions on Mathematical Software (TOMS), 37(3), 2010.



(a) Clough-Tocher (b) Hardy Multiquadrics (c) Thin Plate Spline

(d) IDW (e) IDW - gTPS (f) IDW - lTPS

(g) TBB (h) TBB - gTPS (i) TBB - lTPS

(j) NN (k) NN - gTPS (l) NN - lTPS

Figure 7: Removing shape artefacts while using local functions. Surfaces of the original methods are on the left, modifications using the tangent planes are in the middle, using local TPS is on the right.