

Gabriel Zachmann,  
René Weller,  
André Hinkenjann (Hg.)

## Virtuelle und Erweiterte Realität

11. Workshop der GI-Fachgruppe VR/AR





Berichte aus der Informatik

**Gabriel Zachmann,  
René Weller,  
André Hinkenjann (Hg.)**

# **Virtuelle und Erweiterte Realität**

11. Workshop der GI-Fachgruppe VR/AR

Shaker Verlag  
Aachen 2014



# Vorwort

Sehr geehrte Leserinnen und Leser,

wir begrüßen Sie herzlich zum 11. Workshop Virtuelle und Erweiterte Realität der Fachgruppe VR/AR der Gesellschaft für Informatik e.V. Als etablierte Plattform für den Informations- und Ideenaustausch der deutschsprachigen VR/AR-Szene bietet der Workshop den idealen Rahmen, aktuelle Ergebnisse und Vorhaben aus Forschung und Entwicklung im Kreise eines fachkundigen Publikums zur Diskussion zu stellen.

In diesem Jahr wurden 20 Beiträge in den Kategorien Langbeitrag, Kurzbeitrag und Poster auf dem Workshop eingereicht. Das Programmkomitee hat hieraus 14 Beiträge zur Präsentation und Publikation in diesem Tagungsband ausgewählt. Besichtigungen ausgewählter Forschungseinrichtungen der Universität Bremen runden den wissenschaftlichen Teil des Workshops ab. Darüber hinaus bietet sich dieses Jahr den Teilnehmern des GI-VRAR-Workshops die Gelegenheit, den VRIPHYS-Workshop (Workshop on Virtual Reality Interaction and Physical Simulation) zu besuchen, der dieses Jahr co-located vom 24.-25. September stattfindet.

Unser Dank gilt an erster Stelle den Autoren, die mit ihren qualitativ hochwertigen Beiträgen die fruchtbare Diskussion unter Experten anstoßen. Auch den Mitgliedern des Programmkomitees, die alle eingehenden Artikel begutachteten und den Autoren Anregungen zur Verbesserung gaben, danken wir für ihre Mühe und konstruktive Kritik. Ihre Auswahl verspricht uns spannende Beiträge, unter anderem aus den Themenbereichen Interaktion, Rendering und Displays, Avatare und Gaming, sowie Anwendungen.

Gastgeber in diesem Jahr ist die Universität Bremen. An verschiedenen Instituten der Universität, insbesondere dem Fachbereich für Informatik, forschen zahlreiche Arbeitsgruppen in den Bereichen Virtuelle Realität, Erweiterte Realität, Mensch-Computer-Interaktion, Medieninformatik, Künstliche Intelligenz und Computergraphik. Nachwuchs in diesen Bereichen wird in den Studiengängen Informatik (Bachelor und Master), System-Engineering (Master), Digital Media (Bachelor und Master) ausgebildet.

Wir wünschen den Teilnehmern erkenntnisreiche Tage mit lohnendem Ideenaustausch und vielversprechenden neuen Kontakten.

Bremen im September 2014

Gabriel Zachmann  
René Weller  
André Hinkenjann

## Programmkomitee

Johannes Behr, Fraunhofer IGD  
Manfred Bogen, Fraunhofer IAIS  
Christian-A. Bohn, Wedel University of Applied Sciences  
Wolfgang Broll, Ilmenau University of Technology  
Gerd Bruder, Universität Hamburg  
Guido Brunnett, Chemnitz University of Technology  
Matthias Bues, Fraunhofer IAO Stuttgart  
Ralf Doerner, RheinMain University of Applied Sciences  
Bernd Froehlich, Bauhaus-Universität Weimar  
Chris Geiger, University of Applied Sciences Düsseldorf  
Andreas Gerndt, German Aerospace Center (DLR) Braunschweig  
Paul Grimm, Hochschule Fulda  
Martin Göbel, Hochschule Bonn-Rhein-Sieg  
Andre Hinkenjann, Bonn-Rhein-Sieg University of Applied Sciences  
Thomas Hulin, DLR Oberpfaffenhofen  
Bernhard Jung, Technische Universität Bergakademie Freiberg  
Yvonne Jung, Fachhochschule Fulda  
Rolf Kruse, Fachhochschule Erfurt  
Björn Krüger, University of Bonn  
Torsten Kuhlen, RWTH Aachen University  
Marc Latoschik, Universität Würzburg  
Jens Maiero, Bonn-Rhein-Sieg University of Applied Sciences  
Björn Mellies, University of Bremen  
Daniel Mohr, University of Bremen  
Sina Mostafawy, Fachhochschule Düsseldorf  
Andreas Mühlberger, Universität Regensburg  
Heinrich Müller, Technische Universität Dortmund  
Volker Paelke, University of Hanover  
Christoph Runde, Virtual Dimension Center (VDC) Fellbach  
Mikel Sagardia, DLR Oberpfaffenhofen  
Marco Schumann, Fraunhofer IFF  
Oliver Staadt, Universität Rostock  
Frank Steinicke, Universität Hamburg  
Jörg Stöcklein, Universität Paderborn  
Martin Weier, Bonn-Rhein-Sieg University of Applied Sciences  
Rene Weller, University of Bremen  
Martin Westhoven, Fraunhofer FKIE  
Robin Wolff, DLR Braunschweig  
Uwe Wössner, HLRS Stuttgart  
Gabriel Zachmann, University of Bremen

# Inhalt

<b>A Generic Virtual Reality Flight Simulator</b>	1
Turgay Aslandere, Daniel Dreyer, Frieder Pantkratz and Rene Schubotz	
<b>Ein parametrisches Modell für konfigurierbare interaktive 3D-Produktpräsentationen im Web</b>	13
Ekkehard Beier, Michael Englert, Jonas Etzold, Paul Grimm, Marcel Klomann and Yvonne Jung	
<b>Entwicklung einer gestenbasierten Steuerung und eines virtuellen Tenor-Avatars für eine interaktive Medieninstallation mit Gesangssynthese</b>	25
Jochen Feitsch, Marco Strobel and Chris Geiger	
<b>Picture-based Localisation for Pervasive Gaming</b>	37
Martin Fischbach, Jean-Luc Lugrin, Marc Erich Latoschik and Michael Fendt	
<b>SPIRIT - Ereignisgesteuerte Informationsvermittlung, Inspiration und Unterhaltung im urbanen Umfeld auf Basis mobiler Augmented Reality Technologien</b>	49
Antonia Kampa, Habiburrahman Dastageeri, Martin Storz, Volker Coors and Ulrike Spierling	
<b>”Real-Time Global Illumination im Vergleich: Analyse aktueller Algorithmen und Optimierung durch Blurred Reflective Shadow Maps</b>	61
Valentin Kraft, Sina Mostafawy and Martin Panknin	
<b>Dynamic Emotional States based on Personality Profiles for Adaptive Agent Behavior Patterns</b>	73
Fabian Krueger, Sven Seele, Rainer Herpers, Christian Bauckhage and Peter Becker	
<b>Fast Construction of SAH-based Bounding Interval Hierarchies using Dynamic Parallelism</b>	85
Carl-Feofan Matthes, Adrian Kreskowski, Andre Schollmeyer and Bernd Froehlich	
<b>Innovative, Contact-free Natural User Interaction with Cars</b>	97
Mohammad Razavi, Saber Adavi, Muhammed Zaid Alam, Daniel Mohr and Gabriel Zachmann	
<b>Dynamic Level of Detail for Tiled Large High-Resolution Displays</b>	109
Christian Scheel, Falko Löffler, Anke Lehmann, Heidrun Schumann and Oliver Stadt	
<b>3D-Längenanamorphosen in einem einzigen Renderingschritt</b>	121
Jonas Schell, Tom Vierjahn and Sina Mostafawy	
<b>Camera-Based Pointing Technique Using Dynamic On-Screen Markers</b>	133
David Scherfgen and Rainer Herpers	

<b>Enhancing Rendering Performance with View-Direction-Based Rendering Techniques for Large, High Resolution Multi-Display-Systems</b>	145
Martin Weier, Jens Maiero, Thorsten Roth, Andre Hinkenjann and Philipp Slusallek	
<b>Black Offset Correction on Multiprojector-Display-Systems</b>	157
Timon Zietlow, Marcel Heinz and Guido Brunnett	



# A Generic Virtual Reality Flight Simulator

Turgay Aslandere\*, Daniel Dreyer†, Frieder Pankratz\*, Renè Schubotz†

\* German Aerospace Center (DLR) † Airbus Group Innovations  
38106 Brunswick 81663 Munich

E-Mail: Turgay.Aslandere@dlr.de

\* Technische Universität München (TUM)  
D-85748, Munich

**Abstract:** Flight simulators are used for different purposes, such as pilot training, aircraft design and development. Full-scale flight simulators have high costs and dependency on aircraft type due to hardware constraints. Hence, virtual reality flight simulators are designed. On the other hand, they are generally created only for specific applications, such as helicopter simulators. As a result, these tools can hardly be used as a generic tool which can work with various aircraft simulations. Although, there are certain generic virtual reality applications that can be used for virtual prototyping and ergonomics, they lack realistic flight simulation and environment. In this paper, we present a generic aerospace application which brings a solution to these problems. The architecture of the application is described and a calibration method which, makes the application independent of the physical mock-up and the flight simulator compatible with different aircraft types, is presented. The preliminary results of the first prototype are given as the generic virtual reality flight simulator is used by the aerospace industry for research and development purposes.

**Keywords:** System Architecture and Intelligent environments, 3D Interaction Devices and Interaction Techniques, Flight Simulator

## 1 Introduction

Full flight simulators have drawbacks such as high costs since they have a fixed installation because of the hardware constraints. Several low cost flight simulators have been employed such as [LHH07], so far they are dependent on the aircraft type. The adaption of a mock-up to aircraft types is expensive and difficult with this kind of flight simulators. It is necessary to create a different replica cockpit for each aircraft type to use this kind of simulators as a generic flight simulator. Hence, an alternative flight simulator concept compatible with all kind of aircraft is needed.

In the industry, various virtual reality tools are used for virtual prototyping and ergonomics analysis. The tools which are created for virtual prototyping and evaluation of the ergonomics as Lijing et al. suggested [LWX<sup>+</sup>09] can be used with different aircraft types without a cockpit replica. However, these tools only provide an internal view of the aircraft and lack a flight environment and simulation in general. Therefore, the pilots who use the virtual

prototyping software, are not able to see the world including airports, cities as they look through the aircraft window. Therefore, their evaluation of the aircraft is not satisfying with virtual prototyping tools. As a result, a generic virtual reality flight simulator which is compatible with all kind of aircraft and has complete flight physics and environment is required. The flight simulator can also be used for pilot training purposes along with the aircraft development and testing purposes.

In this paper, we create a generic Virtual Reality Flight Simulator (VRFS). The most central components of the VRFS are discussed introducing a modular approach. Our approach makes the VRFS usable with various scenarios from virtual reality flight training to testing cockpit ergonomics. This approach also makes the VRFS capable of working with various aircraft. However, compatibility with different aircraft requires a perfect alignment of virtual and real world. Therefore, a calibration method is described. The preliminary results with regard to the interaction in the VRFS are discussed.

Our work has two main contributions. First, we describe how to implement a generic VRFS exploiting existing tools, such as desktop flight simulators, using an extensible and adaptable system architecture. Second, we show that the system brings new challenges with regard to interaction techniques. We evaluate interaction with the virtual buttons.

Virtual reality applications have been developed to overcome the high costs of the full-scale flight simulators and dependency on aircraft type. Yavrucuk et al. present a virtual reality helicopter simulator [YKT11]. This simulator is a low cost solution. On the other hand, the application is highly based on the Flight Gear [Per04] software and does not present a generic software architecture. Unlike [YKT11], an interactive real time application with a modular architecture which is not dependent on the virtual environment is demonstrated by Wolff et al. [WPG11]. It is designed for satellite servicing and can also be used for astronaut training and maintenance [WPG11]. Courter et al. proposed an extensible aerospace tool which aims more at walk-through scenarios [CSN<sup>+</sup>10]. A walk-through property is not necessarily needed for flight simulators since pilots usually control the aircraft while they are sitting on a cockpit seat.

## **2 System Overview**

VR (Virtual Reality) applications are complex systems which consist of various components. Zachmann describes the general architecture and basic software components of VR systems [Zac00]. The VRFS also contains basic software components which will be called core modules. These are virtual environment, communication and object handler module. The virtual environment module is responsible for visualization and flight simulation. The communication module provides data exchange with other modules and the object handler is the module that controls the human computer interaction. These are the essential components which are required to run the VRFS with the hardware.

The hardware of VRFS can vary as the system is generic and independent of specific input and output devices. The modules of the VRFS run in high performance workstations

with multi-core processors and GPUs. The first prototype consists of hardware components depicted in Figure 1. An optical tracking system (ART Optical Tracking System [ART12]) for finger and head tracking, a joystick, throttle and pedals with other desktop input devices such as keyboard and mouse are employed as input devices. An optical tracking system is chosen for the first prototype, since it provides precise tracking with minimal latency. Besides, it does not have any cables for connection which allows users to move more easily. It is not only employed for finger and head tracking but also to track the seating buck. Actually, the seating buck is a simplified mock-up with low cost constraints. It includes a cockpit seat which is surrounded by the basic flight hardware, such as joystick and pedals in the first prototype. Most of the interaction is achieved by using virtual hand metaphor [BKLP04] with virtual objects without a more complex physical mock-up. This is necessary to reduce the hardware used in the virtual reality flight simulator and to make it capable of working with all kinds of aircraft including airplanes and helicopters. On the other hand, the interaction with virtual objects is challenging and has some drawbacks such as missing haptic feed-back and real time constraints.

A Head Mounted Display (Nvisor SX60) and stereo headphones are used as output devices. The first prototype is also usable with projection based output. However, projection based displays have active view problem where head tracking cannot be used for more than one person, whereas HMDs do not have this problem since each user has to wear an independent HMD with a head tracking target. Therefore, the projection based systems are supported for future development but they are not used for the first prototype of VRFS. Stereoscopic views which enable binocular oculomotor cues and increase the feeling of presence in the synthetic 3D world of the VRFS are provided for both HMDs and projection based output devices.

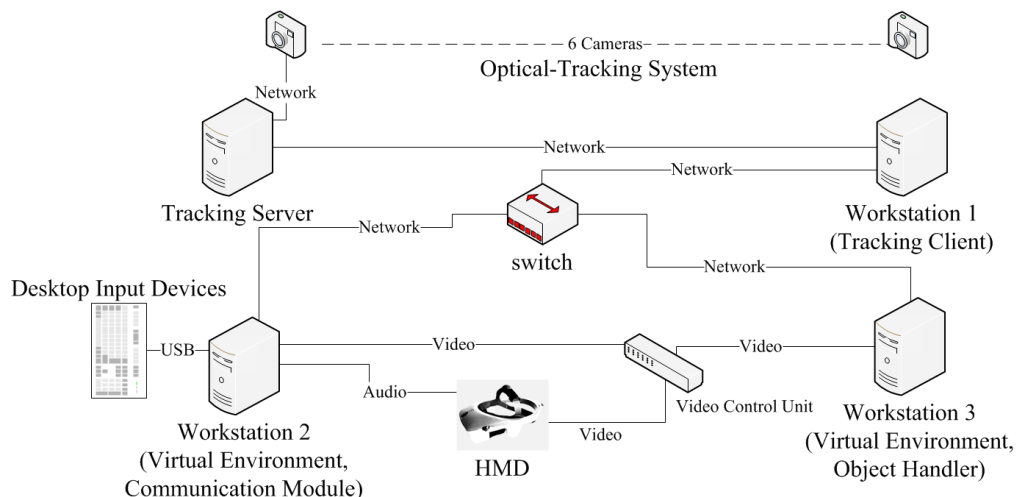


Figure 1: A possible configuration for data flow through the hardware components of the VRFS. In the first prototype, the virtual environment does not allow users to create more than one view port. Therefore, two instances of the virtual environment module are necessary for stereoscopic views.

### 3 System Architecture

The VRFS is a distributed system as the modules are generally located on different computing environments due to requirements of high computing power. Also, network connected computers are needed where multiple users are involved. The communication among the modules is provided by the Local Area Network (LAN). The data flow through the modules would be different where more users are involved or additional modules are executed.

Each module can be located on a different workstation (Figure 1) where high performance is needed. However, end-to-end latency of the distributed system increases with this configuration due to network latency.

#### 3.1 Communication Module

The communication module is the core of the VRFS. Besides the communication, it is also responsible for spatial transformations, calibration of the system and sensor fusion. These tasks are achieved by the communication module to isolate modules of the VRFS from each other. In this manner, additional modules should be only connected to the communication module for further development.

The communication module exploits the Ubitrack framework which makes use of the SRG (Spatial Relationship Graph) concept to setup a dataflow graph. SRG is a graph where the nodes represent coordinate systems or orientation free points on real or virtual objects. The edges of SRG represent transformations between the coordinate systems [EHP<sup>+</sup>08]. The spatial relationship graph specifies all relevant properties of the tracking setup. SRGs enable the automated analysis and synthesis of complex tracking scenarios. Therefore, they improve the usability of VRFS and reduce the sources of potential error related to the tracking setup. A more detailed description of Ubitrack and SRG concept can be found in [PHW<sup>+</sup>11].

Ubitrack can dynamically adapt to changes in the sensor infrastructure. In other words, it incorporates all hardware devices and software components that deliver raw spatial data. The generic sensor API of Ubitrack provides an abstraction from the specific hardware devices while providing sufficient information to make use of existing tracking devices [NWB04]. Therefore, exploiting this capability makes the VRFS independent of the virtual reality input devices. It also offers a tool called Trackman with an editor for graphical planning and analysis of the tracking setup. This editor makes the tracking setup of VRFS accessible by engineers and designers who have no programming experience.

#### 3.2 Virtual Environment Module

The virtual environment module is responsible for the flight simulator engine and visualization of virtual scenarios. The virtual environment is internally represented as a scene graph. It includes virtual world objects and their behavior. Accordingly, the objects such as airports and aircraft are a part of the virtual environment. These objects are expected to look realistic and act according to the law of physics. The virtual environment of the

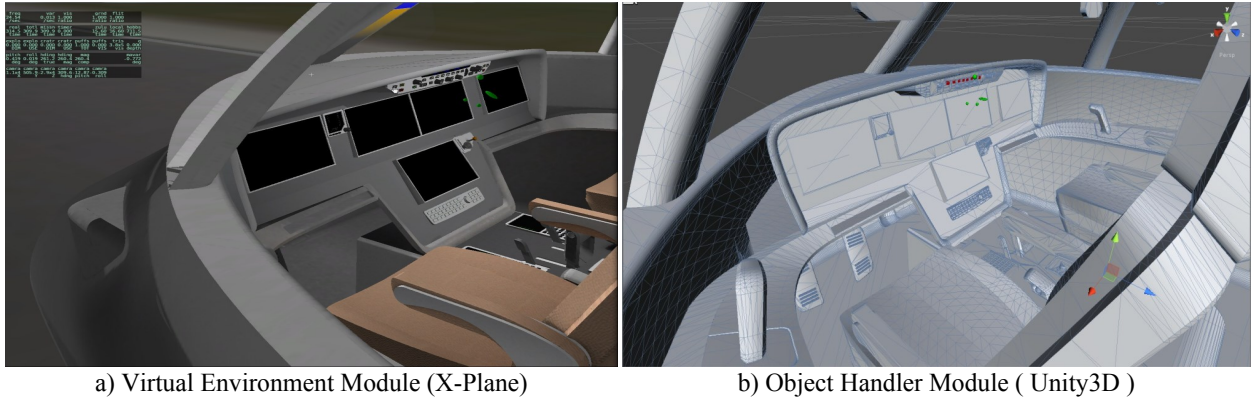


Figure 2: Aircraft geometry in the virtual environment and object handler module.

first prototype can be seen in Figure 2 a) where all properties of the virtual environment such as textures and lighting are assigned. The virtual environment module has a control and interaction sub-module which receives the data from the object handler module through the communication module and assigns it to the virtual world. The flight simulator engine is also generally a sub-module of the virtual environment module. Its responsibility is the simulation flight physics and conditions. The sub-modules of the virtual environment can be plugins or classes depending on the virtual environment that is employed.

The virtual environment module is generally a desktop flight simulator such as X-Plane[xpl12] , Flight-Gear[Per04]. In this case, the control and interaction plugin bridges the components of VRFS with the desktop flight simulator. On the other hand, in-house-developed software which consists of flight simulator and rendering engine and might be preferred for commercial purposes. We use X-Plane 9 in the first prototype. Actually, the full flight simulators (which imitate whole aircraft with motion systems) exploit the similar software. X-Plane is a commercial engineering tool that is used to predict performance of aircraft with high accuracy. It also has Federal Aviation Administration (FAA) certification to be used for pilot training with valid hardware. It provides the flight simulator engine within the virtual environment in the VRFS.

### 3.3 Object Handler Module

The object handler module is responsible for the human-computer interaction and the physics in the aircraft interiors. In other words, what happens in the cockpit or aircraft cabin, is under control of the object handler module. A physics simulation is necessary to simulate correct behavior of virtual objects inside of the cockpit.

The object handler module receives input from the communication module and produces collision data using the collision detection module which is usually embedded into the object handler. The object handler module is required to have an editor that can be used for interaction design. This editor employs the same CAD data of the aircraft as the virtual environment module. However, this geometry does not have textures and colors since it is

only used for collision detection and physics (Figure 2 b)).

The object handler was independent of any other software packages during the design of the VRFS. However, the advantages of commercial software, such as an editor to configure scene graph for the interaction management push us to make the architecture of the object handler compatible with them. In the first prototype of VRFS, we employed Unity3d. It is a commercial software package which offers an editor that can be used for interaction design. Unity3d contains NVIDIA PhysX built-in physics engine that is used to simulate the physics inside the cockpit.

The object handler module makes the interaction independent of the aircraft geometry. We implement predefined aircraft objects (particular instance of a class) without any geometry to achieve this. The CAD data of aircraft must be assigned to predefined objects in the scene graph by the designers and engineers who are going to test their virtual setup in the VRFS. This process does not require any programming knowledge, it is achieved by dragging a CAD object onto the predefined object by the mouse cursor using the graphical user interface.

### **3.4 Data Flow**

The data flow among the modules of the first prototype is shown in Figure 3. In this figure, the input device client collects data from tracking devices, sends it to the communication module, which then computes required coordinate transformations for the tracking data. After that, the communication module sends the transformed tracking data to the object handler and virtual environment module. The computation of the coordinate transformations is a necessity since coordinate systems in the object handler and the virtual environment often have a different orientation and origin. The object handler processes transformed tracking data to compute collisions with the collision detection module. The computed collision data is sent back to the communication module by the object handler. The virtual environment module receives the processed data from the object handler and provides the visual and audio output to the hardware. The processed data includes collision data, transformed tracking data and additional data such as aircraft ID.

The virtual environment module also directly receives sensor data from desktop input devices and assigns it to flight simulator engine. Afterwards, the flight data including location, speed, altitude of the aircraft is synchronized with the main virtual environment, if there is more than one instance of virtual environment module. The sensor data from desktop input devices could be also distributed via communication module. On the other hand, this would increase latency. The latency of the aircraft controls such as a joystick and pedals can result in incorrect navigation of the aircraft.

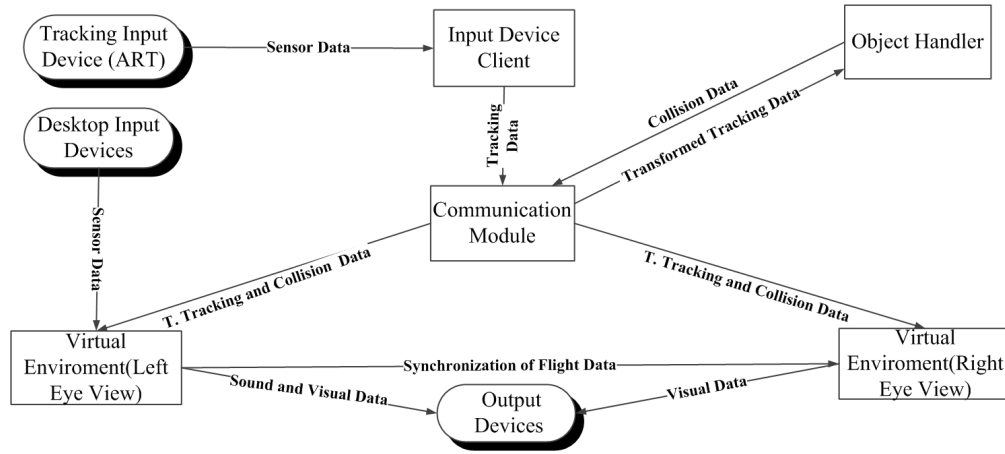


Figure 3: A possible configuration for data flow through the software components of VRFS. This diagram will differ as new modules are added. The flight data is synchronized without using communication module to make the VRFS independent of the virtual environment.

## 4 Calibration and Validation of the System

Virtual seating bucks are commonly used for different purposes such as the investigation of ergonomics during product development [SF08]. However, they require careful calibration to ensure alignment between the virtual and physical world the user interacts with. In this section, we describe our calibration method: "Virtual Seating Buck Calibration (VSBC)" for the first prototype of VRFS.

### 4.1 Virtual Seating Buck Calibration (VSBC)

The cockpit objects in both virtual and real world must precisely match with each other. Otherwise it would be difficult to provide a feeling of presence in the VRFS. And this can have a negative effect on interaction. For this reason, the VSBC is provided to align the virtual environment and real world for exploiting the seating buck. The VSBC is performed in three steps. First of all, the alignment of the virtual and real objects is performed considering geometry and location. Second, the alignment of coordinate systems of virtual and real world is provided. This assures that real and virtual world objects have the same scale and move in the same orientation. Finally, the display system is calibrated. This provides the correct image of the virtual world from the first person point of view.

*The alignment of the real and virtual objects:* The geometry and location of the objects must be precisely the same in the virtual environment and real world. Non-functional geometry, colors, material and the rest of the properties related to cosmetics of the physical mock-up do not have to be similar since, HMD blocks the real world. For this process, a cockpit seat is chosen, although an arbitrary seat may be chosen on which users can be located. The cockpit seat is scanned with a calibrated 3-D scanner and point clouds are obtained. Then, the mesh of the seat is created using open source software which provides marching cubes algorithms. The reconstructed seat is modified by the aircraft designers

according to the design of the aircraft. The usage of a different aircraft seat for each virtual cockpit can be extremely expensive. Therefore, this process (Figure 4) reduces the costs of physical mock-up. Moreover, different aircraft types are tested constantly in virtual reality laboratories and providing a physical mock-up for each aircraft type is difficult.

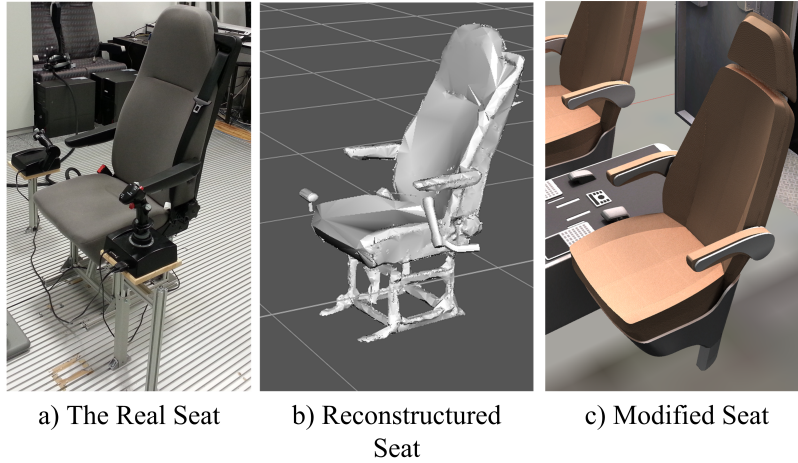


Figure 4: Alignment of the real and virtual seat object.

*The alignment of the coordinate systems:* The orientation and scaling of the coordinate systems of the virtual environment and real world must be identical or additional transformations must be implemented. A point on the seat in the both virtual environment and real world is chosen for this purpose (Figure 5). A tracking target is located on the point to get the exact position of it in the real world. In the virtual environment, the location of this point (which is represented by virtual marker) is determined by measuring the position of the real tracking target on the real seat. Afterwards, transformations are performed in the communication module to match the position and orientation of the virtual marker and the real tracking target. The transformations are carried out using SRGs. In Figure 5, the virtual and real tracking targets are circled in red and the orientation of the virtual environment and real world coordinate systems are illustrated on the backs of both seats. The position and orientation of the head tracking target is computed relative to the tracking target to make the VRFS independent of the room coordinate system. In other words, regardless of orientation and location of the real seat in the room, the user will be sitting on the virtual seat and looking at the same direction in the virtual environment.

*Calibration of the HMD:* Uncalibrated HMD systems can cause distorted perspective-related cues and lead to wrong distance judgments. The method which [KTCR09] suggested for calibration of HMDs in virtual reality environments is used in this work. This method is based on the perception of users that they can estimate correctness of the FOV by repeatedly putting the HMD on and off while they are comparing real objects and virtual objects displayed in the HMD. The FOV in the VRFS system is easily determined since all the measurements necessary for the calibration are known including the location of the user and the size of the objects in both virtual and real world.





Figure 5: Alignment of the coordinate systems.

## 4.2 Validation

The validation was performed by four male lab members whose ages were between 25 and 35. The seat model was a proper validation parameter to test, since the similar geometry was used in the real and virtual world. The subjects were asked to touch different points on the virtual seat (Figure 6). None of the subjects reported any mismatches between the seats in the real and virtual world.

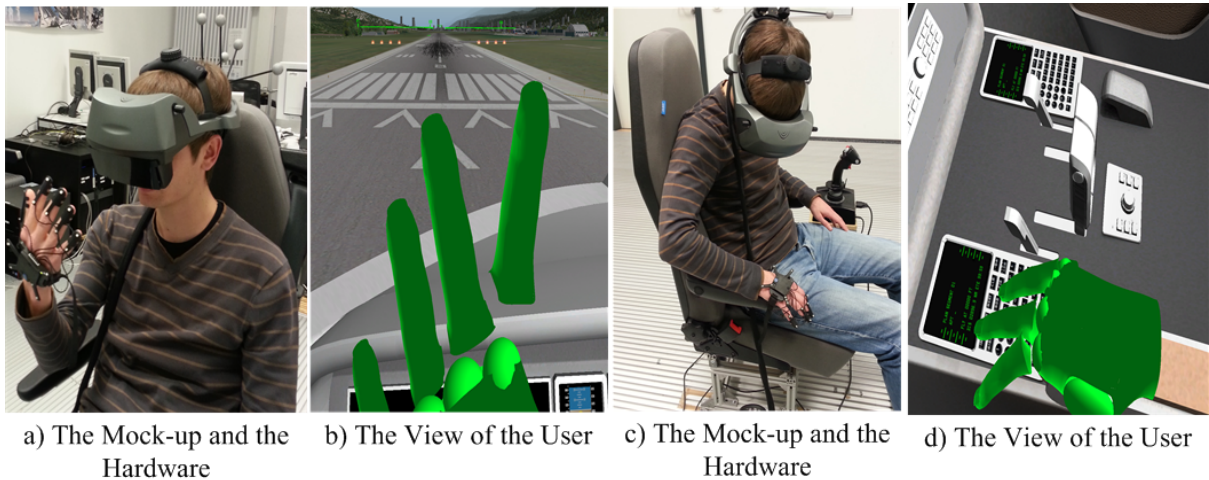


Figure 6: Validation of the Virtual Seating Buck Calibration

## 5 Preliminary Results

We evaluated the effectiveness of virtual hand-button interaction in the VRFS. The experiments were performed with 16 right handed persons whose ages were between 23 and 44. Two of the participants were female. A brief introduction to the VRFS and test procedure was given to participants before they started the experiments. The aim of the introduction was to make sure that the participants would follow the standard behavior. For example, the participants were supposed to put their hands on their knees after touching a button,

and wait for the next instruction. The hand avatar was a scanned hand geometry of a male hand which almost had 50 percentile male hand size (BS EN ISO 7250-1:2010).

The participants touched 7 virtual buttons (The button size:  $15 \times 10 \times 10 \text{ mm}^3$ ) on an FCU (Flight Control Unit) panel with two runs. Each button was pressed once after an instruction. In total, they touched 14 buttons. The order of the buttons that the participants touched, was pseudorandom. Feed-back of the selection is provided visually by changing the color of the virtual hand as haptic feed-back cannot be provided. The color of the hand avatar was red during the collision with a button. Otherwise, it was green. All participants had short breaks between the runs to decrease effect of tiredness. The hits such as touching the same target button twice, touching another buttons while target button was instructed, missing the target button were counted as a miss. When the target button was enabled, it was lit. The results were recorded as a percentage of successful hits. The elapsed time for touching the target button was recorded. Since elapsed time was biased by instructor's latency, this information was not used for measuring the hit rate.

The participants achieved a mean value of 0.77 ( $\sigma = 0.19$ ) hit rate. Some of the participants achieve over 0.90 hit rate. Misses occurred randomly. The participants generally reported that the HMD was heavy or the finger tracking device did not fit them well. One of the participant reported that he was unable to converge stereo images. The missing haptic feed-back was the biggest shortcoming of the VRFS during the experiments. Most of the participants hit a target button more than one time which increased the miss rate during the interaction. Because, they were not aware that they were touching buttons in spite of the visual feed-back and their finger was going through the buttons in the real world. The solution for this miss rate is to use a physical mock-up which gives users more haptic feed-back. On the other hand, adapting the physical mock-up to different cockpits is problematic and expensive. This would also make the VRFS partly dependent on the aircraft type which is an inadmissible property.

It has been observed that the frame rate of the virtual environment is faster than 25 fps during our tests. On the other hand, the frame rate relies on many variables. For example, enabling weather conditions in the virtual environment or increasing the number of the objects in the virtual cockpit for the collision detection would slow down the application. Also, the end-to-end system latency is very dependent on the hardware where modules are located. The latency would increase when additional modules are used for multiple users. Therefore, a detailed latency measurement is not given in this paper. A latency analysis for various configurations of the VRFS and the discussion for the real-time rendering challenges are left as future work.

## 6 Conclusion

This paper presented an outline of a generic distributed virtual reality application which is aimed to meet the needs of the aerospace industry. The most central components of the modular system were discussed. Furthermore, the advantages of the VRFS over its

counterparts, such as its independence from the tracking devices, virtual environment and aircraft type were explained. The calibration and validation of the generic virtual reality application was demonstrated.

The preliminary results show that the VRFS is a promising flight simulator concept in spite of the real time constraint. The VRFS is used as an engineering flight simulator for testing new aircraft concepts at the moment. The virtual hand-button interaction might be sufficient for virtual prototyping but it is not ready for pilot training yet. In the future work, we will evaluate the effect of the hand avatar and the button size on the interaction. Also, instead of the visual feed-back, an electrotactile feed-back might be an alternative during the virtual hand-button interaction. Additionally, we will investigate the interaction with other virtual cockpit objects, such as sliders and touch-screens which will be needed for pilot training.

## 7 Acknowledgements

This work has been supported by Airbus Group.

## References

- [ART12] ART Advanced Realtime Tracking GmbH. *<http://www.ar-tracking.com>*, 2012.
- [BKLP04] Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola, and Ivan Poupyrev. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.
- [CSN<sup>+</sup>10] D. Courter, J.P. Springer, C. Neumann, C. Cruz-Neira, and D. Reiners. Beyond desktop point and click: Immersive walkthrough of aerospace structures. In *Aerospace Conference, 2010 IEEE*, pages 1–8, 2010.
- [EHP<sup>+</sup>08] F. Echtler, M. Huber, D. Pustka, P. Keitler, and G. Klinker. Splitting the scene graph - using spatial relationship graphs instead of scene graphs in augmented reality. In *Proceedings of the 3rd International Conference on Computer Graphics Theory and Applications (GRAPP)*, January 2008.
- [KTCR09] Scott A. Kuhl, William B. Thompson, and Sarah H. Creem-Regehr. Hmd calibration and its effects on distance judgments. *ACM Trans. Appl. Percept.*, 6(3):19:1–19:20, September 2009.
- [LHH07] Zhang Lei, Jiang Hongzhou, and Li Hongren. Pc based high quality and low cost flight simulator. In *Automation and Logistics, 2007 IEEE International Conference on*, pages 1017–1022, 2007.
- [LWX<sup>+</sup>09] Wang Lijing, Xiang Wei, He Xueli, Sun Xiaohui, Yu Jinhai, Zhou Lin, and Sun Gaoyong. The virtual evaluation of the ergonomics layout in aircraft cockpit.

In *Computer-Aided Industrial Design Conceptual Design, 2009. CAID CD 2009. IEEE 10th International Conference on*, pages 1438–1442, 2009.

- [NWB04] Joe Newman, Martin Wagner, and Martin Bauer. Ubiquitous tracking for augmented reality. In *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pages 192–201, Arlington, VA, USA, Nov. 2004.
- [Per04] Alexander R. Perry. The flightgear flight simulator. In *Proceedings of the annual conference on USENIX Annual Technical Conference, ATEC '04*, pages 31–31, Berkeley, CA, USA, 2004. USENIX Association.
- [PHW<sup>+</sup>11] D. Pustka, M. Huber, C. Waechter, F. Echtler, P. Keitler, J. Newman, D. Schmalstieg, and Gudrun Klinker. Automatic configuration of pervasive sensor networks for augmented reality. *Pervasive Computing, IEEE*, 10(3):68–79, 2011.
- [SF08] H. Salzmann and B. Froehlich. The two-user seating buck: Enabling face-to-face discussions of novel car interface concepts. In *Virtual Reality Conference, 2008. VR '08. IEEE*, pages 75–82, 2008.
- [WPG11] R. Wolff, C. Preusche, and A. Gerndt. A modular architecture for an interactive real-time simulation and training environment for satellite on-orbit servicing. In *Distributed Simulation and Real Time Applications (DS-RT), 2011 IEEE/ACM 15th International Symposium on*, pages 72–80, 2011.
- [xpl12] *Laminar Research. X-Plane Manual*, 2012.
- [YKT11] I. Yavrucuk, E. Kubali, and O. Tarimci. A low cost flight simulator using virtual reality tools. *Aerospace and Electronic Systems Magazine, IEEE*, 26(4):10–14, 2011.
- [Zac00] Dipl.-Inform. Gabriel Zachmann. *Virtual Reality in Assembly Simulation Collision Detection, Simulation Algorithms, and Interaction Techniques*. PhD thesis, der Technischen Universitaet Darmstadt, 2000.

# Ein parametrisches Modell für konfigurierbare interaktive 3D-Produktpräsentationen im Web

Ekkehard Beier\*, Michael Englert†, Jonas Etzold†,  
Paul Grimm†, Yvonne Jung†, Marcel Klomann†

\* intelligentgraphics GmbH  
Ehrenbergstr. 11  
98693 Ilmenau

† Hochschule Fulda  
Marquardstr. 35  
36039 Fulda

eb@intelligentgraphics.biz {vorname.nachname}@cs.hs-fulda.de

**Zusammenfassung:** Ein konsistentes Datenmodell zur datenbankgetriebenen Generierung von interaktiven 3D-Produktkonfigurationen und Produktpräsentationen wird in diesem Beitrag vorgestellt. Die schnelle und effektive Erstellung (Authoring) von 3D-Anwendungen stellt heutzutage immer noch eine der großen Herausforderungen dar. Durch die Rückführung aller Interaktionsmöglichkeiten auf elementare Interaktoren ist es möglich, das Verhalten ebenso parametrisch zu beschreiben wie Geometrien und Materialien. Durch eine Hierarchisierung dieser Interaktoren kann zudem komplexes Verhalten ebenso beschrieben werden wie Interproduktbeziehungswissen. So wird es ermöglicht, eine interaktive 3D-Anwendung zu parametrisieren, so dass sie automatisch aus einem ERP-System (Enterprise Resource Planing-System) erzeugt werden kann. Die Beschreibung der erzeugten 3D-Anwendung erfolgt auf einem Treiberkonzept, so dass sie unabhängig von speziellen 3D-Bibliotheken erfolgt. Durch die Verwendung von speziellen Graphiktreibern, die sowohl die Funktionen für das Rendering ebenso wie für die Interaktionen kapseln, erfolgt dann die eigentliche Erstellung der Laufzeitumgebung. Eine beispielhafte Implementierung auf Basis von X3DOM wird im Anschluss vorgestellt.

Die Fokussierung auf den konsequenten Einsatz von Datenbank-Technologien und die Erstellung einer serviceorientierten Infrastruktur erlaubt dabei die Verlagerung des Erstellungsprozesses hin zu den Anwendungsexperten, die nicht notwendigerweise Programmierer sein müssen. Abschließend wird die Anwendung des vorgestellten Datenmodells auf 3D-Produktkonfiguratoren im Bereich der Möbelindustrie vorgestellt, die sich durch umfangreiche Parametrien und Strukturen sowie mögliche Interaktionen zur Produktindividualisierung und durch vielfältige Interproduktabhängigkeiten auszeichnet.

**Stichworte:** Authoring, 3D-Produktkonfiguratoren, X3DOM, HTML5

## 1 Einleitung

Die Visualisierung von Produkten in 3D hat eine Reihe von Vorteilen [BK14]:

- Die Ergebnisse eines ggf. komplexen Konfigurationsprozesses können dargestellt werden und unter Korrektheits- und/oder ästhetischen Aspekten evaluiert werden.

- Im Bereich konfigurierbarer Produkte bzw. Produkt-Design können Konfigurationen bzw. Entwürfe visualisiert werden, bevor sie überhaupt real existieren.
- Die bildliche Darstellung kann in entsprechend hochwertiger Ausführung für weitergehendes verwendet werden, z.B. zur Anreicherung einer elektronischen Bestellung oder für Produktkataloge.

Durch die mobile Verfügbarkeit von 3D-Graphik ergeben sich neue Aspekte einerseits der Anwendbarkeit, etwa durch AR-Szenarien, andererseits auch Anforderungen im interaktiven oder manipulativen Bereich:

- Produkte verfügen in der Regel über interaktive Eigenschaften, welche aus Anwendersicht relevant sind, z.B. im Zusammenspiel mit einer Umgebung.
- Produkte haben oft eine Vielzahl von Ausprägungen, die sich visuell, geometrisch oder strukturell niederschlagen und die für den Endanwender im Kontext der Kauf- oder Entscheidungsfindung ebenfalls relevant sind.

Während sich einerseits industriell relevante Konfigurationstechnologien (siehe auch Abb. 1) im Kontext von ERP/PPS-Systemen (Enterprise Resource Planing-Systeme sowie Produktionsplanungs- und Steuerungssysteme) als Standards etabliert haben und andererseits eine gewisse Konvergenz im Kontext geometrischer Formate und APIs festzustellen ist (z.B. durch die zunehmend durchgängige Verfügbarkeit von WebGL [Khr14], jedoch oftmals spezialisiert nach Anwendungsgebieten wie Virtuelle Realität, Web oder mobiler Lauffähigkeit), so ist die Verbindung dieser beiden Welten (kaufmännische und geometrische) aktuell nicht befriedigend gelöst. Die bekannten Lösungen sind in einer oder mehreren Hinsichten proprietär oder haben oftmals einen starken Projektbezug. Ersteres limitiert die universelle Anwendbarkeit; letzteres beschränkt die industriellen Einsatzmöglichkeiten, die Skalierung im Kontext der Datenerstellung und erfordert dabei ein hohes Qualifikationslevel (z.T. durch Programmierer).

Basierend auf den oben genannten Anforderungen und den Nachteilen der bekannten Lösungen wurde ein neuer Ansatz mit folgenden Schwerpunkten konzipiert:

- Strukturierte, parametrische und visuelle Beschreibung von interaktiven Produkten in einem konsistenten Datenmodell
- Prinzipielle Unabhängigkeit des Datenmodells von einem konkreten 3D-Rendering-Verfahren bzw. Implementations-API und den darin verwendeten Low- Level-Geometrie bzw. -Material-Dateiformaten (ebenfalls Shader-Unabhängigkeit)
- Verwendung einer interoperablen Scripting-Technologie, die sowohl die parametrische Erzeugung beliebig komplexer Produkte erlaubt, sowie die Definition einfacher (elementare eindimensionale Objektbewegungen) wie auch umfangreicher Interaktionen (Multi-Objektbewegungen wie mehrgliedrige Türen, Strukturmanipulationen, etc.)

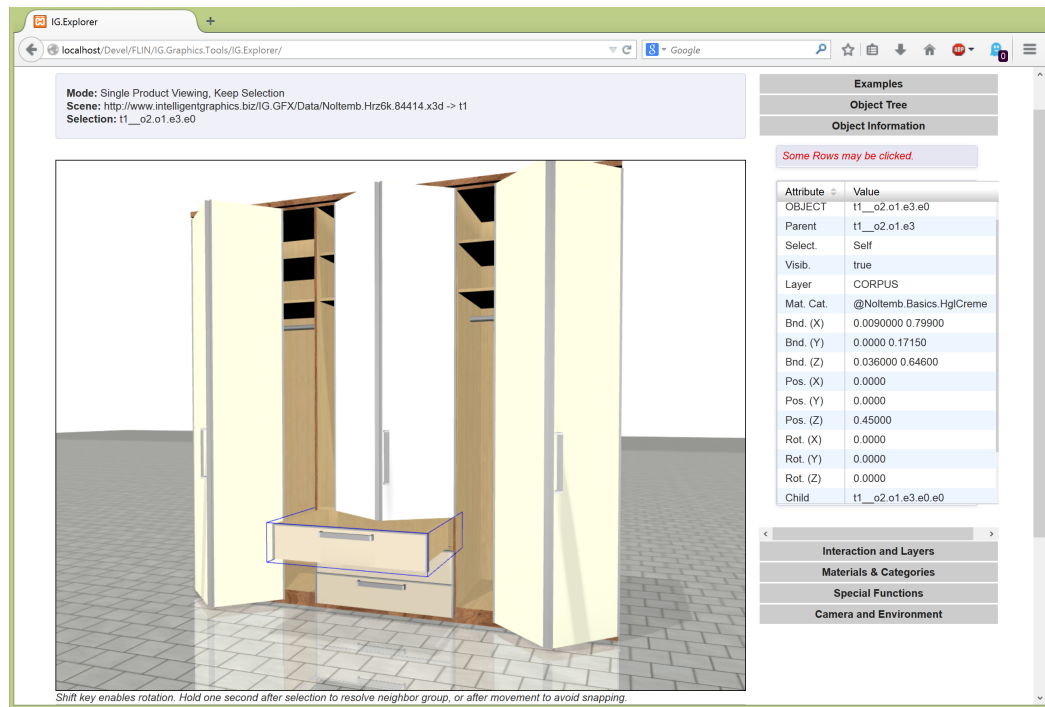


Abbildung 1: Interaktiver 3D-Explorer mit Menü zur individualisierten Produktkonfiguration.

ermöglicht. Hierbei soll ausdrücklich die Einbeziehung von Interproduktbeziehungswissen erfolgen, d.h. Regeln, die den Zustand anderer Objekte und/oder der Umgebung mit einbeziehen

- Fokussierung auf konsequenten Einsatz von Datenbank-Technologien und Erstellung einer serviceorientierten Infrastruktur

Als initiale Anwendungsdomäne soll dabei der Bereich der Möbelmodellierung dienen, da hier einerseits langjährige Erfahrungen vorliegen, andererseits sich dieser durch sehr umfangreiche Anforderungen im Bereich Parametrik und Struktur, sowie Interaktion und Interproduktabhängigkeiten auszeichnet.

Nachfolgend wird das Datenmodell vorgestellt. In Abschnitt 4 wird ein konkretes Backend auf Basis von X3DOM [BJFS12] beschrieben.

## 2 Verwandte Arbeiten

Parametrische Modelle zur Erstellung von 3D-Objekten sind im CAD-Bereich (Computer-Aided Design) schon lange bekannt. Ein erster Versuch eines einfachen macro-parametrischen Modells im Webkontext, das auf CSG-Primitiven (Box, Zylinder, Kegel, Torus usw.) und elementaren Transformationen beruht (wobei die sog. Macro-Information durch die Aufnahme bzw. Wiedergabe der Modelliersequenzen eine dynamische Objektrepräsentation definiert), wurde beispielsweise in [PML13] vorgestellt. Weiterhin gibt es inzwischen einige kommerzielle Plattformen, welche unter Verwendung aktueller W3C-Technologien webba-

sierte CAx-Systeme<sup>1</sup> anbieten, wie etwa GrabCAD [Gra14] für verteiltes Design Review oder Simscale [Sim14] zur cloud-basierten Strukturmechaniksimulation. Neben der reinen 3D-Modellierung gewinnen derzeit auch immer mehr individualisierte Online-Konfiguratoren und 3D-Produktpräsentationen für den Massenmarkt an Bedeutung [BK14].

Oftmals müssen 3D-Szenen für die Präsentation im Web, insbesondere wenn die Anwendung auch auf weniger leistungsfähiger oder gar mobiler Hardware laufen soll, noch speziell aufbereitet werden, indem z.B. ein besonders effizientes Containerformat zur Geometrieübertragung und -repräsentation genutzt wird (vgl. z.B. [BJFS12]) oder auch indem statische Teilgraphen zu einem Mesh zusammengefasst werden, um somit die Anzahl der Draw-Calls zu reduzieren. Ein geeignetes Webservice-Framework, welches automatisiert eine entsprechende Transcodierungs- und Optimierungs-Pipeline für X3DOM vorstellt, wurde etwa kürzlich in [AWJF13] vorgestellt.

X3DOM [BJFS12] ist ein JavaScript-basiertes Open-Source-Framework, das deklarative 3D-Graphik in HTML5 unter Verwendung aktueller Webtechnologien wie etwa CSS3 und WebGL [Khr14] bereitstellt. Analog zur SVG-Integration für 2D-Graphik erweitert X3DOM dabei in deklarativer Weise den HTML-DOM des Webbrowsers um 3D-Objekte, deren Knotendefinition wiederum auf den offenen ISO-Standard X3D [Web13] aufsetzt. Diese lassen sich über Standard-DOM-Scripting (z.B. via `setAttribute()`) interaktiv manipulieren, wodurch einerseits auch Webentwickler ohne Expertenwissen aus dem Graphikbereich in die Lage versetzt werden, 3D-Webanwendungen zu entwickeln, und wodurch andererseits bekannte JavaScript-Bibliotheken (wie z.B. jQuery [jQu14] oder YUI [Yah14]) einfach integriert werden können.

### 3 Parametrisches Modell

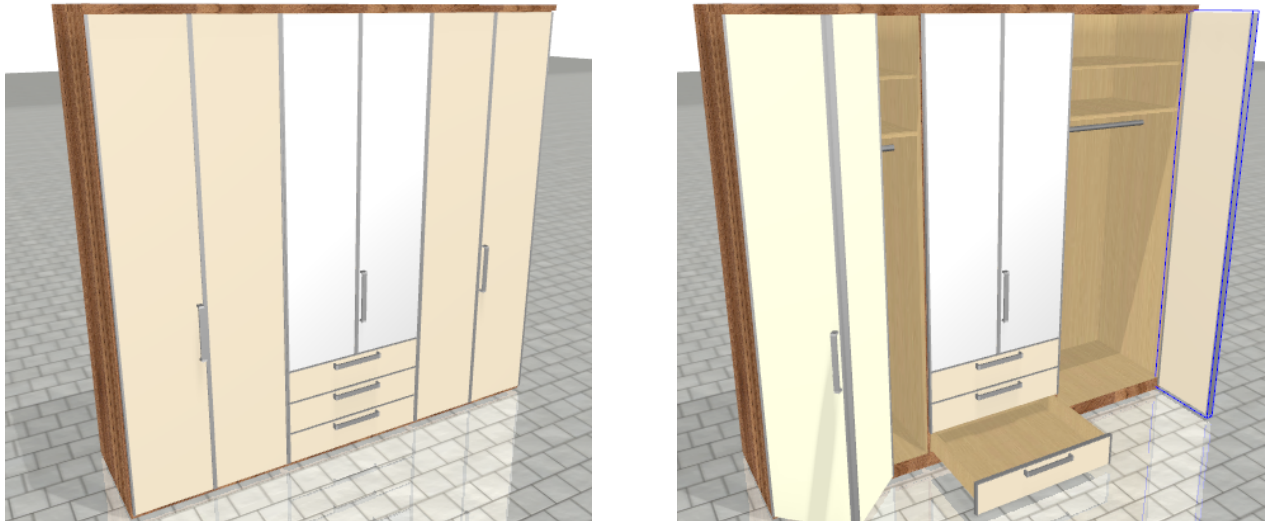
Das parametrische Datenmodell wird dazu genutzt, eine Szene dynamisch zu generieren. Diese dient der möglichst realistischen Darstellung von Produkten und zur Interaktion mit diesen. Produkte bestehen aus einer Vielzahl von Komponenten, die unter Umständen alle eine unterschiedliche Art der Interaktion benötigen, um das reale Verhalten abzubilden. Zu diesem Zweck wurden unabhängige *Interaktoren* entwickelt, die das Verhalten lediglich über die API des Graphiktreibers an das zugrundeliegende Graphiksystem weitergeben. Interaktoren sind Module, die Komponenten eines Produkts um ein bestimmtes Verhalten erweitern.

Ein Produkt wird dabei aus vielen elementaren Bauteilen zusammengesetzt. Diese Bauteile werden in Form eines Baumes strukturiert, bzw. innerhalb eines Szenengraphen strukturiert angeordnet. Aufgrund dieser Struktur können jedem elementaren Bauteil bestimmte Verhaltensweisen in Form von Interaktoren zugewiesen werden. Dies geschieht durch die Deklarationen von XML-Attributen innerhalb von XML-Tags. Allerdings ist hier zu beachten, dass nicht jedem elementaren Bauteil ein Interaktor zugewiesen ist, sondern meistens über-

---

<sup>1</sup>CAD, Computer-Aided Engineering (CAE) usw. (vgl. z.B. [VWBZ09] für eine Einführung)





**Abbildung 2:** Ein Produkt mit mehreren Interaktoren (Schubladen, unterschiedliche Türarten) zum Interagieren mit den Komponenten. Das je aktive Element wird dabei durch die blauen Linien (rechts im Bild) gekennzeichnet.

geordneten Gruppen. So besteht eine Schranktür (s. Abb. 2) beispielsweise aus mehreren Bauteilen, der Interaktor indessen wird der gesamten Gruppe zugewiesen. Beim Selektieren eines elementaren Bauteils dieser Tür wird solange im Szenengraphen nach oben iteriert bis ein Interaktor gefunden und direkt aktiviert wird. Das Verhalten wird nun auf die gesamte Gruppe angewendet.

Um herauszufinden, ob eine Interaktion auf dem entsprechenden Knoten des Szenengraphen liegt oder nicht mehr weiter aggregiert werden soll, werden die drei Parameter “Self”, “Parent” und “None” genutzt. Self bedeutet dabei, dass sich hier ein Interaktor befindet und dieser angewendet werden soll. Parent dagegen weist darauf hin, dass eine Interaktion ausgeführt werden soll, allerdings an einer Stelle weiter oben im Szenengraphen. None bestimmt, dass weder auf diesem Bauteil noch irgendwo weiter oben im Szenengraphen ein Interaktor existiert.

### 3.1 Interaktor-Konzept

Das Interaktor-Konzept basiert auf 18 elementaren Definitionen, anhand derer in Kombination annähernd alle Manipulationen umgesetzt werden können. Dabei kann man die 18 Interaktoren in die zwei Kategorien *Translation* und *Rotation* unterteilen. Für jede der beiden Kategorien existieren dabei wiederum drei verschiedene Formen der Manipulation, die jeweils auf die X-, Y- oder Z-Achse angewendet werden können:

- Range (Wertebereich, in dem Interaktion erfolgt, z.B.  $[-1.0, 1.0]$ )
- Raster (Wertebereich, bei dem in einem festen Raster interagiert wird, z.B. ein Raster von 0.5 mit vorherigem Wertebereich  $\{-1.0, -0.5, 0.0, 0.5, 1.0\}$ )

- Value (klare Definition der Werte, die angenommen werden können, wie beispielsweise  $\{-1.0, 0.0, 0.25, 1.0\}$ )

Betrachtet man dieses Konzept, so ergibt sich folgende Anzahl an Interaktoren:

2 Kategorien · 3 Ausprägungen · 3 Achsen = 18 elementare Interaktoren.

Neben den Wertebereichen, in denen ein Interaktor agieren kann, definiert er auch zusätzlich noch maximale und minimale Grenzen, sowie Snapping-Bereiche. Beispielsweise kann man die Verschiebung auf der X-Achse mittels *XRangeTranslator* auf einen Wertebereich zwischen  $[-1.0, 1.0]$  beschränken und gleichzeitig einen Snapping-Faktor von beispielsweise 0.5 angeben, sodass sich das Objekt stufenlos zwischen  $[-0.5, 0.5]$  bewegt und ab dem Überschreiten dieser Grenzen auf die jeweiligen Wertebereichsgrenzen springt. Dadurch lassen sich Einrastfunktionen oder ähnliches umsetzen.

Alle erwähnten 18 Interaktoren dienen der Einzelobjektmanipulation, was bedeutet, dass jeder einzelne auf ein einzelnes Objekt wie eine Schranktür, eine Schublade, eine Autotür usw. angewendet werden kann. Mit diesen elementaren Interaktionen lassen sich eine Vielzahl von Anwendungsfällen einfach und ohne spezielle Programmierung lösen. Durch Erweiterungen und Kombinationen der elementaren Interaktoren und unter Verwendung des abstrakten Szenengraphen-APIs sind auch komplexe Multi-Objekt-Interaktionen möglich. Beispielsweise muss bei einer Schrank-Klapptür eine Rotation des linken Türteils in eine bestimmte Richtung und gleichzeitig eine Translation erfolgen, während der rechte Türteil in die entgegengesetzte Richtung rotieren muss. Eine solche Interaktion ist in Abb. 2 dargestellt.

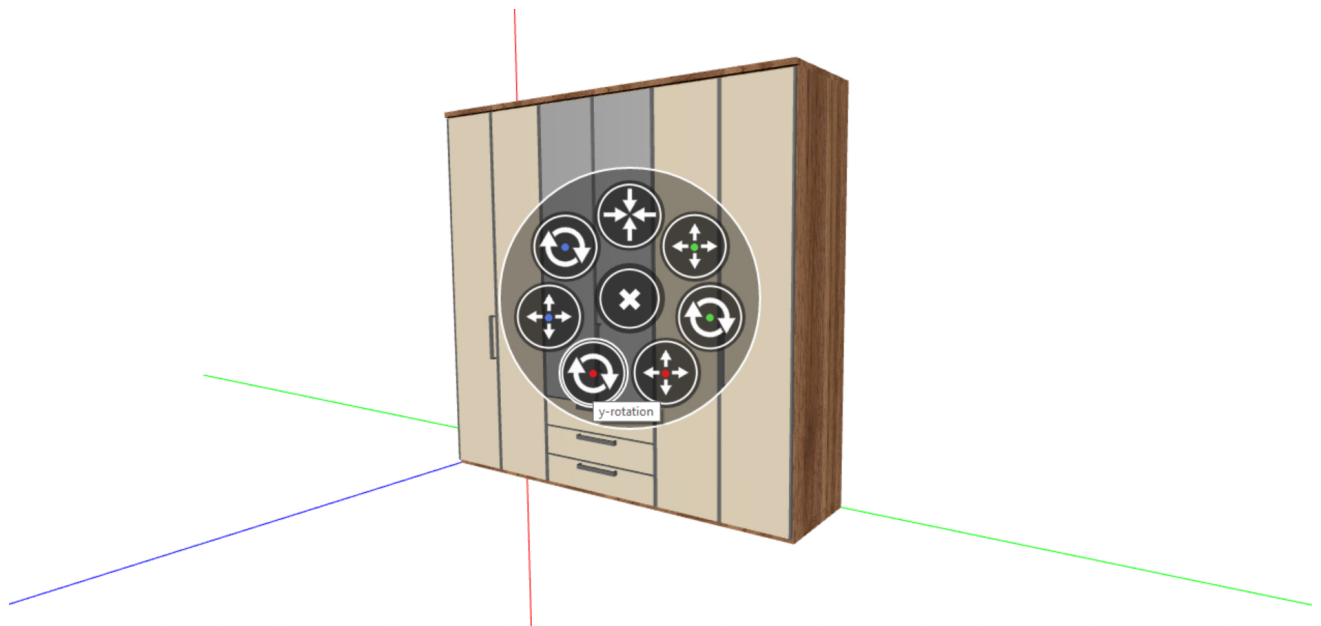
Des Weiteren existiert eine darauf aufbauende User-Interface-Komponente (als Manipulator bezeichnet), mit der auf Basis der Interaktoren mehrere Manipulationen auf ein und dasselbe Produkt oder Bauteil angewendet werden können. Dazu muss das entsprechende Bauteil definieren, welche Aktionen auf ihm ausgeführt werden können und der Manipulator wird dann on-the-fly mit den nötigen Schaltflächen ausgestattet. Beispielsweise kann man für einen gesamten Schrank den Manipulator so definieren, dass er in allen drei Richtungen rotieren und verschieben können soll (siehe Abb. 3). Im Gegensatz zu den Interaktoren, deren Semantik bereits im Rahmen des Modells integriert ist, sind die Manipulatoren noch in der experimentellen Phase und deren modellseitige Integration ist aktuell noch offen.

Durch die Verwendung des X3D-basierten X3DOM-Frameworks stellt sich an dieser Stelle noch die Frage, inwieweit alternativ die Nutzung standardisierter VRML-/X3D-Konzepte (insbes. die Pointing-Device-Sensor-Komponente [Web13]) sinnvoll wäre. Auch wenn in Version 1.6.1 mittlerweile zumindest einige der Pointing-Sensoren rudimentär implementiert sind, so gehört diese Komponente aus mehreren Gründen nicht zum vorgeschlagenen HTML-Profil<sup>2</sup>. Tatsächlich ist das X3D-Sensorkonzept einerseits für Webentwickler wenig zugänglich; stattdessen bietet X3DOM hier die um zusätzliche 3D-Eventinformationen ergänzten Standard-DOM-UI-Events an (z.B. “onmousemove” und “onclick”).

Andererseits bringen sensorbasierte Gizmos, wie in [MPJ<sup>+</sup>14] beschrieben, zusätzliche Probleme mit sich: Um beispielsweise die Unmenge an benötigten ROUTEs, inklusive hier-

---

<sup>2</sup>[http://www.x3dom.org/?page\\_id=158](http://www.x3dom.org/?page_id=158)



**Abbildung 3:** Manipulator, der dynamisch verschiedene Interaktoren auf Produkte anwenden kann.

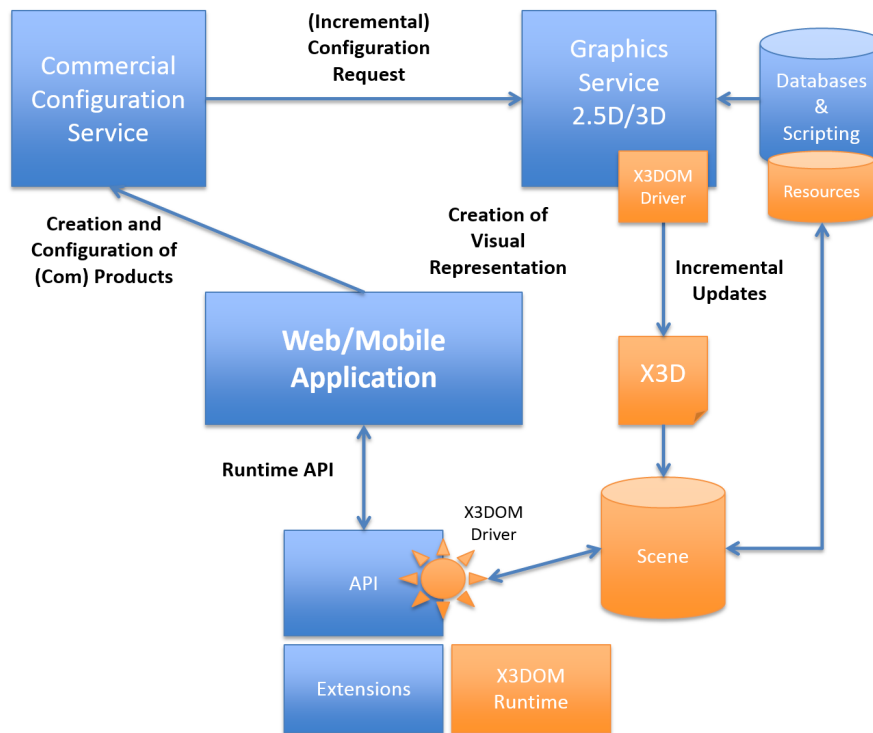
bei notwendiger Wertadaptation, einigermaßen handhabbar zu halten, würde man PROTOs mit internen ROUTEs und Script-Nodes benötigen, welche in X3DOM nicht vorhanden sind. Während der VRML-/X3D-Prototypenmechanismus schwer im DOM-Umfeld abbildbar ist und darüber hinaus die X3DOM-Implementierung um einen wesentlichen Faktor aufblähen würde, wäre ein spezieller Script-Knoten, mit X3D-spezifischem Eventhandling und Datentypen, zudem recht widersprüchlich zum in HTML bereits definierten `<script>`-Tag.

### 3.2 Systemarchitektur

Um den konzeptionellen Schwerpunkten und Anforderungen aus Abschnitt 1 gerecht zu werden, wurde ein dynamisches parametrisches Modell in einer verteilten Architektur konzipiert. Abbildung 4 stellt einen Überblick dieser Architektur dar, die nachfolgend beschrieben wird.

Im Mittelpunkt der Architektur fungiert die *Web/Mobile Application* als Schnittstelle zwischen Anwender und dem System. Auf der einen Seite ermöglicht dies den Zugriff auf das parametrische Modell zur Erstellung und Konfiguration von Produkten über den *Commercial Configuration Service*. Auf der anderen Seite dient es der Visualisierung von Szenenarrangements mit Produkten in einer webbasierten und plattformunabhängigen Anwendung. Dies wird über eine *Runtime API* erreicht (Abb. 4, Mitte), welche die notwendigen Interaktionskonzepte implementiert, so dass das reale Verhalten eines Produktes innerhalb der Webanwendung genutzt werden kann.

Der *Graphics Service 2.5D/3D* benötigt für die Erstellung von Produkten granulare, parametrische und visuelle Beschreibungen aller vorhandenen Bauteilkomponenten, aus denen dann konkrete Konfigurationen unter Auswertung kaufmännischer Beschreibungen abgeleitet werden. Diese Komponenten liegen als wiederverwendbare, universelle oder spezifische Ressourcen vor (*Databases & Scripting*). Im Ergebnis werden in der *Web/Mobile Application* Produkte eingefügt, welche anschließend zu komplexen Planungen arrangiert werden können.



**Abbildung 4:** Überblick über die Systemarchitektur mit High-Level-Komponenten (blau) und Präsentationsteil (orange; hier exemplarisch mit X3DOM umgesetzt).

Auch die dafür notwendigen Informationen sind Teil des hier vorgestellten Datenmodells.

Das parametrische Modell zur dynamischen Beschreibung von Produkten wird dabei in den *Databases & Scripting* Ressourcen abgebildet und durch unterschiedliche Datenbanken realisiert. Dabei werden Asset- und Produktdatenbanken unterschieden. Asset-Datenbanken verwalten Produktkategorien, Geometrien, Materialien, Ebenen etc. und beschreiben damit die Grundelemente zur Erzeugung von Produkten. Product Datenbanken beschreiben Produkte aus diesen Assets und speichern darüber hinaus deren konkretes physisches Verhalten in Form von Parametern, die Komponentenbeschränkungen und Interaktionsmöglichkeiten definieren. Diese strukturierte, parametrische und visuelle Beschreibung von interaktiven Produkten in einem konsistenten Datenmodell unter Berücksichtigung von kaufmännischen Korrektheits- und/oder subjektiven ästhetischen Aspekten ermöglichen eine hochflexible und dynamische Produkterstellung.

Änderungen an einzelnen Parametern eines Produktes und seiner Komponenten resultieren innerhalb der Architektur in Konfigurationsanfragen an den *Commercial Configuration Service*, der diese Änderungen anwendet und über den *Graphics Service 2.5D/3D* visualisiert. Hierbei kommt es entweder zu einem geometrischen Roundtrip, welcher durch client-seitiges Caching optimiert wird oder zu einem nicht-geometrischen Material-Update, welches sehr performant ausgeführt werden kann. Geometrische und visuelle Ressourcen unterliegen hierbei einem zentralen Namensschema, welches letztlich übergreifend asynchrones Laden, Sharing und Caching erlaubt. Somit ergeben sich aus praktischer Sicht hinreichende Optionen für das effiziente, interaktive Arbeiten ohne jedoch die inkrementellen Auswirkungen



Abbildung 5: Veränderung der Materialeigenschaften von Teilen eines Produkts.

von Merkmalsänderungen modellseitig erfassen zu müssen.

Anmerkung: Um eine effiziente Implementierung interaktiver Änderungen zu erlauben, wird aus den Datenbanken im Rahmen eines versionierten Deployments eine laufzeitoptimierte Datenstruktur exportiert, auf welcher letztlich der *Graphics Service 2.5D/3D* arbeitet.

## 4 Beispielhafte Umsetzung mit X3DOM

Zur Umsetzung der beschriebenen Aspekte wurde ein abstraktes, interaktives Szenengraph-API entwickelt, welches lediglich einen JavaScript-Stack erfordert. Dieses API wurde zunächst mittels X3DOM implementiert (sog. *X3DOM-Treiber*). Wie schon in Abschnitt 2 kurz angerissen, erlaubt X3DOM – im Gegensatz zu VRML/X3D – die Einbindung von 3D-Inhalten in HTML-Dokumente ohne die Verwendung spezieller Plugins (welche typischerweise je für eine gegebene Plattform entwickelt werden müssen und zudem immer gewisse Sicherheits- und Installationsprobleme mit sich bringen). Durch die direkte Integration in den HTML-DOM lässt sich die 3D-Szene zudem einfach mit Hilfe von Standard-DOM-Scripting, wie es jeder Webentwickler kennt, verändern, wodurch auch die Verwendung eines Plugin- bzw. X3D-spezifischen Interfaces wie EAI (VRML97 External Authoring Interface) resp. SAI (X3D Scene Access Interface<sup>3</sup>) entfällt. Der prinzipiell abstrakte Datenansatz soll dabei die Unabhängigkeit der in den Datenbanken und Scripts realisierten Daten, wobei es sich in der Regel um immense Investitionen handelt, mit Hinblick auf unterschiedliche Laufzeitumgebungen und -plattformen sicherstellen.

Darüber hinaus unterstützt das X3DOM-Backend die Visualisierung verschiedener Materialien an einem Produkt bzw. dessen Einzelkomponenten. Beim Erstellen der Produkte

<sup>3</sup><http://www.web3d.org/documents/specifications/19775-2/V3.3/index.html>

werden Materialkategorien an die einzelnen Komponenten vergeben. Diese Kategorien betreffen eine oder mehrere Komponenten eines Produkts und können verwendet werden, um gezielt das Material eines Produkts bzw. dessen Teilkomponenten zu beeinflussen. In Abbildung 5 wird ein Schrank dargestellt, dessen Korpus (Innen- und Außenseite) neue Materialeigenschaften zugewiesen bekommt. Der X3DOM-Treiber ermöglicht eine realistische und interaktive Produktdarstellung. Gleichzeitig wird eine Vielzahl von Endgeräten unterstützt, da X3DOM auf offenen Standards wie X3D [Web13], HTML5 und WebGL aufbaut, deren Unterstützung durch die Browserhersteller weiterhin steigt.

Jedes Produkt verwendet für den strukturellen Aufbau genau drei verschiedene Typen von X3D-Knoten. Zum einen handelt es sich um das Inline-Element, welches ein tatsächliches Bauteil per URL referenziert und später dann auch in der Szene sichtbar ist, z.B. ein Brett für die Schranktür. Um jedes Inline-Element wird wiederum ein Transform-Element gehängt, um das Bauteil verschieben, rotieren oder auch skalieren zu können. Als letztes X3D-Bauteil werden noch einfache Group-Elemente verwendet, um mehrere Inline-Elemente effizient zu Gruppen zusammenzufassen. Somit kann man sowohl auf einzelne Bretter, als auch auf Gruppen Interaktoren anwenden.

## 5 Zusammenfassung und Ausblick

Die interaktive Präsentation von großen Produktpaletten wie beispielsweise von konfigurierbaren Möbelkombinationen ist in der traditionellen Vorgehensweise insbesondere durch die Beschreibung der vorkommenden Interaktionsmöglichkeiten sehr aufwendig und benötigt IT-Expertenwissen. In dem hier vorgelegten Beitrag wird ein Weg aufgezeigt, wie dies mit Hilfe des vorgestellten Datenmodells unter Einsatz von Datenbanktechnologien automatisiert werden kann. Weiterhin wurde eine webfähige und plattformunabhängige Präsentationskomponente vorgestellt.

Der Schwerpunkt der bisherigen Konzeption und Implementierung lag im Bereich der Abbildung notwendiger Funktionalitäten. Großes Potential in Bezug auf Performancegewinne wird in der Optimierung insbesondere der Geometriebeschreibung gesehen, um die Anzahl der Graphiktreiber-Aufrufe zu minimieren. Hierfür erscheint (wie in [AWJF13] und [MPJ<sup>+</sup>14] vorgeschlagen) eine Zusammenfassung der Geometriehierarchien, jedoch in Abhängigkeit der verwendeten Interaktoren, vielversprechend, um die aufgebauten Technologien auch auf komplexere Produkte wie z.B. in der Automobilindustrie verwenden zu können.

## Literatur

- [AWJF13] ADERHOLD, A., K. WILKOSINSKA, Y. JUNG und D. FELLNER: *Distributed 3D Model Optimization for the Web with the Common Implementation Framework for Online Virtual Museums*. In: *Proceedings Digital Heritage 2013*, Band 2, Seiten 719–726. IEEE, 2013.

- [BJFS12] BEHR, J., Y. JUNG, T. FRANKE und T. STURM: *Using images and explicit binary container for efficient and incremental delivery of declarative 3D scenes on the web*. In: *Proceedings Web3D 2012*, Seiten 17–25, New York, USA, 2012. ACM Press.
- [BK14] BEIER, EKKEHARD und HENRY KURZ: *Consumer Focused Interactive 3D Visualization, based on SAP IPC / VC*, 2014. European Configuration Work Group (CWG) Conference (Leipzig).
- [Gra14] GRABCAD: *GrabCad Homepage*, 2014. <http://grabcad.com>.
- [jQu14] JQUERY FOUNDATION: *jQuery*, 2014. <http://jquery.com/>.
- [Khr14] KHRONOS: *WebGL Specification*, 2014. <https://www.khronos.org/registry/webgl/specs/latest/1.0/>.
- [MPJ<sup>+</sup>14] MOUTON, CHRISTOPHE, SAMUEL PARFOURU, CLOTILDE JEULIN, CECILE DUTERTRE, JEAN-LOUIS GOBLET, THOMAS PAVIOT, SAMIR LAMOURE, MAX LIMPER, CHRISTIAN STEIN, JOHANNES BEHR und YVONNE JUNG: *Enhancing the Plant Layout Design Process using X3DOM and a Scalable Web3D Service Architecture*. In: *Proceedings Web3D 2014*, New York, USA, 2014. ACM Press.
- [PML13] PAVIOT, THOMAS, CHRISTOPHE MOUTON und SAMIR LAMOURE: *Long Term Control of 3D Engineering Data for Nuclear Power Plants*. In: *Proceedings Web3D '13*, Seiten 205–205, New York, NY, USA, 2013. ACM.
- [Sim14] SIMSCALE: *Simscale Homepage*, 2014. <http://www.simscale.de>.
- [VWBZ09] VAJNA, S., C. WEBER, H. BLEY und K. ZEMAN: *CAx für Ingenieure – Eine praxisbezogene Einführung*. Springer, 2. Auflage, 2009.
- [Web13] WEB3D CONSORTIUM: *X3D Specification*, 2013. <http://www.web3d.org/files/specifications/19775-1/V3.3/Part01/Architecture.html>.
- [Yah14] YAHOO: *YUI*, 2014. <https://yuilibrary.com/>.





# Entwicklung einer gestenbasierten Steuerung und eines virtuellen Tenor-Avatars für eine interaktive Medieninstallation mit Gesangssynthese

Jochen Feitsch, Marco Strobel, Chris Geiger

FH Düsseldorf

Josef-Gockeln-Str. 9

40474 Düsseldorf

E-Mail: {jochen.feitsch,marco.strobel,geiger}@fh-duesseldorf.de

**Abstract:** In diesem Beitrag wird eine interaktive Medieninstallation vorgestellt, die es dem Benutzer ermöglicht sich selbst als Tenor in einem virtuellen Opern-Setting der 20er Jahre zu erleben und als solcher zu singen ohne selbst singen zu müssen. Dabei werden Körpergesten und Mundbewegungen erkannt und analysiert und daraus Gesang erzeugt sowie ein virtueller 3D-Avatar gesteuert. Ziel ist es, eine unterhaltsame und glaubwürdige Nutzererfahrung zu erzeugen, sowohl für den gegebenenfalls musikalisch unerfahrenen Benutzer, als auch für das Publikum. Hierzu werden 3D Ganzkörpertracking mit Gesichtstracking kombiniert, multimodale Interaktions- und Feedbackmethoden verwendet, ein 3D Avatar inklusive Gesichtsmorphing in Echtzeit animiert, und eine Gesangsstimme synthetisiert. Das System wurde vorläufig evaluiert und ist im aktuellen Zustand vornehmlich als Unterhaltungsanwendung einzustufen.

**Keywords:** 3D Eingabegeräte und Interaktionstechniken, Displaytechnologien und Tracking, Avatare und Agenten, Echtzeit-Rendering, Multimodale Interaktion, Human Factors, Innovative Anwendungen, Entertainment/Edutainment, Künstlerische Anwendungen

## 1 Einleitung

Singen ist eine grundlegende und sehr alte Form des menschlichen Ausdrucks. Seit es Menschen gibt, wurde die Stimme verwendet um Klänge zu erzeugen. In der heutigen Kultur wird dem eigenständigen Singen jedoch eine immer geringere Bedeutung zugemessen und wenige Menschen sind befähigt auf höherem Niveau selbst zu singen.

Nachfolgend wird angenommen, dass ein (stiller) Bedarf nach dieser grundlegenden menschlichen Fähigkeit besteht und danach mit dieser auf spielerische Art und Weise zu experimentieren. Dies stellt auch die Motivation der hier präsentierten Arbeit dar: eine unterhaltsame Erfahrung mit dem „Instrument Gesang“ zu bieten.

Forschungsergebnisse im Bereich der Musikpsychologie zeigen, dass die visuelle Darstellung bei Aufführungen wichtig für die Wahrnehmung des musikalischen Ausdrucks ist [BW11]. Überspitzt könnte man sagen, dass beim Hören von Musik das Auge in gewisser

Weise mithört. Auch deshalb fokussiert diese Arbeit nicht auf die realistische Nachbildung des menschlichen Gesangsapparats. In einem komplexen interaktiven audiovisuellen Aufbau soll dem Benutzer vielmehr eine Schnittstelle geboten werden, um sich selbst in die Rolle einer singenden Person versetzen zu können. Hierzu schlüpft er in sein eigenes virtuelles Tenor-Ebenbild und wird befähigt, dieses durch Bewegung und Mimik zu steuern. Mit Hilfe der Gesangssynthese kann dieser virtuelle Tenor zum Singen gebracht werden. Der Benutzer singt mit einer Stimme, die nicht seine eigene ist, was sowohl für ihn, als auch für das Publikum eine unterhaltsame Erfahrung darstellt.



Abbildung 1: Aufbau des Systems

## 2 Related Work

Die Synthese von Gesang ist ein etablierter Forschungsbereich mit langer Tradition und das menschliche Gesangsorgan ist ein komplexes und nicht trivial zu synthetisierendes Instrument. Leider hat die Gesangsstimmensynthese bisher weniger wissenschaftliche Aufmerksamkeit erhalten als die Sprachsynthese.

Sundberg beschreibt 2006 die aktuell bekannten Grundlagen für diesen Forschungsbereich [Sun06]. Systeme, welche die menschliche Stimme als Eingabe verwenden, werden Gesang-zu-Gesang Synthesizer (voice-to-voice) genannt. Eines dieser Systeme ist Vocalistener2, welches eine menschliche Stimme synthetisieren kann indem es eine menschliche Stimme als Eingabe verwendet; Tonhöhe, Dynamik, Phoneme und Klangmuster werden analysiert und basie-

rend auf diesen Mustern wird eine Gesangsstimme erzeugt. In einer aktuelleren Publikation präsentierten die Autoren VocaWatcher, der eine realistische Mimik eines menschlichen Sängers generieren und diese zur Steuerung des Gesichts eines humanoiden Roboters nutzen kann [GNK<sup>+</sup>12]. Dies ist nach unseren Recherchen eines der ersten Systeme, das einen Fokus auf das Imitieren eines realen Sängers basierend auf Akustik und visuellen Ausdrücken legt. Cheng und Huang haben 2010 einen fortgeschrittenen Mundtrackingansatz vorgestellt, welcher Echtzeit-Mundtracking und eine 3D Repräsentation der Bewegung in Echtzeit kombiniert [CH10].

Cano et al. beschreiben ein System, das es erlaubt zwischen der Gesangsstimme des Nutzers und der Stimme eines professionellen Sängers zu wandeln [CLB<sup>+</sup>00]. Dieses System wurde bei Karaoke-Auftritten genutzt.

Das nicht-triviale Abbilden von Gesten auf Sound stellt eine echte Herausforderung dar wenn die Anzahl der Dimensionen von Sensorendaten hoch ist. Fasciani et al. [FW13] nutzt daher den Ansatz eines künstlichen neuronalen Netzwerkes um diese Abbildung zu vereinfachen und die Usability ihres durch die Stimme kontrollierten Interfaces zu erhöhen.

Vergleichbar zu unserem Ansatz ist das „Artificial Singing“ Projekt, ein Gerät, das einen Gesangssynthesizer durch Mundbewegungen steuert, die von einer Webkamera erkannt werden [LHT03]. Der Mund des Nutzers wird verfolgt und Mundparameter wie Höhe, Breite, Öffnung und Rundheit werden auf die Parameter eines Synthesizers für Tonhöhe, Lautstärke, Vibrato und Ähnliches abgebildet.

Bisher existiert kaum kommerzielle Software in diesem Bereich. Yamaha entwickelte Vocaloid ([www.vocaloid.com](http://www.vocaloid.com)), einen Gesangsstimmensynthesizer der Liedtext und Melodie als Eingabe verwendet und daraus eine Gesangsstimme erzeugt. Die Software hatte großen Erfolg in Japan und einige Songs wurden damit produziert. Vocaloid besteht aus einem Editor und einer Reihe von Stimmen mit verschiedenen Eigenschaften.

Durch die Microsoft Kinect ist die Steuerung von Medien über Echtzeit-Interaktion durch den ganzen Körper für viele Menschen zugänglich geworden. Vermehrt in Spielen (wie „Let’s Sing and Dance“, „Just dance“ uvm.) gibt es Ansätze zur Gesangs- oder Bewegungsanalyse, die sich aber meist nur auf den Analysebereich zur Bewertung der Performance beschränken. In anderen Anwendungen (wie beispielsweise im Spiel „Kinect Adventures!“) kann ein vereinfachter Avatar mehr oder weniger direkt in einer 3D Szene gesteuert werden. Hierbei wird oft nur die relative Bewegung zur Steuerung ohne direkte Auswertung betrachtet. Sony Online Entertainment (SOE) hat in den letzten Jahren ein System zur Abbildung der eigenen Mimik auf einen Charakter veröffentlicht (SOEmote: [www.soe.com/soemote/guide/index.vm](http://www.soe.com/soemote/guide/index.vm)). Dieses ermöglicht es soziale Interaktionen mit anderen Spielern in SOE’s Online-Spielen (speziell „Everquest 2“ und in Zukunft „Everquest Next“) in Echtzeit um die eigene Mimik zu erweitern.

### 3 Systemübersicht

Die Installation besteht aus einer 4x4 Videowand mit 46" Monitoren. Für das Tracking kann das sehr exakte Bewegungstrackingsystem MyoMotion von Noraxon ([www.noraxon.com](http://www.noraxon.com)) verwendet werden oder eine Microsoft Kinect 2. Das Gesichtstracking wird mittels einer PrimeSense Carmine (oder einer weiteren Kinect) realisiert. Der Anwender trägt einen Frack, weiße Handschuhe und einen Zylinder um die Erfahrung ein Opersänger aus den 20er Jahren zu sein zu verstärken. Bei Nutzung des MyoMotion Trackingsystems trägt der Anwender zusätzlich fünf portable Sensoren: jeweils einen pro Oberarm, einen pro Unterarm und einen Sensor am Rücken als Ursprungsrotation des Koordinatensystems.

Der Zylinder ist mit einem Knochenschallkopfhörer ausgestattet um die generierte Gesangsstimme „im Kopf“ des Benutzers zu platzieren. Dabei werden Umgebungsgeräusche nicht, wie bei herkömmlichen Kopfhörern, blockiert. Dies ist wichtig um die Hintergrundmusik noch wahrnehmen zu können. Zusätzlich wird ein Latz angelegt an dem mehrere Exciter<sup>1</sup> und Vibrationsmodule angebracht sind um die Vibration der Gesangsstimme auf den Brustkorb des Nutzers zu übertragen (siehe Abbildung 2).

Die Nutzung von Knochenschallkopfhörern und Vibrationsfeedback unterstützt die Illusion selbst zu singen. Die während der Aufführung abgespielte Hintergrundmusik wird zu Beginn durch das Auflegen einer Schallplatte auf ein altes Grammophon gewählt. Das Grammophon dient als Resonanzkörper für einen Exciter der per Computer für die Audioausgabe angesteuert wird. Auf den Schallplatten angebrachte RFID Marker werden durch einen am Grammophon angebrachten RFID Reader ausgelesen und bestimmen so die gesungene Arie (siehe Abbildung 2). Zusätzlich sind am Grammophonarm und an einem der Handschuhe einfache Kontaktschalter angebracht. Beim Auflegen des Grammophonarms wird der Kontakt geschlossen und der Aufführungsmodus, sowie die Hintergrundmusik, werden gestartet. Die Steuerung erfolgt über einen Arduino Nano, der über Bluetooth mit dem System verbunden ist. Der Schalter auf dem Handschuh wird geschlossen wenn Daumen und Zeigefinger zusammengeführt werden und reguliert welches der beiden Vokalsets über das Gesichtstracking angesprochen wird (siehe Abschnitt „Gesichtstracking und Kalibrierung“). Eine ausführliche Beschreibung aller verwendeten Interaktions- und Feedbackmodule beschreiben wir in [FSMG14].

Die Verarbeitung wird auf zwei über Netzwerk verbundenen Computern ausgeführt. Ein Rechner ist für die Audio Hardware, das Gesichtstracking und das Synthesizer Modul zuständig, während der andere für die Hauptapplikation, das Ganzkörpertracking und das Rendering verantwortlich ist. Der Computer, der die Hauptapplikation ausführt, verarbeitet alle Trackingdaten und nutzt diese um einen vom Anwender gesteuerten Opersänger Avatar vollständig in 3D zu animieren. Die Übertragung der Daten an das Synthesizer-Modul erfolgt über OSC (Open Sound Control).

In einem Vorverarbeitungsvorgang kann der Kopf des Tenor-Characters auf den Benutzer

---

<sup>1</sup>Gerät, das normalerweise zur Anregung von Platten oder anderen Schwingungskörpern zur Signalausgabe verwendet wird.



Abbildung 2: Vibrationsweste, Knochenschallkopfhörer und Grammophon

angepasst und modifiziert werden. Die Bewegung der Arme und das Formen von Vokalen mit dem Mund steuert sowohl die Bewegungen des Avatars und erzeugt gleichzeitig den Gesang, wobei hier nur Vokale synthetisiert werden. Ziel ist es, je nach gewähltem Song frei zu singen oder die Originalmelodie des Liedes so gut wie möglich nachzusingen. Je besser das gewählte Lied nachgesungen wird, umso mehr nimmt der virtuelle Character das Aussehen eines bekannten klassischen Tenors an. Für die Bewertung ist hierbei nur die Tonhöhe wichtig, welche durch die Arme reguliert wird; die Vokale der Töne können frei gewählt werden um eine gewisse Individualität zu ermöglichen und die Komplexität der Interaktion zu minimieren.

## 4 Visualisierung

Zusätzlich zu haptischem und akustischem Feedback ist eine visuelle Präsenz wichtig um die Erfahrung eines Operauftrittes zu vervollständigen.

Ein Ziel des Systems ist es, den Nutzer zu integrieren und ihn die Aufführung multisensorisch erfahren zu lassen. Ein Bestandteil dieses individuellen Erlebnisses ist es, einen Avatar zu generieren der den Nutzer visuell widerspiegelt. Hierfür kann zu Beginn das Gesicht des Avatars auf den Nutzer angepasst und durch weitere Parameter modifiziert werden. Um dies zu ermöglichen wurden das FaceGen SDK ([www.facegen.com](http://www.facegen.com)) und eine Benutzeroberfläche zur einfachen Individualisierung implementiert. Mit einer HD Kamera kann der Nutzer ein zweidimensionales Foto seines Gesichts machen und in einem semi-automatischen Kalibrierungsprozess elf Marker für Gesichtsmerkmale darauf platzieren um die Gesichtsstruktur zu identifizieren. Wenn alle Marker gesetzt sind beginnt das System ein 3D Mesh zu erstellen, welches das Gesicht des Nutzers in Form und Farbe repräsentiert. Dieser Kopf wird dann auf das vordefinierte Model eines Tenorkörpers gesetzt. Nach diesem Vorgang kann der Nutzer weiter das Aussehen seines Avatars über mehrere Reglern anpassen. Hierbei können Alter, Geschlecht und ethnische Parameter (afro-amerikanisch, ost-asiatisch, süd-asiatisch oder europäisch) angepasst werden.

Der Anwender steht in Form seines Avatars im Rampenlicht auf einer Bühne in einem großen, abgedunkelten, modellierten Opernsaal. Der Avatar wird animiert um die Bewegungen des Anwenders, sowie dessen aktuelle Gesichtszüge in Echtzeit zu repräsentieren (siehe

Abschnitt „Usertracking“). Während der Aufführung wird ein schwarz-weiß Kamerafilter angewendet welcher alte Kameras über Farblosigkeit, Körnung und Flecken simuliert um das Setting der 20er Jahre nachzuempfinden.

Zusätzlich zum freien Singen können auch vordefinierte Arien nachgesungen werden. Um das Treffen der richtigen Tonhöhe und das Erlernen der Musikstücke zu vereinfachen wurde ein einfaches zeitbasiertes Anzeigesystem entwickelt welches ausstehende Tonhöhenwechsel ein paar Sekunden im Voraus anzeigt. Diese werden in Form von Pfeilen nach unten beziehungsweise oben dargestellt, erscheinen am rechten Bildschirmrand und bewegen sich von dort zum linken Rand. Sie sind so justiert, dass der Tonwechsel geschieht sobald ein Balken am linken Rand erreicht wird. Außerdem wird an diesem Balken eine Korrekturhilfe des aktuell gesungenen Tones dargestellt, welche über ein Symbol für jeweils „zu hoch“, „zu tief“ oder „korrekt“ gekennzeichnet ist.

## 5 Usertracking

Für das Gesichtstracking kann entweder eine PrimeSense Carmine oder eine Microsoft Kinect genutzt werden ohne eine Änderung am System vorzunehmen. Die PrimeSense Carmine 1.09 ist jedoch die geeignetere Wahl aufgrund ihrer besseren Eigenschaften bei nahen Aufnahmen. Für das Ganzkörpertracking wird meistens das Noraxon MyoMotion System verwendet, ein portables, kabelloses und erweiterbares System bestehend aus zwei bis 36 IMU (Inertial Measurement Unit) Sensoren. Dieses liefert dreidimensionale Orientierungsdaten der Gelenke. In früheren Iterationen wurde stattdessen eine weitere Kinect-artige Kamera verwendet. Hierbei entstanden jedoch Probleme, da sich die Infrarotmuster der Sensoren für Gesichtstracking und Ganzkörpertracking störten. Eine Möglichkeit diese Störung zu minimieren ist, das Sichtfeld der beiden Sensoren zueinander so gut wie möglich abzuschirmen. Zusätzlich kann über die „Shake’n’Sense“ [BIH<sup>+</sup>12] Methode eine künstliche Bewegungsunschärfe erzeugt werden, um Übersprechen der beiden Infrarotmuster zu unterdrücken. Die Variante des IMU Systems ist jedoch erheblich schneller und präziser als Tiefen-Sensoren, im Gegenzug bieten sie jedoch keine Möglichkeit der Abbildung einer Raumposition.

In der aktuellen Iteration wurde das neue Microsoft Kinect 2 System getestet. Erste Tests waren hierbei vielversprechend: weder der Einsatz eines anderen Tiefenbildsensors, noch kleinere Verdeckungen störten das Trackingverhalten der Kinect 2 stark. Bis das System stabil implementiert werden kann, sind noch weitere Tests notwendig – aktuell spricht jedoch nichts gegen den Wechsel auf die Kinect 2 zum Ganzkörpertracking.

Das Gesichtstracking-System setzt faceshift ([www.faceshift.com](http://www.faceshift.com)) ein, eine Software deren Ursprungszweck im markerlosen Gesichts Motion Capturing für Character Animationen liegt. Hierbei werden Gesichtstracking-Daten generiert, welche auf einem 3D Benutzerprofil basieren, das für jeden Benutzer vorher angelegt werden muss (siehe Abbildung 3). Dieser Kalibrierungsprozess dauert fünf bis zehn Minuten, in denen verschiedene Gesichtsausdrücke des Nutzers aufgenommen werden und darauf basierend anschließend das Profil des Nutzers berechnet wird.



Abbildung 3: Training des Gesichtstrackings

Sobald faceshift kalibriert wurde folgt ein zusätzlicher Kalibrierungsschritt, der die Genauigkeit der Vokalerkennung weiter erhöht. Dies ist nötig, da nicht jeder Mensch zur Erzeugung des gleichen Lauts auch die gleiche Mundform annimmt. Das System speist ein eigens dafür entwickeltes neuronales Netzwerk über einen interaktiven Prozess mit Nutzerdaten. Hierfür nimmt der Benutzer für mehrere Sekunden unter Bewegung seines Kopfes die entsprechende Mundform an. Es gibt vier Mundformen, die für das neuronale Netzwerk trainiert werden müssen: „A“, „E“, „O“ und der geschlossene Mund. Auf ein zweites Vokalset, welches die gleichen Mundformen nutzt, kann über eine Handgeste umgeschaltet werden (siehe Abbildung 4).

Nach dem Kalibrierungs-Schritt kann in den Afführungs-Modus gewechselt werden in dem das Tracking aktiv gestartet wird. Tracking Daten beinhalten Kopfposen, Armgesten und Körperposition, Blendshapes der Gesichtsmerkmale (auch Koeffizienten genannt), Blickrichtung und optionale manuell festgelegte virtuelle Marker.



Abbildung 4: Bilden der sechs möglichen Vokale

Das Ganzkörper-Skelett-Mapping nutzt die zur Verfügung stehenden Gelenkdaten des Körper-Trackings um diese in Echtzeit auf den Körper des Avatars abzubilden. Weiter werden Kopfpose, Gesichtsmerkmale und Blickrichtung genutzt um den Kopf und die Gesichtszüge des Avatars in Echtzeit zu animieren. Eine Glättung der Daten wird vorgenommen um die Animation zu verbessern. Zusätzlich werden Daten der Hand-, Ellenbogen- und Schultergelenke zur Berechnung der Lautstärke und Tonhöhe verwendet. Die Armstreckung wird mittels der Entfernung der Schulter zum Ellenbogen plus der Entfernung des Ellenbogens zur Hand, geteilt durch die gesamte Entfernung, also von der Schulter zur Hand, berechnet. Der

so ermittelte Wert wird genutzt um die Lautstärke festzulegen, wobei die volle Lautstärke bei ganz gestreckten Armen erreicht wird.

Die Handpositionshöhe (Y-Achse) wird von der Schulterpositionshöhe abgezogen und durch die maximale Armstreckung geteilt. Dieser Wert wird zur Bestimmung der Tonhöhe herangezogen. Beide Berechnungen werden jeweils für beide Arme durchgeführt und am Ende wird der höhere Wert für den Synthesizer verwendet.

## 6 Mapping, „Hamonizer“ und Soundsynthese

Mithilfe von Mundformung und Armgesten steuert der Benutzer verschiedene Parameter der Gesangssynthese. Der Gesang ist komplett synthetisiert um eine größtmögliche Kontrolle und Flexibilität zu gewährleisten.

Die Lautstärke wird direkt über die Armstreckung gesteuert: ein Strecken der Arme erhöht die Lautstärke, ein Anwinkeln verringert sie. Die gesungene Tonhöhe wird dabei über die Armhöhe bestimmt.

Steht die Tonhöhe in direkter Korrelation zur Armhöhe ist es sehr schwer, einen gewünschten Ton zu treffen. Ein ähnliches Verhalten kann zum Beispiel beim berührungslos gespielten Musikinstrument Theremin beobachtet werden. Da eine unterhaltsame Benutzererfahrung im Vordergrund steht und das System auch besonders von musikalisch unerfahrenen Benutzern verwendet werden soll, wäre diese Variante des Mappings kontraproduktiv für das geplante Ziel. Es wurden daher zwei Alternativen entwickelt, um dieses Problem zu lösen:

Methoden eins quantisiert die Armhöhe auf 25 feste Stufen. Zusätzlich werden kleine Armbewegungen verworfen, um ein schnelles und ungewolltes Wechseln zwischen zwei Stufen zu verhindern. Jeder Stufe ist ein konkreter MIDI-Wert zugeordnet, beginnend bei 41 (ca 87 Hz) bis 65 (ca 349 Hz). Diese Methode erlaubt es geübten Benutzern den Gesang umfangreich zu steuern, für ungeübte Anwender ist jedoch die zweite Variante besser geeignet.

Um dem Anwender beim Spielen zu helfen und den spielerischen und unterhaltenden Charakter der Installation damit zu unterstützen, berücksichtigt die zweite Methode zur Tonhöhensteuerung nur die Richtung der Armbewegung: bewegt man die Arme nach oben, wird der nächst höhere, bewegt man die Arme nach unten, der nächst tiefere Ton gespielt. Hierzu wird die Melodielinie des Liedes verwendet (diese liegt als MIDI-Track vor): der nächst höhere beziehungsweise tiefere Ton wird anhand der richtigen Melodie und der aktuell vorliegenden Tonart ermittelt. Auf diese Weise wird immer ein Ton ausgewählt, der hinreichend gut zur Hintergrundmusik der aktuell gesungenen Arie passt.

Der Benutzer kann den gesungenen Vokal durch die Mundformung bestimmen. Das zuvor trainierte neuronale Netzwerk erkennt den wahrscheinlichsten Vokal und sendet diesen an die Sound Synthese. Um ungewollte schnelle Wechsel zwischen zwei Vokalen, hervorgerufen durch Signalrauschen und Ungenauigkeit beim Tracking, zu verhindern, sind fünf aufeinanderfolgende Erkennungen desselben Vokals nötig, um eine Änderung zu bewirken. Dies erzeugt einen kleinen, aber akzeptablen Delay. Um den Vokalsound zu ändern werden die



Filter der Gesangssynthese in Abhängigkeit des erkannten Vokals angepasst. Eine schnelle Interpolation zwischen dem aktuellen und dem erkannten Vokal verhindert Soundartefakte. Die Interpolationszeit darf jedoch nicht zu groß sein, da ein hörbares „Vokal Glissando“ merkwürdig klingt.

Grundlage für die Gesangssynthese ist eine Vokalsynthese via Formanten<sup>2</sup>. Die in dieser Arbeit verwendeten Frequenzwerte für die einzelnen Vokale sind Durchschnittswerte, entnommen aus [PB52]. Die Klangverarbeitung geschieht mittels Max/MSP in Kombination mit Java Script. Die Gesangssynthese verwendet eine additive Synthesetechnik und wird über die Armhöhe des Benutzers per MIDI gesteuert, um die Grundfrequenz des Gesangs zu bestimmen.

Der gewünschte Vokal wird durch Bandpassfilter modelliert. Diese formen die zum entsprechenden Vokal gehörenden Formanten. Während zwei Formanten ausreichend sind, um ein Signal als menschlichen Vokal erkennen zu können, stellten sich drei Vokale als geeigneter heraus, um einen menschlicheren Klang zu erzeugen. Ein vierter Formant bei 2,7 kHz wird verwendet, um die Klangcharakteristik eines Opersängers nachzuahmen. Ein Formant um 3 kHz ist nur im Frequenzspektrum des Gesangs eines ausgebildeten Opersängers oder einer ausgebildeten Opersängerin zu finden und nennen sich „Sängerformant“.

Einen detaillierteren Überblick über das „Harmonizer“ Modul und die Gesangssynthese haben wir in [FSG13b] dargestellt.

## 7 Evaluation

Für eine erste Evaluation testeten sechs Personen das komplette System, inklusive Kalibrieren, Trainieren des neuronalen Netzwerks und Singen zweier Stücke. Vor dem Singen der Stücke konnten das System einige Zeit frei getestet werden. Im Anschluss an den Test wurden die Teilnehmer befragt und füllten den AttrakDiff<sup>TM</sup> (attrakdiff.de) Fragebogen aus. Drei der Testteilnehmer hatten umfangreiche musikalische Kenntnisse (Sänger in Chören beziehungsweise Bands), die anderen drei hatten kaum Vorkenntnisse.

AttrakDiff<sup>TM</sup> wird verwendet um die Attraktivität von interaktiven Produkten zu testen. Zu diesem Zweck wird ein Fragebogen mit gegensätzlichen Attributspaaren für jede Frage ausgefüllt. Die Attribute beziehen sich jeweils auf die verschiedenen zu evaluierenden Kategorien: die pragmatische Qualität (PQ), welche die Gebrauchstauglichkeit, wie effektiv der Benutzer sein Ziel erreichen konnte, beschreibt; die hedonistische Qualität – Stimulation (HQ–S), die zeigt, wie gut das Produkt mit Hilfe von inspirierenden Funktionalitäten, Inhalten, Interaktions- und Präsentationsmöglichkeiten den Benutzer in seinem Bestreben, sich zu verbessern, unterstützt; die hedonistische Qualität – Identität (HQ–I) gibt den Grad der Benutzeridentifikation mit dem Produkt an; und die Attraktivität (ATT) als allgemeine Beurteilung des Produkts.

---

<sup>2</sup>Kurz gesagt sind Formanten die akustische Resonanz des menschlichen Vokaltrakts und werden benötigt um zwischen verschiedenen Vokalen unterscheiden zu können.

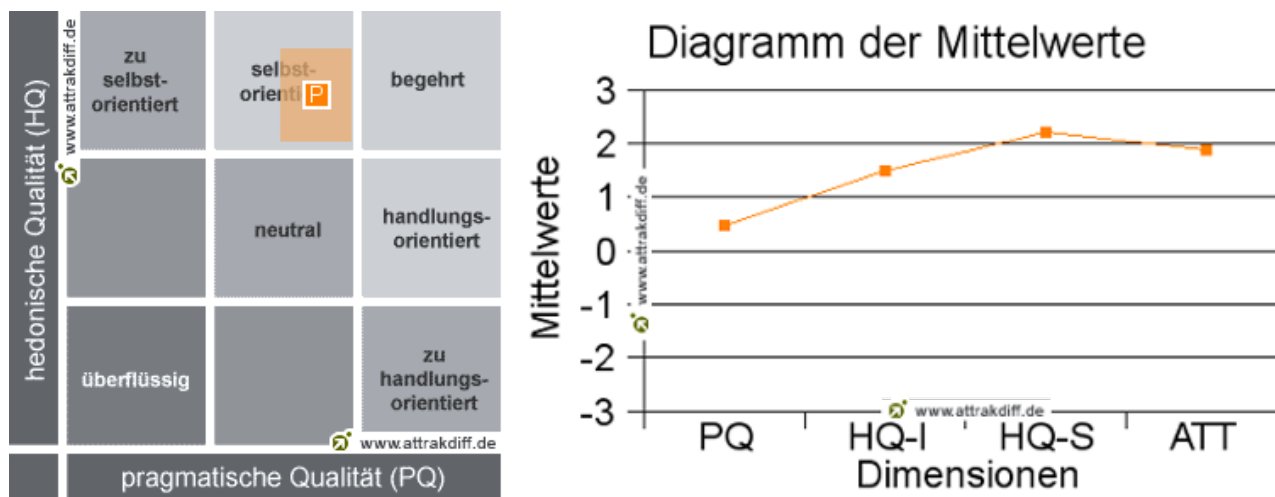


Abbildung 5: AttrakDiff™ (attrakdiff.de) Testergebnisse

PQ und HQ sind voneinander unabhängige Faktoren und tragen etwa zu gleichen Teilen zum ATT Level bei. Die Ergebnisse für die Benutzeroberfläche können im Portfolio gesehen werden (Abbildung 5). Die Installation ist als „selbst-orientiert“ einzustufen. Das bedeutet, die HQ berücksichtigend, dass der Benutzer eine gute Identifikation mit dem Produkt hat und von diesem stimuliert und motiviert wird. Die PQ betreffend wird der Benutzer bis zu einem gewissen Grad von dem System unterstützt, jedoch gibt es noch Optimierungspotential bei der Benutzerfreundlichkeit.

Das halb-transparent orangefarbene Konfidenz-Rechteck zeigt, dass diese Interpretation für die PQ recht eindeutig ist, jedoch ist die Ausdehnung der HQ ein wenig größer und damit weniger aussagekräftig. Dies lässt sich durch die geringen Anzahl an Testpersonen erklären.

Die Durchschnittswerte der evaluierten Kategorien sind in Abbildung 5 dargestellt. Alle vier Kategorien sind überdurchschnittlich bewertet worden. Besonders die HQ-S erreichte eine sehr hohe Punktzahl, und die allgemeine Attraktivität ist ebenfalls stark ausgeprägt, was darauf hinweist, dass das Ziel, eine unterhaltsame Erfahrung für den Benutzer zu erzeugen, erreicht wurde. Die Benutzeridentifizierung mit dem Produkt ist ebenfalls relativ stark, was jedoch ein zu erwartender Nebeneffekt bei einer angenehmen Erfahrung mit einem Produkt ist. Im direkten Vergleich mit den anderen Kategorien ist die PQ eher schwach ausgeprägt. Eine Verbesserung der Benutzbarkeit würde demnach die allgemeine Attraktivität noch weiter steigern.

Um die konkreten Anforderungen zur Verbesserung der Benutzbarkeit der Installation zu analysieren wurden besonders die Interviews ausgewertet. Folgende Aussagen konnten hierbei getroffen werden:

- Allen sechs Personen machte das Spielen Spaß. Drei der Teilnehmer nannten hier eine technische Faszination als Grund für die unterhaltsame Erfahrung.
- Die Soundsynthese wurde einheitlich als zumindest glaubwürdig bezeichnet. Nur ein Teilnehmer erwähnte, dass vereinzelt eine künstliche Klangcharakteristik der Gesangsstimme zu erkennen war.

- Die drei Testpersonen ohne gesangliche Erfahrung konnten sich vorstellen, dass durch die Installation ein Gefühl für die Erfahrung beim echten Singen übermittelt werden konnte. Die drei Testpersonen mit gesanglicher Erfahrung fanden den Grad der Simulation nur bedingt überzeugend.
- Für fünf der Tester war das Spielen sehr schwierig und kompliziert. Sie äußerten sich allerdings zuversichtlich, dass sie mit ein wenig Übung ihre Fähigkeiten im Umgang mit der Installation stark verbessern könnten. Ein Teilnehmer hatte keinerlei Schwierigkeiten das System zu verwenden.
- Eine Verbesserung der Visualisierung würde die Benutzbarkeit stark verbessern.
- Keiner der Anwender fand die Kalibrierungsphase zu lang.

Eine andere interessante Beobachtung war, dass die meisten Teilnehmer einen starken Ehrgeiz entwickelten, besser im Umgang mit dem System zu werden. Offensichtlich ist es einfacher, eine technische Anwendung wie die hier präsentierte Installation zu erlernen als tatsächliches Singen, was zu höherer Motivation beim Lernen führt. Zusammenfassend lässt sich sagen, dass diese vorläufige Evaluation eine positive Tendenz zeigt und das System aktuell als Unterhaltungsanwendung zu klassifizieren ist, aufgrund der geringen Anzahl an Testteilnehmern allerdings statistisch nicht stark aussagekräftig ist.

## 8 Fazit

Die hier präsentierte Installation wurde bereits auf diversen wissenschaftlichen Konferenzen erfolgreich aufgeführt (u.a. auf der Mensch und Computer 2013 in Bremen [FSG13a]). Auch wenn nicht alle hier beschriebenen fortschrittlichen Komponenten gezeigt wurden, waren die Zuschauer von der Darbietung und der Benutzererfahrung positiv beeindruckt. Besonders die jüngst ausgeführte Evaluation war für die weitere Entwicklung des Systems wichtig. Alle Teile des Systems funktionieren wie geplant und beschrieben, allerdings gibt es noch Einschränkungen bezüglich des aufwändigen Aufbaus und der Kalibrierungszeit. Einige Bereiche müssen zudem auf ihre Fehlertoleranz hin optimiert werden bevor umfangreiche Benutzertests durchgeführt werden können. Konkret ist geplant weitere Verbesserungen an der Visualisierung und Spielhilfe vorzunehmen. Zudem funktioniert der Gesangssynthesizer bisher ausschließlich mit Vokalen, weshalb das Hinzufügen von Konsonanten eine spannende Ergänzung der Darbietung wäre.

Die bisherige Resonanz, dass das System unterhaltsam sei, motiviert zu einigen Gedanken für zukünftige Entwicklungen: zum einen sollen zusätzliche Arien hinzugefügt werden, jedoch auch neue Stimmen entwickelt werden, wie zum Beispiel Sopran oder Alt Gesang. Persönliche Charakteristika wie Extraversion und Introversion könnten der Stimme zusätzliches Leben einhauchen. In weiter entfernten Zukunftsszenarien wäre es eventuell möglich, die Stimme eines Sängers aufzunehmen, bedeutende Eigenschaften dieser zu erkennen und auf den Gesangssynthesizer anzuwenden.

## Literatur

- [BIH<sup>+</sup>12] D. Butler, Shahram Izadi, Otmar Hilliges, David Molyneaux, Steve Hodges, and David Kim. Shake'n'sense: Reducing interference for overlapping structured light depth cameras. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 1933–1936, NY, USA, 2012.
- [BW11] Klaus-Ernst Behne and Clemens Wöllner. Seeing or hearing the pianists? A synopsis of an early audiovisual perception experiment and a replication. *Musicae Scientiae*, 15(3):324–342, 2011.
- [CH10] Jie Cheng and Peisen Huang. Real-time mouth tracking and 3d reconstruction. In *Proc. of ICISP*, volume 4, pages 1524–1528, Yentai, China, 2010.
- [CLB<sup>+</sup>00] Pedro Cano, Alex Loscos, Jordi Bonada, Maarten De Boer, and Xavier Serra. Voice morphing system for impersonating in karaoke applications. In *Proc. of ICMC*, Berlin, Germany, 2000.
- [FSG13a] Jochen Feitsch, Marco Strobel, and Christian Geiger. Caruso - singen wie ein tenor. In *Proc. of MUC*, pages 531–534, Bremen, Germany, 2013.
- [FSG13b] Jochen Feitsch, Marco Strobel, and Christian Geiger. Singing like a tenor without a real voice. In *Proc. of ACE*, pages 258–269, Twente, Netherlands, 2013.
- [FSMG14] Jochen Feitsch, Marco Strobel, Stefan Meyer, and Christian Geiger. Tangible and body-related interaction techniques for a singing voice synthesis installation. In *Proc. of TEI*, pages 157–164, Munich, Germany, 2014.
- [FW13] Stefano Fasciani and Stefano Wyse. A self-organizing gesture map for a voice-controlled instrument interface. In *Proc. of NIME*, pages 507–511, Daejeon, Korea, 2013.
- [GNK<sup>+</sup>12] M. Goto, T. Nakano, S. Kajita, Y. Matsusaka, S. Nakaoka, and K. Yokoi. Vocalistener and voca-watcher: Imitating a human singer by using signal processing. In *Proc. of ICASSP*, pages 5393–5396, Kyoto, Japan, 2012.
- [LHT03] Michael J. Lyons, Michael Haehnel, and Nobuji Tetsutani. Designing, playing, and performing with a vision-based mouth interface. In *Proc. of NIME*, pages 116–121, Singapore, Singapore, 2003.
- [PB52] Gordon E. Peterson and Harold L. Barney. Control methods used in a study of the vowels. *The Journal of the Acoustical Society of America*, 24(2):175–184, 1952.
- [Sun06] Johan Sundberg. The KTH synthesis of singing. *Advances in Cognitive Psychology*, 2(2-3):131–143, 2006.

# Picture-based Localisation for Pervasive Gaming

Martin Fischbach, Jean-Luc Lugin, Marc Erich Latoschik, Michael Fendt

Human-Computer Interaction

Universität Würzburg

Am Hubland

97074 Würzburg

E-Mail: martin.fischbach@uni-wuerzburg.de

**Abstract:** Localisation, i.e. determining the position of users or devices, constitutes a key requirement for almost all types of mobile pervasive games. However, current marker-based localisation systems, e.g. based on QR codes, present drawbacks that limit game deployment, scalability and maintainability. In this paper, we propose an alternative to solve these issues and introduce the first steps towards its full realisation. Our approach relies on markerless picture matching using a natural feature detection algorithm. Players reproduce a camera shot of a real-world site to confirm their presence and to progress further in the game. One of the critical requirements is to provide accurate recognition while preserving application responsiveness with a large range of mobile devices and camera resolutions. We developed a proof-of-concept system and determined the best picture resolutions and feature numbers necessary to preserve both accuracy and responsiveness on diverse mobile devices. Our first results demonstrate the feasibility to achieve precise recognition within realtime constraints. We believe such localisation system have the potential to considerably facilitate pervasive game authoring while promoting new types of game mechanics.

**Keywords:** Pervasive Games, Natural Feature Matching, Performance, Localisation

## 1 Introduction

Pervasive gaming describes a category of games that incorporate real-world activities and experiences in a virtual game world [BML05, KGB13]. Due to recent technological advances, pervasive games became increasingly feasible on affordable mobile hardware devices, like smartphones or tablets. Pervasive games allow the exploration of novel game concepts [KLL12] as well as the extension of traditional real or virtual games [GDB<sup>+</sup>07, GSZW<sup>+</sup>13, Inc14, Nia14]. Besides pure entertainment purposes, pervasive gaming also reveals great potential in application areas such as tourism [LGG13], edutainment [WW11, FGEM<sup>+</sup>11], or healthcare [SLB<sup>+</sup>10]. Localisation, i.e. determining the position of users or devices, constitutes a key requirement for almost all types of pervasive games [BOL<sup>+</sup>06]. Most pervasive games rely on players to be physically present in a specific geographical location to progress in the game. The required accuracy of the player's localisation may vary from one game to another. Rough geographic location estimation via GPS, mobile network cell, or WiFi

hotspot information is widely used for pervasive games [BOL<sup>+</sup>06, KGB13]. When the player's location has to be estimated more precisely, artificial fiducial markers (e.g. QR codes) are usually employed [WW11, LGG13]. Although very robust, one major drawback of this approach is that the artificial markers have to be physically placed in the environment by the game authors. They oftentimes require to be frequently maintained to insure their presences and integrity. The placement of such markers is also not always possible or desirable as they may affect the site's aesthetics or offend against prescriptions. In addition, marker-based games have a low scalability due to their difficulty to be duplicated in various locations or to sustain long-term participation [NMW<sup>+</sup>12].

In this paper, we present an alternative to fiducial markers for player localisation based on natural feature tracking techniques. Such techniques are commonly used in pure augmented reality applications and provide a high percentage of recognition in realtime even with mobile devices [WRM<sup>+</sup>08, RRKB11]. Our approach relies on a picture-based localisation technique, where players have to confirm their presence at the right location, by reproducing a given picture with their mobile device. The content and perspective determined by the given photography is said to be a *Natural Reference Point* (NRP) that players should match as closely as possible. Besides the advantage of not requiring a physical marker to be placed in the environment, the creation of a NRP is also straightforward. The device's camera is simply positioned and oriented to specify the perspective the player has to discover and reproduce. The player is guided to the proximity of a NRP using global positioning data visualized via Google Maps. Ideally, other available sensor data, like the orientation obtained from a magnetometer, would be recorded to improve the matching process.

However, outdoor picture comparison using natural feature tracking brings a new set of challenges, especially if the support of a wide range of mobile devices is desired. High resolution pictures with numerous different distinguishable features are commonly required to be efficient. In addition, the utilized algorithms are typically computationally intensive and require a camera calibration process, preferably within an indoor environment. The pervasive game idea renders such ideal conditions impossible to achieve. An ideal solution has to be able to match pictures taken from different cameras and perspectives as well as content affected by environmental conditions (weather, season, and brightness). Moreover, its responsiveness has to support realtime feedback to guide players during their quest of matching the view point. Consequently, the identification of an algorithm capable of running on a multitude of devices with realtime constraints is a first step towards a fast and robust picture-based localisation for pervasive games. In the rest of this paper we will first briefly present related work, before introducing the overall concept and requirements of our approach. We then describe a proof-of-concept system and our empirical methodologies used to identify good candidate picture characteristics that will preserve both responsiveness and accurate recognition. Finally, we discuss the potential impact on pervasive game authoring and gameplay by outlining future improvements and a planned adaption of a commercial game product.

## 2 Related Work

The term *pervasive game* has been used to outline a heterogeneous set of game applications that “extend the gaming experience out into the real world” [BML05, KGB13] or “integrate the physical and social aspects of the real world” [MCMN05], ranging from smart toys through location aware, mobile, video or affective games to augmented reality and augmented tabletop games. Nieuwdorp [Nie07] carefully reviews existing definitions and emphasizes the equality of the terms *pervasive game* and *ubiquitous game*. In fact, she proposes the separation of two perspectives when denoting a game to be *pervasive*: a cultural perspective that describes the real-world gameplay and a technological perspective that describes the technologies supporting the game.

Technologically pervasive games depend on localisation, communication, context detection, augmentation, authoring and game engines, as well as orchestration and surveillance [BOL<sup>+</sup>06]. Although not all of these requirements are present or equally important amongst pervasive games, detecting the position of the user in the real world is essential in most cases. Besides the widely used global or local positioning systems, computer vision approaches provide another common solution for localisation or even tracking. Existent approaches are separated based on the requirement of an artificial fiducial marker. Marker-based approaches commonly utilize QR codes on mobile devices for localisation [WW11, LGG13], if the code’s position is known prior, or for tracking in augmented reality applications [KTC09]. Markerless approaches build upon the detection of natural features in images, which are used to identify scenes or motifs from different perspectives. Compared to marker-based approaches, markerless approaches typically require more processing power [WS09]. Nevertheless augmented reality applications became increasingly suitable on mobile platforms (e.g. [QCE14]), due to both recent advances in affordable mobile computing hardware and the development of highly efficient algorithms especially tailored for devices with limited hardware [WRM<sup>+</sup>08, RRKB11, PR11, PSR13]. These algorithms should also be suitable for picture-based localisation using natural features on mobile devices. Therefore, this paper presents first steps towards the full validation of the feasibility of this concept. We especially focus on the minimum picture resolution and number of matching features required to ensure an efficient comparison while preserving an enjoyable gaming experience.

## 3 Concept

Our main conceptual goal is to provide an alternative to QR codes, that can be utilized for identifying NRPs without the need to deploy an artificial fiducial marker in the environment. We propose to achieve this goal by applying natural feature matching to identify previously recorded reference points and thus confirm the user’s proximity and attention. NRPs can be used in similar fashion to QR codes in pervasive games. For instance, to trigger the output of context information if the player successfully confirmed to be at a specific sight in a touristic game or to trigger the next step in the ongoing story line if the player has

successfully confirmed to be at a designated remote location in a role-playing game. Besides the raw confirmation from the natural feature matching, other available sensor data can be used to support a player’s task of finding and confirming a NRP (see figure 1).



Figure 1: Finding and confirming a NRP: (a) Global or local positioning systems are used to pilot the player into intermediate proximity. (b) Subsequently, a picture in combination with an optional description helps the player to identify the approached NRP. (c) Realtime feedback based on the device orientation obtained from a magnetometer and the quality of the natural feature matching guide the player to the NRP (concept). (d) If a certain threshold of the matching quality is exceeded, the NRP is considered to be confirmed.<sup>1</sup>

To implement the described interaction, the respective sensor data has to be recorded when defining new NRPs and has to be stored on a server. Since the required sensors already have to be present on the players’ mobile devices, authoring is straightforward: in order to place or define a NRP, authors position their device like they want players to discover the NRP and confirm, similar to taking a photo. All necessary data can be recorded automatically without requiring authors to print a marker, deploy it in the environment and store its position on a server manually. Thus the usability of the authoring process is highly improved, promoting the integration of user generated content into pervasive game concepts by making the authoring a game itself. Altogether, the presented concept poses great benefits for pervasive games that rely on the players to discover specific locations or perspectives.

## 4 Proof-of-Concept

As a first step to prove this concept, a natural-featured-based picture comparison for the mobile operating system Android (Version 4.4) has been implemented. This prototype has subsequently been used to empirically determine picture characteristics that result in a good trade-off between accuracy and computing time.

<sup>1</sup>Stencils from <https://www.graffletopia.com/stencils/578>



## 4.1 Implementation

The picture comparison is built upon the open source C/C++ library *OpenCV* [Tea14] for image processing. The access to OpenCV from within the Android application is handled by the *Android NDK*. The comparison of two images is subdivided into three steps: (1) the detection of natural features in each picture, (2) the matching of the detected features between the pictures, and (3) the decision if two pictures are considered to contain the same scene or motif. The detection of natural features is accomplished by the application of the Oriented FAST and Rotated BRIEF (ORB) algorithm [RRKB11]. ORB is optimized for low-power devices, like smartphones, and already included in the OpenCV library. The feature matching strategy uses a *RANdom Sample Consensus* (RANSAC) to assess matches [Lag11, Page 233ff.]. The final step is a binary decision that compares the number of matching features with a threshold.

## 4.2 Evaluation

The described picture comparison process depends upon several parameters, like image sizes or thresholds, that influence the overall performance and robustness. In order to determine optimal values for these parameters the utilized algorithms have been applied for two validation sets:  $V_1$  containing 10 pictures of the same scene and  $V_2$  containing 100 pictures of diverse motifs and scenes. To support the independence of the determined parameter values from a specific device, these pictures have been taken with different smartphones in different resolutions and formats. Figure 2 illustrates a sample comparison between two pictures of  $V_1$ .

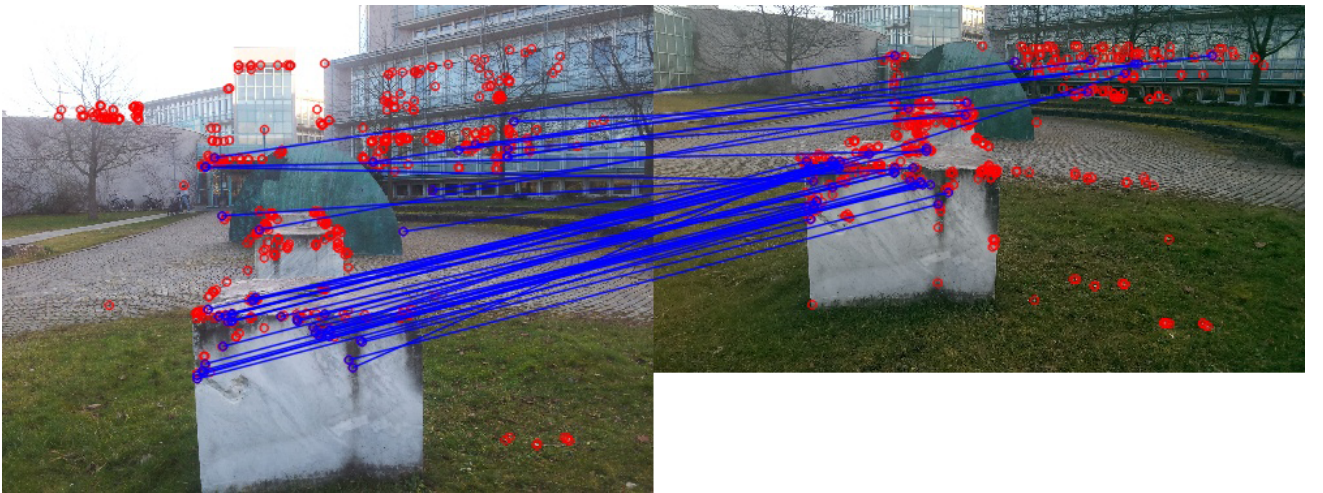


Figure 2: Sample comparison between two pictures taken by different devices. Red circles denote detected natural features. Blue lines connect pairs of matched features.

The first evaluated parameter in this study is the size of the images that should be passed to the picture comparison process to preserve both accuracy and responsiveness. The image resolution is positively correlated with the number of detected features and negatively

correlated with the processing time. Therefore, the most critical aspect is to identify the lowest image resolution that will preserve a high fraction of detected features, necessary for accurate comparison. Figure 3 and 4 show the measured processing time and the number of detected features for the images of  $V_2$ . All images have been resized to several resolutions, by scaling the longer aspect to the desired length.

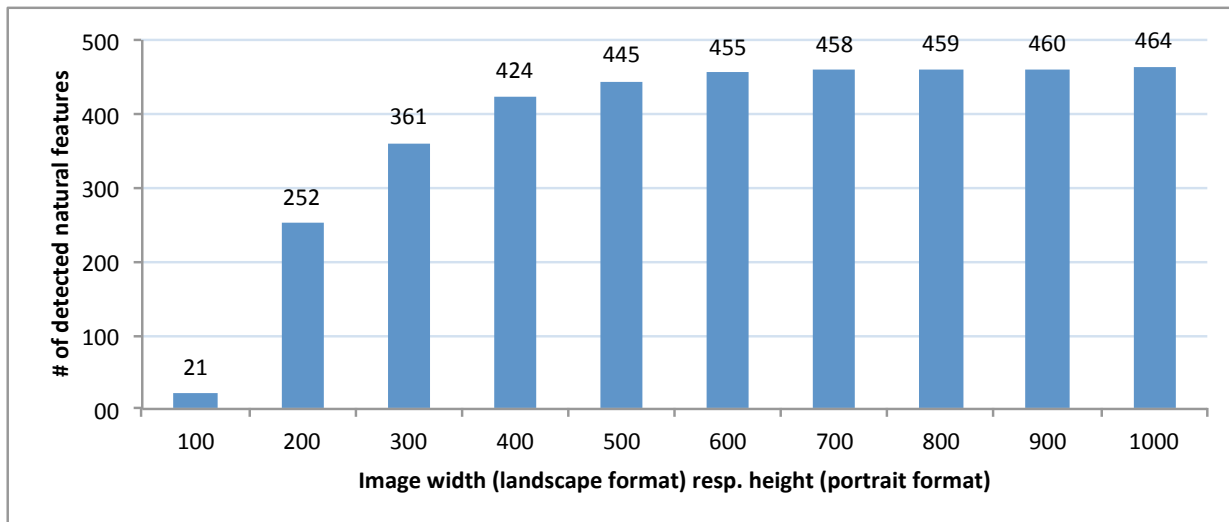


Figure 3: Average number of detected natural features for  $V_2$ . All 100 images have been scaled to all depicted resolutions.

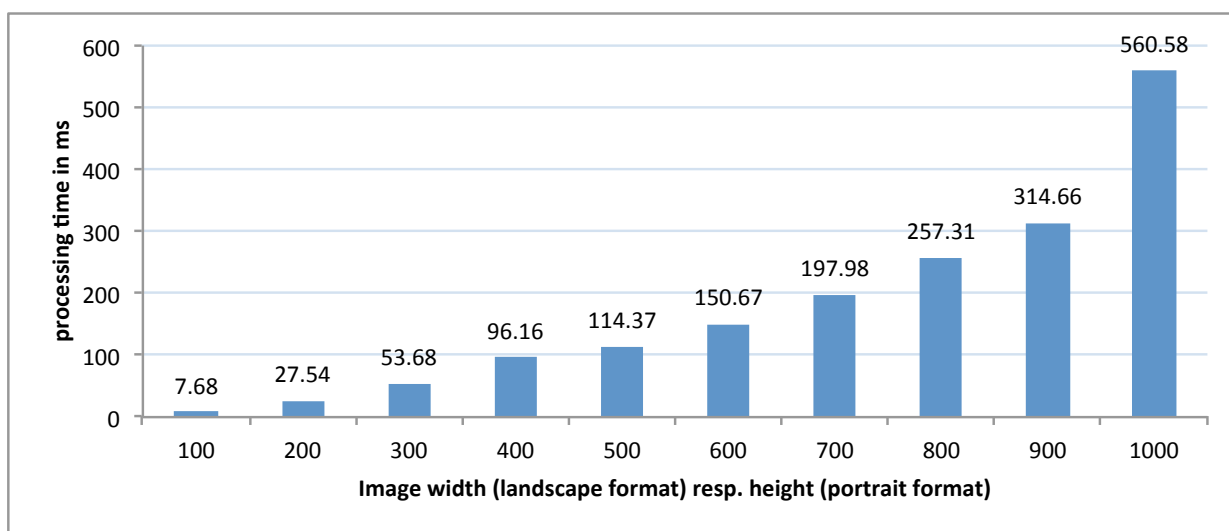


Figure 4: Average processing time for the detection of natural features. The measurements have been taken on a *Samsung Galaxy S2* using  $V_2$ . All 100 images have been scaled to all depicted resolutions.

Up to an image size of 500 pixels a high increase of the number of detected features can be noted. The increase between 500 and 600 pixels is only 2.3%, whereas the corresponding increase in processing time is 24.1%. An image size of 500 pixels is thus considered to be the

best choice for further processing (see table 1). The required processing time of on average  $114.37ms$  is only slightly above the limit of  $100ms$ , for system reactions that are perceived to be instantaneously [Nie94].

Device Name	Camera Resolution	Recommended Resolution
Samsung Galaxy S2	3264x2448 pixels	500x375 pixels
HTC One mini	2688x1520 pixels	500x283 pixels
HTC Wildfire S	2592x1728 pixels	500x333 pixels

Table 1: Recommended image resolutions for natural feature detection.

The second evaluated parameter is the threshold for the final binary decision. To get a lower bound for this threshold, every image from  $V_2$  has been compared with all other images from this validation set, resulting in a total of 9900 direct comparisons (see figure 5). Because every image in  $V_2$  has different content, the obtained quantities of matching features are assumed to be insufficient to denote a matching of two pictures. The highest obtained number of matching features is 44, excluding two outliers at 134 and 144.

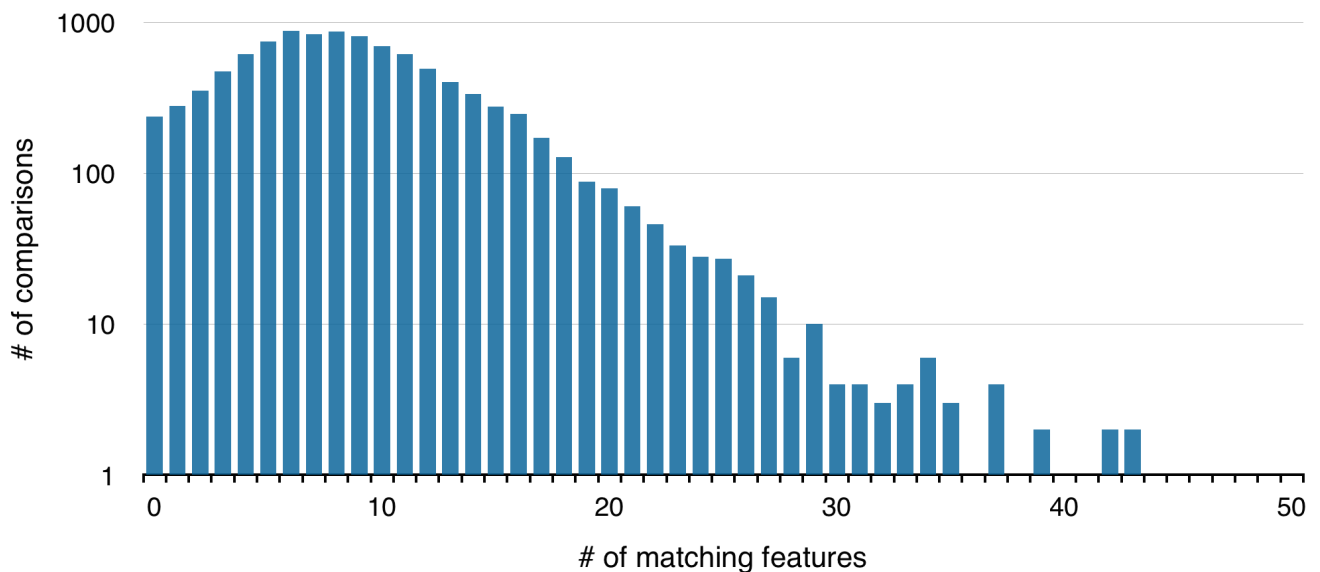


Figure 5: Number of matching features for all comparisons between pictures of  $V_2$ , showing diverse motifs and scenes (negative examples). Two outliers at 134 and 144 are not visualized.

To get an upper bound for the threshold, every image of  $V_1$  has been compared in the same way, resulting in a total of 90 comparisons (see figure 6). Since every image in  $V_1$  shows the same scene, the obtained quantities denote sufficient amounts of accepted matching features. The number of accepted matching features range from 12 to 191.

Because the determined bounds overlap, the positive and negative examples are not completely separable. While optimizing the binary decision by maximizing the corresponding  $F_1$  score [MKS<sup>+</sup>99] would lead to a high accuracy, it does not meet the requirements of

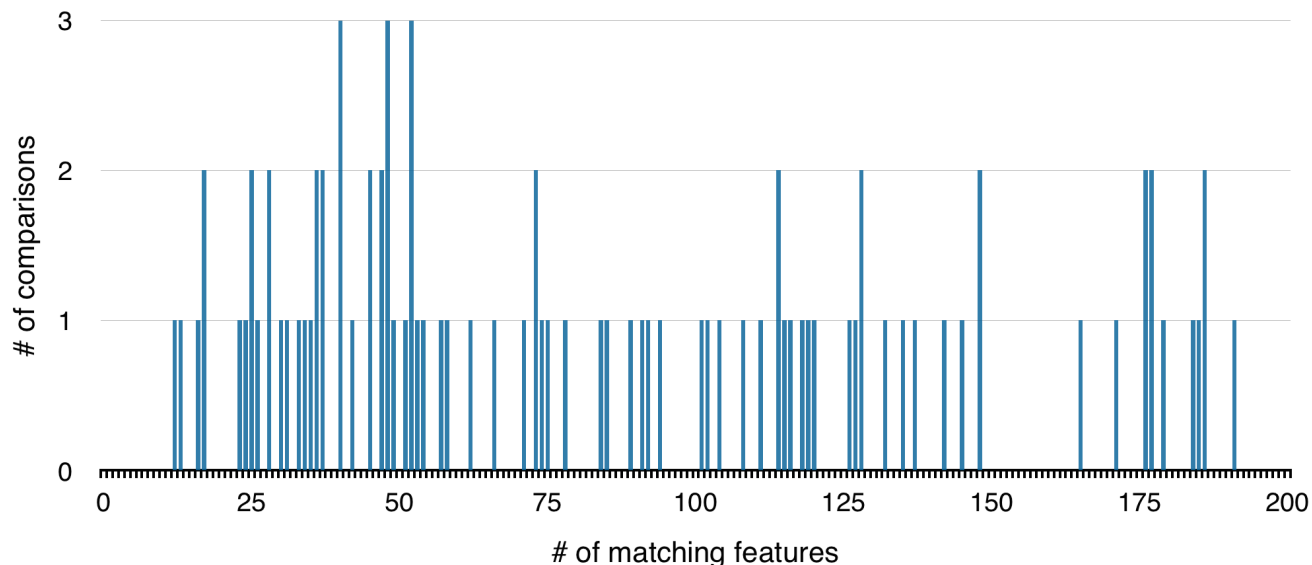


Figure 6: Number of matching features for all comparisons between pictures of  $V_1$ , showing the same scene (positive examples).

pervasive games. It is crucial not to mistake two pictures that do not show the same content to be equal (false positives). Therefore a threshold of 45, just above the highest occurrence of matches between two distinct pictures, is considered to be the best choice.

Even though this choice causes a higher amount of equal images to be denied (false negatives), the application context has the potential to neglect this flaw. Typically users will position their mobile device in the approximate location and orientation of the NRP, while slightly moving it. This allows the application to record several images that all can be checked against the corresponding image stored on the server. Resulting in a higher chance of confirming a NRP despite the presence of false negatives.

## 5 Conclusion

In this paper we proposed an alternative to improve the traditional deployment, scalability, and maintainability limitations of marker-based pervasive games. Our approach relies on markerless picture matching and uses the natural feature detection algorithm ORB, designed for mobile platforms. With our concept, players can localise themselves by reproducing a camera shot of a real-world sight in order to progress further into the game. One of the critical requirements was thus to provide accurate recognition, while preserving application responsiveness with a wide range of mobile devices and camera resolutions. We developed a proof-of-concept system and evaluated the best picture resolutions and number of features necessary to preserve both requirements on diverse mobile devices. Our first results demonstrate the feasibility to achieve precise recognition within realtime constraints.

We believe that many existing QR code-based pervasive games, e.g. [WW11, LGG13], could easily benefit from such an approach. It also has the potential to promote the devel-

opment of novel gameplay concepts, especially for geocaching or role-playing games, where players are meant to discover specific perspectives of a historical or touristic site and the mounting of artificial markers is impossible. Finally, the simplicity of creating new NRPs improves the efficiency of the authoring process and thus could greatly facilitate the integration of user generated content into pervasive gaming concepts.

However, future work will evaluate to what extent the identified minimum resolution and number of features are supporting an enjoyable gaming experience, especially with a large diversity of images. The required degree of accuracy should promote a challenging gameplay without frustrating the player by making the game too difficult or easy. Consequently, the visualization of the degree of similarity as well as the orientation offset will be used to provide realtime feedback during the matching process. Before running a complete usability study with a full game implementation in partnership with a game company, we also wish to evaluate the system robustness to weather, season and brightness variations. The range of mobile devices under test will also be increased to a more representative sample, in order to continue the evaluation of our system's portability and performance.

## References

- [BML05] Steve Benford, Carsten Magerkurth, and Peter Ljungstrand. Bridging the Physical and Digital in Pervasive Gaming. *Commun. ACM*, 48(3):54–57, March 2005.
- [BOL<sup>+</sup>06] Wolfgang Broll, Jan Ohlenburg, Irma Lindt, Iris Herbst, and Anne-Kathrin Braun. Meeting Technology Challenges of Pervasive Augmented Reality Games. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '06. ACM, 2006.
- [FGEM<sup>+</sup>11] Faranak Fotouhi-Ghazvini, Rae A Earnshaw, Ali Moeini, David Robison, and Peter Excell. From E-Learning to M-Learning-the use of Mixed Reality Games as a new Educational Paradigm. *iJIM*, 5(2):17–25, 2011.
- [GDB<sup>+</sup>07] Lyndsay Grant, Hans Daanen, Steve Benford, Alastair Hampshire, Adam Drozd, and Chris Greenhalgh. MobiMissions: the game of missions for mobile phones. In *ACM SIGGRAPH 2007 educators program*, page 12. ACM, 2007.
- [GSZW<sup>+</sup>13] Anke Giebler-Schubert, Chris Zimmerer, Thomas Wedler, Martin Fischbach, and Marc Erich Latoschik. Ein digitales Tabletop-Rollenspiel für Mixed-Reality-Interaktionstechniken. In *Virtuelle und Erweiterte Realität, 10. Workshop der GI-Fachgruppe VR/AR*, Informatik, pages 181–184. Shaker Verlag, 2013.
- [Inc14] Stray Boots Inc. Stary Boots Scavenger Hunts and Walking Tours. Retrieved from [www.strayboots.com](http://www.strayboots.com), 2014.

- [KGB13] Vlasios Kasapakis, Damianos Gavalas, and Nikos Bubaris. Pervasive Games Research: A Design Aspects-based State of the Art Report. In *Proceedings of the 17th Panhellenic Conference on Informatics*, PCI '13, pages 152–157. ACM, 2013.
- [KLL12] Ben Kirman, Conor Linehan, and Shaun Lawson. Blowtooth: A Provocative Pervasive Game for Smuggling Virtual Drugs Through Real Airport Security. *Personal Ubiquitous Comput.*, 16(6):767–775, August 2012.
- [KTC09] Tai-Wei Kan, Chin-Hung Teng, and Wen-Shou Chou. Applying QR Code in Augmented Reality Applications. In *Proceedings of the 8th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '09, pages 253–257. ACM, 2009.
- [Lag11] Robert Laganière. *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 Recipes to Master this Library of Programming Functions for Real-time Computer Vision*. Packt Publishing Ltd, 2011.
- [LGG13] MaríaTeresa Linaza, Aitor Gutierrez, and Ander García. Pervasive Augmented Reality Games to Experience Tourism Destinations. In *Information and Communication Technologies in Tourism 2014*, pages 497–509. Springer International Publishing, 2013.
- [MCMN05] Carsten Magerkurth, Adrian David Cheok, Regan L Mandryk, and Trond Nilsen. Pervasive Games: Bringing Computer Entertainment Back to the Real World. *Computers in Entertainment (CIE)*, 3(3):4–4, 2005.
- [MKS<sup>+</sup>99] John Makhoul, Francis Kubala, Richard Schwartz, Ralph Weischedel, et al. Performance measures for information extraction. In *Proceedings of DARPA broadcast news workshop*, pages 249–252, 1999.
- [Nia14] NianticLabs@Google. Ingress. The game. Retrieved from [www.ingress.com](http://www.ingress.com), June 2014.
- [Nie94] Jakob Nielsen. *Usability engineering*. Elsevier, 1994.
- [Nie07] Eva Nieuwdorp. The Pervasive Discourse: An Analysis. *Comput. Entertain.*, 5(2), April 2007.
- [NMW<sup>+</sup>12] Carman Neustaedter, Victoria Moulder, Ron Wakkary, Tejinder K Judge, and Anthony Tang. Designing Mixed Reality Games to Study Culture, Family Practices, and Social Engagement. 2012.
- [PR11] Christian Pirchheim and Gerhard Reitmayr. Homography-based planar mapping and tracking for mobile phones. In *Mixed and Augmented Reality (IS-MAR), 2011 10th IEEE International Symposium on*, pages 27–36, Oct 2011.

- [PSR13] C. Pirchheim, D. Schmalstieg, and G. Reitmayr. Handling pure camera rotation in keyframe-based SLAM. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 229–238, Oct 2013.
- [QCE14] Inc Qualcomm Connected Experiences. Vuforia - enable your apps to see. Retrieved from [www.vuforia.com](http://www.vuforia.com), July 2014.
- [RRKB11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571, Nov 2011.
- [SLB<sup>+</sup>10] Kevin G. Stanley, Ian Livingston, Alan Bandurka, Robert Kapiszka, and Regan L. Mandryk. PiNiZoRo: A GPS-based Exercise Game for Families. In *Proceedings of the International Academic Conference on the Future of Game Design and Technology*, Futureplay '10, pages 243–246. ACM, 2010.
- [Tea14] OpenCV Developers Team. OpenCV. Retrieved from <http://opencv.org>, June 2014.
- [WRM<sup>+</sup>08] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134. IEEE Computer Society, 2008.
- [WS09] Daniel Wagner and Dieter Schmalstieg. History and future of tracking for mobile phone augmented reality. In *International Symposium on Ubiquitous Virtual Reality*, pages 7–10, 2009.
- [WW11] Bian Wu and Alf Inge Wang. A pervasive game to know your city better. In *Proceedings of the 2011 IEEE International Games Innovation Conference, IGIC '11*, pages 117–120, 2011.





# **SPIRIT - Ereignisgesteuerte Informationsvermittlung, Inspiration und Unterhaltung im urbanen Umfeld auf Basis mobiler Augmented Reality Technologien**

Antonia Kampa<sup>1</sup>, antonia.kampa@hs-rm.de  
Habiburrahman Dastageeri<sup>2</sup>, habiburrahman.dastageeri@hft-stuttgart.de  
Martin Storz<sup>2</sup>, martin.storz@hft-stuttgart.de  
Volker Coors<sup>2</sup>, volker.coors@hft-stuttgart.de  
Ulrike Spierling<sup>1</sup>, ulrike.spierling@hs-rm.de

<sup>1</sup>Hochschule RheinMain, Unter den Eichen 5, 65195 Wiesbaden

<sup>2</sup>Hochschule für Technik Stuttgart, Schellingstr. 24, 70174 Stuttgart

**Abstract:** Im Rahmen des Projektes SPIRIT wird eine innovative, prototypische Anwendung entwickelt, die die Sensorik mobiler Smartphone und Tablets verwendet, um ortsbasiert und kontextabhängig deren Realkamerabild mit eigens vorproduzierten Videoaufnahmen zu überlagern. In der mobilen Augmented Reality Anwendung sollen so ortsbezogene Informationen im Außenbereich mittels GPS, aber auch innerhalb von Gebäuden über eigene Indoor Navigationslösungen getriggert werden. Dargebotene Inhalte werden durch die narrative Metapher einer Geistersuche mit dem technischen Konzept integriert. Somit wird Nutzern der Anwendung durch Interactive Digital Storytelling ein situationsbezogenes unterhaltsames Erlebnis ermöglicht. In diesem Beitrag wird an Hand eines Szenarios ein erster lauffähiger Demonstrator beschrieben, sowie die Motivation für den videobasierten Ansatz erläutert.

**Keywords:** Augmented Reality, Interactive Storytelling, Indoor Navigation, Location-based Services, Story Engine

## **1 Einleitung**

Im Projekt SPIRIT wird eine Anwendung entwickelt, die Besuchern historischer Kulturstätten einen persönlichen Bezug zu Informationen über die Vergangenheit besuchter Orte erlaubt. Eine mobile App verwandelt ein Smartphone oder Tablet in ein „magisches“ Medium und erweckt mittels Augmented Reality (AR) darstellbare „Geister“ in interaktiven Storys zum Leben. Technisch wird im Projekt eine mobile, location-based AR Software für dynamische Video-Inhalte entwickelt. Sie fußt thematisch auf dem Projekt GEIST [KCS+01], dessen Mobile-Game-Design-Konzept des „magischen Equipments“ übernommen wird, mit dem Besucher Kontakt zum „Geist der Vergangenheit“ aufnehmen. Unter Verwendung von Richtungssensoren, Kamera-, GPS- und Indoor Navigations-Daten werden ortsbezogene Informationen multimedial in das Umfeld integriert. Es sollen dazu passende interaktive Storys auf mobilen Geräten dynamisch von einer Engine präsentiert werden. Schauplatz für den ersten Prototyp ist das Römerkastell Saalburg, dessen Museumseinrichtung als Projektpartner beteiligt ist.

Ziel dieses Beitrags ist die Vorstellung des SPIRIT Demonstrators und eine Diskussion des Einsatzes von technischen und erzählerischen Elementen bezogen auf Augmented Reality (AR). Nach einer Einführung der wichtigsten Begriffe des Interactive Digital Storytelling (IDS) folgt eine Beschreibung der SPIRIT Demonstrator AR App, sowie verwandter Projekte und Herausforderungen bei der Erstellung von Konzept und Inhalten. Die mediale Inhaltsstruktur wird vorgestellt und in Hinblick auf die Realisierung als AR App diskutiert. Im Anschluss werden die Anforderungen an die Entwicklungsplattform genannt und bereits implementierte technische Grundlagen des Trackings der SPIRIT Demonstrator App beschrieben.

## 2 Verwandte Anwendungen und Stand der Technik

### 2.1 Szenario und Story-Metapher im SPIRIT Demonstrator

Die konzipierte SPIRIT App „erzählt“ Spaziergängern abhängig von ihrer Position Teile einer Geschichte mit historischem Bezug, in dem sie ad-hoc aneinander gereichte Video-„Schnipsel“ abspielt. Abb. 1 symbolisiert den ersten implementierten Demonstrator – mit Hilfe eines Tablets gelingt einer Spielerin der Kontakt zum Geist eines römischen Soldaten, der sie auffordert, mit ihm das Römerkastell Saalburg zu besuchen.



Abbildung 1: Mockup-Szenariodarstellung des SPIRIT Demonstrator

Mit Hilfe des magischen Equipments interagieren Besucher demnach mit videobasierten virtuellen Figuren. Das Neue an der Konzeption ist weniger die erstmalige Entwicklung einzelner vorhandener Techniken (mobile, ortsbezogene, videobasierte AR App, Entertainment und Wissensvermittlung) als deren Integration. Vorproduziertes Videomaterial wird auf das Kamerabild mobiler Geräte gelegt, um für Besucher eine erweiterte Realität (AR) zu schaffen. Technische Herausforderungen liegen dabei in der glaubwürdigen Darstellung und der Interaktion mit virtuellen Figuren eingebettet in den AR-Kontext.

Für die Anpassung des gezeigten Inhalts auf das Verhalten des Benutzers wird eine Plot-Engine entwickelt. Die Plot-Engine setzt für einen Nutzer innerhalb einer oder mehrerer „Begegnungen“ mit Geistern aus atomaren Story-Elementen ad hoc eine Abfolge zu einem Story-Erlebnis zusammen. Die Plot-Engine reagiert auf User-Input, wie Bewegung, Standortveränderung und Interaktion mit graphischen Elementen (GUI), und berücksichtigt Regeln der Storyworld, die vorab von Autoren definiert wurden. Basierend auf derselben Storyworld erzeugt eine Plot-Engine für verschiedene Benutzerverhalten also verschiedene Story-Abfolgen. Die Repräsentation der Story-Elemente erfolgt im Projekt SPIRIT mit vorproduzierten Videos, Ton und Text, die mit dem Kamerabild des Tablet überlagert werden (AR). Alle Story-Elemente und ihre Repräsentation müssen vorab erstellt werden. Das Ziel im Projekt SPIRIT ist es auch, ein Authoring-Framework zur Verfügung zu stellen, mit dem Inhalte von Nicht-Programmierern, wie z.B. Mediengestaltern, eingepflegt werden können. Zu diesem Zweck wird im späteren Verlauf des Projekts ein Autorentool erstellt.

## **2.2 Verwandte Anwendungen**

Eine Übersicht über den Trend zu AR Spielen gibt [BLH+08]. „Haunted Planet“ betreibt mobile AR für Schlossbesuche mit „Geister-Kontakt“. Im Gegensatz zu SPIRIT beschränkt sich diese Anwendung auf das einmalige Geister-„Aufspüren“ ohne Story-Zusammenhang. Mobile Urban Drama [HKG12] ist eine dänische Gruppe aus Theaterdarstellern und Entwicklern, die verschiedene mobile ortsbasierte Anwendungen im historischen und kommerziellen Kontext erstellt haben. Ohne Einsatz von AR unterstützen dabei reale Schauspieler die Anwendung für Shops und Betriebe vor Ort.

Eine typische Herausforderung im Interactive Storytelling ist das ad-hoc Anpassen eines Geschichtenverlaufs (Plot) an Nutzereingaben. Zwei Beispiele hierfür sind Façade [MS05] und Office Brawl [MSS13]. Beide Anwendungen sind hochinteraktiv, d.h. Nutzer greifen mit einer hohen Frequenz in das Geschehen ein. Beide werden durch Texteingaben gespielt, auf die virtuelle Figuren (Charaktere) sofort verbale Antworten geben. Bei Office Brawl soll vom Spieler ein Konflikt zwischen zwei virtuellen Mitarbeitern gelöst werden. In informellen Tests sind Nutzern dabei menschliche Stimmen glaubwürdiger erschienen als synthetische [MSS13]. In Façade [MS05] soll analog ein Ehekrach geschlichtet werden. Zusätzlich zu einer Desktop-Version wurde Façade auch in einer AR-Version entwickelt [DML+06]. Diese ist ein interaktives Drama mit einer angestrebten zwanglosen, natürlichen Interaktion. Letzteres wird durch reale Requisiten in einem physisch realen, dem Desktop Façade Setting nachempfundenen Raum realisiert.

Android verwendet Java als Programmiersprache und bietet einen direkten Zugriff auf die notwendigen Hardwarekomponenten, wie beispielsweise Kamera, Magnetsensoren und OpenGL. Anpassungen waren hierbei im Bereich der Videodarstellung notwendig, da der Android MediaPlayer keine Unterstützung für beispielsweise den Alphakanal bietet. Die Transparenz musste manuell implementiert werden siehe. Dadurch sind flexiblere Effekte und Skalierungen zur Laufzeit sowie generelle Anpassungen an unsere Anforderungen möglich.

Andere verbreitete AR-Frameworks (Wikitude, Metaio, AndAR) bieten dagegen nur bedingt nutzbare Lösungsansätze für unsere Anforderungen.

Das Video wird vom „Android MediaPlayer“ abgespielt und auf eine Textur gerendert. Diese Textur wird daraufhin mit OpenGL ES 2.0 Shadern bearbeitet und dargestellt. Beim Erreichen bestimmter Positionen wird dabei ein Referenzfoto des aktuellen Kamerabildes erstellt, das zur Positionierung des Videos auf dem Display verwendet wird. Bei Veränderungen des Blickwinkels oder der Entfernung wird das Video entsprechend skaliert und positioniert.

### 3 Interactive Digital Storytelling

Interactive Digital Storytelling folgt in seiner Bedeutung im Projekt SPIRIT einer Arbeitshypothese, bei der einzelne Nutzer durch die Interaktion mit einer Applikation eine Story „erleben“ [Spi12]. Mit Story bzw. Geschichte meinen wir das Ergebnis eines Ablaufs von Ereignissen, die nicht nur Objekte, sondern „menschliche“ Umstände, Beziehungen und Gefühle betreffen und erfunden oder historisch recherchiert („wahr“) sein können. *Interaktiv* bezieht sich auf die Erstellung des Handlungsablaufs der Story, den der Benutzer aktiv mitbestimmen kann.

#### 3.1 Interactive Storytelling allgemein

Beim Storytelling legt ein Autor eine strikte Abfolge von Szenen in einem linearen Szenario der Story fest. Beim Interactive Storytelling im Projekt SPIRIT brechen wir diese lineare Struktur auf und erstellen dagegen eine Story-World mit sinnvoll annotierten, atomaren Elementen. Auf Basis dieses Story-Modells können wir dynamisch geeignete Szenen auswählen, die einen neuen Handlungsablauf (Plot) der Story ergeben. Im Projekt GEIST [SGB+02], [KCS+01] definierte eine „Story-Engine“ nicht konkret die nächste abzuspielende Szene, sondern entschied lediglich auf Basis des Story-Modells, welche narrative (erzählerische) Funktion nach Propp [Pro58] die nächste Szene erfüllen soll. Die Entscheidung wurde an eine Scene-Engine weiter gegeben, die ihrerseits eine passende Szene auswählte. Wurde keine passende Szene gefunden, musste die Story-Engine das Story-Modell erneut konsultieren und eine alternative narrative Funktion herausuchen.

Eine Story-Engine generiert ad-hoc unter Berücksichtigung von Nutzereingaben Antworten, die eine interaktive Story ergeben. Sie plant eine Handlungsabfolge aus mehreren erzählerischen Elementen, die nicht der Reihenfolge der vom Autor vorgegebenen Story entsprechen muss [KCS+01]. Eine Story Engine koordiniert dynamisch den Ablauf einer Story auf ihrem strukturell höchsten Level [SGB+02]. Hierbei ergibt sich ein Problem für Autoren, die meist Nicht-Programmierer sind, eine Story in geeignete, maschinell lesbare kleinere Einheiten aufzubrechen und zu annotieren. Bei [HKG12] geben Autoren alle Verzweigungen zwischen den Einheiten der Story vor. Solch ein vollständiges so genanntes „Branching“ funktioniert nur, wenn alle Entscheidungsmöglichkeiten von Benutzern im Ablauf bekannt sind. Bei umfangreichen Storys müssten sehr viele Verzweigungen konstruiert werden, um alle möglichen Nutzereingaben zu berücksichtigen. Somit müssten auch Verzweigungen für unwahrscheinliche Kombinationen von Nutzereingaben erstellt werden, die möglicherweise

niemals oder selten auftreten. Daher wird in der Fachgemeinschaft des Interactive Storytelling manuelles „Branching“ nach Möglichkeit vermieden und Verzweigungen dynamisch erzeugt.

### **3.2 Interactive Storytelling in SPIRIT**

Im Projekt SPIRIT muss die geplante Plot-Engine verschiedene Auftritte von Geistern mit Nutzereingaben unter Berücksichtigung von Vorbedingungen und Effekten für die Story koordinieren. Außerdem sollen Auftritte von Geistern – visualisiert durch Videoschnipsel – auch proaktiv initiiert werden können. Im Demonstrator werden vorerst direkte Verzweigungen (Branching) benutzt. Im Verlauf des Projekts SPIRIT wird dieser Ansatz erweitert, um dynamisch Handlungsabfolgen aus möglichst kurz erzählbaren Teilen der Story zusammenstellen zu können. Dabei wird nicht komplett auf Verzweigungen verzichtet, da im Projekt SPIRIT nicht so oft Antworten generiert werden müssen wie zum Beispiel in Façade [MS05].

Um eine Brücke zwischen den Autoren, die Nicht-Programmierer sind, und der Erzeugung von maschinenlesbaren, annotierten, atomaren Handlungen zu schlagen, werden wir im weiteren Verlauf des Projekts SPIRIT ein Autorentool erstellen.

## **4 Herausforderungen und Lösungsansätze beim Einsatz von Animation und Video in AR Anwendungen**

### **4.1 Glaubwürdigkeit in der erweiterten Realität**

Die Theorie des Uncanny Valleys [Mor05] deutet an, dass der Versuch, animierte Charaktere und Roboter visuell nahezu menschengleich und realistisch zu gestalten, einen negativen Effekt erzeugt, wenn die Animation nicht realistisch gelingt. Das Resultat ist eine geringere Glaubwürdigkeit sowie Akzeptanz. In informellen Untersuchungen sind Nutzern von Office Brawl [MSS13] aufgenommene, menschliche Stimmen glaubwürdiger erschienen als synthetische Stimmen. In AR Façade [DML+06] wird nach eigenen Aussagen versucht, Glaubwürdigkeit durch einen visuellen Abstraktionsgrad (Cartoon) zu erzeugen. Im Projekt SPIRIT wollen wir eine glaubwürdige Erfahrung durch die ad-hoc Kombination von kurzen, live gedrehten Videoschnipseln für die Repräsentation von aneinander gereihten Story-Elementen erreichen. Da der Effekt alberner Spukgeschichten vermieden werden soll, wurden Cartoon-Repräsentationen im Vorfeld von den Partnern abgelehnt. Die resultierende Glaubwürdigkeit der Charaktere soll im Projekt durch Entwicklung von darauf abgestimmten non-linearen Video-Produktionsverfahren verbessert und anschließend evaluiert werden.

### **4.2 Darstellung von virtuellen Charakteren in der erweiterten Realität**

In diesem Abschnitt werden verschiedene Aspekte der Darstellung der Geistmetapher in unserem SPIRIT Demonstrator beschrieben. Damit die AR Repräsentation der nicht realen Geister möglichst konsistent sind, wollen wir Logikbrüche für den Besucher vermeiden. Für den ersten

Entwurf eines Geister-„Stils“ wurden zunächst Eigenschaften identifiziert, die Geistern zugeschrieben werden, um sie in unsere Geistmetapher mit aufzunehmen: Unschärfe, Transparenz, Schweben und Wabern, sowie eine leicht violette Färbung.

Alle Auftritte der virtuellen Charaktere müssen als Video vorproduziert werden. Dabei wird der Nachteil gegenüber Animationen in Kauf genommen, dass Videos weniger Freiheitsgrade wie z.B. Transformation, Skalierung und Translation erlauben, und deswegen nur schwer an Änderungen wie Blickwinkel dynamisch angepasst werden können. Weiterhin müssen alle Möglichkeiten dafür vorab eingeplant werden, um in einer Produktion ausreichend Videomaterial zu erstellen. Unser „Geist“ im ersten Demonstrator schaut immer in eine vorgegebene Richtung, beim Dreh meist in die Kamera und in der AR App wird er auf einem Billboard abgebildet, um immer den Benutzer anzusehen. Versucht der Benutzer um den Geist zu gehen, wird dieser immer in seine Richtung ausgerichtet.

Der Benutzer könnte für seinen Input mehr Zeit benötigen, als das Videomaterial lang ist. Um Stillstände der Videos zu vermeiden wurden nach dem Vorbild von Office Brawl [MSS13] neutrale Videos produziert, in denen der Geist merklich auf den Benutzer wartet. Unterschiedliche Sequenzen aus Video und Ton in unserem Projekt können nicht ad hoc ohne Brüche in Bild und Ton aneinander gereiht werden wie vergleichsweise reiner Text, da Körperhaltung und Stimmlage von Aufnahme zu Aufnahme variieren. Unsere Vorüberlegungen für eine glaubwürdige Geistmetapher hierzu waren den Darsteller des Geists in der immer gleichen Körperhaltung jedes Video beenden und beginnen zu lassen, was aber beim Video-Dreh nicht realisiert werden konnte. Auf dieser Grundlage musste das Anschluss-Problem in der Postproduktion der Video Schnipsel gelöst werden. Wir brauchten Zwischensequenzen im Video und Ton, die zu jedem Zeitpunkt im Videomaterial auftreten können. Als glaubwürdige Erklärung für diese Zwischensequenzen benutzen wir neben der Geistmetapher die Metapher des „magischen Equipment“ als Übertragungsmedium mit variablem Empfang. Ähnlich dem gestörten Empfang von Fernseher und Radio, bei dem Störeffekte wie Verzerrungen und Aussetzer in Bild und Ton auftreten, erzeugten wir mittels Filter der Postproduktionssoftware Störungen im Videomaterial (siehe Abb.2). Diese Störungen fügen wir ad hoc zu Beginn jedes Video Schnipsels ein, um konsistente Übergänge zu schaffen.



Abbildung 2: Effekte für den „gestörten Empfang bei einer Geistübertragung“

### **4.3 Interpretation von Bewegungen im realen Raum und Nutzereingaben in AR**

In diesem Abschnitt werden verschiedene Möglichkeiten der Interpretation von Bewegungen der Benutzer im Raum und Eingaben der Nutzer diskutiert. Die aktuelle Umsetzung im SPIRIT Demonstrator sowie die geplante Umsetzung im Projekt SPIRIT wird erläutert.

Zu Beginn des SPIRIT Demonstrators haben wir uns ähnlich der App „Haunted Planet“ [BLH+08] für ein „Aufspüren“ unseren Geists entschieden. Dem römischen Geist wird eine zufällig bestimmte, dem Benutzer aktuell nahe GPS-Koordinate zugewiesen. Nähert sich der Benutzer dieser Koordinate, beginnt das „magische Equipment“ die „Verbindung zum römischen Geist“ aufzubauen und das „Signal zum Geist“ wird immer stärker. In diesem Schritt wird das halbtransparente Video schrittweise in das Kamerabild eingeblendet und an einen bestimmten Punkt im Kamerabild verankert (markerloses Tracking), um dadurch eine erweiterte Realität (AR) zu schaffen. Damit Geister zukünftig an der richtigen Stelle und Größe im Setting dargestellt werden, wollen wir im Projekt SPIRIT das Einblenden der vorproduzierten Videos an bestimmten historischen Schauplätzen durch markerloses Tracking auslösen.

Wie auch in AR Façade [DML+06] umgesetzt, wollen wir Entscheidungen auf dem Story-Level mit einer Story-Engine realisieren; somit sind die Verzweigungspunkte für den Benutzer nicht offensichtlich. Im Projekt SPIRIT wollen wir diese Nebenläufigkeit durch bereits implementiertes markerloses Tracking unterstützen, sowie durch weitere Eingabemöglichkeiten wie z.B. Sprache.

Bei AR Façade [DML+06] wurde festgestellt, dass die meisten der 80 Teilnehmer in einer Demonstration sprachen, ohne zu den virtuellen Charakteren zu sprechen. Sie überlegten laut oder sprachen zu den Entwicklern oder den Zuschauern. Wenn man nun die Sprache und Bewegungen der Benutzer in eine AR Erfahrung einbeziehen will, steht man somit vor zwei Problemen. Zum einen muss man bestimmen, ob eine Handlung vom Benutzer für die Story gedacht war, und zum anderen, ob man die außerhalb der Story gemachten Eingaben interpretieren will. Kommentare oder Hilfen könnten parallel zur Story angezeigt werden. Im Projekt SPIRIT wollen wir das magische Equipment auf diese Art und Weise nicht nur Metapher für ein Übertragungsgerät nutzen, sondern auch als Begleiter, der für Fragen und Anmerkungen offen steht und Hilfestellung geben kann. Auf einen Modus für ein Tutorial außerhalb des eigentlichen Erlebnisses könnten wir dadurch verzichten.

### **4.4 Interaktion mit realen Gegenständen in AR Räumen**

Eine andere Frage betrifft, wie man mit Interaktion mit realen Gegenständen im Raum umgeht, da reale Gegenstände nicht von virtuellen Charakteren, jedoch vom Benutzer manipuliert werden können. In den ersten aus AR Façade gezogenen Lehren [DML+06] wird der Vorteil beschrieben, dass die benötigte Infrastruktur des AR Systems durch allgemeines Referenzieren von realen Objekten im Raum vereinfacht wurde. Das System interpretiert somit keine detaillierten Interaktionsformen des Benutzers mit Objekten, wie ansehen, anfassen, nehmen, benutzen, weglegen.

Das Problem, dass AR Charaktere keine realen Gegenstände manipulieren können, bleibt noch ungelöst. In AR Façade nehmen die AR Charaktere Weingläser auf, indem ihre Handhaltung so verändert wird, als ob sie ein Weinglas halten: Das reale Weinglas bleibt stehen und die Hand des AR Charakters bleibt leer. Es wäre auch denkbar gewesen, dass ein animiertes, virtuelles Weinglas aus dem Nichts entsteht, sobald Getränke zubereitet wurden.

Der Benutzer kann reale Objekte in der erweiterten Realität (AR) manipulieren. In AR Façade [DML+06] werden ferngesteuerte Computer als Telefon und Anrufbeantworter verwendet, sodass beim Abnehmen des Hörers der Benutzer die Stimme aus dem realen Telefon hören kann, bzw. bei Abrufen einer Nachricht eine Audiodatei im realen Anrufbeantworter ausgegeben und nicht über den üblichen Audio-Output abgespielt wird. Die Macher von AR Façade [DML+06] erhoffen sich davon ein realeres Erlebnis für den Benutzer. Im Projekt SPIRIT wollen wir in Zukunft ähnlich diesem Ansatz reale Objekte in die AR Erfahrung unserer App einbeziehen.

Bei Interaktionen zwischen Benutzern und virtuellen Objekten oder Charakteren fehlt haptisches Feedback, deswegen halten die Macher von AR Façade [DML+06] eine solche Interaktion für unnatürlich für den Benutzer. Als Lösung für dieses Problem im Projekt SPIRIT schlagen wir hier einen Reality Mix vor, in dem ein virtuelles Objekt ein reales Objekt durch markerloses Tracking überlagert. Am Beispiel des Weinglases beschrieben könnten Benutzer ein reales leeres Weinglas nehmen, das mit einer Aufnahme eines gefüllten, virtuellen Weinglases überlagert wird. Als Interaktion leert man das virtuelle Weinglas nach jedem zum Mund führen nach und nach.

## **5 AR-Konzept**

### **5.1 Anforderungen an die Entwicklungsplattform**

Die Voraussetzungen für die Darstellung von videobasierter Augmented Reality auf mobilen Geräten ist die plattformabhängige Unterstützung von Videos auf Texturen. Die hierfür benötigten Methoden werden ab Android 4.0.3 (Min. Api Level 15) zur Verfügung gestellt. Weitere Voraussetzungen sind die Unterstützung von OpenGL ES 2.0, Sensorik (Kompass, Lagesensoren und GPS), Kamera sowie die notwendigen Codecs für die Wiedergabe der Video- und Audioformate. Zwar kann die hohe Gerätefragmentierung den Entwicklungsprozess verlängern, allerdings traten Probleme nur im hardwarenahen Bereich, z.B. OpenGL und vereinzelt auf. Laut Android 4.4 Compatibility Definition Document (Stand: 27. November 2013) ist die Unterstützung diverser Codecs, wie z.B. VP8 verpflichtend und kann daher als vorausgesetzt betrachtet werden.

### **5.2 Eingesetzte Verfahren des Image- und GPS-Trackings**

Ziel des Projekts SPIRIT ist es, videobasierte Augmented Reality berührungsfrei zu realisieren. Bei der Implementierung der Trigger erwies sich eine Kombination aus GPS- und Kompassdaten



als zu ungenau. Alternative Vorgehensweisen mussten entwickelt und getestet werden. Daher wurde ein Image-Tracking-Verfahren entwickelt, das die Kamerabilder vor Ort mit Vergleichsbildern der Ziele auf Gemeinsamkeiten hin überprüft. Markante Merkmale, sogenannte Keypoints, werden dabei aus dem Bild extrahiert und abgeglichen. Die Fläche, die dabei für jeden Keypoint betrachtet wird, nennt man Patch. Für jeden dieser Keypoints des Kamerabilds wird ein Gegenpunkt im Vergleichsbild gesucht. Je größer die Übereinstimmung, desto kleiner die Differenz (distance). Hierzu wurde der Algorithmus ORB (Oriented FAST and Rotated BRIEF [RRK+11]) der Open-Source Bibliothek OpenCV (Computer Vision) verwendet. Ausschlaggebend für das Ergebnis sind somit insgesamt vier Werte; Anzahl der zugewiesenen Vergleichspunkte, minimale Differenzen (Min\_dist), maximale Differenzen (Max\_dist) und durchschnittliche Differenzen (Avg\_dist). Allerdings erwies sich ein ungefilterter Abgleich der Vergleichspunkte als sehr unzuverlässig, da sehr viele falsche Zuordnungen stattfanden und die Ergebnisse somit nicht aussagekräftig waren. Daraufhin wurden drei Filtermethoden implementiert. Im ersten Schritt erfolgt ein „symmetrischer Abgleich“. Es sollen nur Vergleichspunkte, die beim Vergleich von Bild 1 mit Bild 2, aber auch in entgegengesetzter Reihenfolge übereinstimmen, betrachtet werden. Vergleichspunkte ohne diese Eigenschaft wurden entfernt. Folgend wurden im zweiten Schritt Vergleichspunkte mit schlechten Bewertungen eliminiert. Dazu zählen diejenigen, deren Max\_dist über dem doppelten Wert des Min\_dist liegt. Jedoch kann es auch hierbei zu Zufallstreffern kommen, wie z.B. auf dem Bild befindliche ähnliche aber nicht angepeilte Elemente, wie im Hintergrund befindliche Fensterrahmen. Um auch diese Fehlerquelle zu entfernen, wurde der RANSAC Filter [FB81] verwendet. Entfernte Punkte können somit als Ausreißer erkannt und gelöscht werden.

Maßgebliche Parameter für das Ergebnis und Performance des Image-Trackings ist die Anzahl der Keypoints und die Auflösung der Vergleichsbilder. (Tab.1) zeigt einen Ausschnitt der Testtabelle für Messwerte mit variierten Keypoints. Die Bilder hatten dabei eine Auflösung von 400x300 Pixel für das Kamerabild und 175x395 Pixel für das Vergleichsbild.

Messungen	Keypoints	Minimale Zeit	Maximale Zeit	Durchschnitt
100	500	439 ms	1246 ms	468 ms
100	750	527 ms	701 ms	571 ms
100	2500	1120 ms	1437 ms	1194 ms
100	7500	1604 ms	1941 ms	1699 ms

Tabelle 1: Auszug aus den Testergebnissen der zeitlichen Messung bei ansteigenden Keypoints

Ab 2500 Keypoints konnten zuverlässig erfolgreiche Zuweisungen erfolgen. Zudem ist es notwendig, die übereinstimmenden durchschnittlichen Differenzen (Avg\_dist) als Parameter festzulegen. Experimentell erwiesen sich im Innenbereich Zuweisungen ab einem Wert von 30 Avg\_dist, im Außenbereich ab 35 als erfolgreich.



Abbildung 3: Ausschnitte aus Original mit hervorgehobenem Patch:  
links (400x30px) mit 8x8 Pixel Patch, Mitte (1600x120px) mit 64x64 Pixel Patch, rechts  
(1600x120px) mit 8x8 Pixel Patch

Die gewählte geringe Auflösung des Vergleichsbildes hat zwei Vorteile, eine deutlich bessere Performance beim Ermitteln der Keypoints sowie eine geringere Fehleranfälligkeit bezüglich der Eindeutigkeit der Keypoints. Dies liegt daran, dass bei kleinen „Patchgrößen“ bei der Berechnung der Keypoints bei geringer Auflösung viele Informationen einbezogen werden können siehe (Abb.3, links). Bei größeren Auflösungen müssten für vergleichbare Ergebnisse sehr große Patchgrößen verwendet werden siehe (Abb.3, Mitte), da ansonsten nur sehr wenige Informationen berücksichtigt werden können siehe (Abb.3, rechts). Da bei hoher Auflösung exponentiell mehr Pixel beachtet werden müssen und die Qualität der Ergebnisse auch bei reduzierter Auflösung gegeben ist, wird die Auflösung generell gegebenenfalls automatisch verringert. Entsprechend werden ausschließlich Bilder in geringer Auflösung verwendet.

(Abb.4) stellt das eingesetzte Verfahren visuell beim Vergleich des Haupteingangs des Römerkastells Saalburg dar. Hierbei wurden 8 Vergleichspunkte (Row\_Count) erkannt. Das beschriebene Verfahren erwies sich als zeitaufwändig, da lediglich 1-4 Bilder pro Sekunde verglichen werden konnten. Zur Steigerung der Performance kann per GPS eine Filterung der vorher definierten Bilder stattfinden, und somit ein Vergleich mit lediglich in nahem Umkreis liegenden Zielen erfolgen.



Abbildung 4: Darstellung der Zuweisung von Vergleichspunkten

Bei bereits im Voraus festgelegten Vergleichsbildern sind dynamische Umwelteinflüsse zu berücksichtigen, wie tageszeitabhängige Lichtverhältnisse, Witterungsbedingungen oder auch jahreszeitliche Faktoren beim Pflanzenwachstum. Dies sollte bei der Auswahl der

Vergleichsobjekte berücksichtigt werden. Daher werden für den Außenbereich mehrere angepasste Vergleichsbilder verwendet. Im Innenbereich kann dies vernachlässigt werden.

## **6 Fazit und Ausblick**

In diesem Beitrag haben wir verschiedene Design-Ideen und Konzepte für videobasierte AR diskutiert und Lösungen genannt, die im Projekt SPIRIT in einem ersten Demonstrator umgesetzt wurden oder im weiteren Verlauf noch umgesetzt werden sollen.

Durch eine glaubwürdige Metapher der Interaktion mit „Geistern“ sowie mit einem „magischen Equipment“ sollen möglichst durch Design-Entscheidungen eventuelle bleibende technische Begrenzungen abgepuffert werden. Darüber hinaus werden Interactive Storytelling Lösungen entwickelt, die eine dynamische Anpassung von vorgefertigtem Video-Material an verschiedene Situationen von Nutzern ermöglichen.

Das beschriebene AR-Verfahren eignet sich, um ohne Platzierung eigens erstellte Marker (QR-Codes) erfolgreich Videos zu triggern. Videodateien können so an Georeferenzen gebunden werden, ohne den historischen Standort Saalburg selbst zu verändern. Damit konnte den Anforderungen des Römerkastells Saalburg nachgegangen werden, da aufgrund des Status des Kastells als UNESCO-Weltkulturerbes keine eingreifenden Änderungen möglich sind.

Die Performance der Videodarstellung auf den von uns benutzten mobilen Geräten reicht aus, um auch HD-Videos mit üblichen 24 Frames pro Sekunde darzustellen. Empfehlenswert ist die Beobachtung der weiteren Entwicklung des Android-Mediaplayers bzgl. der Unterstützung des VP8 Alpha-Kanals (<http://wiki.webmproject.org/alpha-channel>), da wir dadurch einen direkten Zugriff auf den Alpha-Kanal erhalten und somit die Dateigröße der Videos verringert wird. Speziell berücksichtigt wurde ein möglichst hoher Grad an Unabhängigkeit der Ausgabehardware. Die evolutionäre Entwicklung über PC, Laptop, Netbook und schließlich zum Tablet, Smartphone und „See-through glasses“ wird noch nicht als abgeschlossen angesehen [LFM+14]. Derzeit werden Tablets als Ausgabemedium verwendet, zukünftig könnten aber auch Smartphones, „See-through glasses“ und weitere „Wearables“ berücksichtigt werden.

## **Danksagung**

Das Projekt SPIRIT wird unter dem Förderkennzeichen 03FH035PA3/B3 vom BMBF gefördert. Dank gilt K. Stöbener und dem Saalburgmuseum für die Unterstützung bei der Inhaltsgestaltung.

## **Literatur**

- [BLH<sup>+</sup>08] Wolfgang Broll, Irma Lindt, Iris Herbst, Jan Ohlenburg, Anne-Kathrin Braun, and Richard Wetzel. Toward Next-Gen Mobile AR Games. *IEEE Comput. Graph. Appl.* 28, 4 (July 2008), 40-48, 2008.
- [DML<sup>+</sup>06] Dow, S., Mehta, M., Lausier, A., MacIntyre, B., & Mateas, M. Initial lessons from AR Façade, an interactive augmented reality drama. In *Proceedings of the 2006*

*ACM SIGCHI international conference on Advances in Computer Entertainment Technology*, p. 28, 2006.

- [FB81] Martin A. Fischler and Robert C. Bolles: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In *Proceedings of Communications of the ACM* 24 (6), pp. 381–395, 1981.
- [HKG12] Frank Allan Hansen, Karen Johanne Kortbek, and Kaj Grønbaek. Mobile Urban Drama: interactive storytelling in real world environments. *New Review of Hypermedia and Multimedia*, Taylor & Francis, 18(1-2):63-89, 2012.
- [KCS<sup>+</sup>01] Ursula Kretschmer, Volker Coors, Ulrike Spierling, Dieter Grasbon, Kerstin Schneider, Isabel Rojas, and Rainer Malaka. Meeting the Spirit of History. In *Proceedings of the International Symposium on Virtual Reality, Archaeology and Cultural Heritage, VAST 2001*, Glyfada, Greece, pp. 161-172, 2001.
- [LFM<sup>+</sup>14] D. Lanman, H. Fuchs, M. Mine , I. McDowall, M. Abrash. Put on your 3D glasses now: the past, present, and future of virtual and augmented reality. In *Proceedings SIGGRAPH '14*, 2014.
- [Mor05] Mori, M., On the Uncanny Valley. *Proceedings of the Humanoids-2005 workshop: Views of the Uncanny Valley*. Tsukuba, Japan, 2005.
- [MS05] Michael Mateas and Andrew Stern. Structuring Content in the Façade Interactive Drama Architecture. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE 2005*, pp. 93-98, 2005.
- [MSS13] Wolfgang Müller, Ulrike Spierling, and Claudia Stockhausen. Production and Delivery of Interactive Narratives Based on Video Snippets. In *Interactive Storytelling, Proceedings of ICIDS 2013*, Springer LNCS 8230:71-82, 2013.
- [Pro68] Vladimir Propp. *Morphology of the Folktale*. The American Folklore Society and Indiana University (Translation 1968, orig. 1928), Second Edition, University of Texas Press, 1968.
- [RRK11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of IEEE Computer Vision (ICCV 2011)*, pp. 2564-2571, 2011.
- [SGB<sup>+</sup>02] Ulrike Spierling, Dieter Grasbon, Norbert Braun, and Ido Iurgel. Setting the scene: playing digital director in interactive storytelling and creation. *Computers & Graphics*, Vol. 26(1):31-44, 2002.
- [Spi12] Spierling,U. ‘*Implicit Creation*’ – *Non-Programmer Conceptual Models for Authoring in Interactive Digital Storytelling*. Dissertation. University of Plymouth, Plymouth, UK, 2012.

# Real-Time Global Illumination im Vergleich: Analyse aktueller Algorithmen und Optimierung durch Blurred Reflective Shadow Maps

Valentin Kraft\*, Sina Mostafawy\*, Martin Panknin†

\* Fachbereich Medien der FH Düsseldorf

Josef-Gockeln-Str. 9

40474 Düsseldorf

valentin.kraft@online.de

sina.mostafawy@fh-duesseldorf.de

† redPlant GmbH

Elisabethstr. 101

40217 Düsseldorf

martin.panknin@redplant.de

**Abstract:** Realistische Beleuchtungsdarstellung im Echtzeitbereich hat in den letzten Jahren immens an Bedeutung gewonnen. Durch den Einsatz im VR-/AR-Bereich und in der immer größer werdenden Spieleindustrie sind eine Vielzahl an neuen Methoden und Algorithmen erforscht und umgesetzt worden. Ziel dieser Arbeit ist es, diese Thematik tiefgehend zu evaluieren und mögliche Erweiterungen zu erforschen. Daher haben wir folgende zweistufige Vorgehensweise gewählt, deren Ergebnisse in dieser Arbeit zusammengefasst werden: Die Arbeit gibt eine Übersicht über aktuell relevante Verfahren, indem eine repräsentative Auswahl an Algorithmen nach ihren Stärken und Schwächen analysiert und geordnet wird. Daraufhin wird eine neue, die temporale Kohärenz verbessernde Erweiterung des Reflective-Shadow-Maps-Algorithmus vorgeschlagen, die im Gegensatz zu bisherigen Verfahren einfach implementierbar ist, und eine Erweiterung des Imperfect-Shadow-Maps-Algorithmus, die eine alternative Form des Point Sample Renderings vorschlägt, präsentiert.

**Keywords:** Global Illumination, Blurred Reflective Shadow Maps, Real-Time Rendering

## 1 Einleitung

Die effiziente algorithmische Nachbildung des physikalischen Lichttransports ist für die Computergrafik essentiell, jedoch auch überaus komplex. *Global Illumination* (GI) ist dabei eine wichtige Komponente, um die Computergrafik plausibel erscheinen zu lassen, da sie die meisten Lichtphänomene impliziert. Dabei muss sie oftmals in Echtzeit generiert werden, um interaktive Simulatoren betreiben zu können.

Aktuelle Real-Time-GI-Algorithmen sind oftmals hochkomplex und besitzen feine Unterschiede in ihren Möglichkeiten und Eigenschaften, die häufig nur Experten bekannt oder auf Anhieb ersichtlich sind. Mitunter ist die Auswahl eines geeigneten Real-Time-GI-Algorithmus für einen spezifischen Anwendungsfall somit nicht trivial.

Um die Auswahl eines geeigneten Algorithmus in der Praxis zu erleichtern, werden in einem ersten Schritt einige repräsentativ ausgewählte Real-Time-GI-Algorithmen kurz charakteri-

siert und ihre jeweiligen Stärken und Schwächen umrissen. Dies macht eine Gliederung der Algorithmen nach ihren Eigenschaften und generellen Ansätzen möglich.

Anschließend wird in Kap. 4 eine eigene Methode zur Verbesserung der temporalen Kohärenz von Reflective Shadow Maps [DS05] präsentiert, die im Gegensatz zu den bisherigen Ansätzen außerordentlich schnell umsetzbar ist. Abschließend wird eine Änderung des Imperfect-Shadow-Maps-Algorithmus [RGK<sup>+</sup>08] präsentiert, die eine Alternative zum Point Sample Rendering durch Point Primitives bietet.

## 2 Related Work

### 2.1 The State of the Art in Interactive Global Illumination

[RDGK12] geben in ihrer Arbeit eine differenzierte und ausführliche Übersicht über den aktuellen Stand der Forschung (State of the Art) im Bereich der interaktiven Global Illumination. Sie umreißen dabei die grundlegenden Theorien, Methoden und Algorithmen und vergleichen diese in verschiedenen Kategorien miteinander. Darüber hinaus resümieren sie über offene Probleme im Bereich der Real-Time-GI-Forschung und geben somit einen guten Gesamtüberblick über das gesamte Forschungsfeld.

### 2.2 Reflective Shadow Maps

Der Reflective Shadow Maps (RSM) Algorithmus [DS05] ist eine GPU-basierte Technik zur effizienten Generierung von VPLs aus Sicht der direkten Lichtquellen. Für jede direkte Lichtquelle der Szene wird eine RSM erstellt. Diese ist eine spezielle Shadowmap, die neben den üblichen Tiefeninformationen zusätzlich die World-Space-Positionen, Normalen und Reflected-Flux-Informationen aller sichtbaren Oberflächen speichert. Sie kann so interpretiert werden, dass jedes RSM-Textel ein VPL repräsentiert. In der Praxis beleuchten jedoch nicht alle VPLs jeden Oberflächenpunkt. Vielmehr werden diese für jeden Oberflächenpunkt durch ein Importance Sampling ausgewählt. Jedes VPL strahlt dabei mit dem Reflected Flux, sprich der einmal diffus reflektierten Energie der direkten Lichtquelle, in die Szene. Aus der Beleuchtung eines jeden Oberflächenpunktes durch eine genügende Anzahl an VPLs, die ein Deferred Rendering erfordert, ergibt sich somit die durch die direkte Lichtquelle hervorgerufene indirekte Beleuchtung.

### 2.3 Imperfect Shadow Maps

Imperfect Shadow Maps (ISMs) [RGK<sup>+</sup>08] bilden eine effiziente Möglichkeit, indirekte Schatten für VPL-basierte Verfahren zu simulieren, indem sie eine Sichtbarkeitsprüfung pro VPL vollziehen. ISMs sind niedrig aufgelöste Shadow Maps, die pro VPL generiert werden.

Da das Rendern einer vollständigen Shadow Map pro VPL aufgrund der großen Anzahl der VPLs in Echtzeit nicht möglich wäre, greifen Ritschel et al. [RGK<sup>+</sup>08] auf eine Punktwolke als alternative Szenenrepräsentation zurück, die sich durch ein Sampling der gesamten

Geometrie ergibt. Diese Punktsamples werden auf die ISMs verteilt und innerhalb dieser gerendert. Daraus ergeben sich lückenhafte Shadow Maps, die in einem „Push & Pull“-Verfahren [MKC08] näherungsweise rekonstruiert werden. Auch wenn sich dadurch sehr grobe Shadow Maps ergeben, reichen diese aufgrund der großen Anzahl an VPLs für eine visuell überzeugende Simulation der indirekten Schatten aus.

### **3 Übersicht ausgewählter Real-Time-GI-Algorithmen**

Ausgehend von einer tiefgehenden Analyse einiger ausgewählter Algorithmen lassen sich diese charakterisieren und ihre jeweiligen Stärken und Schwächen herausstellen. Die aus [RDGK12] entnommene Tab. 1 reduziert diese Erkenntnisse auf eine Bewertung in fünf Kategorien und kann somit eine zusammenfassende Übersicht liefern. Sie wurde nach steigendem Implementierungsaufwand sortiert, um zu zeigen, dass, zumindest für diese Auswahl an Techniken, ein Zusammenhang zwischen dem Implementierungsaufwand und der erreichten GI-Qualität besteht.

#### **3.1 Charakterisierung nach Stärken und Schwächen**

##### **3.1.1 Screen-Space Directional Occlusion (SSDO)**

Die SSDO-Technik [RGS09] ermöglicht die zum jetzigen Zeitpunkt schnellste, da im Screen-Space berechnete, GI-Approximation. Dadurch eignet sie sich insbesondere für rechenschwache Systeme. Ein weiterer Vorteil des SSDO-Algorithmus ist der geringe Implementierungsaufwand, weshalb sich dieser auch für zeitkritische Projekte eignet.

Die Screen-Space-Berechnung bedingt dabei teils grobe Artefakte, wie z.B. fehlerhafte indirekte Beleuchtung. Diese Artefakte und die Beschränkung auf die rein diffuse Lichtreflexion erlauben somit nur eine geringe GI-Qualität. Die SSDO-Technik kann somit tendenziell eher nicht alleinige Quelle einer qualitativ anspruchsvollen GI-Simulation sein. Gut geeignet ist sie jedoch für den Einsatz als Zusatz- bzw. Unterstützungslösung für andere GI-Algorithmen, wie z.B. dem Cascaded Light Propagation Volumes (CLP) Verfahren.

##### **3.1.2 Image Space Photon Mapping (ISPM)**

Das Image Space Photon Mapping (ISPM) Verfahren [ML09] basiert auf dem Photon-Mapping-Algorithmus und erbt von diesem eine hohe physikalische Genauigkeit. Durch den teilweisen Gebrauch von Raytracing ermöglicht sie eine unter den hier aufgeführten Techniken einzigartige Vielfalt an möglichen Lichttransportwegen und simulierbaren Phänomenen. Neben hochgradig spekularen Oberflächen, beliebigen BSDFs, reflektiven und refraktiven Kaustiken ermöglicht sie sogar Phänomene wie die refraktiven Kaustiken indirekter Beleuchtung. Darüber hinaus bietet sie eine beliebig hohe Anzahl indirekter Reflexionsschritte. Probleme hat der ISPM-Algorithmus z.B. bei dünner Geometrie. Darüber hinaus können innerhalb des Originalalgorithmus nur Punktlichter und kleine Area Lights simuliert werden.

### 3.1.3 Reflective Shadow Maps (RSMs)

RSMs [DS05] sind ein Ansatz zur effektiven Verteilung der VPLs über die GPU-Rasterisierung. Sie sind schnell implementierbar, für vollständig dynamische Szenen geeignet und simulieren die Global Illumination effizient. Sie finden Anwendung in den CLP-, VCT-, SII-, ISPM- und ISM-Algorithmen.

Aufgrund ihrer Beschränkung auf die rein diffuse Reflexion und die Ignorierung der sekundären Sichtbarkeit sind sie nicht in der Lage, qualitativ hochwertige GI-Simulationen zu generieren.

Es ist eine Vielzahl von optimierenden Erweiterungen des RSM-Algorithmus bekannt, die vor allem die VPL-Auswahl und die Singularitäten, die durch einzelne ungünstig positionierte VPLs entstehen können, zu verbessern suchen.

### 3.1.4 Imperfect Shadow Maps (ISMs)

Der ISM-Algorithmus [RGK<sup>+</sup>08] ermöglicht die Sichtbarkeitsprüfung der VPLs und damit die Simulation indirekter Schatten in Echtzeit. Im Vergleich zum RSM-Algorithmus verbessert sich so die Qualität der GI-Simulation, es erhöht sich jedoch auch der Implementierungsaufwand. ISMs können zudem für die Schattenberechnung des direkten Lichts komplexer Area Lights genutzt werden.

Allerdings sind sie für nur in sehr geringem Maße spekulare Materialien geeignet und neigen aufgrund ihrer Unvollständigkeit zu *Light Leaking* oder falschen Schatten bei sehr feinen Objekten. Da die Punktrepräsentation nicht szenengrößenabhängig generiert wird, stellen große Szenen ein Problem dar.

### 3.1.5 Bidirectional Reflective Shadow Maps (BRSMs)

Die Bidirectional Reflective Shadow Maps (BRSMs) [REH<sup>+</sup>11] kombinieren RSMs und ISMs zu einem ausgereiften Algorithmus, der sowohl das indirekte Licht als auch die indirekten Schatten durch ein bidirektionales Importance Sampling bzw. die Adaptivität des ISM-Geometriesamplings exakter berechnet als die RSM- bzw. ISM-Ansätze. Dadurch sind beliebig große bzw. komplexe Szenen problemlos umsetzbar. Auf diese Weise erzeugt der BRSM-Algorithmus die derzeit qualitativ hochwertigste Echtzeit-GI-Simulation auf Basis von VPLs. Eingeschränkt wird die Skalierbarkeit einzig durch die Notwendigkeit einer Verteilungsfunktion, für deren Berechnung alle Triangles der Szene durchlaufen werden müssen. Darüber hinaus ist der Algorithmus nicht in der Lage, (stark) spekulare Reflexionen zu berechnen. Die Implementierung des Algorithmus ist durchaus komplex.

### 3.1.6 Cascaded Light Propagation Volumes (CLP)

Die CLP-Technik [KD10] basiert auf einer Voxelstruktur und ist somit eine Discrete Ordinate Method (DOM). Sie interpretiert die Lichtenergie in physikalisch korrekter Art und Weise als ein im Raum kontinuierlich verteiltes Feld. Aufgrund dessen ist die Simulation von



Methode	Speed	Quality	Dynam.	Scalab.	Implem.	Transport
RSM <sup>1</sup> [DS05]	*****	*****	*****	*****	*****	LDDE
SSDO <sup>2</sup> [RGS09]	*****	*****	*****	*****	*****	LDDE
SII <sup>1</sup> [DS06]	*****	*****	*****	*****	*****	LD{SD}E
IR <sup>1</sup> [Kel97]	*****	*****	*****	*****	*****	LD <sup>+</sup> {S D}E
ISM <sup>1</sup> [RGK <sup>+</sup> 08]	*****	*****	*****	*****	*****	LD{SD}E
ISPM <sup>2,3</sup> [ML09]	*****	*****	*****	*****	*****	L{S D} <sup>+</sup> E
CLP <sup>4</sup> [KD10]	*****	*****	*****	*****	*****	LD <sup>+</sup> {D V S}E
SSBC <sup>1,2</sup> [NED11]	*****	*****	*****	*****	*****	LD <sup>+</sup> {S D}E
BRSM <sup>1</sup> [REH <sup>+</sup> 11]	*****	*****	*****	*****	*****	LD{SD}E
VCT <sup>4</sup> [CNS <sup>+</sup> 11]	*****	*****	*****	*****	*****	LD{D V S}E

Tabelle 1: Vergleich nach Relevanz ausgewählter Real-Time-GI-Algorithmen auf Basis von [RDGK12] nach Geschwindigkeit, Qualität, Dynamik, Skalierbarkeit, Implementierungsaufwand und simulierbaren Lichttransportwegen. Der VCT-Algorithmus [CNS<sup>+</sup>11] wurde nachträglich bewertet und der Tabelle hinzugefügt. Die hochgestellten Ziffern geben den grundlegenden Ansatz des Algorithmus an (1: VPL, 2: Screen-Space, 3: Photon Mapping, 4: Discrete Ordinate Methoden (DOM)).

Lichtstreuungseffekten in Echtzeit, wie z.B. Nebel, möglich. Große und komplexe Szenen stellen hinsichtlich der Performanz kaum ein Problem dar. Spekulare Oberflächen können ohne Weiteres simuliert werden, jedoch können sie im Gegensatz zur VCT-Technik kein indirektes Licht reflektieren.

Probleme können bei sehr dünnen Oberflächen, die nicht vernünftig durch eine Zelle des Geometrievolumens abgedeckt sind, auftreten. Resultat sind *Light Leaking* und *Self-Illumination*. Darüber hinaus ist es möglich, dass es zu Fehlern im Geometrievolumen kommt, wenn Teile der Geometrie weder durch die RSM der Lichtquellen noch durch die Kamera erfasst werden. Ein weiterer, DOM-typischer, Nachteil ist der Verlust der Richtungsinformation des indirekten Lichts aufgrund der „Propagation Phase“. Das Resultat ist *Lightsmearing*.

### 3.1.7 Voxel Cone Tracing (VCT)

Das Voxel Cone Tracing (VCT) Verfahren [CNS<sup>+</sup>11] basiert ebenfalls auf einem DOM-Ansatz, nutzt allerdings einen ungewöhnlichen Hybrid-Ansatz, indem es Quasi-VPLs über eine RSM verteilt und die Lichtenergie anschließend durch ein approximiertes Raytracing, das Cone Tracing, einsammelt. So ermöglicht das VCT die bisher qualitativ hochwertigste Real-Time-GI-Simulation. Der Algorithmus produziert keine Noise; im Gegenteil sind Berechnungen umso schneller, je weicher bzw. niederfrequenter das Ergebnis sein soll. So ermöglicht die VCT-Technik Soft Shadows, „weiche“ Illumination und theoretisch z.B. auch Depth of Field auf sehr effiziente Art und Weise. Das Cone Tracing sorgt dabei dafür, dass auch hochfrequente Signale respektive Reflexionen realisiert werden können. Insbesondere

erlaubt das VCT als einzige Technik die spekulare Reflexion indirekter Highlights. Die Voxelstruktur erlaubt dabei einen Level-of-Detail-Ansatz, der für eine Verarbeitung beliebig großer bzw. komplexer Szenen sorgt. Echtzeit-Frameraten werden dabei durch die vollständig GPU-basierte Berechnung sichergestellt.

Schwierigkeiten hat der Algorithmus, ähnlich wie die CLP-Methode, bei sehr dünner Geometrie. Ein weiterer Nachteil ist der teils sehr hohe Speicherbedarf bei großen Szenen. Darüber hinaus ist auch das Cone Tracing nicht voll-automatisch; so zeigen z.B. sehr weite Cones Artefakte, während sehr enge Cones die Performanz negativ beeinflussen. Der wohl größte Nachteil des Voxel Cone Tracings ist jedoch die äußerst aufwendige Implementierung.

## 4 Blurred Reflective Shadow Maps



Abbildung 1: Oben: World-Position-, Depth-, Reflected-Flux- und Normal-Buffer der RSM. Unten: Eine leichte Weichzeichnung des Reflected-Flux- bzw. Normal-Buffers sorgt für eine sichtbare Verbesserung der temporalen Kohärenz der GI-Simulation.

Da die VPL-Algorithmen wie alle GI-Algorithmen auf Näherungen bzw. die Diskretisierung konstant verteilter Energie angewiesen sind, können sich durch eine zu grobe Abtastung sichtbare Artefakte ergeben. Im Zusammenhang mit VPL-Algorithmen wird dies meist als *Flickering* bezeichnet. Der Grund für das Flickering ist eine niedrige temporale Kohärenz; bereits kleine Änderungen an der Position oder Ausrichtung eines Lichtes führen dazu, dass sich alle korrespondierenden VPLs dieser Änderung anpassen. Je nach der Auflösung der RSM und der Anzahl der VPLs äußert sich dies in sichtbarem Flickering. Eine einfache Methode zur Verbesserung der temporalen Kohärenz ist es, einfach die Anzahl der VPLs zu erhöhen. Da dies jedoch zu erhöhtem Rechenaufwand führt, werden i.d.R. Importance-Sampling-basierte Verfahren bevorzugt, die die Auswahl der VPLs verbessern, indem z.B. VPLs mit einem großen Beitrag für das aktuelle Bildergebnis bevorzugt ausgewählt werden

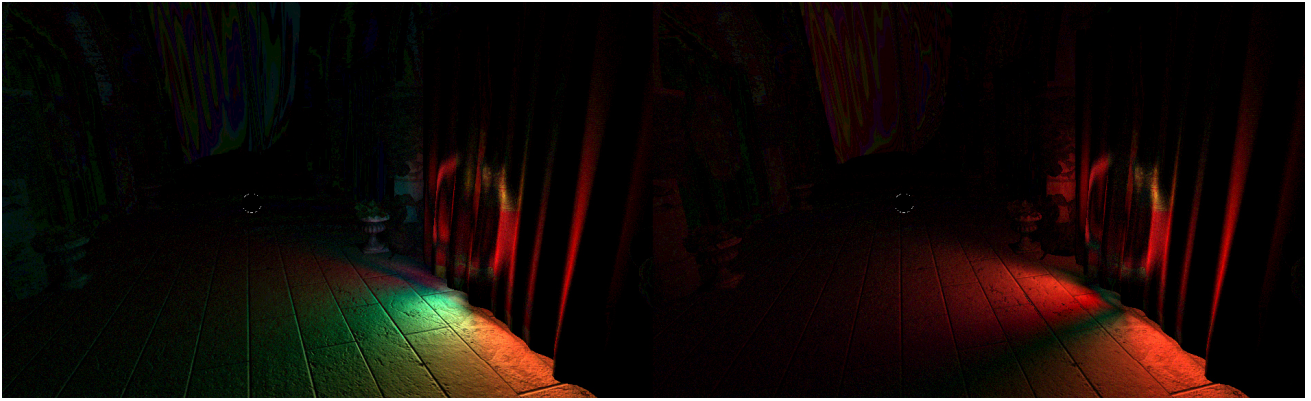


Abbildung 2: Die Differenz zweier Frames innerhalb der Bewegung eines indirekt leuchtenden Spotlights als Maß temporaler Kohärenz. Links: ohne Blurred RSM, rechts: mit Blurred RSM. Deutlich zu sehen ist, dass die temporale Kohärenz bei der Blurred RSM steigt, da die Unterschiede zwischen den Frames hier weniger gravierend und niedrigerfrequent ausfallen (Werte um den Faktor 20 verstärkt).

[REH<sup>+</sup>11]. Andere Arbeiten clustern mehrere VPLs zu Quasi-Area-Lights [PKD12] oder nutzen erweiterte Samplingverfahren [BBH13]. Diese Verfahren sind jedoch allesamt aufwendig in ihrer Implementierung. Im Folgenden wird eine schnell umsetzbare Methode vorgestellt, die die temporale Kohärenz der RSM-basierten VPLs deutlich verbessert und dafür nur sehr geringe zusätzliche Berechnungen benötigt.

Da die Texel der RSM die Position, Farbe und Orientierung der VPLs bestimmen, kann es mitunter zu starken Änderungen der VPL-Parameter bei Kanten bzw. hochfrequenten Mustern kommen. Unter diesen Bedingungen ist es möglich, dass sich der Wert eines Pixels innerhalb eines Frames stark ändert, was wiederum zu einer plötzlichen sichtbaren Änderung im Bild führt. [DS05] führen zu diesem Zweck ein Quasi-Importance-Sampling an Objektkanten ein. Hier muss an Stellen starker Änderung die RSM in höherer Auflösung generiert werden. Da bei scharfen Kanten eine erhöhte RSM-Auflösung jedoch nur wenig Verbesserung bringt, wird hiermit ein alternatives Verfahren vorgestellt.

Dieses stützt sich auf einen Blur Shader, der sowohl den Reflected-Flux- als auch den Normal-Buffer der RSM leicht weichzeichnet (Abb. 1). Da eine exakte Positionierung der VPLs auf der Geometrie erfolgen soll, darf der World-Position-Buffer hierbei nicht weichgezeichnet werden. Der Blur führt dabei zu einem weicheren Übergang an Objektkanten und verhindert somit effektiv Flickering. Da kleine Änderungen der Ausrichtung bzw. der Farbe eines VPLs visuell kaum wahrnehmbar sind und indirektes Licht i.d.R. ohnehin niederfrequent ist, entsteht somit eine von einer unbearbeiteten RSM ununterscheidbare Lösung (Abb. 3), die die temporale Kohärenz jedoch klar verbessert (Abb. 2).

Blur Shader können in effektiver Art und Weise auf der GPU implementiert werden, sodass der Rechenaufwand für eine Weichzeichnung eines Buffers äußerst gering ist. Implementiert wurden die Blurred RSMs in Unity3D; hier wurde ein schneller Unity-interner Gaußscher Blur Shader eingesetzt, der für die Weichzeichnung der 512\*512 Pixel messenden

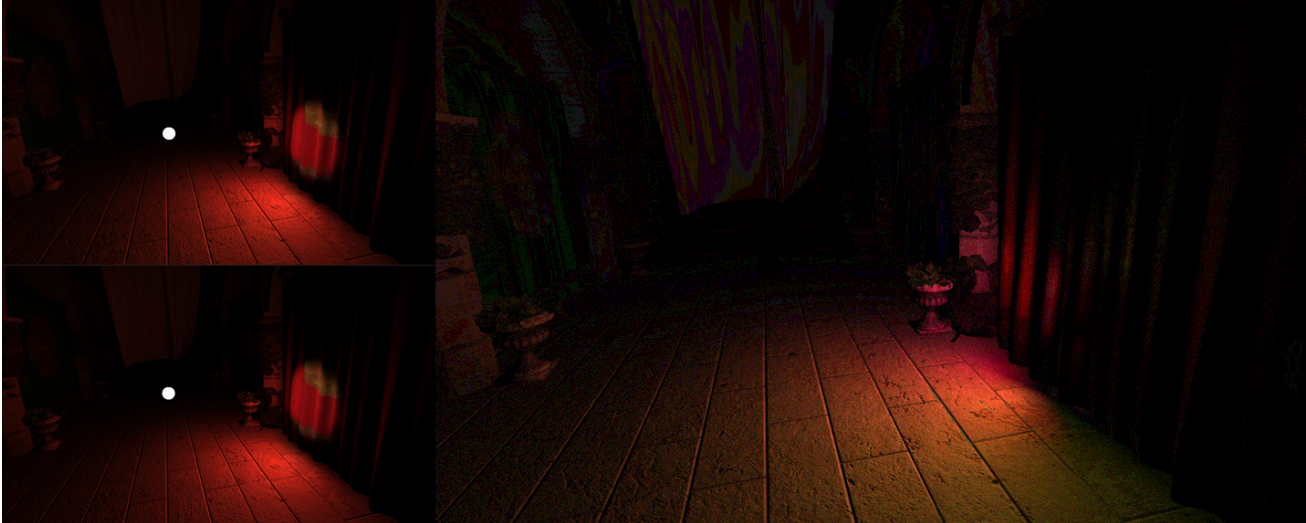


Abbildung 3: Vergleich des Renderings mit (links oben) und ohne (links unten) Blurred RSM (Blur-Radius: 1 Pixel, RSM-Größe: 512\*512 Pixel). Rechts: Differenz der beiden Bilder (Werte um den Faktor 20 verstärkt). Da indirektes Licht ohnehin niederfrequent ist, erzeugt die Verwendung eines Blur Shaders ein nur minimal anderes Bildergebnis, das von der Referenz optisch ununterscheidbar ist.

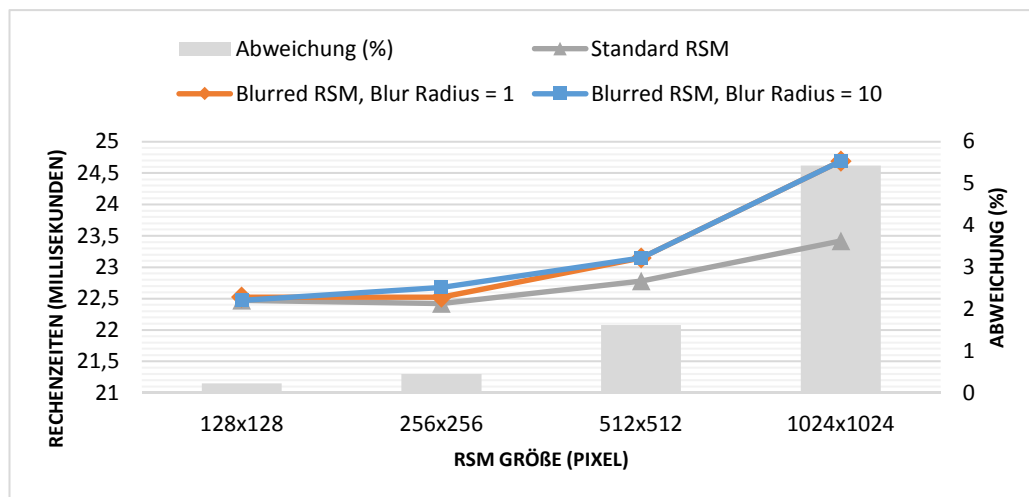


Abbildung 4: Rechenzeiten pro Frame mit Standard RSMs und Blurred RSMs im Vergleich. Gemessen wurde auf einer AMD Radeon HD 7870 in der Crytek Sponza Szene mit einer direkten Lichtquelle und 260 VPLs zur Simulation des indirekten Lichts. Gut ersichtlich ist, dass die Rechenzeiten des Blur Shaders im Wesentlichen unabhängig von dem Blur-Radius sind, mit wachsender RSM-Größe jedoch stark ansteigen. Für die üblichen RSM-Maße von 512\*512 Pixeln erhöht sich die Gesamt-Rechenzeit bei der Nutzung einer Blur Shaders jedoch nur um ca. 1,5%.

Reflected-Flux- und Normal-Buffer eine erhöhte Gesamtrechenzeit von lediglich ca. 1,5% pro Frame benötigte (Vgl. Abb. 4). Die Rechenzeiten sind dabei im Wesentlichen unabhängig von der Kernelgröße des Gauß-Filters. Obwohl größere Filterradien zu immer höherer zeitlicher Kohärenz führen, werden die Normalen- und Farbwerte bei zu großen Radien nivelliert, was zu Detailverlust durch eine Einheitsfarbe und Gleichrichtung der VPLs führt. Für RSMs der Größe 512\*512 Pixel erzeugten Filterradien zwischen 1 und 2 Pixeln die aus unserer Sicht besten Ergebnisse. Hierbei wird die temporale Kohärenz maßgeblich verbessert, während das Ergebnis von der Referenz für den Anwender ununterscheidbar bleibt (Abb. 3, 2). Dabei kann sich an Abb. 1 als optische Referenz orientiert werden.

Da bei einer Weichzeichnung die Summe der Farbwerte konstant bleibt, bleibt auch der Reflected Flux, den die Farbwerte des Buffers repräsentieren, konstant. Das Weichzeichnen des Reflected-Flux-Buffers ist somit auch eine physikalisch konsistente Operation und führt keinerlei Bias ein. Die Blurred RSMs sind somit ein schnell umsetzbares, effizientes und effektives Mittel zur Verbesserung der temporalen Kohärenz des RSM-Algorithmus.

## 5 ISM Point Rendering mit Quad Primitives

Der ISM-Algorithmus ist darauf angewiesen, die Punktsamples der Geometrie aus Sicht eines jeden VPLs jeweils in eine parabolisch verzerrte Shadow Map zu rendern. Ritschel et al. [RGK<sup>+</sup>08] nutzen dafür die *GL\_Points*-Methode, die die Samples als Punkte variierender Größe darstellt. Während Ritschel et al. [RGK<sup>+</sup>08] die Geometriesamples in einem Preprocessing-Schritt generieren, nutzen Barák et al. [BBH13] in einem alternativen Ansatz hierfür einen Tessellation Shader, der zur Laufzeit die sichtbaren Triangles sampelt und so eine effiziente Abtastung, auch in volldynamischen Szenen, ermöglicht. Bei der Verwendung eines Tessellation Shaders zur Samplegenerierung oder wenn keine entsprechende Point Rendering Methode zur Verfügung steht, stellt sich jedoch die Frage, wie die Punktsamples in eine ISM gerendert werden können.

Hilfreich kann hier die Nutzung eines Geometry Shaders sein, der in der Lage ist, ein Quad Primitive um einen Samplepunkt zu erstellen. Dies kann im World- oder View-Space geschehen, wichtig ist jedoch, dass das Quad in Richtung des negativen Forward-Vektors der Kamera und somit der VPL-Normale entgegengesetzt orientiert ist. Die Positionen der Quad-Vertices im Worldspace sind dabei durch Gl. 1 gegeben, wobei  $\vec{v}$  die Position des Geometriesamples im World-Space,  $s$  den Skalierungsfaktor des Quads,  $\delta$  die Distanz zwischen Geometriesample und aktuellem VPL,  $\vec{r}$  den right-Vektor des VPLs und  $\vec{u}$  den up-Vektor des VPLs darstellt.

$$\vec{v}_{1,2,3,4} = \vec{v} \pm \frac{s}{\delta} \cdot \vec{r} \pm \frac{s}{\delta} \cdot \vec{u} \quad (1)$$

Die Größe der Quads nimmt dabei im Gegensatz zum klassischen ISM-Algorithmus [RGK<sup>+</sup>08]

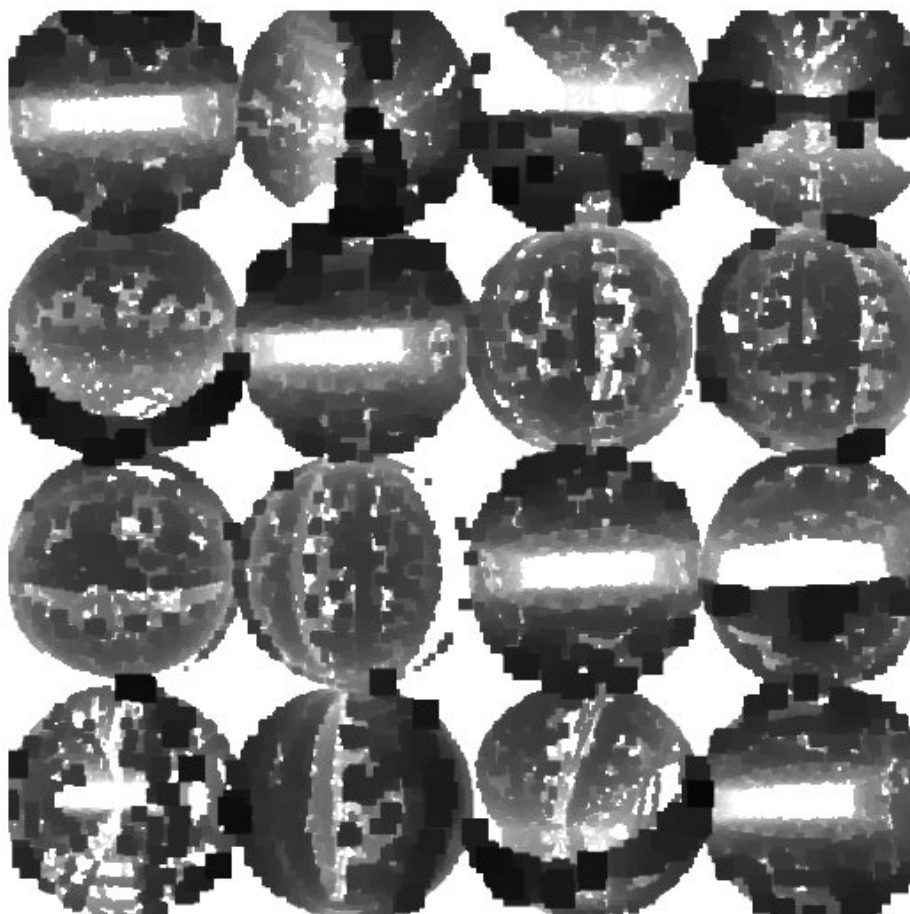


Abbildung 5: Die Punktsamples der ISMs können auch durch Quad Primitives, die innerhalb eines Geometry Shaders erstellt werden, gerendert werden. – [BBH13], S. 6.

linear statt quadratisch mit der Sample-VPL-Distanz ab. Dies führte in unserem Fall zu besseren Ergebnissen, da eine quadratische Abnahme sowohl bei sehr kurzen ( $\delta < 1$ ), als auch bei sehr weiten Distanzen zu extreme Quad-Maße lieferte. Innerhalb des Geometry Shaders kann nun auch die parabolische Projektion effizient umgesetzt werden, indem jedes Vertex des Quad Primitives entsprechend parabolisch verzerrt wird.

Das Pointsample Rendering durch Quad Primitives führt somit zu alternativen ISMs wie in Abb. 5 dargestellt, die insbesondere bei der Samplegenerierung durch einen Tessellation Shader sinnvoll ist.

## 6 Fazit

In dieser Arbeit wurde eine Übersicht über aktuell relevante Real-Time-Global-Illumination-Algorithmen gegeben und diese hinsichtlich ihrer Stärken und Schwächen bewertet, was vor allem für die Praxis relevant ist.

Außerdem wurde eine auf einer Weichzeichnung der RSM fußende Methode zur Verbesserung der temporalen Kohärenz vorgestellt, die im Gegensatz zu bisherigen Ansätzen schnell und

einfach umzusetzen ist. Blurred RSMs können bei richtiger Anwendung die Anzahl der für eine überzeugende GI-Simulation nötigen VPLs verringern bzw. Flickering reduzieren, ohne einen sichtbaren Qualitätsverlust der GI herbeizuführen.

Die Darstellung von Punktsamples durch Quad Primitives ermöglicht das Rendering von ISMs auch ohne Point Rendering Methode bzw. bei Nutzung eines Tessellation Shaders, der den ISM-Algorithmus deutlich beschleunigen und vereinfachen kann.

## Literatur

- [BBH13] Tomáš Barák, Jiří Bittner, and Vlastimil Havran. Temporally coherent adaptive sampling for imperfect shadow maps. *Computer Graphics Forum (Proceedings of EGSR 2013)*, 32(4):87–96, 2013.
- [CNS<sup>+</sup>11] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum (Proc. of Pacific Graphics 2011)*, 2011.
- [DS05] Carsten Dachsbacher and Marc Stamminger. Reflective shadow maps. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games, I3D '05*, pages 203–231, New York, NY, USA, 2005. ACM.
- [DS06] Carsten Dachsbacher and Marc Stamminger. Splatting indirect illumination. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, I3D '06*, pages 93–100, New York, NY, USA, 2006. ACM.
- [KD10] Anton Kaplanyan and Carsten Dachsbacher. 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*, pages 99–107, New York, NY, USA, 2010. ACM.
- [Kel97] Alexander Keller. Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97*, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
- [MKC08] Ricardo Marroquim, Martin Kraus, and Paulo Roma Cavalcanti. Special section: Point-based graphics: Efficient image reconstruction for point-based and line-based rendering. *Comput. Graph.*, 32(2):189–203, April 2008.
- [ML09] Morgan McGuire and David Luebke. Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of the 2009 ACM SIGGRAPH/EuroGraphics conference on High Performance Graphics*, New York, NY, USA, August 2009. ACM.

- [NED11] Jan Novák, Thomas Engelhardt, and Carsten Dachsbacher. Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 119–124, New York, NY, USA, 2011. ACM.
- [PKD12] Roman Prutkin, Anton Kaplanyan, and Carsten Dachsbacher. Reflective shadow map clustering for real-time global illumination. In Carlos Andújar and Enrico Puppo, editors, *Eurographics (Short Papers)*, pages 9–12. Eurographics Association, 2012.
- [RDGK12] Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. The state of the art in interactive global illumination. *Comput. Graph. Forum*, 31(1):160–188, February 2012.
- [REH<sup>+</sup>11] Tobias Ritschel, Elmar Eisemann, Inwoo Ha, James D.K. Kim, and Hans-Peter Seidel. Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. *Computer Graphics Forum (presented at EGSR 2011)*, 2011.
- [RGK<sup>+</sup>08] Tobias Ritschel, Thorsten Grosch, Min H. Kim, Hans-Peter Seidel, Carsten Dachsbacher, and Jan Kautz. Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Trans. Graph. (Proc. of SIGGRAPH ASIA 2008)*, 27(5), 2008.
- [RGS09] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating Dynamic Global Illumination in Screen Space. In *Proceedings ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2009.



# Dynamic Emotional States Based on Personality Profiles for Adaptive Agent Behavior Patterns

Fabian Krueger\*, Sven Seele\*, Rainer Herpers\*\*‡, Christian Bauckhage†, Peter Becker\*

\* Institute of Visual Computing  
Bonn-Rhein-Sieg University of Applied Sciences  
Grantham-Allee 20  
53757 St. Augustin  
E-Mail: sven.seele@h-brs.de

† University of Bonn  
53115 Bonn

\* York University  
Toronto, M3J 1P3, Canada

‡ University of New Brunswick  
Fredericton, E3B 5A3, Canada

**Abstract:** Realism and plausibility of computer controlled entities in entertainment software have been enhanced by adding both static personalities and dynamic emotions. Here a generic model is introduced that allows findings from real-life personality studies to be transferred to a computational model. Adaptive behavior patterns are enabled by introducing dynamic event-based emotions. The advantages of this model have been validated using a four-way crossroad in a traffic simulation. Driving agents using the introduced model enhanced by dynamics were compared to agents based on static personality profiles and simple rule-based behavior. The results show that adding a dynamic factor to agents improves perceivable plausibility and realism.

**Keywords:** intelligent virtual agents; traffic simulation; personality modeling

## 1 Introduction

While graphical fidelity continues to increase, virtual environments often suffer from unrealistic and incomprehensible behavior of computer controlled entities (agents), e.g., predictable behavior. When modeling large groups of entities, simple agents may be acceptable or necessary to meet limitations of computation resources. However, when simulating a small number of agents within the focus of a user's attention, these agents have to make plausible decisions if they are to display behavior with a high degree of realism. In gaming applications, incorrect behavior may be annoying but tolerable and predictability might even be desired, but both can be counterproductive in other cases, e.g., training or learning applications.

Agents with personality profiles potentially increase the realism and thus the level of immersion achieved by a virtual reality application. This work uses a generic model to map personality studies and profiles to a standardized form for use in software applications. This model allows real-life studies to be used when modeling behavior of autonomously interacting agents. Adding personality to an agent makes briefly observed decisions appear more consistent and more realistic. Observing a single agent for an extended period of time,

however, still reveals implausible behavior because it does not adapt to its environment. In a deterministic system a static personality always leads to the same action in the same situation. When a user observes agents experiencing events that are expected to invoke emotions, these agents are expected to adapt their behavior. To add this aspect, we introduce a model for adding emotions to agents, thus enabling adaptive behavior.

## 2 Related Work

Modeling personality and emotion is an integral part of the field of virtual humans and affective agents research. They are employed in a wide variety of applications, e.g., digital storytelling, human-computer interaction, education, and entertainment (e.g., [A<sup>+</sup>00, S<sup>+</sup>10]). In most cases, a user observes and/or interacts with a specific virtual presence over a prolonged time span; perhaps even across multiple sessions. Creating believable agents in such applications requires (a) consistent behavior based on a personality and (b) the ability to convey emotions through the agent's expressions and decisions [Ort03].

While trying to understand why people make the decisions they do, many researchers have investigated correlations between personality and performance in specific tasks (e.g., [M<sup>+</sup>00], [Her09]). Integrating personality profiles into the decision making processes of an agent can result in more plausible and realistic behavior (e.g., [SHB12]). The way personality influences the performance of a person is highly dependent on the specific task [M<sup>+</sup>00]. To model personality for virtual humans, the Five Factor Model (FFM) is most commonly used because of its descriptive nature and the fact that only five traits<sup>1</sup> are sufficient to define a personality [A<sup>+</sup>00, KMT07]. Since even within one model, different scales for each trait exist, e.g., NEO FFI [CM92] and the Big Five Questionnaire [CBBP93], the development of a generic model for computational use of personality profiles would be useful.

For emotions, the OCC model [OCC88] is the method of choice for many researchers [KMT07]. The model includes 22 types of emotions that can be either positive or negative. However, its scale makes the model complex and, depending on the application, there are alternatives for representing emotions. For example, the PANAS model reduces the number of dimensions by only considering positive and negative affect [WCT88].

## 3 Modeling Adaptive Behavior

To model adaptive behavior of agents, we propose two steps: (1) Adding a personality for consistent behavior patterns. (2) Adding a dynamic emotion model that enables adapting to events in the environment. Emotions are short-lived feelings caused by experienced events, unlike mood, which is a lasting state of feeling born from complex cognitive processes [Tha89]. Therefore, in our model, emotions are caused by predefined events, influence the behavior for a limited time, and fade until normal behavior, as defined by the personality, is restored.

---

<sup>1</sup>The five traits are: openness, conscientiousness, extraversion, agreeableness, and neuroticism.

### 3.1 Modeling Personality Profiles

If incorporated into the decision making processes, personality profiles can make agent behavior more persistent. Creating these profiles from real-life studies allows linking personalities to specific behavior patterns (e.g., driving behavior [Her09]). Then agents are not only able to act persistently but can emulate realistic behavior patterns. Thus, we introduce a model to represent complete personality studies independent of the applied personality model.

#### 3.1.1 Formal Description

There are many personality models with varying numbers of dimensions to which the personality is mapped. To be as generic as possible, the use of any number of dimensions is supported. The definition of a general personality model is given by:

$$\begin{aligned} \mathbf{p} &= \langle p_1, \dots, p_n \rangle \in \mathbb{R}^n; & D &= \{d_1, \dots, d_n\}, \forall d \in D : d : \mathbb{R}^n \rightarrow \mathbb{R} \\ P &= \{\mathbf{p}_0, \dots, \mathbf{p}_k\} \subseteq \mathbb{R}^n; & \mathcal{P} &= 2^P \setminus \emptyset \end{aligned} \quad (1)$$

A personality profile  $\mathbf{p}$  is a sequence of numbers  $p_1$  to  $p_n$ , each describing the value of a single dimension. Each dimension can be accessed by functions  $d_1$  to  $d_n$  from a set  $D$ . A finite set of profiles is denoted as  $P$  and the power set of  $P$  without the empty set is denoted as  $\mathcal{P}$ . Using the average value  $\text{avg}(P, d)$  of a dimension  $d$  over all profiles  $\mathbf{p}$  in a set  $P$

$$\text{avg} : \mathcal{P} \times D \rightarrow \mathbb{R}, (P, d) \mapsto \frac{\sum_{\mathbf{p} \in P} d(\mathbf{p})}{|P|} \quad (2)$$

and the standard deviation of the same variables

$$\text{stdDev} : \mathcal{P} \times D \rightarrow \mathbb{R}, (P, d) \mapsto \sqrt{\frac{\sum_{\mathbf{p} \in P} (d(\mathbf{p}) - \text{avg}(P, d))^2}{|P|}}, \quad (3)$$

the following equation calculates the z-score for a single dimension  $d$  of a personality profile  $\mathbf{p}$  in reference to a set of personality profiles  $P$ .

$$\text{zScore} : \mathbb{R}^n \times \mathcal{P} \times D \rightarrow \mathbb{R}, (\mathbf{p}, P, d) \mapsto \frac{d(\mathbf{p}) - \text{avg}(P, d)}{\text{stdDev}(P, d)}. \quad (4)$$

Note that  $\mathbf{p}$  does not have to be inside the set  $P$ . The z-score is an important measurement for this model, as it provides a relative measurement in reference to the average, enabling relative comparisons between values that are independent of the used personality study.

Additionally, a comprehensive model for personality study results is given by:

$$\begin{aligned} \mathcal{S} &= \langle P, K, l, h \rangle \text{ with } P = \{\mathbf{p}_0, \dots, \mathbf{p}_n\} \\ \mathcal{K} &= \langle Q, \mathbf{p}, \mathbf{z} \rangle, \text{ with } Q \subseteq P, \forall d \in D : l \leq d(\mathbf{p}) \leq h \wedge d(\mathbf{z}) = \text{zScore}(\mathbf{p}, P, d) \\ K &= \{\mathcal{K}_0, \dots, \mathcal{K}_m\}, \text{ with } \bigcup_{\mathcal{K}_i \in K} Q_i \subseteq P \end{aligned} \quad (5)$$

$$l, h \in \mathbb{R}, \forall d \in D \wedge \forall \mathbf{q} \in P : l \leq d(\mathbf{q}) \leq h$$

The study  $\mathcal{S}$  consists of a set of personality profiles  $P$  containing all the profiles  $\mathbf{p}_0$  to  $\mathbf{p}_n$  that were determined for the test subjects. A set  $K$  of classifications of profiles is also part of the study. Such a class  $\mathcal{K} \in K$  consists of a single personality profile  $\mathbf{p}$  representing the class, the tuple of z-scores of that profile  $\mathbf{z}$  and a set of personality profiles  $Q \subseteq P$  associated with this class. These classifications categorize the entire set of profiles in a much smaller set while preserving the identifying properties of those profiles. Techniques to identify such classifications can be found in the literature (e.g., [HR06]).  $l$  and  $h$  represent the theoretical minimum and maximum values of the given study (e.g., NEO-FFI: 0 to 48).

### 3.1.2 Utilizing the Model for Decision Making

How this model is used to generate plausible decisions in a computer application strongly depends on the data provided by studies. For example, a study by Herzberg indicates a correlation between three classes of personality profiles and driving behavior [Her09]. By creating profiles from these prototypes with minor random variations or by using real profiles from studies and mapping them, a set of personality profiles can be designed and assigned to autonomous driving agents to make their behavior believable. For decision making, it is reasonable to combine the numerous dimensions of a personality profile into a single value for a specific decision task. The transformation depends on the study used to correlate personality to an agent's behavior. In the given example, one classification is interpreted as aggressive, one as careful, and one as resilient. Thus, for a value expressing an agent's wariness, a continuous function would map the prototype profile of the first class to a low value, the second to a high value, and the third to a medium value.

## 3.2 Modeling Emotions

The next step is to add a dynamic component to the model in the form of emotions. It is introduced in four steps: Representing, experiencing, and fading emotions and influencing behavior. Emotions will affect behavior by temporarily influencing the personality profile. Similar to personality profiles, an emotional state is represented with multiple dimensions. The given model utilizes a two dimensional approach of positive and negative emotions. Those are used for the following reasons: They are accessible, comprehensible, and most events causing emotional changes can be mapped intuitively. They provide enough utility while keeping the complexity moderate. However, the model can be adjusted to utilize any number of emotional dimensions if necessary.

### 3.2.1 Representing Emotions

Unlike personality dimensions, a dimension of the emotional state is defined as a tuple:

$$\mathbf{d} = \langle d_1, d_2, d_3, d_4 \rangle \in \mathbb{R}^4 \text{ with } 0 \leq d_1, d_2, d_3 \leq 1, d_4 > 0 \quad (6)$$

The value  $d_1$  denotes the current value of the emotion. The other values control the fading. Because the variables that control the fading are stored into each emotion dimension, they

may differ for each dimension (e.g., positive emotions may regress faster than negative ones). The base value  $d_2$  denotes the value from which the fading started. The parameter for the linear part of the fading  $d_3$  specifies the reduction in each fading step. The parameter for the exponential part of the fading  $d_4$  controls the time the current value stays almost the same before starting to be reduced linearly as explained in Section 3.2.3. All values can be accessed using the following functions:

$$\begin{aligned} \text{current} : \mathbb{R}^4 \rightarrow \mathbb{R}, (\mathbf{d}) \mapsto d_1; \quad \text{linF} : \mathbb{R}^4 \rightarrow \mathbb{R}, (\mathbf{d}) \mapsto d_3 \\ \text{base} : \mathbb{R}^4 \rightarrow \mathbb{R}, (\mathbf{d}) \mapsto d_2; \quad \text{expF} : \mathbb{R}^4 \rightarrow \mathbb{R}, (\mathbf{d}) \mapsto d_4 \end{aligned} \quad (7)$$

A set function only sets the current and base values of  $\mathbf{d}$ :

$$\text{set} : \mathbb{R}^4 \times \mathbb{R} \rightarrow \mathbb{R}^4, (\mathbf{d}, v) \mapsto \langle v, v, \text{linF}(\mathbf{d}), \text{expF}(\mathbf{d}) \rangle \quad (8)$$

An emotional state is constructed as a tuple of dimensions  $\mathbf{d}$  from (7). In our work a two-dimensional emotional state is applied:

$$\begin{aligned} \mathcal{E} = \langle \mathbf{d}_1, \mathbf{d}_2 \rangle \in \mathbb{R}^4 \times \mathbb{R}^4 \\ E = \{\text{ne}_e, \text{pe}_e\} \text{ with } \text{ne} : \mathbb{R}^4 \times \mathbb{R}^4 \rightarrow \mathbb{R}^4, \mathcal{E} \mapsto \mathbf{d}_1, \text{pe} : \mathbb{R}^4 \times \mathbb{R}^4 \rightarrow \mathbb{R}^4, \mathcal{E} \mapsto \mathbf{d}_2 \end{aligned} \quad (9)$$

The negative emotions can be accessed with  $\text{ne}(\mathcal{E})$  and the positive emotions with  $\text{pe}(\mathcal{E})$ . Since the dimensions are independent of each other, their influence on behavior can be modeled separately for each dimension. For example, when modifying a personality based on the FFM, negative emotion could make agents more neurotic, while positive emotions do not make them less neurotic, but do make them more agreeable.

### 3.2.2 Experiencing Emotions

Emotions are experienced by an agent who is involved in an incident defined as emotionally relevant by the application. In this model, experience of an emotion is modeled as the modification of the current emotional state of an agent controlled by an incident  $\mathbf{i}$  (see (10)). It has the same number of elements as there are dimensions of emotions, containing one numeric value for each dimension to be experienced by an agent. The functions  $\text{ni}(\mathbf{i})$  and  $\text{pi}(\mathbf{i})$  are used to access the incident's values of negative and positive emotions respectively.

$$\mathbf{i} = \langle i_1, i_2 \rangle \in \mathbb{R}^2, \quad \text{ni} : \mathbb{R}^2 \rightarrow \mathbb{R}, (\mathbf{i}) \mapsto i_1, \quad \text{pi} : \mathbb{R}^2 \rightarrow \mathbb{R}, (\mathbf{i}) \mapsto i_2 \quad (10)$$

The perception of such an incident is influenced by the personality profile of the perceiving agent. As intuitively assumed and also implied by neuroticism, the dimension of emotional stability (John and Srivastava [JS99]), studies found correlations between the personality of individuals and their proneness to perceiving specific emotions (e.g., see [WC92]). With this in mind, the perception in this model is a function incorporating the personality and severity of the incident. To set how much a dimension of a personality influences which emotional dimension, an n-tuple is introduced for each emotional dimension:

$$\mathbf{s}_{ne} = \langle s_1, \dots, s_n \rangle \in \mathbb{R}^n, \quad \mathbf{s}_{pe} = \langle s_1, \dots, s_n \rangle \in \mathbb{R}^n \quad (11)$$

Here  $n$  is the number of personality dimensions used. With equation (11) it is possible that certain combinations of personality  $\mathbf{p}$  and correlation tuples  $\mathbf{s}_{ne}$  and  $\mathbf{s}_{pe}$  lead to negative experience (e.g., a perceived incident with  $ni(\mathbf{i}) > 0$  could lower the negative emotion  $ne(\mathcal{E})$  instead of increasing it). Therefore, parameters  $l_i$  and  $c_i$  are introduced in definition (12). The value  $l_i$  denotes the minimum percentage of an incident  $\mathbf{i}$  that will be experienced.  $c_i$  denotes a global scalar for experiencing emotions that provides a tool for calibrating the values generated by this model.

$$l_i \in \mathbb{R} \text{ with } 0 < l_i \leq 1, \quad c_i \in \mathbb{R} \text{ with } 0 < c_i \quad (12)$$

In (13) the function  $perceive(a, \mathbf{i})$  updates the emotional state  $\mathcal{E}_a^T$  of an agent  $a$  from the set of existing agents  $A$  according to an incident  $\mathbf{i}$  by creating a new emotional state  $\mathcal{E}_a^{T+1}$ . The new values of the emotional dimensions  $ne(\mathcal{E}_a^{T+1})$  and  $pe(\mathcal{E}_a^{T+1})$  are set to the values of the old states  $ne(\mathcal{E}_a^T)$  and  $pe(\mathcal{E}_a^T)$  increased by the maximum of the percentage of the given emotion of  $\mathbf{i}$  as set by  $l_i$  and the sum of the personality influence set by  $\mathbf{s}_{ne}$  and  $\mathbf{s}_{pe}$ .

$$\begin{aligned} & \text{perceive} : A \times \mathbb{R}^2 \rightarrow \mathbb{R}^4 \times \mathbb{R}^4, (a, \mathbf{i}) = \mathcal{E}_a^{T+1} \text{ with} \\ & ne(\mathcal{E}_a^{T+1}) = \text{set}(ne(\mathcal{E}_a^T), \text{current}(ne(\mathcal{E}_a^T)) + \max\{l_{ne}, c_{ne}\}) \\ & \quad l_{ne} = l_i \cdot ni(\mathbf{i}), \quad c_{ne} = c_i \cdot ni(\mathbf{i}) \cdot \sum_{\mathbf{d} \in D} d(\mathbf{s}_{ne}) \cdot d(\mathbf{p}_a) \\ & pe(\mathcal{E}_a^{T+1}) = \text{set}(pe(\mathcal{E}_a^T), \text{current}(pe(\mathcal{E}_a^T)) + \max\{l_{pe}, c_{pe}\}) \\ & \quad l_{pe} = l_i \cdot pi(\mathbf{i}), \quad c_{pe} = c_i \cdot pi(\mathbf{i}) \cdot \sum_{\mathbf{d} \in D} d(\mathbf{s}_{pe}) \cdot d(\mathbf{p}_a) \end{aligned} \quad (13)$$

### 3.2.3 Fading Emotions

Since emotions are handled as short-lived feelings and a direct reaction to perceived events, they also have to fade over time. Thus, a fading function is called in small intervals (e.g., 1 s) to regress the emotional state, starting when no incident  $\mathbf{i}$  was experienced in the last time step. The function, as shown in (14), takes a single emotional dimension  $\mathbf{d}^T$  and creates a new one  $\mathbf{d}^{T+1}$  for the next time step, setting the current value to the maximum of either zero, the linear fading function  $\text{lin}(\mathbf{d}^T)$ , or the exponential function  $\text{exp}(\mathbf{d}^T)$ .

$$\begin{aligned} & \text{fade} : \mathbb{R}^4 \rightarrow \mathbb{R}^4, (\mathbf{d}^T) \mapsto \mathbf{d}^{T+1} = \langle d_1, \text{base}(\mathbf{d}^T), \text{linF}(\mathbf{d}^T), \text{expF}(\mathbf{d}^T) \rangle \\ & \text{current}(\mathbf{d}^{T+1}) = d_1 = \max\{0, \text{lin}(\mathbf{d}^T), \text{exp}(\mathbf{d}^T)\} \\ & \text{exp} : \mathbb{R}^4 \rightarrow \mathbb{R}, (\mathbf{d}) \mapsto \frac{\text{current}(\mathbf{d})^2}{\text{base}(\mathbf{d}) + 10^{-\text{expF}(\mathbf{d})}}, \quad \text{lin} : \mathbb{R}^4 \rightarrow \mathbb{R}, (\mathbf{d}) \mapsto \text{current}(\mathbf{d}) - \text{linF}(\mathbf{d}) \end{aligned} \quad (14)$$

The fading process is exemplified in Fig. 1, which shows fading curves for different emotional dimensions ( $\mathbf{d}_1$  to  $\mathbf{d}_4$ ). The linear function simply subtracts the provided parameter  $\text{linF}$  from the current value. In the beginning, the exponential function reduces the value slowly because  $\text{current}(\mathbf{d})$  and  $\text{base}(\mathbf{d})$  start the fading at the same value (see  $\text{set}(\mathbf{d}, v)$  in (8)), keeping the value at nearly the same level for a period of time determined by  $\text{expF}$ . While

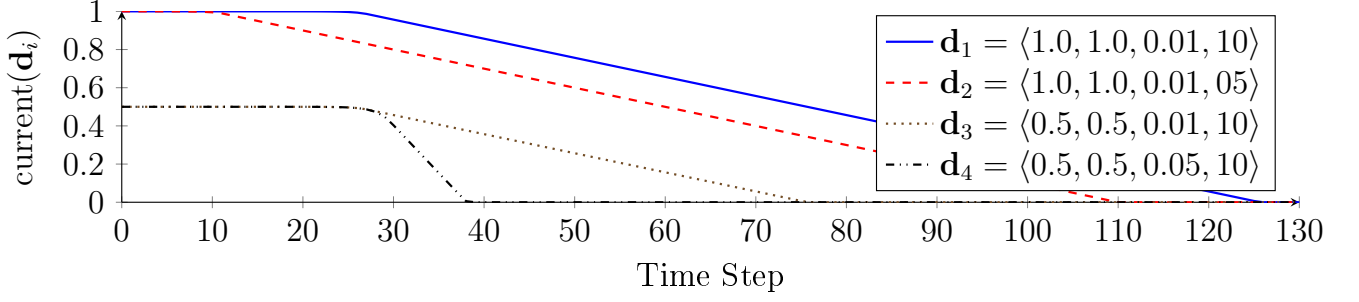


Figure 1: Fadings of emotions for emotion dimensions  $\mathbf{d}_1$  to  $\mathbf{d}_4$ .  $\mathbf{d}_2$  starts linear fading earlier than the others because of a lower value of  $expF(\mathbf{d}_2)$ .  $\mathbf{d}_4$  fades the fastest because of the highest value of  $linF(\mathbf{d}_4)$ . The fading of  $\mathbf{d}_1$  and  $\mathbf{d}_3$  is identical, only the initial values differ.

the numerator keeps getting smaller with each call of  $fade(\mathbf{d})$ , the denominator stays the same until  $set(\mathbf{d})$  is called again, which effectively resets the fading to start anew. When  $current(\mathbf{d})$  gets small enough that  $exp(\mathbf{d})$  reduces it more than  $lin(\mathbf{d})$ , the linear function takes over the fading, operating as a break to assure a steady change in the behavior.

### 3.2.4 Utilizing Emotions

To influence behavior, a function is introduced to temporarily affect a personality profile based on the emotional state. As these profiles already influence behavior, it is not necessary to change the decision making process. Also by having them integrated in the personality profiles, emotions can be switched on or off dependent on the application's needs.

The effect of emotions on a personality profile is described with n-tuples, where n is the number of dimensions of the personality model used. For each dimension of emotion, one tuple has to be created, e.g., the two n-tuples  $\mathbf{t}_{ne}$  and  $\mathbf{t}_{pe}$  in (15). The values in each n-tuple are coefficients for the current value of the corresponding emotion. The values  $d(\mathbf{t}_{ne})$  and  $d(\mathbf{t}_{pe})$  are used for calculating the effect on the personality dimension  $d(\mathbf{p})$ .

$$\mathbf{t}_{ne} = \langle t_{ne_1}, \dots, t_{ne_n} \rangle \in \mathbb{R}^n, \quad \mathbf{t}_{pe} = \langle t_{pe_1}, \dots, t_{pe_n} \rangle \in \mathbb{R}^n \quad (15)$$

To represent the influence of emotions, the static personality profile  $\mathbf{p}_a$  of an agent  $a$  is transformed into a dynamic profile  $\mathbf{b}_a = \langle b_1, \dots, b_n \rangle \in \mathbb{R}^n$  with

$$\begin{aligned} \forall d \in D : d(\mathbf{b}_a) &= \text{clamp}(d(\mathbf{p}_a) + n(\mathcal{E}_a, d) + p(\mathcal{E}_a, d), l, h), \quad l, h \in \mathcal{S} \\ n : \mathbb{R}^4 \times \mathbb{R}^4 \times D &\rightarrow \mathbb{R}, (\mathcal{E}, d) \mapsto \text{current}(ne(\mathcal{E})) \cdot d(\mathbf{t}_{ne}) \\ p : \mathbb{R}^4 \times \mathbb{R}^4 \times D &\rightarrow \mathbb{R}, (\mathcal{E}, d) \mapsto \text{current}(pe(\mathcal{E})) \cdot d(\mathbf{t}_{pe}) \end{aligned} \quad (16)$$

For each dimension  $d$  of the profile, the dynamic value  $d(\mathbf{b}_a)$  is determined by taking the static value  $d(\mathbf{p}_a)$  and adding each emotion dimension multiplied by the corresponding value of the tuples from definition (15). As this can cause values outside the boundaries set by the personality study  $\mathcal{S}$ , the function  $\text{clamp}(x, min, max)$  is used to limit the result  $x$  to valid values between  $min$  and  $max$ . The resulting dynamic  $\mathbf{b}_a$  is structurally identical to the static  $\mathbf{p}_a$  and can be used analogously in an application.

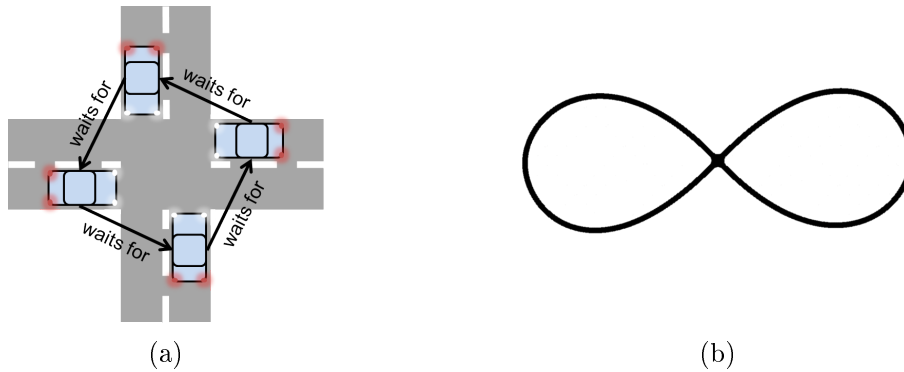


Figure 2: Evaluation scenario. (a) Deadlock situation at an unregulated four-way crossroad. If four agents arrive at the same time at all roads there is no rule to resolve this deadlock situation. (b) Layout of the road used for the test scenario.

## 4 Evaluation Scenario

To evaluate the models of dynamic personality profiles and emotions, they were applied to the specific scenario of an unregulated crossroad with the priority-to-the-right system. Problems may occur as soon as agents arrive at the crossroad from at least three sides at the same time, resulting in a deadlock (see Fig. 2 (a)).

The road network is set up as a closed system (see Fig. 2 (b)). A specified number of agents are randomly distributed across the network of 2 km (two loops of 500 m with two driving directions). The maximum velocity was set to 50 km/h. Agents are assigned profiles based on studies by Herzberg and Roth [HR06, Her09], which connected three personality prototypes to driving behavior. The prototype profiles were  $\langle 0.3, -0.12, -0.06, -0.54, -0.6 \rangle$  for undercontrolled,  $\langle 0.9, -0.75, -0.09, -0.06, -0.24 \rangle$  for overcontrolled, and  $\langle -0.84, 0.6, 0.15, 0.48, 0.66 \rangle$  for resilient agents.<sup>2</sup> Since the value bounds for these studies are not provided, the prototype profiles are randomized based on the given z-score values thus generating different personalities that still belong to one of the three classes. From the generated profiles, a “politeness” factor  $\varphi$ , adapted from [KTH07], is derived to model traffic-related decision making processes. In accordance with [HR06, Her09], the calculated politeness is low (0.22) for undercontrolled, medium (0.45) for overcontrolled, and high (0.75) for resilient agents<sup>3</sup>.

Experienced emotions influence the politeness value. Positive emotions may be invoked in an agent if another agent allows it to cross the junction. Since these emotions would only take effect after crossing the intersection and fade away before arriving there again, only negative emotions, experienced while waiting, were considered for this scenario. [WC92] found that out of the five dimensions of the FFM, only neuroticism influences the perception of negative emotions. Thus,  $\mathbf{s}_{ne}$  was set to  $\langle 0.333, 0, 0, 0, 0 \rangle$  and the other perception parameters were set to  $l_i = 0.1$  and  $c_i = 0.5$ . The time step for experiencing and fading emotion was one second. The effect of negative emotion was set to  $\mathbf{t}_{ne} = \langle 1, 0, -0.1, -0.75, -0.3 \rangle$ .

<sup>2</sup>The profile dimensions are:  $\langle$ neuroticism, extraversion, openness, agreeableness, conscientiousness $\rangle$ .

<sup>3</sup>The minimum politeness value is 0 and the maximum is 1.



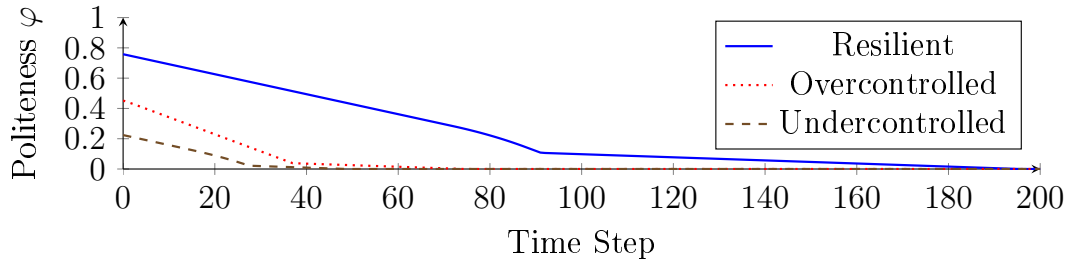


Figure 3: Politeness  $\varphi$  over time for each personality prototype when perceiving incidents while waiting. It decreases with increasing waiting time depending on the personality profile.

Three behavioral types of agents were considered: rule-based (RB), static personality-based (PB) and dynamic emotion-based (EB) agents. RB agents will strictly follow the applicable traffic rules. When detecting a deadlock, the PB and EB agents will utilize their personality profiles through the politeness factor  $\varphi$  to decide which of the involved agents will take action to resolve the deadlock. The involved agent with the highest politeness will give way to the agent waiting to its left, resolving the circular dependency graph. Additionally, the EB agents will react to waiting times by experiencing negative emotion. When waiting to cross, they perceive an “emotion incident”  $\mathbf{i}_1 = \langle 0.2, 0 \rangle$  during each time step. Fig. 3 shows the progression of the politeness  $\varphi$  of the class prototypes of agents when waiting as first in line at the crossroad. Each agent type was simulated for 60 min with 30 and 60 agents randomly distributed across the road network. Each combination of type and number of agents was simulated ten times.

## 5 Results and Discussion

When simulating RB agents, deadlocks occurred in every single run within only a few min, decreasing the flow on the road to zero (cf. [SHB12]). Therefore, only results for the PB and EB agents are reported. Fig. 4 (a) shows how long agents had to wait as first in line at the crossroad for each test run. During each test run, at least one of the PB agents waited for about 2.5 min when simulating 30 agents and up to 8 minutes when simulating 60 agents. In contrast, none of the EB agents waited more than 1.5 minutes, which is more plausible behavior. Fig. 4 (b) depicts the average times for all test runs showing that the maximum values are not outliers. Taking both graphs into account, waiting times of PB agents are overall longer than those of the EB agents. Despite their static profiles, the PB agents’ waiting times vary strongly depending on initial position on the road and routing choices at the crossroad. Because their personalities and thus their behavior were dynamically adapted, EB agents showed more consistent results across the different test runs; even doubling the number of agents from 30 to 60 hardly changes waiting times. In comparison, the waiting times of up to 8 min with 60 PB agents are implausible (see Fig. 4 (a)).

The difference in behavior can be identified by looking at the distribution of yields performed by the agents in Fig. 5. Each yield was performed by one of the agents to resolve

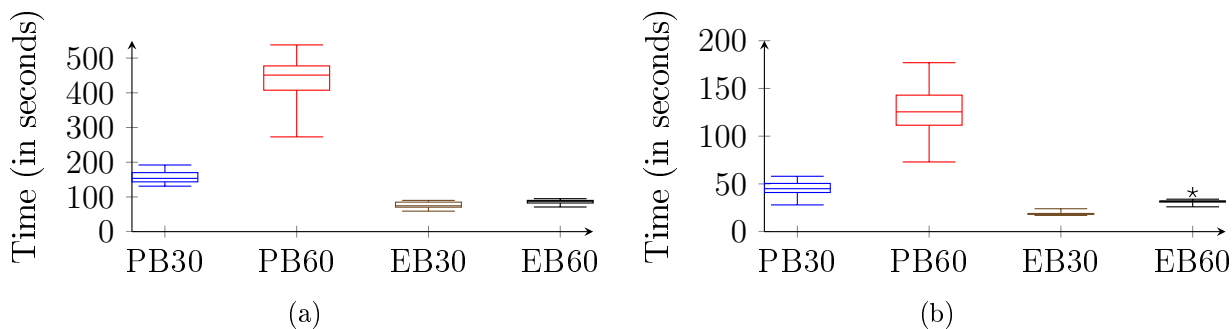


Figure 4: (a) Maximum waiting time of a single agent over all test runs and configurations. (b) Average waiting time of all agents over all test runs and configurations.

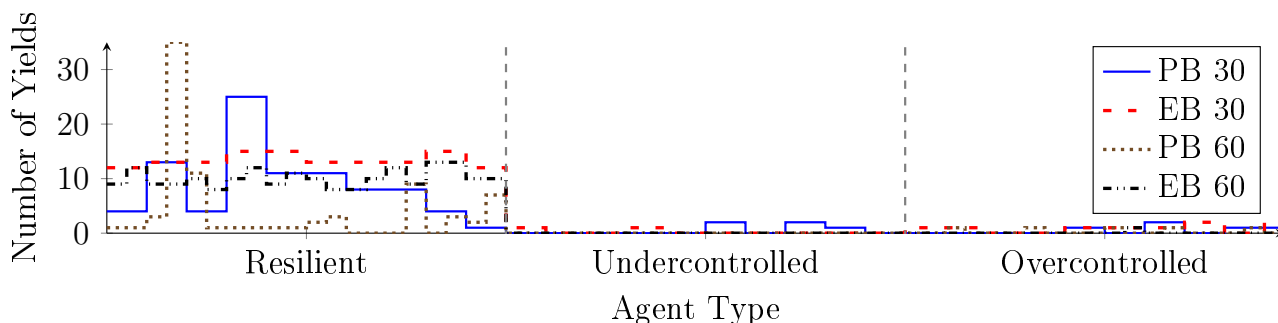


Figure 5: Yields per agent of one sample test run per scenario (PB 30, EB 30, PB 60, EB 60). Agents are sorted by personality prototype.

an identified deadlock. The graph shows the number of yields per agent performed in one test run. For clarification, agents were sorted by their personality prototype. Within every scenario the structure of the results was similar in all runs; therefore a single run was chosen for each scenario configuration (PB 30, EB 30, PB 60, EB 60) as a representative.

Resilient agents performed the most yields in all scenarios. Only in rare cases do over- or undercontrolled drivers yield. These results are consistent with our interpretation of [Her09]. However, one particular PB agent, from the group of resilient drivers, yields often while others barely yield at all. For the chosen samples, the agent with the most yields in PB 30 performed 25% of all yields and in PB 60 40%. However, for the EB agents, yields were distributed between the resilient agents. In EB 30, the agent with the most yields performed about 11% of all yields and in EB 60 6%. In an ideal distribution, each of the 10 (20) resilient agents in EB 30 (EB 60) would perform 10% (5%) of all yields.

A factor to judge the plausibility of the behavior is the number of consecutive yields of an agent. A driver that willingly increases his/her waiting time by yielding to another driver in a deadlock situation would be expected to do so only a few times before running out of patience. For all runs of the EB configurations, the highest number of consecutive yields was 2 or 3. In contrast, consecutive yields in the PB configurations were between 5 and 7 (8 and 19) for 30 (60) agents. This behavior is not only implausible but at the same time explains the longer waiting times for PB agents. Both are improved by EB agents.

## 6 Conclusion and Future Work

In this work we extended work presented in [SHB12] and argued that adding personality profiles to agents and utilizing them in their decision making process results in more plausible observable behavior. We introduced a model to map personality studies to a uniform structure for efficient use in computer applications, making the profiles and the corresponding behavior more realistic. To further improve the integration of personality profiles, an emotion model was integrated, thus allowing adaptive behavior. By adding the emotions to a background layer, the complexity of the decision making process remains unchanged.

To evaluate the proposed models, a generic traffic simulation scenario consisting of a four-way crossroad with the priority-to-the-right system has been applied. Integrating personality profiles into agents that otherwise strictly follow traffic rules was confirmed to be an advantage. While rule-based agents were not able to resolve occurring deadlocks, agents with personality profiles were able to cope with the situation, but showed different behavior. When simulating agents with static profiles, the “most polite” agent waived its right of way multiple times in a row to resolve the situation resulting in prolonged waiting times, which observers would consider implausible. When negative emotions caused by waiting at the crossroad were introduced, the agent’s emotional state changed and in turn its politeness was decreased. Thus, during the simulation of agents with dynamic profiles, deadlock-resolving yields were evenly distributed across agents of the *resilient* category, the group with the “most polite” drivers. The results showed more plausible behavior because consecutive yields by a single agent and unrealistic waiting times for that agent were prevented.

While the presented models create more plausible behavior in the given application scenario, proving that the behavior is also realistic requires comparison to real traffic data. So far we do not have access to such data. As a possible solution to this issue, subjects could provide this data by completing reference tasks in a driving simulator. Alternatively, subjects could be asked to subjectively judge the plausibility of the generated behavior. Additionally, further evaluations are necessary to show that the same positive results can be observed for other scenarios, including those not related to road traffic.

## References

- [A<sup>+</sup>00] E. André et al. Integrating models of personality and emotions into lifelike characters. In *Affective interactions*. pp. 150–165. Springer, 2000.
- [CBBP93] G. V. Caprara, C. Barbaranelli, L. Borgogni, and M. Perugini. The “big five questionnaire”. *Personality and Individual Differences*, 15(3):281 – 288, 1993.
- [CM92] P. T. Costa and R. R. McCrae. *Revised NEO Personality Inventory (NEO PI-R) and NEO Five-Factor Inventory (NEO FFI): Professional Manual*. Psychological Assessment Resources, 1992.

- [Her09] P. Herzberg. Beyond accident-proneness: Using Five-Factor Model prototypes to predict driving behavior. *Research in Personality*, 43(6):1096–1100, 2009.
- [HR06] P. Herzberg and M. Roth. Beyond resilient, undercontrollers, and overcontrollers? An extension of personality prototype research. *European Journal of Personality*, 20(1):5–28, January 2006.
- [JS99] O. John and S. Srivastava. The big five trait taxonomy: History, measurement, and theoretical perspectives. *Handbook of personality*, 2:102–138, 1999.
- [KMT07] Z. Kasap and N. Magnenat-Thalmann. Intelligent virtual humans with autonomy and personality: State-of-the-art. *Intelligent Decision Technologies*, 1(1-2):3–15, 2007.
- [KTH07] A. Kesting, M. Treiber, and D. Helbing. General Lane-Changing Model MOBIL for Car-Following Models. *Transportation Research Record*, 1999(1):86–94, January 2007.
- [M<sup>+</sup>00] G. Matthews et al. Individual differences: Personality and Mood. In *Human Performance: Cognition, Stress, and Individual Differences*, chapter 15, pages 265–285. Psychology Press, 2000.
- [OCC88] A. Ortony, G. Clore, and A. Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, Cambridge, UK, 1988.
- [Ort03] A. Ortony. *Emotions In Humans And Artifacts*, chapter On Making Believable Emotional Agents Believable. MIT Press, 2003.
- [S<sup>+</sup>10] A. Signoretti et al. Increasing the efficiency of NPCs using a focus of attention based on emotions and personality. In *2010 Brazilian Sym. on Games and Dig. Ent. (SBGAMES)*, pages 171–181, 2010.
- [SHB12] S. Seele, R. Herpers, and C. Bauckhage. Cognitive Agents for Microscopic Traffic Simulations in Virtual Environments. In *Int. Conf. on Entertainment Computing*, pages 1–8, 2012.
- [Tha89] R. E. Thayer. *Modern Perspectives on Mood*. Oxford University Press, New York, USA, 1989.
- [WC92] D. Watson and L. A. Clark. On traits and temperament: general and specific factors of emotional experience and their relation to the five-factor model. *Journal of Personality*, 60(2):441–476, 1992.
- [WCT88] D. Watson, L. A. Clark, and A. Tellegen. Development and validation of brief measures of positive and negative affect: the PANAS scales. *Journal of personality and social psychology*, 54(6):1063–70, June 1988.

# Fast Construction of SAH-based Bounding Interval Hierarchies using Dynamic Parallelism

Carl-Feofan Matthes, Adrian Kreskowski, Andre Schollmeyer, Bernd Froehlich



**Abstract:** We present a fast and efficient algorithm for the construction of a SAH-based BIH on the GPU. Our approach uses a novel asynchronous processing scheme which launches kernels for the necessary subtasks directly on the GPU. This avoids the communication overhead between CPU and GPU and optimizes the GPU’s workload. The SAH is employed for all hierarchy levels which results in very efficient BIHs. Our results show that our algorithm processes hundreds of thousands primitives at interactive frame rates. This enables interactive ray tracing of dynamic scenes even with changing geometry as is necessary for skeletal animation or adaptive tessellation.

**Keywords:** GPU, ray tracing, dynamic parallelism, spatial data structures, bounding interval hierarchy

## 1 Introduction

In virtual reality environments, the spatial perception and the degree of immersion depend on the visual quality of the rendering system. In terms of visual quality, most ray tracing systems outperform even modern rendering systems [LSB<sup>+</sup>14] [She12] because they provide global illumination effects such as reflections, shadows or caustics. In general, interactive ray tracing systems depend on a spatial data structure to minimize costs for intersection tests. In most cases, hierarchical acceleration data structures such as kd-trees, bounding volume hierarchies (BVH) or bounding interval hierarchies (BIH) are employed. However, the generation of these data structures is a costly task and it is often performed during preprocessing which prevents using ray tracing for dynamic scenes.

There are various approaches to mitigate this limitation. Multi-level hierarchies based on grids [KBS11] or a combination of BVH and kd-trees [RDS<sup>+</sup>10] have been introduced to

deal with moving, but rigid objects. Most recent approaches use modern graphics hardware to accelerate the generation process. In particular, some subtasks such as sorting can be performed in parallel on the GPU. However, frequent communication with the GPU is necessary to control the subdivision process. This communication interrupts the kernel execution and represents a bottleneck in most approaches.

In this paper, we present a GPU-based approach for the efficient generation of a full SAH-based BIH. Our approach exploits recent graphics hardware developments to eliminate the communication overhead between CPU and GPU. In particular, one of the latest key features of Nvidia graphics processors, *dynamic parallelism*, allows us to spawn new computation kernels directly on the GPU. Our implementation shows that our approach generates very efficient BIHs for hundreds of thousands primitives at interactive frame rates.

The main contribution of our approach is a novel multi-level parallel algorithm for generating a full SAH-based BIH on the GPU. Our asynchronous processing scheme avoids the communication overhead typically arising in other GPU-based implementations. The resulting hierarchies are very efficient for ray tracing because our SAH is used at all hierarchy levels. Our scheme generates BIHs on-the-fly which allows VR-applications to use ray tracing as rendering method for dynamic scenes, even if the object geometry changes, e.g. for skeletal animation or adaptive tessellation. This may highly increase visual quality, spatial perception and the degree of immersion in virtual reality applications.

## 2 Background

In general, there are a variety of acceleration data structures for ray tracing. Ray classification schemes [AK87] suffer from their enormous memory requirements and are not considered in this work. In most cases, spatial data structures such as kd-trees, BVHs or BIHs are used. However, most algorithms for the on-the-fly generation of these data structures focus either on the BVH or the kd-tree. For constructing a SAH-based BVH, Wald [Wal07] shows a fast, yet not interactive approach. Lauterbach et al. [LGS<sup>+</sup>09] present a hybrid approach which does not use a full SAH, but results in an almost optimized hierarchy. Furthermore, interactive updates (instead of rebuilding from scratch) have been used to increase performance [WBS07]. For kd-trees, there are also highly parallelized algorithms, both CPU-based [SSK07] as well as GPU-based approaches by Danilewski et al. [DPS10] and Zhou et al. [ZHWG08]. The latter both use different node stages to optimize the amount of parallelism for the corresponding node size.

When constructing an efficient acceleration data structure for ray tracing, it is important to find the best split for each node. Havran [Hav00] shows that clipping empty space in the upper levels of a hierarchy may significantly increase rendering performance. In our algorithm, a Surface Area Heuristic (SAH) is used to find the optimal splitting planes for all node stages.

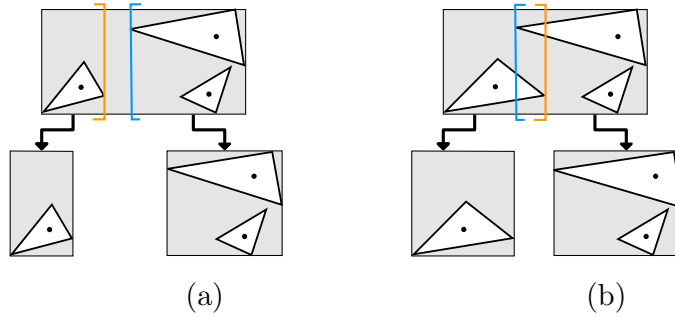


Figure 1: The Figures (a) and (b) illustrate the split of a BIH node. The bounds of the resulting child nodes are spatially adjusted with respect to the split direction and the primitives contained in the child nodes. This avoids instancing of primitives, but may result in overlapping bounding boxes, as shown in (b).

## 2.1 Surface Area Heuristic

The selection of splitting planes directly affects the quality of the acceleration data structure. The Surface Area Heuristic (SAH) proposed by MacDonald et al. [MB90] is a suitable estimate for the split quality and ray tracing efficiency. Given split candidate  $x$ , the cost function  $C(x)$  is

$$C(x) = C_t + C_i \frac{A_L(x)N_L(x) + A_R(x)N_R(x)}{A_P} \quad (1)$$

where  $A_P$  identifies the surface area of the node to be split. The surface areas of the left and the right child are represented by  $A_L$  and  $A_R$ , whereas  $N_L$  and  $N_R$  are the number of primitives in the left and right child. The constants  $C_t$  and  $C_i$  reflect the costs for traversal and intersection, respectively.

In order to accelerate the evaluation of the SAH, MacDonald et al. allow the use of a limited number of equally spaced splitting plane candidates. In our algorithm, we employ 31 splitting plane candidates per axis in order to determine the best split.

## 2.2 Bounding Interval Hierarchy

A bounding interval hierarchy is a spatial acceleration data structure which was introduced by Wächter and Keller [WK06]. While its construction properties are similar to a BVH, they report that its traversal is as efficient as for a kd-tree. The construction of BIHs is generally faster compared to other spatial data structures which makes them suitable for ray tracing of dynamic scenes. The main reason for this is that spatial partitioning schemes need to handle primitives which overlap the volume boundaries. In BIHs, the bounding boxes of a node are adjusted to contain all primitives completely, as shown in Figure 1. Thus, instancing is not required which is a major difference in comparison to kd-trees. As a result, the memory requirements can be predicted and primitives can be sorted in-place which minimizes the need for dynamic memory allocation.

```

1 WHILE nodes IN queue {
2   IF split NOT viable {
3     store (node)
4   } ELSE {
5     FOREACH axis {
6       FOREACH primitive IN node {
7         bin_primitive()
8       }
9     }
10    FOREACH bin {
11      evaluate_costs()
12    }
13    split(node)
14    FOREACH primitive IN node
15      assign primitive to left_child or right_child
16
17    FOREACH primitive IN left_child
18      update_max_bound(left_child)
19
20    FOREACH primitive IN right_child
21      update_min_bound(right_child)
22
23    enqueue_child_nodes()
24  }
25 }

```

Figure 2: General pseudo code description for building a SAH-based BIH.

The construction of a BIH starts with a single node containing all primitives. This node is recursively subdivided until the termination criteria are satisfied. The pseudo code in Figure 2 summarizes the tasks which have to be performed, most of which can be run in parallel as separate GPU kernels.

### 2.3 Dynamic Parallelism

For better understanding, we give a brief overview of the terminology used to describe our asynchronous processing scheme.

A *thread* is the smallest working unit, which maps directly to a processing unit of the GPU. Threads execute programs, referred to as *kernels*, in parallel. A collection of threads is called a *block*. In order to run kernels, one or more blocks are launched together in a *grid*. An important unit of the CUDA programming model [NVI14] that we utilize in our algorithm is the *warp*. A warp is a small set of threads (currently 32) within a block which perform the same operations in parallel at the same time and are therefore synchronized implicitly.

*Dynamic parallelism* is an essential feature of Nvidia’s graphics processors which was introduced by GK110. In contrast to prior graphics hardware, it allows a kernel to launch an own child kernel with dynamically adjustable parameters such as block and grid size, to distribute the resources according to upcoming task. Parent kernels are able to wait for the results of their own child kernels. We exploit this capability to implement a management



kernel which distributes workload according to the upcoming subtasks without interrupting the GPUs workflow by returning to the CPU to reconfigure launch parameters.

### 3 BIH Construction Using Dynamic Parallelism

This section describes the asynchronous processing scheme of our algorithm. As shown in Figure 2, the BIH construction consists of a number of parallelizable subtasks, which can be executed as separate kernels. In our system, these kernels are launched by a management kernel which represents the management layer of our algorithm. The workflow of the management kernel is shown in Figure 3.

In our management kernel, we utilize warps as functional entities for several reasons. First, warps are synchronized implicitly. Therefore, instructions can be assumed to be executed in a locked-step fashion for all 32 threads in a warp. Second, specialized intra-warp instructions, e.g. shuffle instructions, can be used to share and compute information within warps efficiently.

All BIH nodes are stored in an array which is organized in blocks of size 32. Henceforth, we will refer to such a block of nodes as *chunk*. Each chunk is associated with a *chunk header* which indicates the actual number of nodes contained in the chunk. All accesses to a chunk are protected by the corresponding chunk header.

In the management kernel, there is no synchronization between warps, which means all warps work independently of each other. Each chunk of nodes is processed by a warp. Within one warp, each thread fetches exactly one node from its chunk. Given the number of management warps  $W_L$ , the index of the current chunk  $B(w_i)$  to be processed by warp  $w_i$  is computed by

$$B(w_i) = W_L H_i + i, \quad 0 \leq i < W_L \quad (2)$$

where  $H_i$  is the current iteration of  $w_i$ , which is equal to the number of chunks already processed by  $w_i$ . After each iteration, a warp moves  $W_L$  chunks forward along the queue. Nevertheless, each warp processes only one chunk per iteration.

At the end of each iteration, memory has to be reserved to store the child nodes in consent with all other warps in the management layer. A chunk of size 32 can generate up to 64 child nodes. Therefore, a warp may need to reserve up to two chunks. The reservation of free chunks is realized by means of atomic additions to a global variable which identifies the index of the first free chunk. In general, warps reserve the appropriate number of chunks with respect to the number of child nodes produced.

In order to infer the storage location for child nodes, we compute the prefix sum across the number of child nodes of all threads in a warp. For parallel prefix sums and their applications, we refer to [Ble90]. After child nodes have been stored in the reserved chunks, the corresponding chunk headers are adjusted such that their value represents the number of valid nodes in the chunk. As soon as no more child nodes are produced, reserved chunk headers are finalized and the warp moves to the next chunk.

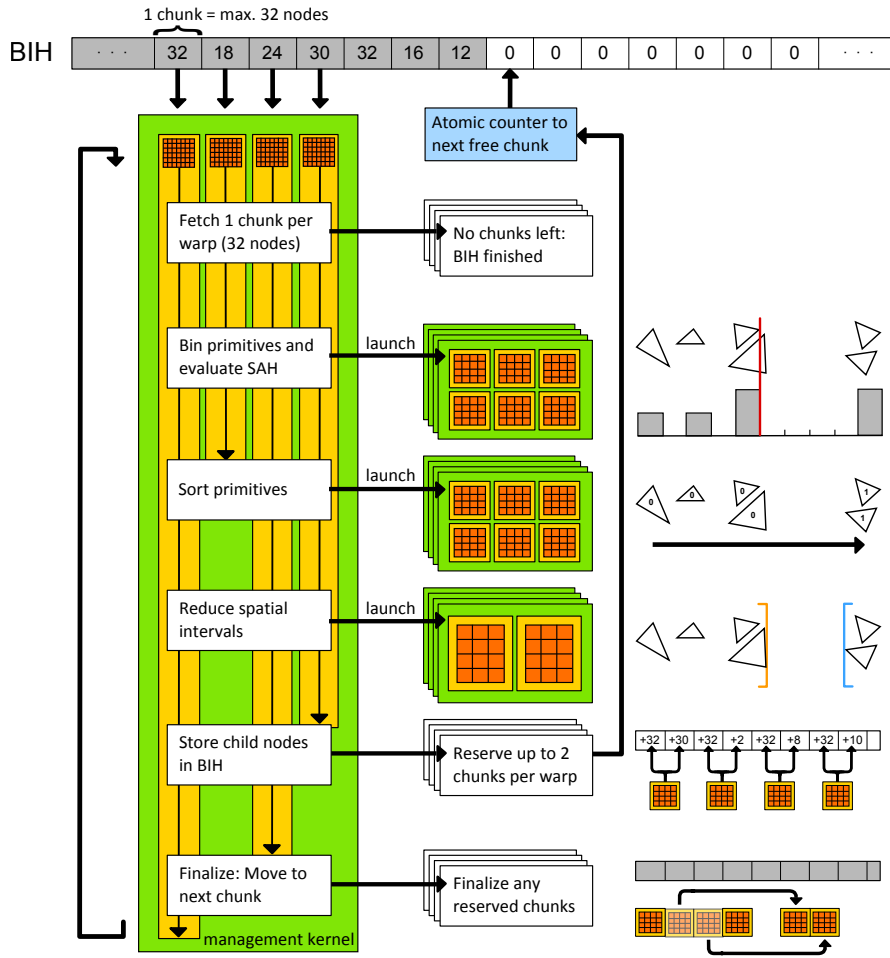


Figure 3: This Figure illustrates the workflow of our Large Node Stage. Each kernel (green) consists of thread blocks (orange). In this example, our management kernel is executed with four asynchronously working warps ( $W_L = 4$ ). Each of these warps processes up to 32 nodes in parallel and launches the necessary child kernels. After splitting, the first thread of each warp reserves chunks for storing the child nodes by means of an atomic counter (blue).

Since warps in our management layer work asynchronously and communication takes place only by means of atomic additions, it is possible that one warp stores children in a reserved chunk, while another warp already processes the nodes contained in that chunk. The amount of valid nodes stored in a chunk is determined by the corresponding chunk header. Of course, a thread reads and processes its corresponding node only if it is valid. However, a warp may end the current iteration and advance to the next chunk only if the chunk header of its current chunk has been finalized, indicating that no further nodes will be added to the chunk, and all nodes have been processed.

In summary, warps of our management layer are coupled very loosely and the BIH is constructed in a highly asynchronous manner. Its construction takes place without any explicit synchronization points. As soon as a warp has processed its current chunk, it moves on to the next one immediately. For these reasons, warps in the management layer can be located in different computational blocks and all streaming multiprocessors can be utilized from the beginning.

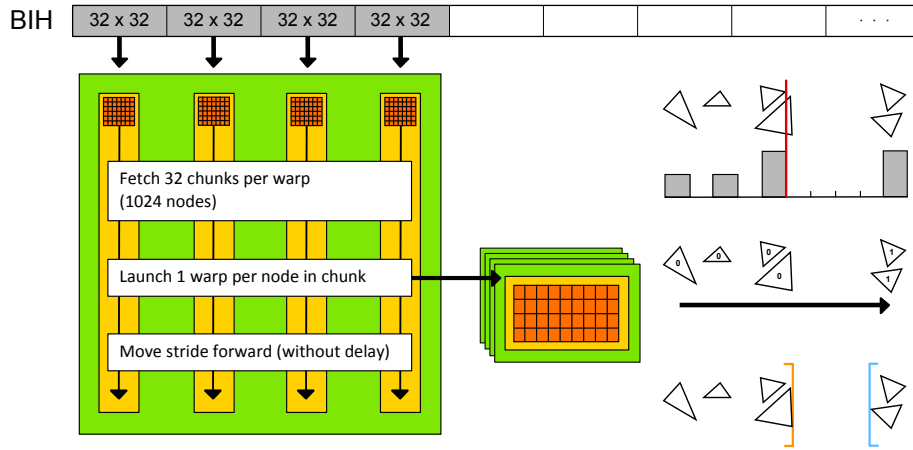


Figure 4: This Figure illustrates the workflow of our Small Node Stage. In contrast to the LNS, each warp processes 32 chunks (1024 nodes). For each node, one single warp is launched which performs all necessary subtasks.

## 4 Node Stages

As the depth of the acceleration structure increases, the amount of nodes which have to be processed grows exponentially. At the same time, the number of primitives per node decreases quickly. For this reason, we adjust the level of parallelism in order to better distribute the parallel processing power to the granularity of the problem. Similar to [ZHWG08], our management layer is organized in two node stages – a *Large Node Stage (LNS)* and a *Small Node Stage (SNS)*. Our Large Node Stage is designed to work on a relatively small amount of nodes containing many primitives, whereas our Small Node Stage invests its capacities into processing thousands of nodes with a small number of primitives per node in parallel.

### 4.1 Large Node Stage

The LNS uses dynamic parallelism extensively. As there are few nodes in the queue with very high primitive counts, three specialized kernels are launched – one for every major task of the construction process, as shown in Figure 3. The amount of parallelism of these grids is dynamically adjusted to the granularity of the tasks. First, the best split is determined by a kernel specialized to generate and evaluate the SAH for 31 splitting plane candidates per axis. This results in 32 bins per axis which can be sorted efficiently using intra-warp instructions. Similarly, a specialized kernel is launched to sort all primitives of a node in parallel with respect to the splitting plane and the splitting axis. The third kernel is optimized for the parallel adjustment of the child nodes’ bounding boxes.

### 4.2 Small Node Stage

The most important difference between our two node stages is that each warp in the management layer of the Small Node Stage fetches 32 chunks. This means that every management thread is responsible for one chunk (32 nodes) per iteration.

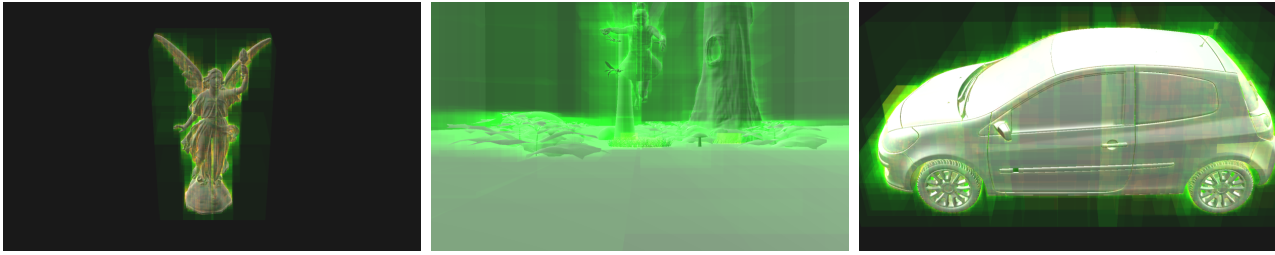
(a) *Lucy*(b) *Fairy Forest*(c) *Clio*

Figure 5: This Figure shows the models used for performance tests.

In order to process all nodes in the chunk at once, each thread launches one single grid. The number of warps in that grid equals the number of nodes in the chunk. In contrast to the LNS, all major tasks of the construction are performed in this grid, as indicated in Figure 4.

If  $W_S$  represents the number of management warps in the Small Node Stage,  $1024 \cdot W_S$  nodes are processed in each iteration. In comparison, the SNS processes considerably more nodes at once than the LNS. However, for reasons of efficiency, nodes may only enter our Small Node Stage if the number of primitives per node is less than 1024.

Another important difference between our two node stages is the location where administrative tasks, e.g. storing child nodes, are performed. In the Small Node Stage, these administrative tasks are moved into the newly launched child grid. This has the advantage that warps in the management layer can proceed to the next chunk immediately after the child grid is launched.

## 5 Results and Discussion

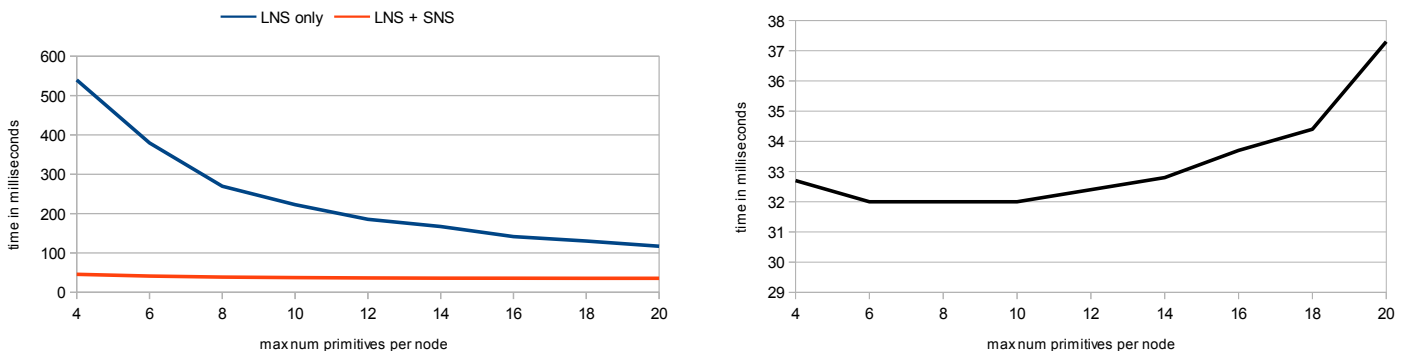
All of our results were measured on an Intel Core i7-3820 CPU with 32 GiB of RAM and a Nvidia GTX Titan GPU. For ray casting, a naïve GPU implementation with no further optimizations was used. The window resolution was set to 1664x1024.

The BIH construction time depends on the number of warps in the management layer. More specifically, we observe that the optimal construction time is achieved when the number of warps is close to the number of streaming multiprocessors (SMX) of the graphics hardware. However, since a thread block can only run on one streaming multiprocessor, we distribute all warps of our management layer into distinct blocks. In our implementation, the number of management warps for both node stages  $W_L$  and  $W_S$  was set to 16 (close to the 15 SMX of the GK110).

For performance measurements, we used the three models shown in Figure 5. Table 1 shows the corresponding number of triangles and construction times, whereas Figure 6 shows the construction and ray casting performance for different configurations for the model *Fairy Forest*. We used this model to find the optimal settings for the combined performance for construction and ray casting performance. The primitive threshold was used as an additional

Model	#Triangles	[ZHWG08]	[DPS10]	[LGS <sup>+</sup> 09]	Our approach
Lucy	79k	n.a.	n.a.	n.a.	25 ms
Fairy Forest	175k	77 ms	57 ms	488 ms	45 ms
Clio	257k	n.a.	n.a.	n.a.	56 ms

Table 1: This Figure shows the construction times of our algorithm and other approaches. Unfortunately, a direct comparison with the other approaches is not possible, because different data structures (kd-trees and BVH) and heuristics have been used and the tests were performed on slower hardware (GeForce 8800 ULTRA, GTX285 and GTX280). In comparison to our approach, only [LGS<sup>+</sup>09] and [DPS10] use a SAH for all node stages. Zhou et al. [ZHWG08] use a SAH only in the lower hierarchy levels which significantly reduces costs.



(a) BIH construction using our algorithm

(b) Ray casting performance using the BIH

Figure 6: These diagrams show the times for construction and ray casting of the model *Fairy Forest* in dependence of the maximal number of primitives per leaf node. Obviously, smaller leaf nodes are optimal for ray casting (6b). In addition, we observe that the construction time is almost independent using both node stages (6a) which proves the efficiency of our multi-stage concept.

termination criterion to avoid degeneracies and to evaluate the efficiency of our node stages.

Unfortunately, a direct comparison to existing work is quite involved. A performance comparison would require a reimplementation of other approaches, because the used graphics hardware lacks the necessary capabilities we require. Furthermore, the acceleration data structures produced by these approaches are different. Danilewski et al. [DPS10] utilize five highly optimized node stages, which seems advantageous compared to our system. However, our system avoids the communication overhead, but measuring this performance gain is also quite involved and remains future work. Zhou et al. [ZHWG08] use much slower hardware, but refrain from using a SAH for all hierarchy levels which highly reduces construction costs.

An interesting observation is that the frame rate is almost entirely dependent on the ray casting times for our models. This is because the BIH construction performs almost independent of the maximal number of primitives per node. Furthermore, we evaluated the efficiency of our node stage concept. We limited the number of maximal primitives per

node to measure the scaling of the Large Node Stage. Using only the Large Node Stage, the construction times increase exponentially, which indicates that too few threads need to handle an exponentially growing number of child nodes. Using both node stages, the construction time is almost constant with respect to the number of primitives in the leaf nodes which shows that the Small Node Stage is capable of dealing with a larger number of nodes.

Since the size of a chunk (32 nodes) is chosen relatively large, chunks are usually not completely filled. The reserved chunks of a warp are filled entirely only if the number of child nodes of an iteration is a multiple of the chunksize or zero. Therefore, the efficiency of our asynchronous management scheme comes with increased memory consumption. An evaluation of our memory requirements revealed that the overhead introduced by our management layer was in average 44% for our models.

Our work focuses on the construction of bounding interval hierarchies for triangle meshes. It is conceivable that our approach can be generalized and used for the construction of different acceleration data structures. Furthermore, our current implementation of the BIH construction can be extended to every primitive type that provides a bounding box, or even whole objects.

## **6 Conclusion and Future Work**

In this paper we explore new possibilities which dynamic parallelism provides for the generation of acceleration data structures. In our work we focus on the management layer we developed. Our algorithm exploits the implicit synchronization of threads within a warp to avoid costly explicit synchronization of thread blocks. The warps of our management layer are loosely coupled which leads to a highly asynchronous and efficient construction of a bounding interval hierarchy. Our results show that our implementation efficiently utilizes dynamic parallelism and constructs SAH-based BIHs for hundreds of thousands of primitives at interactive frame rates. This fast and complete rebuild of the acceleration data structure allows for real-time ray tracing of dynamic scenes.

Although we use only two node stages, we observe that the utilization of an additional Small Node Stage accelerates the BIH construction considerably compared to utilization of a single node stage. We conjecture that employing a higher number of stages, similar to [DPS10], would lead to even better adjustment to node granularities and increased BIH construction performance.

In this work, we construct the BIH per frame from scratch. However, for small changes in scene topology an iterative approach for updating existing hierarchies may suffice for interactive ray tracing performance. Furthermore, it is conceivable to rebuild the acceleration structure only after several frames and to update it on a per-frame basis.

The memory overhead introduced by our algorithm cannot be neglected. Reserving large chunks of memory reduces the probability of concurrent access to the atomic variables, which effectively lowers the waiting time for every working warp. It would be desirable to profile

different chunk sizes in order to find the optimal parameters that balance memory overhead against atomical access collisions.

Our results show that SAH-based BIHs can be generated on-the-fly for medium-sized scenes. This allows for interactive ray tracing of dynamic scenes which may highly increase the visual quality, spatial perception and degree of immersion in virtual reality applications.

## References

- [AK87] James Arvo and David Kirk. Fast ray tracing by ray classification. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 55–64, New York, NY, USA, 1987. ACM.
- [Ble90] Guy E Blelloch. Prefix sums and their applications. In *Synthesis of parallel algorithms*, pages 35—60. Morgan Kaufmann Publishers Inc., 1990.
- [DPS10] Piotr Danilewski, Stefan Popov, and Philipp Slusallek. Binned SAH Kd-Tree Construction on a GPU. Technical report, Saarland University, Computer Graphics Lab, 6 2010.
- [Hav00] Vlastimil Havran. *Heuristic Ray Shooting Algorithms*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000.
- [KBS11] Javor Kalojanov, Markus Billeter, and Philipp Slusallek. Two-level grids for ray tracing on gpus. *Comput. Graph. Forum*, 30(2):307–314, 2011.
- [LGS<sup>+</sup>09] Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David P. Luebke, and Dinesh Manocha. Fast bvh construction on gpus. *Comput. Graph. Forum*, 28(2):375–384, 2009.
- [LSB<sup>+</sup>14] Felix Lauer, Simon Schneegans, Andreas-Christoph Bernstein, Andre Schollmeyer, and Bernd Froehlich. guacamole - an extensible scene graph and rendering framework based on deferred shading. In *7th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS'14)*. Springer, 2014.
- [MB90] David J. MacDonald and Kellogg S. Booth. Heuristics for ray tracing using space subdivision. *Vis. Comput.*, 6(3):153–166, May 1990.
- [NVI14] NVIDIA Corporation. NVIDIA CUDA C programming guide, 2014. Version 6.0.
- [RDS<sup>+</sup>10] Martin Reichl, Robert Dunger, Alexander Schiewe, Thomas Klemmer, Markus Hartleb, Christopher Lux, and Bernd Frohlich. Gpu-based ray tracing of dynamic scenes. *JVRB*, 7, 2010.

- [She12] Denis Shergin. Unigine engine render: Flexible cross-api technologies. In *ACM SIGGRAPH 2012 Computer Animation Festival*, SIGGRAPH '12, pages 85–85, New York, NY, USA, 2012. ACM.
- [SSK07] Maxim Shevtsov, Alexei Soupikov, and Alexander Kapustin. Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. *Comput. Graph. Forum*, 26(3):395–404, 2007.
- [Wal07] I. Wald. On fast construction of sah-based bounding volume hierarchies. In *Interactive Ray Tracing, 2007. RT '07. IEEE Symposium on*, pages 33–40, Sept 2007.
- [WBS07] Ingo Wald, Solomon Boulos, and Peter Shirley. Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Trans. Graph.*, 26(1), January 2007.
- [WK06] Carsten Wächter and Alexander Keller. Instant ray tracing: The bounding interval hierarchy. In Tomas Akenine-Möller and Wolfgang Heidrich, editors, *Rendering Techniques*, pages 139–149. Eurographics Association, 2006.
- [ZHWG08] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 126:1–126:11, New York, NY, USA, 2008. ACM.



# Innovative and Contact-free Natural User Interaction with Cars

Mohammad Razavi, Saber Adavi, Muhammed Zaid Alam  
Daniel Mohr and Gabriel Zachmann

E-Mails: {razavi, sabera, zaid, mohr and zach}@informatik.uni-bremen.de

**Abstract:** Within the last two decades, the vehicle industry has majorly changed the way humans interact with cars and their embedded systems that provide aid and convenience for the passengers. Today, instead of using the ordinary physical button for each function, cars have multifunctional control devices with hierarchical menus, which demands the visual attention of the driver and also, they are getting progressively complex.

In our approach, we introduce a contact-free, multimodal interaction system for automobiles to make interactions more natural, attractive, and intuitive. We designed an interactive car driving simulation in which various car functions such as radio, windows, mirror, and cabin lights were integrated. They are controlled by a combination of speech, natural gestures, and exploiting the visibility of objects in the car. This yields a heavily decrease in visual demand and improves robustness and user experience.

**Keywords:** Natural user interaction; automotive user interfaces; gesture based interaction; speech based interaction.

## 1 Introduction

In the past 20 years, the car industry has majorly changed the way humans interact with vehicles and their embedded systems that provide aid and convenience for the passengers. Cars are more than the individual means of transport; Many people value their cars as personal spaces and they spend significant time in their cars while commuting to work. Besides the functionalities offered to operate and drive the car, vehicles have become a place for information access, communications, media consumption, and personal entertainment. Many users have become accustomed to these technologies and they do not want to miss them while driving. As most of the technology in the car is digital, cars have become interactive spaces and human factors play a central role in their design and the resulting user experience [SDKS10].

Overall, we see that these developments are an exciting trend: on one hand complex and difficult tasks are taken over by technologies and ease the primary driving task for the user. On the other hand, we see that new interaction needs arise from the use of mobile and embedded technologies in the automotive context [SDKS10]. For drivers, this means a trend towards more complicated devices that leads them away from the driving, i.e., primary task. These distractions significantly slower reaction times, made more eye movements away from the roadway and may cause car accident [LCH13]. This was one of the important motivations for this work to design a novel method to make the interaction with certain main functions of the car naturally and more intuitively to cause a less cognitive load for the driver's brain. We felt this change would make the drivers concentrate on the road better, hence it could reduce the fatal accidents. Furthermore, the driver and the task of driving should not be

the only focusing for car design, but it needs to create positive experiences for drivers and passengers as well. Thus, the driver assistance systems should be designed to empower human capabilities and maintain "the joy of driving", rather than as a means to provide assistance or help [KHL<sup>+</sup>12]. This fact motivates us to design new contact-free interactions and user experiences with the functions of the car, more attractive and interesting without touching of the components.

In this paper, we implemented a simulation software which user can naturally interact with. Trying to make it as natural as possible, we aim to use devices that user doesn't need to touch or wear. We designed a 3D car model in a way that user can interact with different parts of it such as mirrors, windows, radio, etc. Through our research, we performed user studies in order to evaluate system performance alongside its user experience by doing quantitative analysis on data.

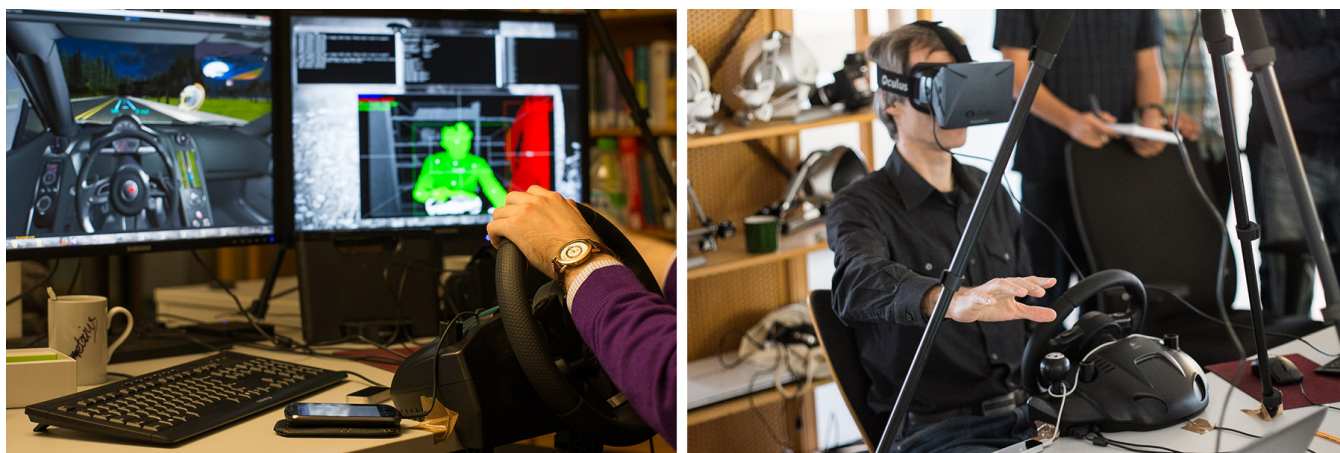


Figure 1: Operating the simulator. **On the left** user is driving in the simulator in debug mode. **On the right** user is interacting with the system while wearing the Oculus Rift.

## 2 Related Work

Cars are no longer mainly mechanical objects, rather they are complex computer systems with very particular input and output devices, and mobile functionality [TBK06]. So far, using physical buttons is the most common way to use these functions, however, the number of these control buttons must be limited.

As one solution, the car industry has tried to map the direct interaction devices down to a single multifunctional controller and a hierarchical menu structure. For instance, BMW iDrive [NDEK09], Audi MMI and Mercedes COMMAND APS are controllers that provide access to most of the functions in a unified way. This approach has been adopted from the computer domain [TBK06].

Several prototypes of the in-car systems have been proposed by researchers. In the Bullseye system [WKL12], input gestures can be made by the driver without regard to the widgets' location and he can interact with the in-car touchscreen without gazing into it to find out the exact location of the items which causes less distraction in comparison to the currently existing systems.

To reduce the driver's distraction, several solutions are pointed out by Pfleging et al. [PSS12]. They deployed an eye-tracker, which supports highlighting of the last gaze position and detects the driver's attention between the real world and the interaction with a screen in the car. That way, they could reduce the time for the attention switch. Also, they reduced the required visual attention by mounting a multitouch screen on the steering wheel [PSS12].

Asif et al. [AHB10] described a tactile information presentation in the car. They mounted vibration actuators into a belt (or probably later integrated into a seat belt) to transmit vibration patterns to driver to declare directional instructions from the navigation system. They use vibration patterns and reports differences in user preference and the measured performance of the information transmission.

With the recent advances in technology, car drivers, nowadays, are offered with a wide variety of in-vehicle systems, i.e., route guidance systems, climate controls, music players. Based on the several research studies, interacting with such in-vehicle systems, while driving highly challenges drivers' attention on the primary task of driving [JST<sup>+</sup>08, Gre04, LBCK04] and the majority of available application programs requires extensive learning periods and adaptation by the user to a high degree [AMS<sup>+</sup>01].

Based on Jæger et al. [JST<sup>+</sup>08], the aim of designing Natural User Interaction metaphors, is to plan a novel interaction structure based on gestures in which no button is touched. Besides gestures, the use of speech input provides a more robust interaction detection and also an interesting alternative for people with certain disabilities [MALR04]. We used speech synthesis, e.g., text to speech [MAR<sup>+</sup>01] and some colour indicators to show the user the state of interaction and provide system feedback.

Miller's law [Mil56] reveals that the working memory allows remembering only five to nine numbers. Thus, LaViola et al. [LK11] presume that the number of created gestures that one can remember could be around seven [JPF13]. Despite the constraints mentioned above, the operation concept should, as far as possible, be generic, easy to learn as well as interactively explorable, and, above all, intuitive [MAR<sup>+</sup>01].

Freehand gestures have not yet incorporated in the car industry for the sake of few issues. One of the problems is the ambiguity; in order to trigger a function, an activation gesture is required to prevent non-intended input. Furthermore, in most use cases, gestures induce an abstract function matching so the user needs to learn how to perform a specific action to trigger a function [BR12].

## **2.1 Our Interaction Design**

Since humans have a tendency to talk to machines [Gra03] as a natural communication way, we proposed a combination of visual and auditory input and output ways for a more robust and natural communication between the car and the driver. As input channel, freehand gestures [MEMS11] with single hand, was used so that the driver has no physical tactility to the controls while driving with the other hand. Also speech recognition was used as supplementary input with no ordinary push-to-talk button in the way that it always listens to what the driver says and is waiting for the driver's command keyword.

Using hand gestures was the key feature in our metaphors, giving the driver a new form to interact. Based on Pavlovic et al. [PSH97] the use of hand gestures provides an attractive alternative to cumbersome interface devices for human-computer interaction (HCI). Given the advancements towards self-driving, it is more than likely that self-driven cars would be the new way of travelling. When the user is free of the driving task, he could find using our

defined gestures more attractive and more natural.

At the point of output, we claim that a natural visual presentation requires a seamless integration of displays into their environment [BR12]. Therefore, head-up display was used to render virtual contents, and the vehicle’s state directly on the windshield. Considering that, if displayed virtual information occludes imminent information from the real world, or if the driver’s perception is reduced due to too much virtual information, the driver’s recognition of danger is reduced [TBK06].

### 3 System Design and Architecture

First of all, we decided to use Unity game engine because it covers most our needs such as supporting different I/O devices (Oculus Rift, Kinect, Leap Motion), multiple programming languages, network connectivity, being cross platform, etc.

We combine various input devices to get a superior motion capture accuracy which is crucial for using interaction metaphors. We used a Kinect for skeleton tracking, Leap Motions for hand posture and gesture detection and microphone with Google speech recognition API. Our overall system has a modular design. By using a middleware software, all components are able to send and receive messages to each other over the network.

Steering-wheel buttons were used to provide an alternative and middleware-independent way of performing the specified interactions for testing purposes to provide an approximated comparison between natural and conventional interactions (see Fig. 2). The middleware

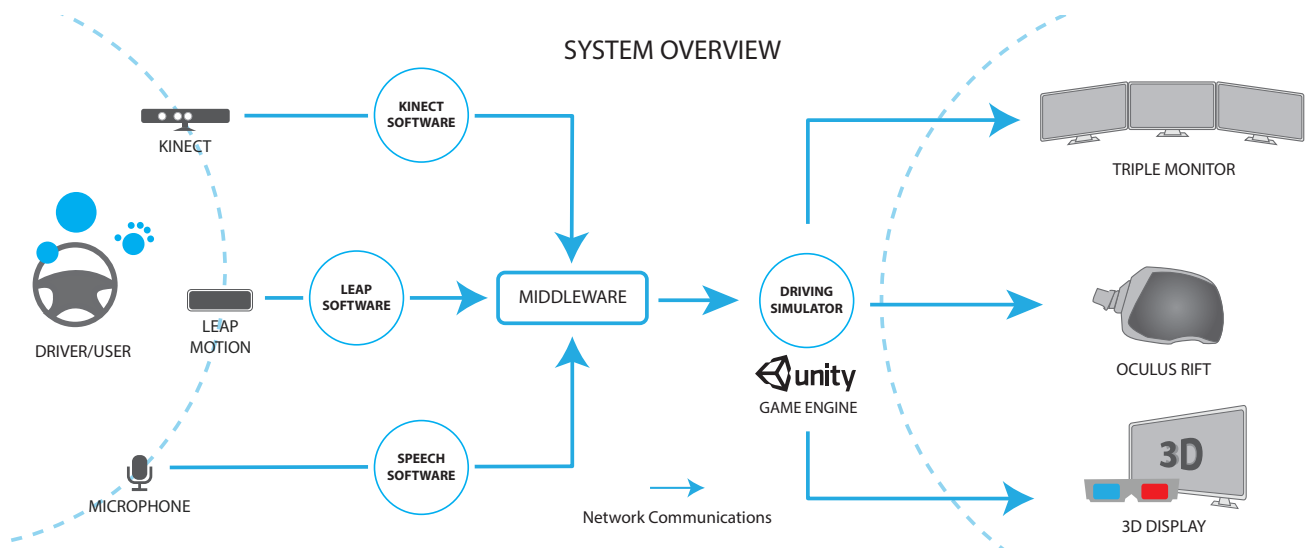


Figure 2: Overview of our virtual environment system components. The driver interacts with the input devices on the left. The middleware is an intermediary component between input devices and the game engine. The game engine renders the simulator environment to the output devices on the right.

application is developed as the centerpiece of the distributed system. It acts as an abstract middle layer that hides detail about input/output device and their applications from the game engine. It derives the interaction metaphors from received data and forwards the proper response. It is modular and configurable, i.e., in order to change the metaphors, we can alter the rules of the configuration file without recompiling the application.

We used off-the-shelf Kinect device in order to fulfill hand tracking, swipe gesture recognition, and hands pointing. We also used Leap Motion controller to perform a precise grab detection by tracking hand finger. A microphone was used to retrieve user's voice in order to recognize the voice commands.

We simulated the environment using Unity v4.2 engine as a racing game, containing the streets of University of Bremen's campus as the racetrack. To mimic the real world traffic, we added several AI cars into the scene and the collision handling between all the cars.

As the system output, three types of devices were used in different contexts. By putting the Oculus Rift - Virtual Reality 3D headset on, users can get fully immersed in the simulation environment. Apart from that, system can support a triple monitor display to provide a super wide field of view for the user in the car. Additionally, our system can be displayed on a 3D TV.

## 4 Interaction Metaphors

Generally, there are two significant groups of devices inside a car: primary and secondary. The primary devices are the ones that influence directly the heading and speed of the car including usually steering wheel and brake pedals. Interacting with these devices constitutes an adequately complex primary task that requires considerable amount of mental load. Therefore, they must be directly mapped from the real world into our driving simulator environment by using the regular gaming steering wheel device with pedals. Additionally, such primary devices must generate immediate feedback whenever they are used in a way that user feels driving in a real car with rather exact measures, e.g., rotation of the steering wheel transforms directly and uniformly into car rotation and virtual steering wheel.

The secondary tasks are not associated directly to the driving and are mandatory to keep the safety of the car. Therefore, direct access should be granted, but not as direct as for primary controls. In that case, there are more possibilities to design natural interactions to perform them. Due to the safety issues, designing secondary in-car interaction systems have to be extremely user-centred [TBK06].

These input devices cannot request drivers to use both hands at the same time in order to interact with them, since at least, one hand must be on the steering wheel while driving. So, all input devices that require both hands, such as two handed joysticks and tablet computers, are not acceptable. Furthermore, additional input equipment such as data gloves or ring mice are not suitable to do the interaction while driving.

In addition, they should require as little visual attention as possible and should not require an explicit learning phase. As a result, only limited cognitive and motor capabilities are available for the interaction task and should cause minimal distraction from the driving task.

In order to display the interaction feedback and information to the driver, we decided to show information on the windshield using head-up display (HUD) technology. Yet, it also brings new questions concerning the safety problems such as cognitive capture, cognitive tunneling [Tuf97] and its element design demands specific principles [HCTH13]. The importance of HUDs in cars will grow significantly, as soon as it is technically feasible to project large amounts of information in high resolution onto the windshield.

Speech input could be a proper interaction type in the car since it does not need visual distraction. However, it has its own drawbacks. Even though it only uses non-visual channel,

users need a feedback to know whether the system understands their voice command or not [Eck13]. In addition, speech recognition systems are not completely accurate, also, they are sensitive to background noise and people accents. We decided to integrate voice commands into our interactions as well as hand gestures, to create a multimodal approach and gain more accuracy and less false positives while they are being used together.

Since using natural interaction has become a new concept in the vehicle industry, we proposed and developed novel natural user interaction metaphor that can be used in the future in cars. To do so, we avoided using the buttons, any artificial or unnatural element for interaction. Besides, the interactions should be easily understandable and useable, minimize the cognitive load of the driver, enhance the driving experience and be intuitive.

The interactions are derived from how we commonly interact within our environment in daily life. For example, we point at an object to show our interest, then we grab it and interact with it. Considering that, no learning phase is needed for the users they can interact with car components in a freehand way.

All the interactions are backed by the visual and audio feedback to notify the driver, the current state of the system. A timeout function with configurable time length is applied in order to overcome staying in an interaction and end it.

While gestures are prone to various problems like the “Gorilla Arm Syndrome”, our metaphors avoids such problems. They are short, do not require continuous interaction and shun long interaction time.

#### 4.1 Proposed Metaphors

We chose four common secondary interactions in the car by taking limitation of time and present technology into account:

**(a). Radio/music player:** Interaction begins by pointing at the car radio/music player for two seconds. Doing the grasp gesture with one hand and moving it up or down will increase or decrease the volume. Likewise, moving the hand to left or right will go to previous or next track. Opening the hand indicates the end of the interaction (see Fig. 3). User has to use the speech commands to pause or play the music.

**(b). Three mirrors in car:** Interaction begins by pointing at the target mirror (rear, left or right mirror) and using mirror’s name as voice command to select it. In order to make the mirror follow user’s hand movements (s)he should grasp and move the hand up or down and left or right. The mirror will rotate to the corresponding directions. The interaction ends by opening the hand.

**(c). Two windows:** Likewise, interaction triggers by pointing at left or right windows, saying their name, doing the grasp gesture and pulling the hand up or down. It will pull the windows up or down. The interaction ends by opening the fist.

**(d). cabin light:** The voice command “cabin light on” will turn the light on, and for turning it off user has to say “cabin light off”.

Due to the limitations of the first generation Kinect, we used Leap Motion to achieve a sufficient finger tracking accuracy which is crucial for grasp and fist opening detection.

#### 4.2 Interface For Natural User Feedback

The human machine interaction occurs in the "medium" user interface. It should be fluid, iterative [Nie93] and its learning curve should be as less as possible to make it natural to the

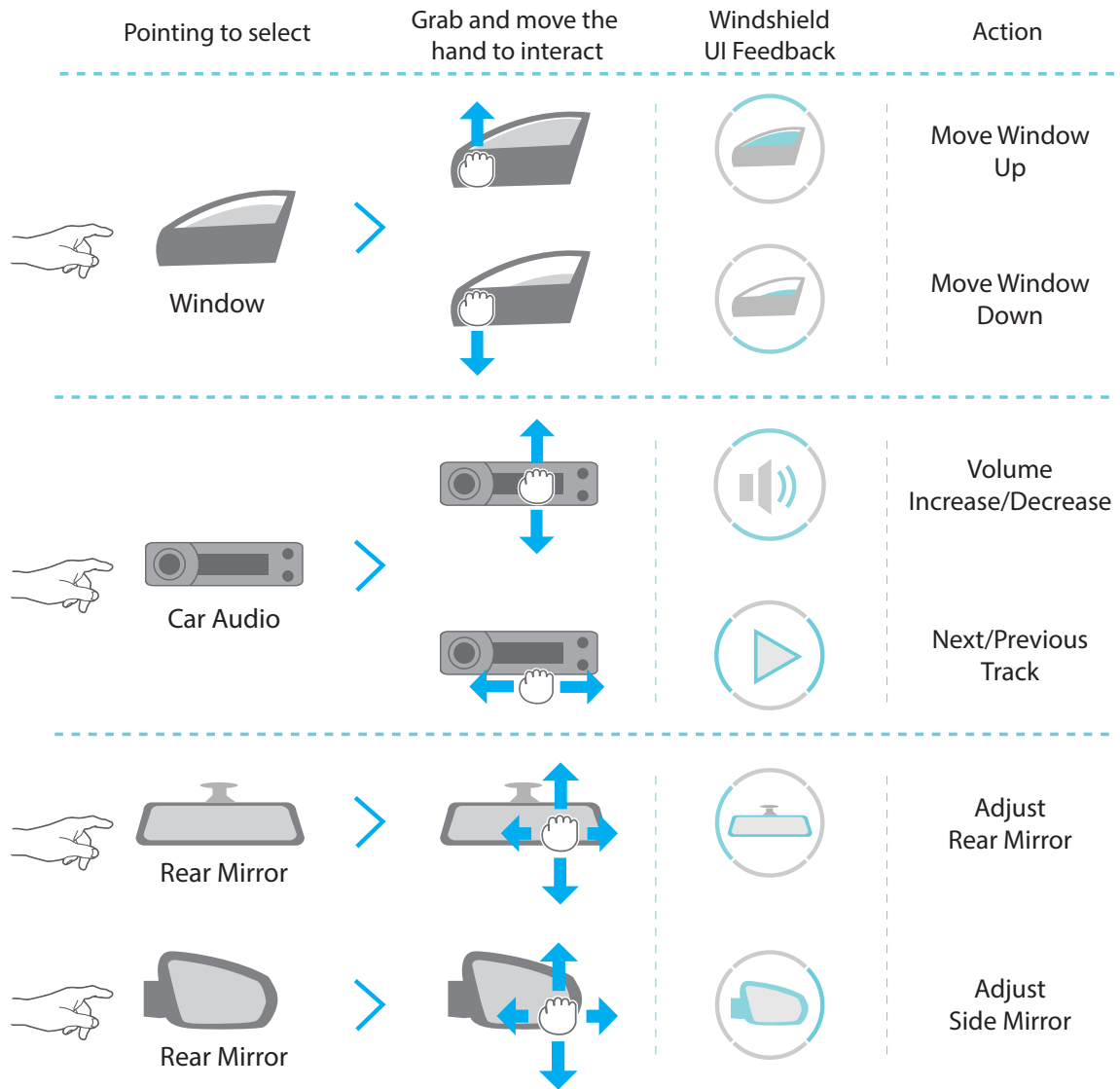


Figure 3: Our interaction metaphors with the corresponding UI feedback

user. Therefore, user interface design is a matter of compromise and trade off [May99] and making every user satisfy with the design is far fetched.

The user interface should be simple, adaptable and provide an overview of all ongoing interactions. Additionally, it is important to group and arrange the interface elements according to their function such that the cognitive load is minimized while retaining all necessary information in front of the user [HN07].

The aim of the UI is to provide a visual feedback of the ongoing interactions to the driver. We found out that the optimum position of the GUI elements is the centre of the windshield above the steering wheel. The design must be minimalistic in a way that it does not obstruct the driver's main line of sight.

The proposed UI involves a circular layout. The currently selected object located in the center and the interactions mapped around it in four quadrants of the circle's border representing the four possible interaction directions. When we point to the object, both the object icon and the circle are first made semi-transparent in gray becoming selected, it turns to a futuristic blue with increased opacity.

We chose only the required parts of the information system to be displayed such as speedometer which is always there, and various warning signs and current shift notifications that appear for a certain amount of time. We also integrated navigation, which is displayed right above the speedometer and a text based direction guidance system to the UI.

## 5 User Studies

For our user studies we used a PC with an Intel Core i7 4770 CPU, Nvidia GeForce 780 GTX, and Windows 7 64-bit. In order to have a comparison between natural and unnatural interactions, we used a steering wheel with control buttons. Two Leap Motions on each sides of the steering wheel, one Kinect on top and a microphone in front of the user act as input devices for our system. We used three displays to have wide view and enough freedom for users to interact with different virtual in-car components such as mirrors, windows, music player, and cabin light (see Fig. 4). We also used in game sounds such as car feedback and music tracks. We planned to use Oculus Rift too, but as some users felt dizzy during tests, we skipped it. We like to conduct a user study with Oculus Rift V2, as most of the problems are resolved, claimed by its developer. To evaluating the system, volunteer students with



Figure 4: Experimental hardware setup that was used for prototyping.

driving experience were asked to try it. Sex, age and their experience in working with devices such as Leap Motion and Kinect were considered, as it may result in different feedback from users. Everyone had previous experience with computer games. The testing group consists of 14 users, 11 males and 3 females. Each user performed 3 different interactions with the system, both by using natural interaction (NUI) and conventional, unnatural interactions (UNUI). For UNUI's, each action is performed by pressing a button. At the end there were totally 42 tasks for NUI and UNUI methods.

Each person had to follow a few steps in order to complete the evaluating tasks. At first, a tutorial video was shown. In the video, user got used to the devices, technology and how to interact with them. Following this, the user tried the system in order to get familiar



with the environment and clarify any misunderstanding of how the system functions. Once training is done, the user is asked to perform the actual test. The whole process of the user interacting with the system, as well as in-game action was recorded for further analysis. The user was asked to perform 4 different scenarios; each of them consisted of three acts triggered by voice command from user for both NUI and UNUI methods.

After system testing is finished, participants will be asked to fill out a questionnaire to get user’s feedback on usability, cognitive load and distraction. At the end, video footages of in-game driving and interactions were reviewed.

There were totally 6 collisions done by users; two of them happened while user tried to interact with the car in the natural way (NUI). This number is 4 when users used steering wheel buttons for interacting (UNUI). By collision we mean hitting the other cars or the road edges. Sometimes it happened that system recognized interaction as false positive. Interestingly, the amount of false positives for NUI method is 3 times compared to UNUI which was 7 times. Also, it showed that the user needed to take her eyes off the road for about 1.0 seconds using NUI compared to 0.4 seconds for UNUI which.

	Collisions	Distraction time	False Positives
NUI	2	1.0 sec	3
UNUI	4	0.4 sec	7

Table 1. The results for collisions, distraction time and falsely recognized interactions for NUI and UNUI interactions.

Based on the User Experience Questionnaire (UEQ) [LSH06] we divided questions into six dimensions that are called *perspicuity*, *efficiency*, *dependability*, *stimulation*, *novelty*, and *attractiveness*. The questionnaire consists of twenty-five 7-point items with scores between -3 to +3 that measure the above-mentioned dimensions [LHS08]. By averaging all values from questions for respective dimension, we can do statistical analyses on the data.

As expected, the NUI method achieved good results in attractiveness and efficiency. It shows that users are eager to use natural interaction instead of the conventional ways, especially when they found how easy it makes the interacting. Based on the perspicuity, it’s not clear enough for users how interactions work. In particular the Leap Motion needs some experience in order to get the best results. Users didn’t find the system novel enough, something that was expected to be high. It may be due to the users’ background because all of them were familiar with the technologies that we used and it wasn’t novel enough to them. Stimulation is close to what we expected, as most of the users found the idea exciting. Figure 5 shows the final results of the questionnaire. Overall we achieved fairly well ratings for attractiveness and efficiency. In our opinion, the hardware limitation of input devices is the main reason for perspicuity, dependability and novelty being too low.

## 6 Conclusion and Future Work

This paper aimed to introduce contact-free multimodal interaction possibilities which could be equipped in automobiles for enhancing the interactions making it more natural, attractive and intuitive.

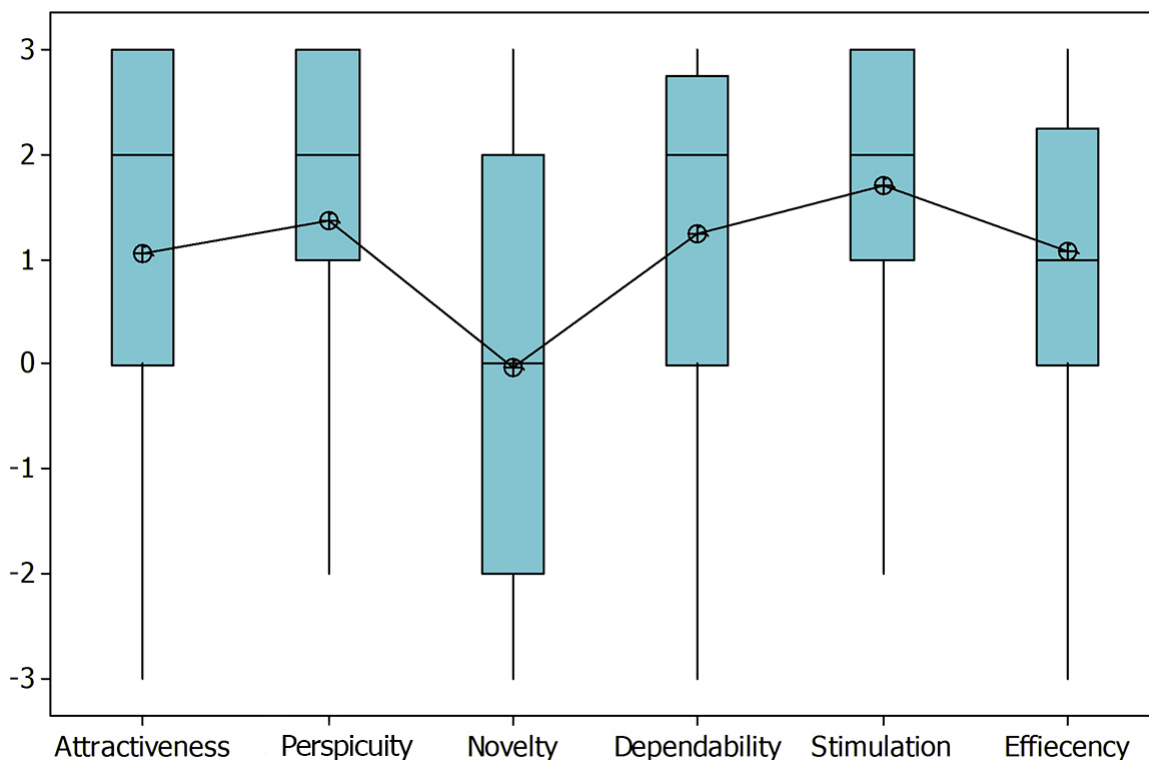


Figure 5: We observe a wide range in questionnaire answers about NUI; however, their aggregation is close to what we expect. Most of the users found our system attractive and easy to learn. Security and efficiency are lower, mostly due to hardware limitations. Though, we assume users familiarity with technologies such as Kinect, Leap Motion and similar systems, caused novelty to score less than other dimensions. In the graph the box shows Divergence, box inline shows Median and dots represent Mean values.

In this work an interactive car simulation game was designed using Unity game engine in which various car applications like radio, windows, mirror, and the cabin lights were designed to be controlled using gestures and voice interactions. We used a modular software architecture that allows us to combine a variant modules of different input and output devices. This also made our system very flexible for future interactions.

The proposed project was implemented using Microsoft Kinect for detecting pointing posture, Leap Motion for grab and swipes, Oculus Rift for enhancing the real gaming experience, and an Android application with Google API for detecting voice commands. We used certain distinct voice commands and gestures specific to certain interaction so that the system is more robust, and reduce the user confusions while using the gestures. The proposed prototype, turned out to be pretty satisfactory with respect to the goals set during the research sessions before the implementation. Though, we had certain setbacks due to working efficiency of certain devices used.

Refining current interactions to getting close to the goals and continued development of integrating more in-car features to the prototype, e.g., navigation system, handling and making phone calls and etc. Furthermore, we are interested to use integrating gaze detection technology to detect the driver's focus and adjust the intensity and the transparency of the visual contents displayed on the HUD. Another feature that we plan to implement in this

project is the tracking user's head. Finally, we like to broaden our testing group, inviting people with different background to get our user study closer to an universal one.

## References

- [AHB10] Amna Asif, Wilko Heuten, and Susanne Boll. Exploring distance encodings with a tactile display to convey turn by turn information in automobiles. In *6th Nordic conference on human-computer interaction: Extending boundaries*, pages 32–41. ACM, 2010.
- [AMS<sup>+</sup>01] Frank Althoff, Gregor McGlaun, Björn Schuller, Peter Morguet, and Manfred Lang. Using multimodal interaction to navigate in arbitrary virtual vrml worlds. In *Workshop on Perceptive user interfaces*, pages 1–8. ACM, 2001.
- [BR12] Nora Broy and Sonja Rümelin. Natural visual user interfaces—beyond input modalities. *Int'l Conference on Automotive User Interfaces*, 2012.
- [Eck13] Ronald Ecker. *Der verteilte Fahrerinteraktionsraum*. PhD thesis, lmu, 2013.
- [Gra03] Michael A Grasso. The long-term adoption of speech recognition in medical applications. In *Computer-Based Medical Systems, 2003. 16th IEEE Symposium*, pages 257–262. IEEE, 2003.
- [Gre04] Paul Green. *Driver distraction, telematics design, and workload managers: Safety issues and solutions*. Society of Automotive Engineers, 2004.
- [HCTH13] Cheng-Hung Huang, Chun-Wen Chao, Tienwei Tsai, and Ming-Hui Hung. The effects of interface design for head-up display on driver behavior. *Life Science Journal*, 10(2), 2013.
- [HN07] Marika Hoedemaeker and Mark Neerincx. Attuning in-car user interfaces to the momentary cognitive load. In *Foundations of Augmented Cognition*, pages 286–293. Springer, 2007.
- [JPF13] Jean-François Jégo, Alexis Paljic, and Philippe Fuchs. User-defined gestural interaction: A study on gesture memorization. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on*, pages 7–10. IEEE, 2013.
- [JST<sup>+</sup>08] Mads Gregers Jæger, Mikael B Skov, Nils Gram Thomassen, et al. You can touch, but you can't look: interacting with in-vehicle systems. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 1139–1148. ACM, 2008.
- [KHL<sup>+</sup>12] Martin Knobel, Marc Hassenzahl, Melanie Lamara, Tobias Sattler, Josef Schumann, Kai Eckoldt, and Andreas Butz. Clique trip: Feeling related in different cars. In *Designing Interactive Systems Conference*, pages 29–37. ACM, 2012.
- [LBCK04] Terry C Lansdown, Nicola Brook-Carter, and Tanita Kersloot. Distraction from multiple in-vehicle secondary tasks: vehicle performance and mental workload implications. *Ergonomics*, 47(1):91–104, 2004.
- [LCH13] David Libby, Alex Chaparro, and Jibo He. Distracted while driving a comparison of the effects of texting and talking on a cell phone. In *Human Factors and Ergonomics Society Annual Meeting*, volume 57, pages 1874–1878. SAGE Publications, 2013.
- [LHS08] Bettina Laugwitz, Theo Held, and Martin Schrepp. *Construction and evaluation of a user experience questionnaire*. Springer, 2008.

- [LK11] Joseph J LaViola and Daniel F Keefe. 3d spatial interaction: applications for art, design, and science. In *ACM SIGGRAPH 2011 Courses*, page 1. ACM, 2011.
- [LSH06] Bettina Laugwitz, Martin Schrepp, and Theo Held. Konstruktion eines fragebogens zur messung der user experience von softwareprodukten. In *Mensch & Computer*, pages 125–134, 2006.
- [MALR04] Gregor McGlaun, Frank Althoff, Manfred Lang, and Gerhard Rigoll. Robust video-based recognition of dynamic head gestures in various domains—comparing a rule-based and a stochastic approach. In *Gesture-Based Communication in Human-Computer Interaction*, pages 180–197. Springer, 2004.
- [MAR<sup>+</sup>01] Gregor McGlaun, Frank Althoff, Hans-Wilhelm Rühl, Michael Alger, and Manfred Lang. A generic operation concept for an ergonomic speech mmi under fixed constraints in the automotive environment. In *Proc. of HCI*, 2001.
- [May99] Deborah J Mayhew. The usability engineering lifecycle. In *CHI'99 Extended Abstracts on Human Factors in Computing Systems*, pages 147–148. ACM, 1999.
- [MEMS11] Angela Mahr, Christoph Endres, Christian Müller, and Tanja Schneeberger. Determining human-centered parameters of ergonomic micro-gesture interaction for drivers using the theater approach. In *3rd Int'l Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 151–158. ACM, 2011.
- [Mil56] George A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.
- [NDEK09] Bernhard Niedermaier, Stephan Durach, Lutz Eckstein, and Andreas Keinath. The new bmw idrive—applied processes and methods to assure high usability. In *Digital Human Modeling*, pages 443–452. Springer, 2009.
- [Nie93] Jakob Nielsen. Iterative user-interface design. *Computer*, 26(11):32–41, 1993.
- [PSH97] Vladimir I Pavlovic, Rajeev Sharma, and Thomas S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19:677–695, 1997.
- [PSS12] Bastian Pfleging, Stefan Schneegass, and Albrecht Schmidt. Multimodal interaction in the car: combining speech and gestures on the steering wheel. In *4th Int'l Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 155–162. ACM, 2012.
- [SDKS10] Albrecht Schmidt, Anind K Dey, Andrew L Kun, and Wolfgang Spiessl. Automotive user interfaces: human computer interaction in the car. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 3177–3180. ACM, 2010.
- [TBK06] Marcus Tonnis, Verena Broy, and Gudrun Klinker. A survey of challenges related to the design of 3d user interfaces for car drivers. In *3D User Interfaces, 2006. 3DUI 2006. IEEE Symposium on*, pages 127–134. IEEE, 2006.
- [Tuf97] Daniel R Tufano. Automotive huds: The overlooked safety issues. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 39(2):303–311, 1997.
- [WKL12] Garrett Weinberg, Andrew Knowles, and Patrick Langer. Bullseye: An automotive touch interface that's always on target. In *4th Int'l Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 17–19, 2012.

# Dynamic Level of Detail for Tiled Large High-Resolution Displays

Christian Scheel\*, Falko Löffler<sup>†</sup>, Anke Lehmann\*, Heidrun Schumann\*, Oliver Staadt\*

\* Institute of Computer Science  
University of Rostock  
18051 Rostock  
christian.scheel@uni-rostock.de  
heidrun.schumann@uni-rostock.de  
oliver.staadt@uni-rostock.de

<sup>†</sup> arivis AG  
18055 Rostock  
f.loeffler@arivis.com

\* Faculty of Civil Engineering  
TU Dresden  
01062 Dresden  
anke.lehmann@tu-dresden.de

**Abstract:** Large high-resolution displays (LHRDs) combine high pixel density with a large display surface area. Level-of-Detail (LOD) methods can be used to accelerate view-dependent rendering of large data sets on such display systems. We present a new method for dynamic, view-dependent level-of-detail rendering for tiled display walls. Based on the user’s tracked head position and orientation, we determine the visible display tiles. Subsequently, we calculate the suitable LOD for each visible tile based on its location in the user’s field of view. We demonstrate the utility of our approach in a collaborative visualization setting for high-resolution digital terrain data.

**Keywords:** large high-resolution displays, level of detail, real-time rendering

## 1 Introduction

Large high-resolution Displays (LHRD) are becoming increasingly popular environments for visualizing complex data. LHRDs can be designed as CAVE systems [Kuh14, DDS<sup>+</sup>09], multi-monitor desktop setups or arrays of LCD panels or projectors [SFF<sup>+</sup>00, CSWL02]. For more details on LHRDs see Ni et al. [NSS<sup>+</sup>06]. The combination of high pixel density and large display surface area allows the user to explore high-resolution data sets without explicit zooming. The user can observe small details by simply stepping closer to the display and the larger context is visible by stepping further back. In remote collaboration sessions, where two LHRDs are connected over a network, it is often necessary to transfer big data sets between the participating sites. To reduce the amount of data that has to be transmitted, foveated rendering for reducing the amount of data can be used. Foveated rendering selectively renders and transfers the data that the viewer can perceive at any given time, which requires only a subset of the display surface if the viewer is located close to the display [AKS<sup>+</sup>08, GFD<sup>+</sup>12]. One possibility to reduce the data is the use of level-of-detail (LOD) rendering. Dynamic LOD rendering reduces the complexity of the data with respect to the point of view in a virtual 3D scene.

In LHRD settings, we cannot assume a fixed distance between the user and the display.

Therefore, a dynamic LOD approach has to consider the current position and viewing direction of the user with respect to the display surface. For example, if the user is located close to the display, we may need an LOD level with a sub-pixel screen-space error, while users will not perceive errors of multiple pixels if they are further away. Furthermore, users might not see information presented on peripheral tiles of a LHRD if they are located near the display.

For that reason, we propose to combine dynamic LOD approaches with foveated rendering. We introduce a novel approach that uses head-tracking data for dynamic LOD calculations to speed up rendering, to save computing capacity, and for data reduction during transmission. We divide the LHRD surface into non-overlapping tiles. For tiled LCD walls, these tiles are usually identical to the size of individual LCD panels, but we can consider tiles in a more general sense. For each of these tiles, a screen-space error will be calculated dynamically in relation to the tracking information from the user. The screen-space error metric has to ensure that there is no perceivable difference between the original high-resolution data set and the LOD rendered data and has to be evaluated in real time.

We demonstrate the utility of our approach using a terrain rendering application. It requires a multiresolution representation, because terrain data mostly consist of height maps and can have millions of data points. Large objects such as terrains have the problem that always a part of the object is near the viewer and another part is far away. So it is a problem to choose one LOD for the whole object. View-dependent LOD methods solve this problem by using different LOD levels for the same object [Paj98, Eri00, BGP09]. For interactive or real-time rendering, multiresolution data structures are needed. We refer to Pajarola et al. for more details [PG07].

## 2 Our Approach

If the viewer is located close to the display surface, only a subset of LHRD tiles is visible. Here one tile refers to a single LCD panel. The number of visible tiles increases with the distance between the LHRD and the viewer. Thus, it is not necessary to render data for all tiles when the viewer is close. Conversely, if the viewer is further away, the number of visible tiles increases, but the level of detail for each visible tile may decrease.

In our approach, a tile has only one a single LOD level. See Figure 1 for an illustration of our prototype architecture. This has the advantage that differences between LOD levels are masked by the display bezels. In a first step, the visibility of tiles with respect to the user position and viewing direction is calculated, and the tiles are then classified into (i) tiles in the user's central field of view, (ii) tiles in the periphery, and (iii) tiles outside of the user's field of view.

Then, the appropriate screen-space error is determined for each visible tile, so that there is no visible error for the user. The screen-space error for a tile depends on the position and view direction of the user in relation to the position of the tile. Based on this screen-space error the matching LOD level is chosen. Finally, the data will be rendered with the chosen

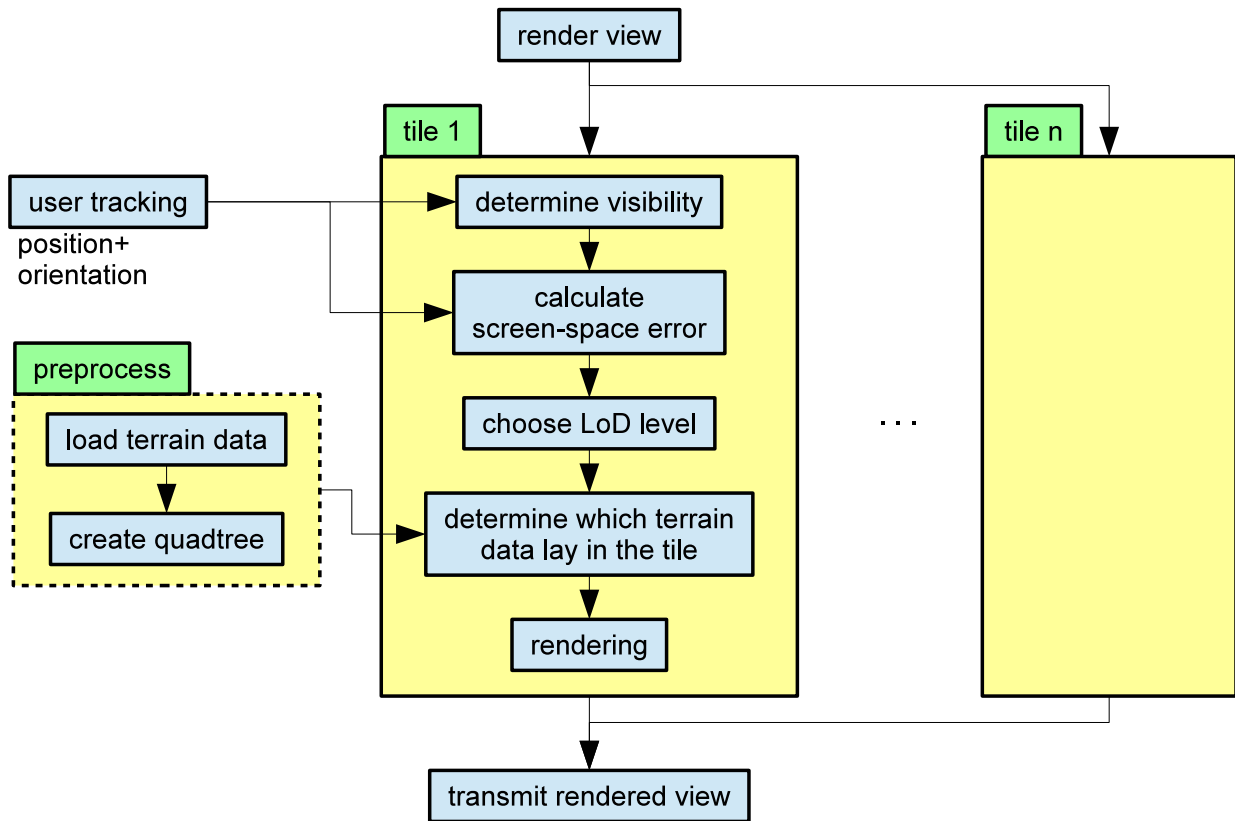


Figure 1: Architecture of our system.

LOD level and, depending on the communication scenario, the reduced rendered view, will be transmitted to the other communication side.

We consider two communication scenarios for data transmission between two communication sites  $A$  and  $B$ . One scenario is a presenter scenario where as example side  $A$  is a presenter and the other side  $B$  is the audience. In this case Side  $A$  only needs to see his own view, because the presenter doesn't need the views of the audience. But it's necessary to transmit the rendered view from side  $A$  to side  $B$  so that the audience can follow the presenter's explanations.

Another scenario is the collaborative work between side  $A$  and  $B$ . Here each side has to render their own view and each side has to know what the other side is seeing. So each side have to transmit his rendered view to the other side. See Radloff et al. [RLSS12] for more details on the two scenarios.

## 2.1 Tile Visibility

A tile can either be in focus (i.e., in the central view of the user), in the periphery or outside of the field of view. We track the user's head position and orientation to determine tile visibility. Based on the position  $P$  and the gaze vector  $\vec{g}$  obtained from a tracking and with the physical characteristics of the LHRD (i.e., position, shape, size), the intersection point  $I$  between the gaze vector and the LHRD can be calculated, see Appendix A.  $I$  is the central focus of the user on the LHRD. It can be possible that the user's line of sight is not

intersecting with the display surface.

Note that even if the focus of the user is outside of the display surface, it is necessary to render those tiles, which are in the periphery of the user's current field of view. Therefore, the angles  $\alpha_i$  between the line of sight  $\vec{s}$  and the vectors  $\vec{c}_i$  between the middle points of the tiles to the user position  $P$  are determined.

If  $\alpha > 0^\circ$  and  $\alpha \leq 90^\circ$  then the corresponding tile is in the periphery of the user. If  $\alpha = 0$  then it is the central focus point and if  $\alpha > 90^\circ$  than a tile is outside the field of view.

If we only use the middle point of a tile for visibility testing, we need to guarantee an error bound for the whole tile. So instead of testing for  $\alpha > 90^\circ$ , a more conservative angle of  $60^\circ$  is chosen. For a more precise visibility calculation of a tile we would need to calculate the smallest possible angle between the line of sight and the lines from the user position  $P$  to all points inside the tile.

## 2.2 LOD Assignment

We calculate the screen-space error  $\rho$  using Eq. 1 based on Ware [War13]:

$$\theta = 2 \cdot \arctan\left(\frac{h}{2 \cdot d}\right), \quad (1)$$

where  $\theta$  is the visual angle and is approximately one minute of arc for the foveal region of the human eyes with the highest acuity.  $h$  is the physical pixel height on the display and  $d$  is the distance between the eye and a pixel on a display. Rewriting Eq. 1 yields

$$1 = \frac{2 \cdot d \cdot \tan\left(\frac{\theta}{2}\right)}{h}. \quad (2)$$

Instead of  $\theta$  in Eq. 2, we use a tolerance angle  $\theta_t$ :

$$\theta_t = \frac{1 + (1 - \cos(\alpha_i)) \cdot 20}{60}. \quad (3)$$

Since we are using a threshold of  $\alpha = 60^\circ$ ,  $0^\circ \leq \alpha_i \leq 60^\circ$ , Eq. 3 implies than  $\frac{1}{60} \leq \theta_t \leq \frac{11}{60}$ . If a user looks straight ahead to the center of a tile,  $\alpha = 0^\circ$  and  $\theta_t = \frac{1}{60}$  which corresponds to the limit of visual acuity. For a tile in peripheral vision,  $\alpha$  becomes bigger and  $\theta_t$  can be increased because visual acuity is also reduced in the periphery.

For the calculation of  $\rho$ , we change Eq. 2 by replacing  $\theta$  with  $\theta_t$  as follows:

$$\rho = \frac{2 \cdot d \cdot \tan\left(\frac{\theta_t}{2}\right)}{h} \quad (4)$$

We assume that the data used for LOD rendering is available in a multiresolution representation. Thus, we need to determine the correct multiresolution level for each visible tile. At the borders of the tiles the data can be cut off for this tile. For all parts of the object inside a tile, the LODs in the multiresolution data structure have to be determined based on the calculated screen-space error. This is a *cut* through the multiresolution data structure. This cut is done for each tile and than one cut for the whole multiresolution data structure



over the LHRD is constructed that consist of all the cuts from the tiles. If more then one cut exists for the same part of the data the cut with the smaller error (higher details) will be chosen.

### 3 Implementation and Results

In this section we describe our prototype implementation and experimental results. The architecture of our prototype is shown in Section 2 in Figure 1. In a preprocess the terrain data will be load and the quadtree and tiles will be created. At runtime the following main steps are carried out for each tile.

1. Determine tile visibility
2. Calculate screen space error for the tile
3. Determine LOD level if tile is visible.

#### 3.1 Implementation

Our terrain rendering method is based Löffler et al. [LSS10]. In our implementation, a quadtree is used as the central data structure. A quadtree node comprises its child nodes and surface patches. A patch is a collection of triangles which is optimized for GPU rendering. We further use the *Chunked LOD* algorithm from Thatcher [Ulr02] for LOD management.

Additionally, we implemented a *feature selector* connected to a tracked input device. This feature selector corresponds to a fixed field of view of 20°, which allows the user to choose the highest LOD level for a tile inside of the field of view of the feature selector. This feature selector can be used to highlight parts of the terrain regardless of the current position of the user.

#### 3.2 Prototype Setup

We implemented our prototype in C++, using the Vrui toolkit for rendering [Kre08]. Our rendering cluster consists of one master and six rendering nodes. Each rendering node is equipped with two NVIDIA GTX 260 graphics boards, and each graphics board is connected to two displays.

We use 12-camera OptiTrack tracking system from NaturalPoint. The tracked volume in front of the LHRD was approximately 3.4 m wide, 3.0 m deep, and 3.0 m tall.

The LHRD is a tile LCD wall comprising 24 with a resolution of  $1920 \times 1200$  pixels for each panel, which are arranged is six columns and four rows (see Figure 2). The total resolution of the system is approximately 55 gigapixels.

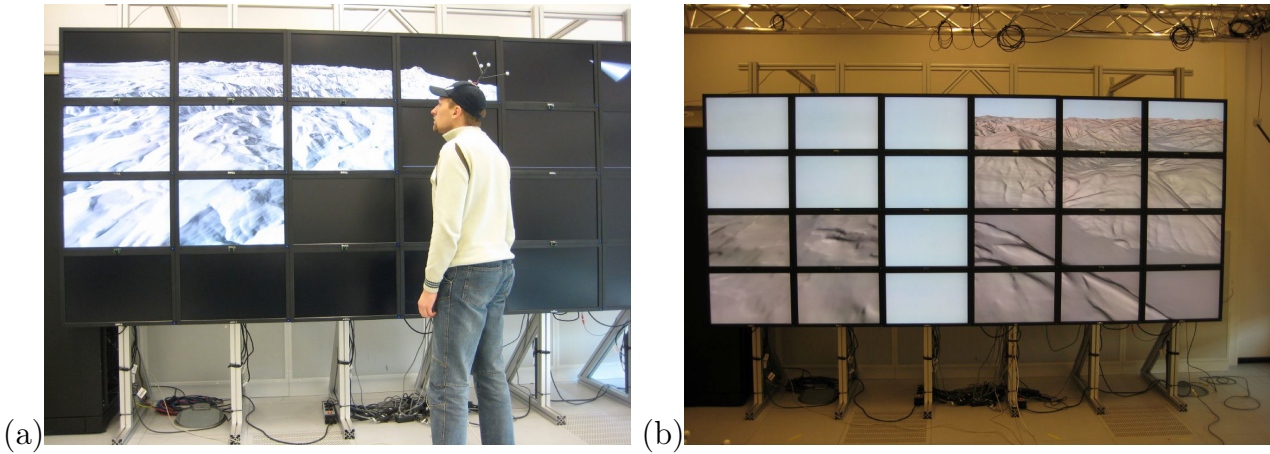


Figure 2: a: One user is focusing on the upper left corner. b: Two users are focusing on different regions of the display (the users are not shown to avoid occlusion).

### 3.3 Results

Figure 2a shows the results for a single user focusing on the left side of the display. Figure 2b illustrates two simultaneous users. One user focuses on the right side of the display and the second user on the lower left corner. Note that, for illustration purposes, the users are not visible to avoid occlusion.

For our first experiment, we used a terrain data set with  $8096 \times 8096$  points. Results are listed in Table 1. The table includes the distance of the user, the frame rate (fps) and the active tiles. The user is always looking at the center of the LHRD, except in the penultimate row. That the frame rates not change significant, except in the last line, is caused through the fact that the frame rate depends on the slowest node. And in our implementation the work balance in the cluster is that each node only cares about the four tiles that are connected to him. So at the moment we have no advantage, from the nodes that have less or nothing to do, for rendering.

The fact that in the last and in the penultimate line in Table 1 the same number of tiles are active but the frames per second change, is a result of our system architecture. In the last line two active tiles belongs to one node of our rendering cluster. In the penultimate line all four active tiles belong to one rendering node. So in the penultimate line a node in our rendering cluster has to render four active tiles and in the last line only two active tiles. This explains the different frame rates.

For the second experiment, we used a terrain model with a resolution of  $4096 \times 4096$  data points. Results are shown in Table 2. The table includes additional to Table 1 the minimal and maximal screen space error of all tiles, which is calculated for the LOD selection. The frame rate is slowing down from the first row to the penultimate row because the user is stepping closer to the Powerwall. So the minimal screen space error has to decrease to achieve that it's look, in the users perception, as there is no error in the display. Note that in the last row the frame rate is increasing, because of the same reason as described for the first experiment above.









user distance	fps	active tiles
3m	14	
2,5m	12	
2m	11-13	
1,5m	11-15	
1m	11-16	
0,5m	12	
0,2m	12-20	
0,2m	30	

Table 1: Results with a data set with  $8096 \times 8096$  data points. The user is always looking at the center of our Power-wall, except in the penultimate row.








user distance	fps	active tiles
minimal error		
maximal error		
3,0m	30,0	
3,01		
37,06		
2,5m	28,7	
2,45		
33,23		
2,0m	27,5	
2,16		
17,09		
1,5m	20,0	
1,59		
16,81		
1,0m	20,8	
1,38		
11,9		
0,5m	20,0	
1,16		
6,12		
0,2m	37,7	
0,49		
—		

Table 2: Results with a data set with  $4096 \times 4096$  data points.

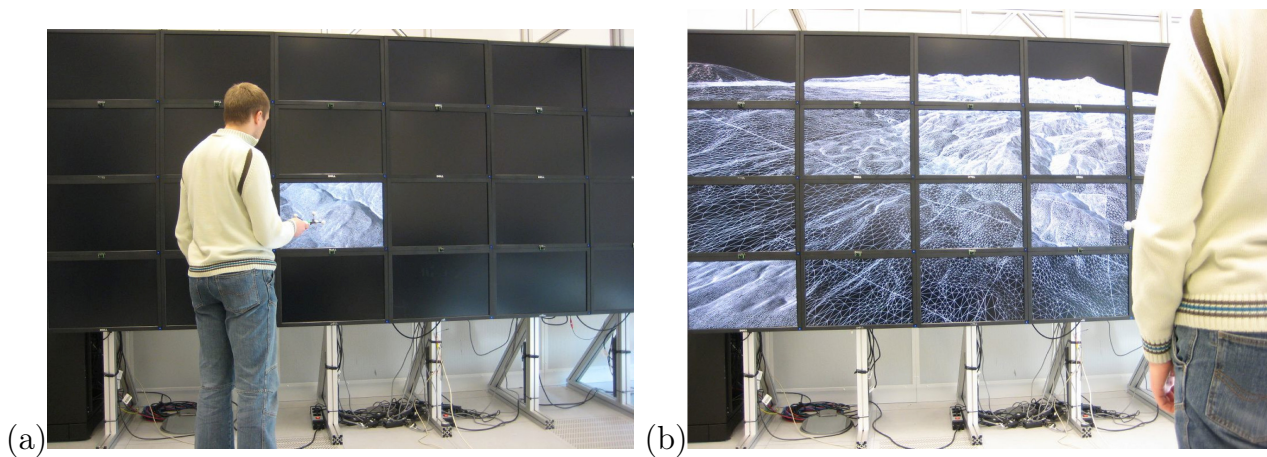


Figure 3: a: Only the tile selected through the feature selector is activated. b: The user is looking in the above right corner and the feature selector points to the lower left corner.

The Perception of switching tiles on and off, for a user, depends on the assumed field of view. In our implementation it's possible to see the switching in the periphery, but because it's in the periphery our experiments have shown that it's not disturbing very much. To avoid a disturbance through the switching fully, a bigger field of view have to be assumed. But this would lead to higher computational costs and a higher data transmission rate.

Figure 3a depicts the use of the feature selector. Figure 3b shows the result when a user is looking at the LHRD and when the feature selector is used in addition.

## 4 Conclusions

We have presented a concept to do dynamic LOD in real-time on LHRDs for multiple user which depends on the distance and gaze vector of a user and on the position of the LHRD. This can be useful for exploring large data sets like terrains on LHRDs. With our approach we can reduce the computing power and the data transmission amount.

If displays with bigger physical sizes are used all four corners of a display could be checked instead of only the middle point. Otherwise it is possible that a tile is deactivated too early.

We have only used head tracking for determining the gaze vector of a user. So if the user's head points in one direction and his eyes look in another direction the determined gaze vector from the head tracking is pointing to a wrong direction. This leads to a wrong rendering on the LHRD. For more accurate results eye tracking is necessary.

An important point for our future work would be to evaluate our results. With the help of user studies we could evaluate if a user is aware of switching tiles on and off and that the display bezels conceal the change between different levels of details. An also interesting point would be to measure the system latency, because a user can change his view direction very fast.

We also need to improve the work balance on our cluster to speed up the rendering. At the moment a node in the cluster is only doing the work for his four displays. So if a node has nothing to do because all of his displays are switched off, the node should be used for

the rendering work of the other nodes.

## 5 Acknowledgments

This work was supported by the German Research Foundation (DFG) within the research training group GRK 1424 MuSAMA. We would also like to thank Steffen Girke for his implementation and testing work and the anonymous reviewers for their valuable comments and suggestions.

## References

- [AKS<sup>+</sup>08] Benjamin A. Ahlborn, Oliver Kreylos, Sohail Shafii, Bernd Hamann, and Oliver G. Staadt. Design and implementation of a foveal projection display. *International Journal of Image and Graphics*, 8(2):243–261, 2008.
- [BGP09] Jonas Bösch, Prashant Goswami, and Renato Pajarola. Raster: Simple and efficient terrain rendering on the GPU. *Eurographics Areas Papers*, pages 35–42, 2009.
- [CSWL02] Han Chen, Rahul Sukthankar, Grant Wallace, and Kai Li. Scalable alignment of large-format multi-projector displays using camera homography trees. In *Proceedings of the conference on Visualization’02*, pages 339–346. IEEE Computer Society, 2002.
- [DDS<sup>+</sup>09] Thomas A DeFanti, Gregory Dawe, Daniel J Sandin, Jurgen P Schulze, Peter Otto, Javier Girado, Falko Kuester, Larry Smarr, and Ramesh Rao. The starcave, a third-generation cave and virtual reality optiportal. *Future Generation Computer Systems*, 25(2):169–178, 2009.
- [Eri00] Carl M Erikson. *Hierarchical levels of detail to accelerate the rendering of large static and dynamic polygonal environments*. PhD thesis, University of North Carolina, 2000.
- [GFD<sup>+</sup>12] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3D graphics. *ACM Trans. Graph.*, 31(6):164:1–164:10, November 2012.
- [Kre08] Oliver Kreylos. Environment-independent vr development. In *Advances in Visual Computing*, pages 901–912. Springer, 2008.
- [Kuh14] Torsten Kuhlen. Virtuelle Realität als Gegenstand und Werkzeug der Wissenschaft. In Sabina Jeschke, Leif Kobbelt, and Alicia Dröge, editors, *Exploring Virtuality*, pages 133–147. Springer Fachmedien Wiesbaden, January 2014.

- [LSS10] Falko Löffler, Sebastian Schwanke, and Heidrun Schumann. A hybrid approach for high quality real-time terrain rendering and optimized a-priori error estimation. In *GRAPP*, pages 233–238, 2010.
- [NSS<sup>+</sup>06] Tao Ni, G.S. Schmidt, O.G. Staadt, M.A. Livingston, R. Ball, and R. May. A survey of large high-resolution display technologies, techniques, and applications. In *IEEE Virtual Reality*, pages 223–236, 2006.
- [Paj98] Renato Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In *Visualization'98. Proceedings*, pages 19–26. IEEE, 1998.
- [PG07] Renato Pajarola and Enrico Gobbetti. Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer*, 23(8):583–605, 2007.
- [RLSS12] Axel Radloff, Anke Lehmann, Oliver Staadt, and Heidrun Schumann. Smart interaction management: An interaction approach for smart meeting rooms. In *8th International Conference on Intelligent Environments (IE)*, pages 228–235, June 2012.
- [SFF<sup>+</sup>00] Daniel R Schikore, Richard A Fischer, Randall Frank, Ross Gaunt, John Hobson, and Brad Whitlock. High-resolution multiprojector display walls. *Computer Graphics and Applications, IEEE*, 20(4):38–44, 2000.
- [Ulr02] Thatcher Ulrich. Rendering massive terrains using chunked level of detail control. *SIGGRAPH Course Notes*, 3(5), 2002.
- [War13] Colin Ware. *Information visualization: perception for design*. Elsevier, 2013.

## A Intersection Point

Based on the position  $P$  and the gaze vector  $\vec{g}$  obtained from a tracking the user's line of sight  $\vec{s}$  is calculated as

$$\vec{s} = P + \lambda \vec{g}. \quad (5)$$

If the LHRD is in a single plane and assuming that the center of the LHRD is the origin of our world coordinate system, the LHRD plane can be described parametrically with the parameters  $u$  and  $t$  as  $E$ :

$$E = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + u \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + t \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} u \\ 0 \\ t \end{pmatrix} \quad (6)$$

To obtain the intersection point between the line of sight and the LHRD we have to solve  $\vec{s} = E$ . This leads to the following system of equations:

$$P_x + \lambda \cdot \vec{g}_x = u \quad (7)$$

$$P_y + \lambda \cdot \vec{g}_y = 0 \quad (8)$$

$$P_z + \lambda \cdot \vec{g}_z = t \quad (9)$$

To determine  $\lambda$  we only have to solve Eq. 8:

$$\lambda = \frac{-P_y}{\vec{g}_y}. \quad (10)$$

Based on Eq. 5 and Eq. 10, we can calculate the intersection point  $I$  using

$$I = P + \frac{-P_y}{\vec{g}_y} \cdot \vec{g}. \quad (11)$$





# 3D-Längenanamorphosen in einem einzigen Renderingschritt

Jonas Schell<sup>1,2</sup>, Tom Vierjahn<sup>3</sup>, Sina Mostafawy<sup>1,3</sup>

<sup>1</sup> FH Düsseldorf    <sup>2</sup> Urbanscreen GmbH & Co. KG    <sup>3</sup>rmh new media GmbH  
Josef-Gockeln-Str. 9                      Am Deich 86                      Gilbachstraße 29a  
40474 Düsseldorf                      28199 Bremen                      50672 Köln

**Abstract:** 3D-Längenanamorphosen sind verzerrte 2D-Bilder, die sich von einer bestimmten Perspektive betrachtet scheinbar zu einem 3D-Objekt verwandeln. Durch die Computergrafik ist es ein Leichtes geworden, 3D-Anamorphosen zu erzeugen. Doch das derzeit angewandte Verfahren bietet noch viel Raum für Optimierungen. So werden im Moment noch zwei verlustbehaftete Rendervorgänge zur Erzeugung der 3D-Längenanamorphose benötigt. Diese Arbeit legt den Grundstein für ein neues, schnelleres und qualitativ besseres Verfahren zur Erzeugung von 3D-Anamorphosen mittels gängiger 3D-Programme. Durch geeignete Vorverzerrung der 3D-Szene kann die Anamorphose mit nur einem verlustfreien Rendervorgang erzeugt werden. Auch die Integration in Echtzeit-Anwendungen ist dank der einfachen Berechnung möglich. Kombiniert mit einem Personen-Tracking-System sind so zum Beispiel auch interaktive Augmented-Reality-Anwendungen möglich.

**Keywords:** 3D-Anamorphose, Projection-Mapping, Mixed/augmented reality, Rendering, Media arts

## 1 Einleitung

Schon Leonardo da Vinci hat sich, zeitgleich mit der Entdeckung der Perspektive [And08], an Anamorphosen versucht und bis heute ist die Faszination für sie nicht versiegt. Manche sprechen sogar von einer Renaissance: „Anamorphosis is having a revival. Post-Renaissance Europe was the setting for its heyday, but anamorphosis is again finding favor in postmodern culture.“ [Top00]. Anamorphosen sind geschickt verzerrte Bilder, die nur von einem bestimmten Standpunkt aus oder durch den Blick in ein optisches Hilfsmittel ihr unverzerrtes Geheimnis enthüllen. Einige der bekanntesten Anamorphosen unserer Zeit stammen vom britischen Künstler Julian Beever [Bee]. Dieser macht immer wieder durch faszinierende 3D-Längenanamorphosen von sich reden. In tagelanger Arbeit malt er seine Kunstwerke in Fußgängerzonen auf der ganzen Welt. Bedingt durch die digitale Revolution ist es heute nicht mehr als eine Fingerübung, eine Vielzahl an Anamorphosen zu erzeugen. Die Mathematik und Programme dafür existieren schon längere Zeit [HNG00] [Ken]: Wo in der Renaissance noch in mühevoller Präzisionsarbeit mit Hilfe von Pinsel und Farbe aufwendige Anamorphosen gemalt wurden, gibt es heute 3D-Programme und Projektoren, mit denen man nicht nur Stilleben, sondern auch anamorphotische Filme zeigen kann (Abb. 1a, 1b).



(a)



(b)

Abbildung 1: Projection Mapping Arbeit „555 Kubik“ 2009 von Urbanscreen auf der Hamburger Kunsthalle [Urb]. a) Vom idealen Standpunkt aus betrachtet. b) Abweichend vom idealen Standpunkt.

Die digitalen Werkzeuge zur Erzeugung von 3D-Längenanamorphosen sind in jedem gängigen 3D-Programm vorhanden, doch wurden diese Werkzeuge nie speziell für die Erschaffung von Anamorphosen ausgelegt. Die vorliegende Arbeit gibt dem Anwender ein Werkzeug an die Hand, welches speziell für die Erzeugung von 3D-Längenanamorphosen geschaffen wurde und daraufhin optimiert ist. Dadurch erzielt man ein gleiches oder besseres Resultat mit geringerem Zeitaufwand und einfacherem Workflow als mit den aktuellen Werkzeugen.

Bei dem hier vorgestellten Verfahren handelt es sich, anders als bei [DC11], nicht um eine neue Art der Anamorphose. Das Ziel ist auch nicht die Erschaffung eines realen anamorphotischen 3D-Objektes [HC07]. Trotzdem haben die Arbeiten einen gemeinsamen Nenner: Alle verzerren 3D-Geometrie, um dadurch eine Anamorphose zu erzeugen. Doch dient die 3D-Verzerrung bei dem hier vorliegenden Verfahren nur der effektiveren Erzeugung einer 3D-Längenanamorphose.

Im folgenden Abschnitt wird das zur Zeit standardmäßig angewandte Verfahren zur Erzeugung von 3D-Längenanamorphosen erklärt. Da dieses Renderverfahren aus drei aufeinanderfolgenden Arbeitsschritten besteht, wird es im Folgenden mit „Drei-Schritte-Rendering“ bezeichnet. Anschließend werden Probleme und Einschränkungen des „Drei-Schritte-Rendering“ aufgezeigt. Vor der mathematischen Herleitung wird die Herangehensweise des neuen Verfahrens skizziert und das Konzept dargelegt. Die grundlegende Idee ist die Verkürzung des „Drei-Schritte-Rendering“ auf einen einzigen Rendervorgang. Dementsprechend wird das neue Verfahren „Ein-Schritt-Rendering“ genannt. Auf die Herleitung folgt die Erläuterung der praktischen Anwendung und anschließend der Vergleich beider Verfahren. Der letzte Abschnitt gibt einen Ausblick auf Anwendungs- und weitere Entwicklungsmöglichkeiten.

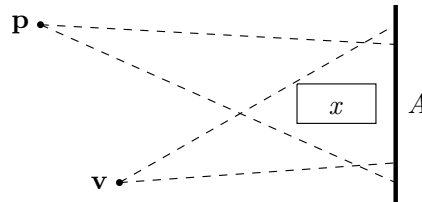


Abbildung 2: Nachgebildete 3D-Szene. Gestrichelt, die Sichtpyramiden der Kameras an Position  $\mathbf{p}$  und  $\mathbf{v}$ .

## 2 Das bisherige „Drei-Schritte-Rendering“

Zusammengefasst besteht das „Drei-Schritte-Rendering“ aus der Kombination von Rendering, Texture-Mapping und nochmaligem Rendering. Dazu wird eine reale Umgebung zuvor digital nachgebildet und die Szene für das Verfahren vorbereitet.

Gehen wir von einer simplen 3D-Szene (Abb. 2), mit  $\mathbf{p}$  als Projektorposition in der realen Szene und  $\mathbf{v}$  als idealer Betrachtersposition aus. Die reale Projektionsfläche wird durch  $A$  repräsentiert. Der Quader  $x$  vor der Projektionsfläche steht stellvertretend für alle Objekte, für welche die Illusion der Perspektive durch die 3D-Anamorphose erzeugt werden soll. Befinden sich die Objekte vor der Projektionsfläche, erscheint es dem Betrachter, als würden diese vor der Fläche schweben, analog dazu erscheint es dem Betrachter, als würden die Objekte in der Projektionsfläche versinken, wenn diese sich in der 3D-Szene, von  $\mathbf{v}$  aus gesehen, hinter der Projektionsfläche  $A$  befinden (siehe auch Abb. 1a). Hat man die gewünschten Veränderungen (Animation, Modeling, etc.) am Quader  $x$  abgeschlossen, werden zwei virtuelle Kameras an den Stellen  $\mathbf{p}$  und  $\mathbf{v}$  in der Szene platziert und auf die Projektionsfläche gerichtet. Dem folgt das „Drei-Schritte-Rendering“:

### 1. Rendern aus Sicht der Betrachter-Kamera:

Im ersten Schritt wird aus der Betrachterperspektive  $\mathbf{v}$  die Szene gerendert und das Ergebnis abgespeichert. In diesem Renderdurchgang erzeugt man das Bild, welches der Betrachter später von der Position  $\mathbf{v}$  aus sehen soll (Abb. 3a). Wie bei einem normalen Rendering wird auch hier die Beleuchtung berechnet, was je nach Szene viel Zeit in Anspruch nehmen kann. Für diesen Schritt muss die Projektionsfläche aus der Szene entfernt werden.

### 2. Texture-Mapping auf die Projektionsfläche aus Sicht der Betrachter-Kamera:

Das im Schritt 1 erzeugte Bild wird nun wieder von der Betrachterkamera auf die Projektionsfläche projiziert.

### 3. Rendern aus Sicht der Projektor-Kamera:

Durch das Texture-Mapping haben wir nun ein flaches 2D-Bild auf der Projektionsfläche. Nun wird von der Projektorposition  $\mathbf{p}$  ein weiteres Mal gerendert. Dafür werden alle Objekte bis auf die Projektionsfläche  $A$  aus der Szene entfernt. Das Ergebnis (Abb. 3b) ist ein Bild, welches von  $\mathbf{p}$  wieder auf die Projektionsfläche projiziert, die

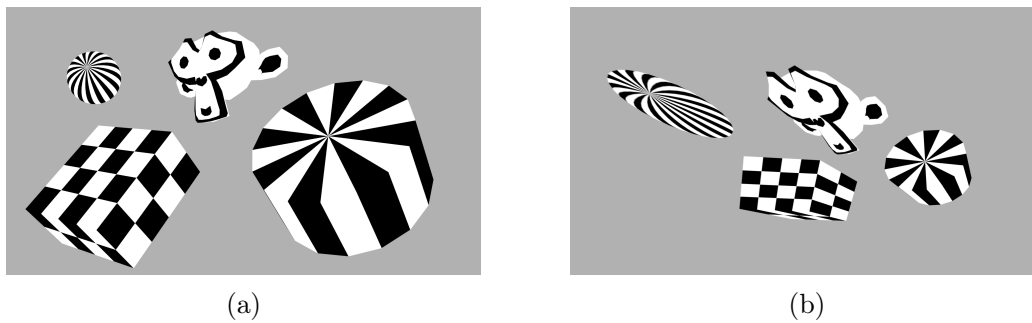


Abbildung 3: a) Ergebnis des in Schritt 1 gerenderten Bildes aus der Betrachter-Perspektive. b) Ergebnis des in Schritt 3 gerenderten Bildes aus der Projektor-Perspektive.

perspektivisch passende 3D-Anamorphose für den idealen Betrachter  $\mathbf{v}$  erzeugt. Da hier keine Beleuchtungsberechnung erfolgt, ist dieser Rendschritt im Vergleich zum ersten deutlich kürzer.

### 3 Probleme des „Drei-Schritte-Renderings“

**Zeitaufwand** In Zeiten von immer aufwendigeren 3D-Effekten und knapp bemessenen Zeitrahmen ist Renderzeit kostbar und jede Einsparung willkommen. Ein Verfahren, das zwei Rendervorgänge benötigt, ist daher nicht sehr effizient. Auch wenn der zweite Rendschritt, da keinerlei Beleuchtungsberechnung stattfindet, deutlich schneller abläuft, ist es doch ein weiterer Arbeitsschritt, der zusätzlich Zeit und zusätzlichen Datenspeicher in Anspruch nimmt. Ebenso ist das Vorbereiten der 3D-Szene für den zweiten Rendschritt, sprich Entfernen der Geometrie und Einstellen der Beleuchtung, sofern nicht automatisiert, eine weitere Belastung für das Zeitbudget.

**Qualitätsverlust** Das doppelte Rendern des „Drei-Schritte-Renderings“ ist in vielen Fällen verlustbehaftet. Die Kombination von Schritt 1 und 2 führt zu einem Auflösungsverlust und dadurch zu einem Verlust der Schärfe. Nehmen wir folgende Situation an (Abb. 4): Die Betrachterkamera  $\mathbf{v}_1$  ist senkrecht auf die Projektionsfläche gerichtet. Dann erfährt das in Schritt 1 gerenderte Bild durch das im Schritt 2 durchgeführte Texture-Mapping eine gleichmäßige Skalierung auf die Projektionsfläche.

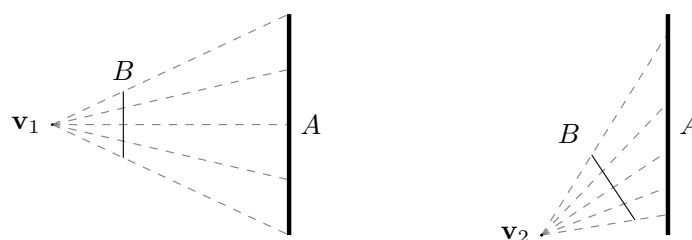


Abbildung 4: Projektion der Bildfläche  $B$  auf die Projektionsfläche  $A$ . Links mit gleichmäßiger und rechts mit ungleichmäßiger Verzerrung auf der Projektionsfläche.

Weicht die Betrachterkamera  $\mathbf{v}_2$  aber von der senkrechten Position ab, so erfährt das Bild eine in einer Richtung immer stärker werdende Streckung, welche zu einem Verlust an Auflösung pro Fläche führt. Man kann diesem Problem natürlich mit einer höheren Auflösung beim 1. Rendschritt von der zuvor festgelegten, optimalen Betrachterposition aus entgegenwirken, es aber nie umgehen. Die höhere Auflösung hat zudem eine deutlich längere Renderzeit sowie die Bearbeitung und Speicherung größerer Datenmengen zur Folge.

**Zusätzliche Fehlerquellen** Das Ändern der 3D-Szene für den zweiten Renderdurchgang und die Einrichtung des Texture-Mappings bergen Fehlerquellen: So könnte das Entfernen der Geometrie in der Vorbereitung für den zweiten Rendervorgang vergessen werden oder Fehler bei den Beleuchtungseinstellungen für den zweiten Rendervorgang zu einer erneuten Beleuchtung des gemappten Bildes führen. Auch ein Bildfehler im ersten Rendervorgang, der erst nach dem zweiten Rendern entdeckt wird, bedeutet einen erheblichen Mehraufwand, da beide Rendervorgänge wiederholt werden müssen.

#### 4 Ein neuer Ansatz: Das „Ein-Schritt-Rendering“

Die Lösung des dargestellten Problems hat sicherlich viele mögliche Wege. Inspiriert von ähnlichen Arbeiten ist der nachfolgende Ansatz entstanden [HC07] [DC11].

Wir gehen von einer 3D-Szene aus, welcher eine reale Szene zu Grunde liegt (Abb. 5). In dieser 3D-Szene repräsentiert  $A$  die Projektionsfläche der realen Szene und  $\mathbf{p}$  und  $\mathbf{v}$  die realen Positionen des Projektors und des idealen Betrachters. Dazu kommen  $\mathbf{x}_1$  und  $\mathbf{x}_2$ , welche das Objekt darstellen, das durch eine 3D-Längenanamorphose für den Betrachter in der realen Szene an dieser Stelle erscheinen soll. Die Geraden durch die Punkte  $\mathbf{v}$  und  $\mathbf{x}_1$  sowie durch  $\mathbf{v}$  und  $\mathbf{x}_2$  stellen die Linien dar, auf denen das Objekt projiziert werden kann, ohne dass sich eine sichtbare Veränderung für den Betrachter  $\mathbf{v}$  ergibt. Dieser Effekt wird in Abbildung 6 skizziert. Alle drei Objekte wirken für den Betrachter gleich groß, da sie perspektivisch korrekt verzerrt sind. Projiziert man  $\mathbf{x}_1$  und  $\mathbf{x}_2$  von  $\mathbf{v}$  aus auf die Projektionsfläche, so erhält man  $\mathbf{s}_1$  und  $\mathbf{s}_2$  (Abb. 5). Die Projektion  $\mathbf{s}_1\mathbf{s}_2$  ist für den Betrachter deckungsgleich und damit scheinbar identisch mit dem Objekt  $\mathbf{x}_1\mathbf{x}_2$ . Diesen Gedanken können wir weiterführen: Wenn wir auf der Projektionsfläche das Abbild  $\mathbf{s}_1\mathbf{s}_2$  anzeigen, erscheint es dem Betrachter als Objekt  $\mathbf{x}_1\mathbf{x}_2$ . In unserem Fall wollen wir das Abbild  $\mathbf{s}_1\mathbf{s}_2$  durch einen Projektor  $\mathbf{p}$  erzeugen. Wieder nutzen wir die gleiche Erkenntnis wie bei der Projektion des Objektes  $\mathbf{x}_1\mathbf{x}_2$  und

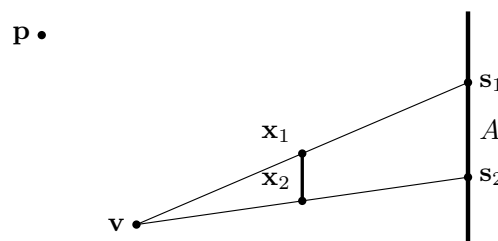


Abbildung 5: Stark vereinfachte 3D-Szene.

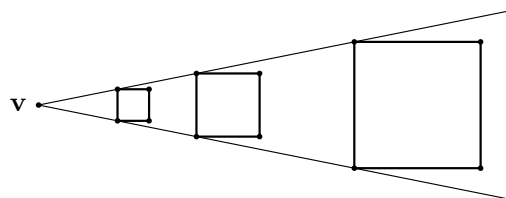


Abbildung 6: Alle drei Objekte erscheinen dem Betrachter gleich groß.

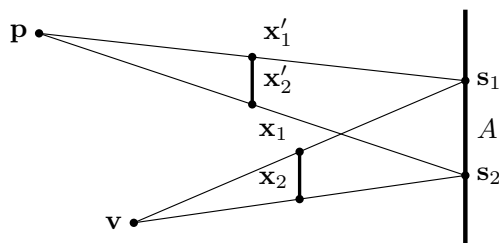


Abbildung 7: Erzeugung des vorverzerrten Objektes  $\mathbf{x}'_1\mathbf{x}'_2$ .

projizieren ein neues Objekt  $\mathbf{x}'_1\mathbf{x}'_2$  auf den Geraden von  $\mathbf{p}$  durch die Punkte  $\mathbf{s}_1$  und  $\mathbf{s}_2$  (Abb. 7). Wird nun von  $\mathbf{p}$  aus das Objekt  $\mathbf{x}'_1\mathbf{x}'_2$  gerendert und das entstandene Bild von  $\mathbf{p}$  auf die Projektionsfläche  $A$  projiziert, erhält man das gewünschte Abbild  $\mathbf{s}_1\mathbf{s}_2$ , das vom Betrachter nicht vom Objekt  $\mathbf{x}_1\mathbf{x}_2$  unterschieden werden kann.

Demnach muss das Objekt  $\mathbf{x}_1\mathbf{x}_2$  so vorverzerrt werden, dass es als Objekt  $\mathbf{x}'_1\mathbf{x}'_2$  von  $\mathbf{p}$  aus wie gewünscht gerendert werden kann.

## 5 Herleitung

Die Vorverzerrung lässt sich bestimmen, indem eine Verzerrungsmatrix bestimmt wird, die die Sichtpyramide  $\mathbf{v}\mathbf{s}_1\mathbf{s}_2$  in  $\mathbf{p}\mathbf{s}_1\mathbf{s}_2$  überführt (Abb. 7). Dazu werden lediglich affine Transformationen verwendet. Die Berechnung der Verzerrungsmatrix erfordert die vorherige Transformation der Szene (Abb. 8) in eine vordefinierte Ausgangslage. Diese wird durch eine Kombination von Rotation und Translation erreicht. Mit Erreichen der definierten Ausgangslage finden wir folgende Situation vor. Die Projektionsfläche befindet sich in der  $x/y$ -Ebene. Außerdem befindet sich der Punkt  $\mathbf{p}_{Lot}$ , welcher der Lotfußpunkt des Lotes von  $\mathbf{p}$  auf der Projektionsfläche  $A$  ist, im Ursprung des Welt-Koordinatensystems (Abb. 9).

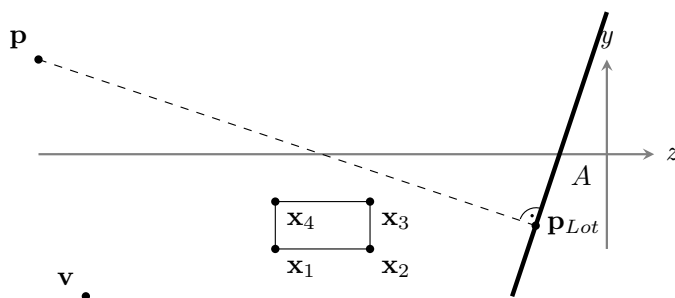


Abbildung 8: 3D-Szene vor der Transformation in die Ausgangslage

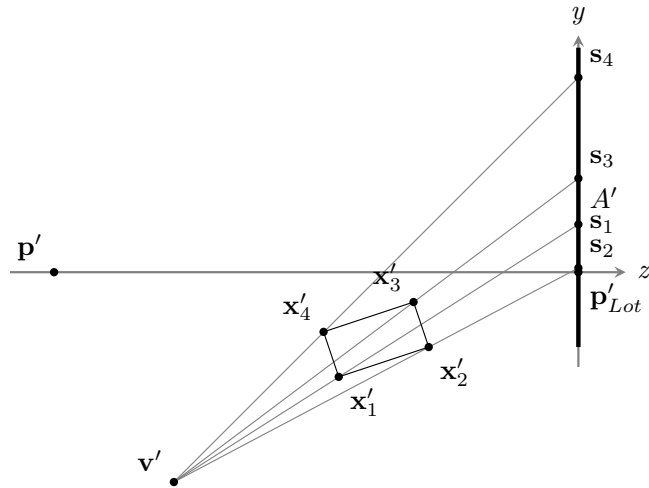


Abbildung 9: 3D-Szene nach der Transformation in die Ausgangslage

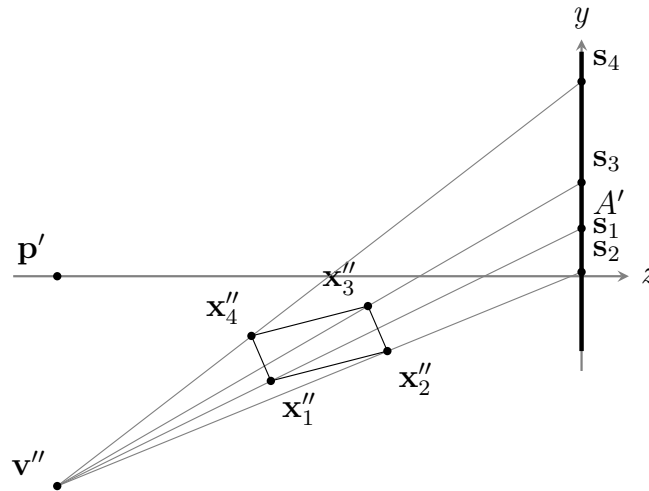


Abbildung 10: Betrachter und Geometrie nach Anwendung der Skalierung

## 5.1 Skalierung

Nachdem sich die Projektionsfläche in der  $x/y$ -Ebene befindet, können wir den Skalierungsfaktor der  $z$ -Komponente ermitteln. Das Ziel der Skalierung ist es, die Szene so zu skalieren, dass Projektor  $\mathbf{p} = [x_{\mathbf{p}} \ y_{\mathbf{p}} \ z_{\mathbf{p}}]^{\top}$  und Betrachter  $\mathbf{v} = [x_{\mathbf{v}} \ y_{\mathbf{v}} \ z_{\mathbf{v}}]^{\top}$  den gleichen Abstand zur Projektionsfläche  $A'$  haben (Abb. 10). Aus den  $z$ -Komponenten  $z_{\mathbf{p}}$  und  $z_{\mathbf{v}}$  wird die Skalierung wie folgt errechnet:

$$scale_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{z_{\mathbf{p}}}{z_{\mathbf{v}}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

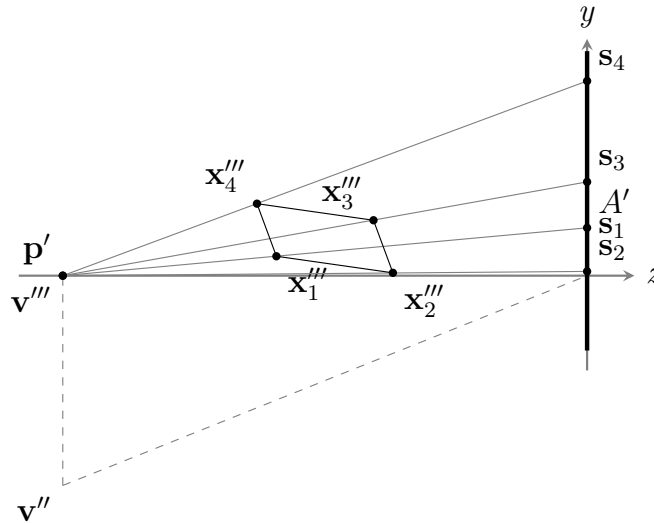


Abbildung 11: Betrachter und Geometrie nach Anwendung der Scherung.

## 5.2 Scherung

Um den Betrachter nun auf die Projektorposition zu transformieren, ist eine Scherung nötig. Die Scherung findet parallel zur  $x/y$ -Ebene statt und kann in zwei Scherungen mit den Koeffizienten  $shear_x$  und  $shear_y$  aufgeteilt werden. Mit Hilfe des Strahlensatzes lassen sich  $shear_x$  und  $shear_y$  wie folgt berechnen (Abb. 11):

$$shear_x = \begin{bmatrix} 1 & 0 & -\frac{x_v}{z_v} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad shear_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\frac{y_v}{z_v} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

## 5.3 Finale Verzerrungsmatrix bestimmen

Durch die Anwendung der oben erstellten Matrizen auf  $\mathbf{x}'_1$  erhält man folgende Gleichung,

$$\begin{pmatrix} x''_1 \\ y''_1 \\ z''_1 \\ 1 \end{pmatrix} = shear_x \cdot shear_y \cdot scale_z \cdot \begin{pmatrix} x'_1 \\ y'_1 \\ z'_1 \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & -\frac{x_v}{z_v} & 0 \\ 0 & 1 & -\frac{y_v}{z_v} & 0 \\ 0 & 0 & \frac{z_p}{z_v} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x'_1 \\ y'_1 \\ z'_1 \\ 1 \end{pmatrix} \quad (3)$$

um den wie gewünscht verzerrten Punkt  $\begin{bmatrix} x''_1 & y''_1 & z''_1 & 1 \end{bmatrix}^T$  zu berechnen.

## 6 Anwendung

Nachdem im vorangegangenen Abschnitt die Berechnung der Verzerrungsmatrix erläutert wurde, beschäftigt sich dieser Abschnitt mit der allgemeingültigen Integration des „Einschritt-Rednerings“ in bestehende 3D-Software. Die Vorgehensweise ist für jede Software



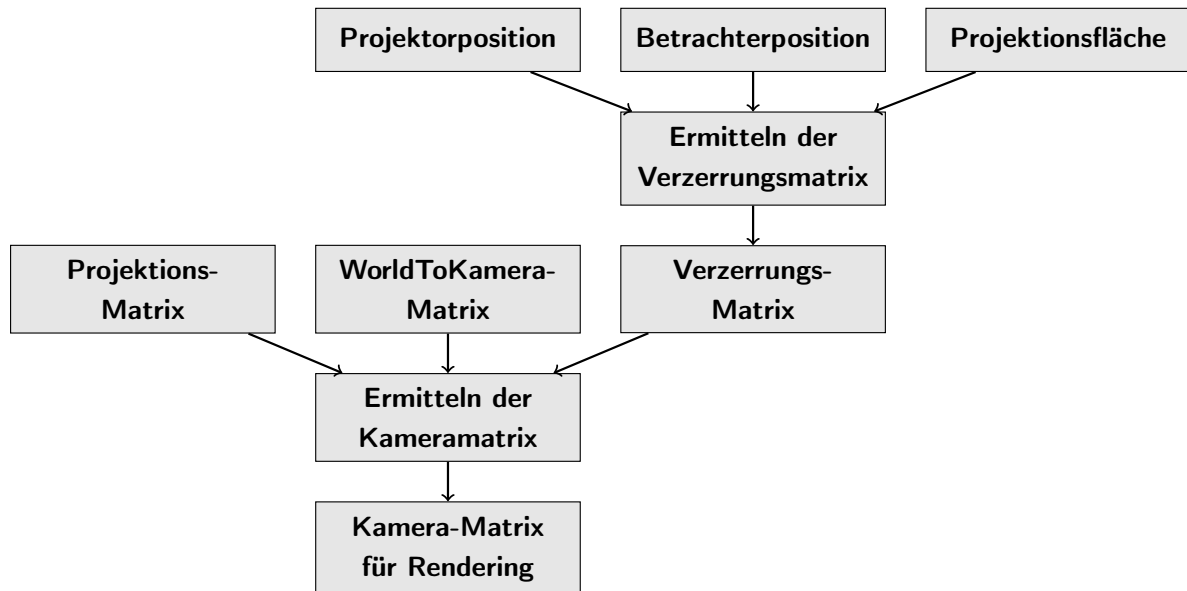


Abbildung 12: Allgemeingültige Implementierung in bestehende 3D-Software.

gültig (Abb 12): Zunächst wird mithilfe der Projektor- bzw. Betrachterposition und der definierten Projektionsfläche die Verzerrungsmatrix errechnet.

In den meisten 3D-Programmen wird eine virtuelle Kamera durch die Verwendung der *WorldToCamera-Matrix* und der *Projection-Matrix* nachgebildet. Beide Matrizen werden zur sog. *Camera-Matrix* multipliziert.

Die zuvor berechnete Verzerrungsmatrix wird nun zur *Camera-Matrix* der Projektor-Kamera multipliziert, sodass sich folgende Transformationsreihenfolge ergibt: 1. Verzerrungsmatrix 2. *WorldToCamera-Matrix* 3. *Projection-Matrix*. Die daraus resultierende Gesamtmatrix überschreibt dann die *Camera-Matrix* der Projektions-Kamera und sorgt somit für die korrekte Verzerrung der 3D-Szene während des Rendervorgangs.

Die Vorgehensweise bei der Implementierung des „Ein-Schritt-Rednerings“ ist grundsätzlich sowohl für klassische 3D-Software (Blender, 3ds Max, Maya usw.) als auch für Echtzeit-Software und Game-Engine (Unity, VVVV usw.) identisch. Aus eigener Erfahrung ist die Implementierung in letztere aber einfacher umzusetzen, da viele Echtzeit-Programme schon die benötigten Tools zur Manipulation der Kamera-Matrizen zur Verfügung stellen.

## 7 Gegenüberstellung

**Qualität des gerenderten Bildes** Zur Beurteilung beider Verfahren bezüglich der Qualitätsunterschiede nutzen wir eine Testszene (Abb. 13). Das folgende Rendering findet mit einer für den Anwendungsfall nicht unüblichen XGA Auflösung von 1024x768 Pixeln statt. Aus der Testszene wird zunächst ein Bild durch Anwendung des „Drei-Schritt-Renderings“ und eins durch Durchführung des „Ein-Schritt-Renderings“ gerendert. Als Ergebnis erhalten wir zwei aus Sicht der Projektor-Kamera gerenderte Bilder. Diese Ergebnisbilder werden nun von der Projektor-Kamera mittels Texture-Mapping auf die virtuelle Fassade gemaped. Dieser Schritt dient dazu, die normalerweise in real durchgeführte Projektion digital

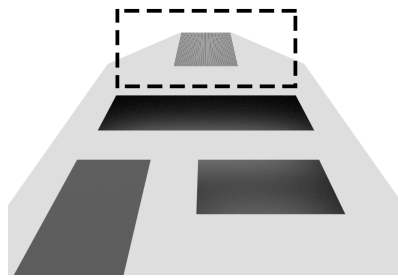


Abbildung 13: Testszene aus Betrachterposition. Im oberen Teil der virtuellen Fassade, gestrichelt umrandet, die Projektionsfläche mit sehr feinem Muster.

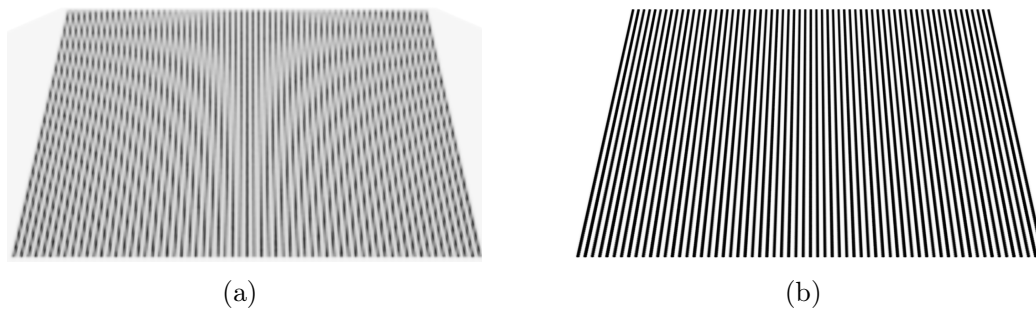


Abbildung 14: Ausschnitte der Projektion auf der virtuellen Fassade (gestrichelt umrandet in Abb. 13). a) Ergebnis des „Drei-Schritte-Rendering“ mit deutlich sichtbarem Alias-Effekt. b) Ergebnis des „Ein-Schritt-Rendering“ ohne sichtbaren Alias.

nachzubilden. Um die Vergleichsbilder zu erhalten, wird zuletzt die Projektion auf der virtuellen Fassade durch eine Kamera an der Betrachterposition aufgenommen (Abb. 14a und 14b). Für diesen Schritt wird eine Auflösung von 4096x3072 verwendet, damit dieses keine Auswirkung auf die Bildqualität hat. Bei der Untersuchung des Bildausschnittes des „Drei-Schritte-Renderings“ (Abb. 14a) sieht man einen deutlichen Alias-Effekt. Beim „Ein-Schritt-Rendering“ hingegen ist kein sichtbarer Alias-Effekt zu sehen (Abb. 14b).

**Zeitaufwand** Auch wenn belastbare Ergebnisse der Renderzeiten noch ausstehen, zeigen erste Tests einer rudimentären Implementierung des „Ein-Schritt-Renderings“ in klassischer 3D-Software (3ds Max, Blender) eine deutliche Verkürzung der Renderzeit. Der maßgebliche Grund für die kürzere Renderzeiten ist der folgende: Beim hier vorgestellten Verfahren kann sofort mit der Auflösung des später verwendeten Projektors gerendert werden. Im Gegensatz

Tabelle 1: Renderzeiten des „Ein-Schritt-Renderings“ bzw. „Drei-Schritte-Renderings“, bei unterschiedlichen Auflösungen. Gerendert wurde eine kurze Animation in 3ds Max.

Verfahren	XGA(1024x768)	2K(2048x1536)	4K(4096x3073)
Ein-Schritt-Rendering	2.49 min	8.33 min	28.49 min
Drei-Schritte-Rendering	3.13 min	9.28 min	31.52 min

dazu muss beim „Drei-Schritte-Rendering“ der erste Rendervorgang in einer deutlich höheren Auflösung gerendert werden, um dem in Abschnitt 3 beschriebenen Auflösungsverlust entgegen zu wirken. So ist es in der Praxis nicht unüblich, einen Film, welcher später mit einem XGA (1024x768) Projektor projiziert wird, im ersten Rendschritt des „Drei-Schritte-Renderings“ mit der vierfachen Projektorauflösung, also 2K, zu rendern. Wie viel mehr Auflösung beim ersten Rendervorgang benötigt wird kann natürlich nicht pauschalisiert werden, doch zeigt die Praxis, dass eine Vervielfachung der Auflösung einen häufig verwendeten Mittelwert darstellt. Der Tabelle (Tab. 1) kann man entnehmen, dass sich die Renderzeit bei einer Vervielfachung der Auflösung um den Faktor 3,3 erhöht.

Nicht zu vernachlässigen ist zudem die Zeitersparnis bei der Erkennung von Fehlern: Diese werden im neuen Verfahren direkt erkannt und können sofort korrigiert werden. Genauso sind schnelle Testrenderings zur Überprüfung des finalen Bildes möglich. Im alten Verfahren musste zuerst der einen Fehler verursachende Arbeitsschritt ausfindig gemacht und anschließend korrigiert werden. Die Wiederholung des gesamten „Drei-Schritte-Renderings“ war gegebenenfalls notwendig.

## 8 Diskussion und Ausblick

In diesem Paper wurde das neue „Ein-Schritt-Rendering“ zur Erzeugung von Längenanamorphosen vorgestellt. Ganz ersetzen kann es das bisherige „Drei-Schritte-Rendering“ dennoch nicht: Das neue Verfahren unterstützt momentan nur planare Projektionsflächen. Dabei ist es dem bisherigen in Qualität und Renderzeit deutlich überlegen. Aus praktischer Erfahrung machen planare Projektionsflächen einen nicht zu unterschätzenden Anteil der bespielten Flächen aus. Neben Projektionen auf planare Flächen wie Wände und Böden kann das „Ein-Schritt-Rendering“ auch für Monitore genutzt werden, welche fast ausnahmslos planar sind. Nichtsdestotrotz bleibt eine Umsetzung des „Ein-Schritt-Renderings“ für beliebig geformte Projektionsflächen eine spannende Herausforderung für künftige Arbeiten.

Das neue Verfahren eignet sich nicht nur für 3D-Umgebungen wie Blender, 3DsMax, Maya und andere. Ein weiterer Einsatzbereich sind 3D-Echtzeitanwendungen. Da lediglich eine weitere Matrixmultiplikation erforderlich ist, kann das „Ein-Schritt-Rendering“ sehr leicht in einer Echtzeit-Anwendung implementiert werden und würde dort kaum Performance kosten. Das „Drei-Schritte-Rendering“ dagegen benötigt zwei Rendschritte und ist somit ineffizienter. Darüber hinaus muss mit einer schlechteren Bildqualität gerechnet werden. Um die Qualität des „Ein-Schritt-Renderings“ zu erreichen, müsste der erste Rendschritt mit einer höheren Auflösung durchgeführt werden. Dies hat jedoch längere Renderzeiten und ein größeres Datenvolumen zur Folge, welche sich negativ bei der Performance bemerkbar machen.

Bisher gibt es nur statische Anamorphosen, die den Betrachter in eine vorbestimmte Position zwingen. Im Gegensatz dazu passt sich eine dynamische Anamorphose dem Betrachter an und wird somit immer korrekt betrachtet [SB07]. Bei einer dynamischen Anamorpho-

se müsste die Verzerrungsmatrix für jede Änderung der Betrachterposition neu errechnet werden. Angesichts der einfachen Berechnung stellt dies aber keine Schwierigkeiten dar. Die Ermittlung der Betrachterposition kann mittels Personen- oder besser Head-Tracking durchgeführt werden. Auch mehrere simultane Betrachter der Anamorphose wären unter Einsatz von Shutter-Brillen oder anderer optischer Trennungs-Techniken möglich. Durch ein Tracking-Verfahren verleiht man der Anamorphose Interaktivität. Zusätzlich erreicht man die Ergänzung um einen weiteren *depth cue*, der selbst-indizierten Tiefenparallaxe, welche den 3D-Eindruck nochmals verbessert.

Ausgehend von dem oben genannten Setup lassen sich eine Vielzahl möglicher Anwendungen entwickeln, die von Computerspielen bis zu Kunst- und Museumsinstallationen reichen. Diese Arbeit mag ein Beitrag dazu sein, Menschen mit Anamorphosen in ihren vielfältigen Formen und Ausprägungen zum Staunen zu bringen.

## Literatur

- [And08] K. Andersen. *The geometry of an art: the history of the mathematical theory of perspective from Alberti to Monge*. Springer, 2008.
- [Bee] J. Beever. Pavement drawings. Url: <http://www.julianbeever.net>, Letzter Zugriff: 30. Juli 2014.
- [DC11] F. De Comité. A new kind of three-dimensional anamorphosis. *Proceedings of Bridges 2011: Mathematics, Music, Art, Architecture, Culture*, 2011.
- [HC07] D. Hansford and D. Collins. Anamorphic 3d geometry. *Computing*, 79(2-4):211–223, 2007.
- [HNG00] J.L. Hunt, B.G. Nickel, and C. Gigault. Anamorphic images. *American Journal of Physics*, 68(3):232–237, 2000.
- [Ken] P. Kent. Anamorph me! Url: <http://www.anamorphosis.com/software.html>, Letzter Zugriff: 30. Juli 2014.
- [SB07] F. Solina and B. Batagelj. Dynamic anamorphosis. 2007. University of Ljubljana, Faculty of Computer and Information Science.
- [Top00] D. Topper. On anamorphosis: Setting some things straight. *Leonardo*, 33(2):115–124, 2000.
- [Urb] Urbanscreen. Offizielle internetseite. Url: <http://www.urbanscreen.com/>, Letzter Zugriff: 30. Juli 2014.

# Camera-Based Pointing Technique Using Dynamic On-Screen Markers

David Scherfgen, Rainer Herpers

Bonn-Rhein-Sieg University of Applied Sciences

Institute of Visual Computing

Grantham-Allee 20

53757 Sankt Augustin, Germany

E-mail: {david.scherfgen|rainer.herpers}@h-brs.de

**Abstract:** A cost-efficient alternative to outside-in tracking systems for pointing interaction with large displays is to equip the pointing device with a camera, whose images are matched to display content. This work presents the “Dynamic Marker Camera Tracking” (DMCT) framework for display-based camera tracking. It accounts for typical display characteristics and employs dynamic on-screen markers that follow the camera. An example marker implementation and a tracking recovery method are presented. DMCT can measure pointing locations with sub-millimeter precision in large tracking volumes and computes 6-DoF camera poses for 3D interaction. 60 Hz update rate and 24 ms latency were achieved. DMCT’s main limitation is the visible marker interfering with display content. In experiments, the prototype’s pointing efficiency was comparable to that of an OptiTrack system.

**Keywords:** pointing interaction, inside-out tracking, large displays, display lag

## 1 Introduction

Large and high-resolution display systems consisting of multiple LCDs or video projectors have become increasingly widespread. Advances in graphics and display hardware have made them a popular tool for data visualization, VR, simulation and product design.

A frequently used and intuitive means of interacting with large displays is “direct pointing”. The user points at a location of interest on the display surface using a pointing device, prior to triggering an action, i.e. “clicking”. Absolute pointing techniques provide a natural and convenient pointing experience and facilitate the eye-hand coordination [KBSR07].

Direct pointing requires a pointing device whose task is to determine the exact pointing location (see Figure 1). Often, the 6-DoF pose (six degrees of freedom) of the interaction device is of interest as well, e.g. for recognizing spatial gestures such as moving the device closer to the display in order to zoom in. Alongside ergonomic aspects, typical requirements for such devices are a high pointing precision, a large operating/tracking space, a high update rate and a low latency (less than 40 ms [PS10]).

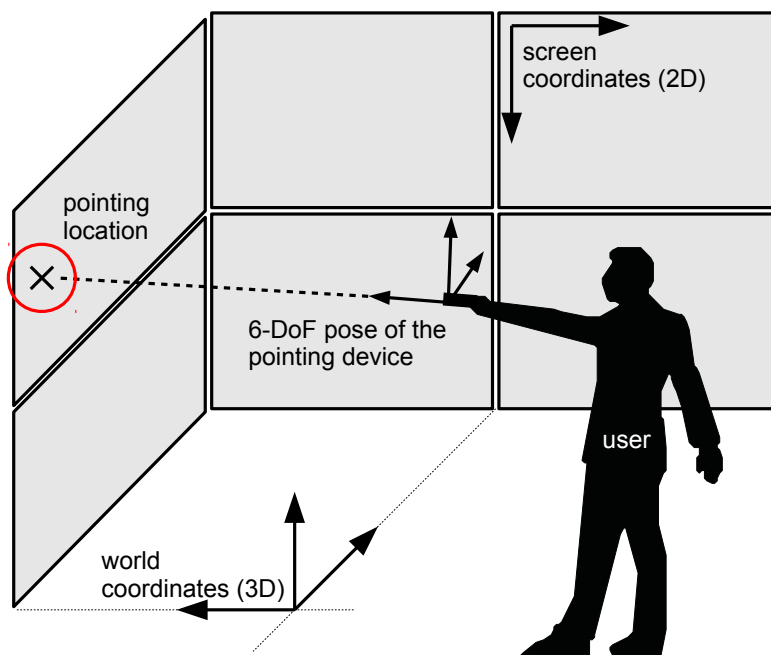


Figure 1: A user points at a display using a pointing device. Both the 2D pointing location and the 6-DoF pose of the device can be used for interacting with an application.

### 1.1 Outside-in tracking

The traditional tracking solution for interaction devices is to use an optical outside-in tracking system. The interaction device is equipped with markers that reflect or emit infrared light towards static tracking cameras. The tracking system computes the 6-DoF pose of the interaction device, from which the pointing location can be determined.

Fast and robust outside-in tracking solutions exist. Lightweight wireless interaction devices can be built easily. Multi-user support is achieved via distinct marker configurations. However, such tracking systems tend to be expensive and do not scale well. A larger tracking volume requires the use of additional cameras. They need to be recalibrated frequently, since even small changes in the camera setup reduce their accuracy. A weakness of such systems is their relatively low rotational precision [RBG01], which affects the pointing precision.

### 1.2 Inside-out camera tracking

Inside-out camera tracking is a cost-efficient alternative to outside-in tracking. In such systems, the interaction device contains a camera that continuously captures images from the device's point of view. So-called “beacons” (often LEDs or printed fiducial markers) are placed at known locations within the environment and detected in the camera images. When at least four beacons are detected and identified, their image coordinates allow to reconstruct the camera's 6-DoF pose. A popular example of an inside-out optical tracking system is the Nintendo Wii Remote, where two infrared LEDs with a fixed separation act as beacons.

### 1.3 Display-based inside-out camera tracking

A special case of inside-out tracking is display-based tracking. It is based on the assumption that the user usually orients the interaction device towards a display. Prominent features of the display content replace physical beacons. See section 2 for an overview of related work.

Capturing and processing camera images of dynamic display content is challenging. One problem involved is “display lag”. Display devices pre-process images before showing them, which causes a delay. Consequently, camera images show display content that has been rendered several frames ago. Further, LCD pixels cannot change instantly (pixel response time), thus camera images can show a mixture of two or more previous display images.

Currently existing display-based tracking systems are mostly proofs of concept and do not yet achieve a convincing performance in terms of precision, latency, update rate and support for multiple displays. The potential of such systems is therefore difficult to judge.

## 2 Related Work

Existing display-based camera tracking approaches can be categorized as follows:

- “Active” approaches rely on artificial marker images that are displayed on top of the application content. These markers are visible to the user, which can be a disadvantage.
- “Semi-active” approaches display tracking markers that can (almost) not be perceived by the user but can still be tracked in camera images. [CPSL10, GSHB07]
- “Passive” approaches do not display artificial markers. Instead, they continuously capture screenshots of the entire display content and extract image features. At the same time, they extract features from the camera images. Feature correspondences between the two images allow the computation of the camera pose. [BFL12, BBF12]

This work focuses on active approaches, since semi-active and passive approaches must usually analyze the display content on the pixel level, in real-time. This does not scale well to high-resolution display systems. Further, passive approaches require a sufficient amount of image features to be visible to the camera, which generally cannot be guaranteed.

An approach named “Direct Pointer” has been presented by Jiang et al. [JOMS06]. A visual marker (cursor), simulating a laser pointer, is shown on the display and tracked in the camera images. The main advantage of this technique is its small, non-intrusive marker. The system is able to continuously move the cursor towards the pointing location without estimating the complete camera pose. An update rate of 30 Hz has been achieved. However, only a 2D pointing location for a single display is computed.

“Marker-Based Display Registration” by Pears et al. [PJO09] uses a large marker consisting of four square shapes. These are detected in the images captured by a smartphone camera, then a homography is computed from the correspondences between display locations

and camera image locations (perspective-n-point problem, PnP). The homography allows to determine the pointing location in display pixel coordinates.

Similarly, Jeon et al. [JHKB06] use a “marker-cursor” in an interaction framework for large displays. It resembles a mouse pointer and functions similarly to Pears’ marker.

Both systems require the marker to be completely visible to the camera. This is achieved by continuously moving the marker to the current pointing location. Pears et al. additionally transform the marker image using the previously computed homography so that the marker’s size, orientation and shape are mostly unaffected by the camera position and viewing angle. For an observer, the marker looks as if it was projected from the camera onto the display.

In order to compensate for display lag, Pears et al. embed a synchronization time code into the marker that changes with each update step, which results in flickering. When the system fails to detect the marker for a number of consecutive frames, it moves it to the display center, where it must be “caught” by the user in order to continue.

The achieved latency, precision and accuracy have not been published. The update rates are 8 Hz (Pears et al.) and 10-15 Hz (Jeon et al.). It is to be expected that the systems easily lose track of the marker because they often cannot update its position quickly enough to follow the user’s motion. On today’s hardware, the systems would certainly be faster. Nevertheless, a problem with both implementations is that the pose estimation is based on a single marker with a small set of feature points. Pears et al. use only four points, which is the minimum for computing a homography. Consequently, robustness against noise or partial occlusion is not given. Multi-display support is not mentioned for either system.

### 3 Dynamic Marker Camera Tracking (DMCT)

This section presents “Dynamic Marker Camera Tracking” (DMCT), a modular framework for active display-based camera tracking and pointing with 6-DoF information.

DMCT builds on Pears’ idea of projecting a marker image from the camera onto the displays. The almost constant appearance of the marker in the camera image facilitates its detection from within a large tracking volume. DMCT adds support for multiple displays, introduces a tracking recovery method and optimizes latency and update rate by carefully timing image captures. Moreover, its markers can be dynamic, i.e. they can adapt to the current situation by changing their size, shape, arrangement or color/transparency.

#### 3.1 Marker concept

A marker consists of an internal state and two methods that must be specified, which are essential parts of the marker tracking cycle (see Figure 4):

- An “update” method that, given an input state, yields a new state, a marker image and a set of feature points within it. The image defines the marker’s visual appearance. It is later projected from the last known camera pose onto the displays (ray casting) where it is then overlaid to the display content and captured by the camera (Figure 2).



- A “detect” method that, given an input state and a camera image showing the marker in that state, detects and locates a subset of its feature points (ideally, all of them). The camera pose is then calculated using the features’ image coordinates (PnP solving).

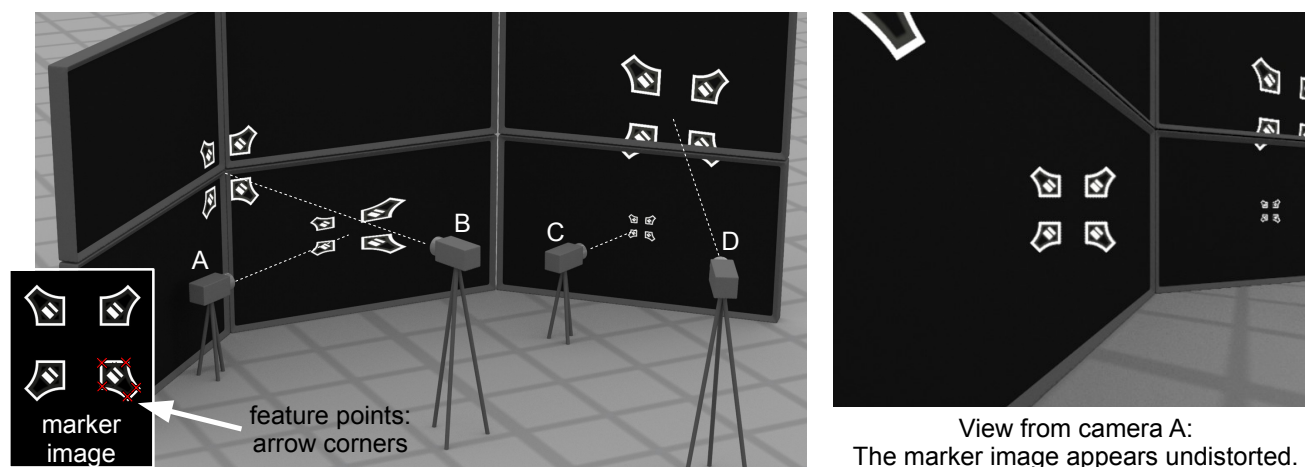


Figure 2: A marker image (lower left) is projected from a set of cameras onto multiple displays (left). The projection results in perspective distortions (e.g. see camera A), however the image appears undistorted from the camera’s perspective (right).

The marker used for evaluation consists of binary fiducial markers (at least four) arranged on a circle around the pointing location (Figure 3, left). Feature points are located at the fiducial centers. The fiducial count, size and arrangement radius are configurable. For identification, each fiducial contains a unique ID, which is scanned during marker detection. A camera pose can be determined from at least four detected fiducials. By using more than that, robustness is increased against partial occlusion or interruptions due to display bezels.

A strong black and white contrast found in the fiducials allows for reliable detection even in noisy camera images. However it also makes the marker more prominent and potentially distracting. It was observed that enough fiducials are usually still detected with 60% opacity. Thus, an adaptive opacity technique has been designed that dynamically controls each fiducial’s opacity: When a fiducial is detected, its opacity is decreased, otherwise it is increased. Flickering is suppressed by allowing only small changes, except when too many fiducials have become undetectable. The adaptive opacity technique attempts to adjust the marker’s transparency towards the maximum at which it is still detectable (Figure 3, right). The required opacity could also be directly estimated from the frame buffer content beneath the fiducials. However, DMCT avoids reading from the frame buffer, as it is typically slow.

### 3.2 Recovery concept

Fast camera motion can cause the system to “lose” the marker. For such cases, and for initialization (when the camera pose is not known yet), DMCT applies a recovery concept. As in the marker concept, a recovery image with feature points is shown on all displays,

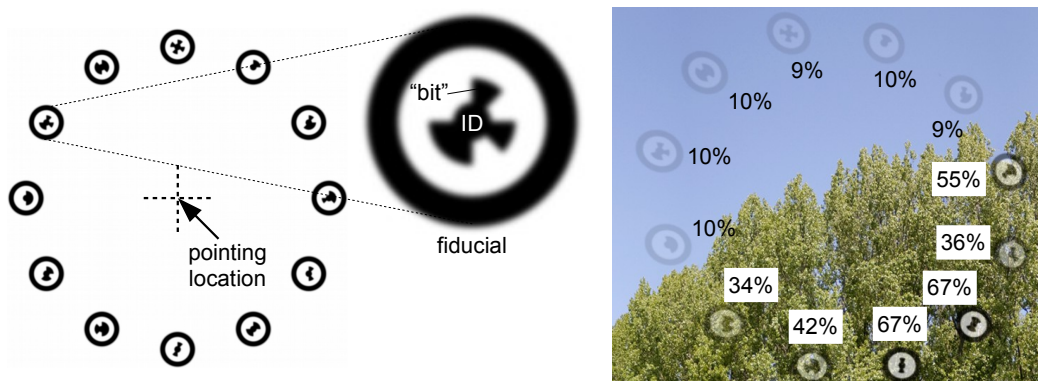


Figure 3: The example marker consists of circularly arranged fiducials (left). In order to make the marker less visible, the fiducials’ opacities are controlled dynamically (right).

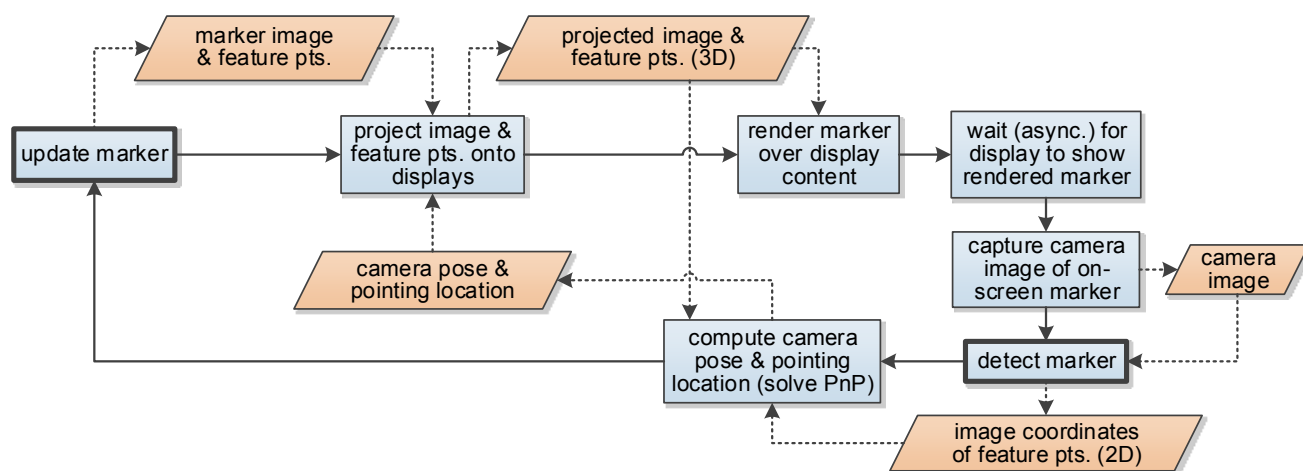


Figure 4: Overview of the marker tracking cycle. It consists of continuously updating, projecting, rendering, capturing and then detecting the marker.

however without projecting it. The system starts showing the recovery image after being unable to detect the marker for e.g. 100 ms. Assuming the pattern is detected immediately, it will be visible for several frames only (depending on display lag). When at least four of the pattern’s feature points were detected, the system determines the camera pose and continues using the marker. An example recovery implementation is depicted in Figure 5.

### 3.3 Displays, screens and screen setup calibration

Visualization environments can consist of multiple displays. These are modeled by “screens”, each representing one physical monitor’s display surface. DMCT mainly targets LCD monitors, therefore screens are assumed to be rectangular, planar and without distortion. For optimal update rate and latency, it is recommended that all displays be synchronized.

DMCT requires each screen’s exact 3D location and orientation in order to project marker images onto them via ray casting. A screen setup calibration procedure is used for this purpose. It is based on photographs of the displays showing special calibration markers (see

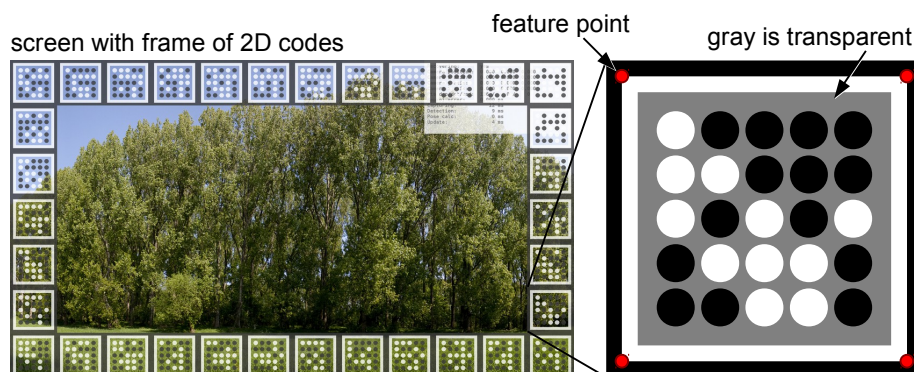


Figure 5: This example recovery pattern consists of a frame of 2D codes around each display. Codes contain an ID and a CRC value. Detecting a single code suffices to resume tracking.

Figure 6), whose inner corners provide feature points. Using Zhang’s widely used camera calibration technique for planar targets, each photograph yields estimates of the relative 3D transforms between pairs of visible screens. One reference screen’s absolute transform is defined arbitrarily by the user. Finally, the estimated overall display arrangement and the camera calibration parameters are refined using Levenberg-Marquardt optimization by minimizing the global reprojection error of all feature points in all photographs.

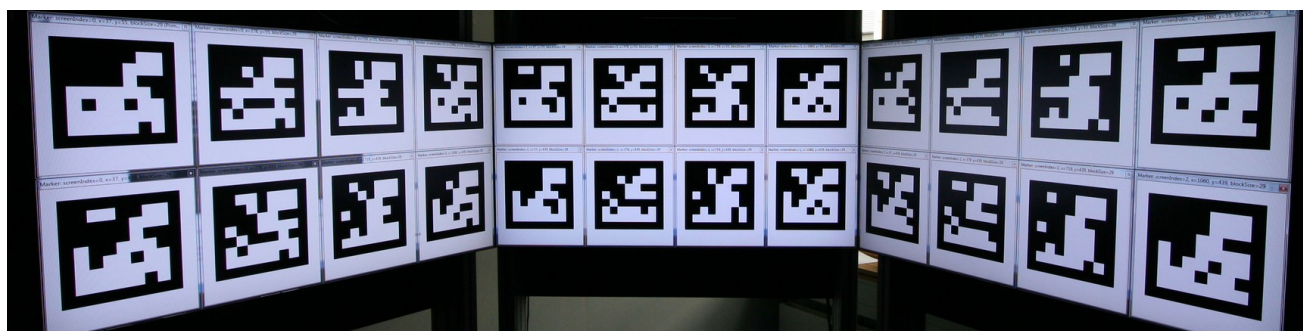


Figure 6: Calibration markers being shown on three displays. The exact 3D display arrangement is found by processing high-resolution photographs taken from different perspectives.

### 3.4 Camera and temporal calibration

In order for a camera to be suitable for DMCT, it should support short exposure times such as 1 ms in order to limit the impact of motion blur. For synchronizing the camera to the displays, it is important to have control over the precise moment of image capture (to millisecond precision). Webcams and smartphones usually do not allow this. The lens’ field of view should be neither too wide (large projected marker) nor too narrow (tracking is lost easily). The prototype’s lens/camera combination provides a 43° horizontal field of view.

The DMCT approach relies on being able to capture camera images of specific video frames that have previously been rendered and output to the displays. For this reason, the

display devices’ characteristics must be taken into account:

- Display lag is compensated by delaying the image capture by a duration  $D$  (capture delay), measured from the v-sync signal following the rendering of the video frame. With displays that require a high  $D$  value, the system is more likely to lose marker tracking after fast camera motion. Therefore, low-delay displays are preferable.
- High pixel response times result in “ghosting”, i.e. a video frame remaining partially visible during the following frame(s) due to the display’s pixels reacting slowly. This makes reliable marker detection difficult, since “old” and “new” marker images are both visible and may overlap. The proposed solution to this problem is to display the same marker image for  $R$  consecutive frames (frame repetition) before capturing a camera image. Essentially, this grants the display more time to produce a clear image. However,  $R$  limits the achievable update rate of the system, which is  $\frac{F}{R}$ , where  $F$  is the display’s refresh frequency, e.g. 60 Hz. Thus, a low pixel response time is desirable.

Proper  $D$  and  $R$  values are determined in “temporal calibration”. This involves showing on all displays a repeating countdown, e.g. from +5 to  $-5$ , and pointing the camera towards the displays. The countdown updates every  $R$  frames. At “0”, a camera image is captured with a delay of  $D$ , while the countdown proceeds. The camera image is shown to the user who then adjusts  $D$  and  $R$  until the image reliably shows clearly visible “0”s (Figure 7).

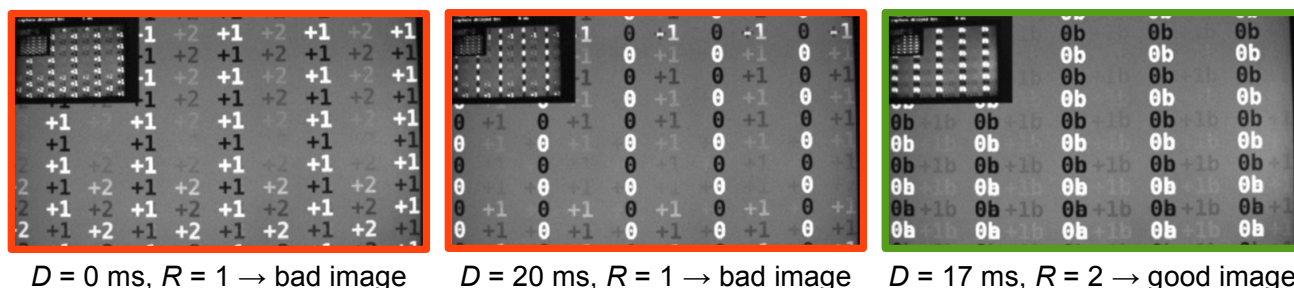


Figure 7: Camera images captured during temporal calibration. A 17 ms capture delay and a frame repetition of 2 result in images showing clear “0”s, while the first two images exhibit ghosting. Repeated frames are distinguished via letters, e.g. “0a” and “0b” for  $R = 2$ .

## 4 Implementation

The DMCT system has been implemented in C++ using OpenCV for image processing. The implementation includes a VRPN server that streams the 6-DoF data to client applications. A “mouse emulator” application allows to control the mouse pointer under Windows for non-fullscreen applications, using a transparent “always-on-top” window for the marker. Rotation is translated to mouse wheel motion, i.e. the user can scroll by turning the pointing device.

Figure 8 shows the pointing device prototype built around a MATRIX VISION mvBlue-FOX 220aC camera. The current implementation requires a USB cable.

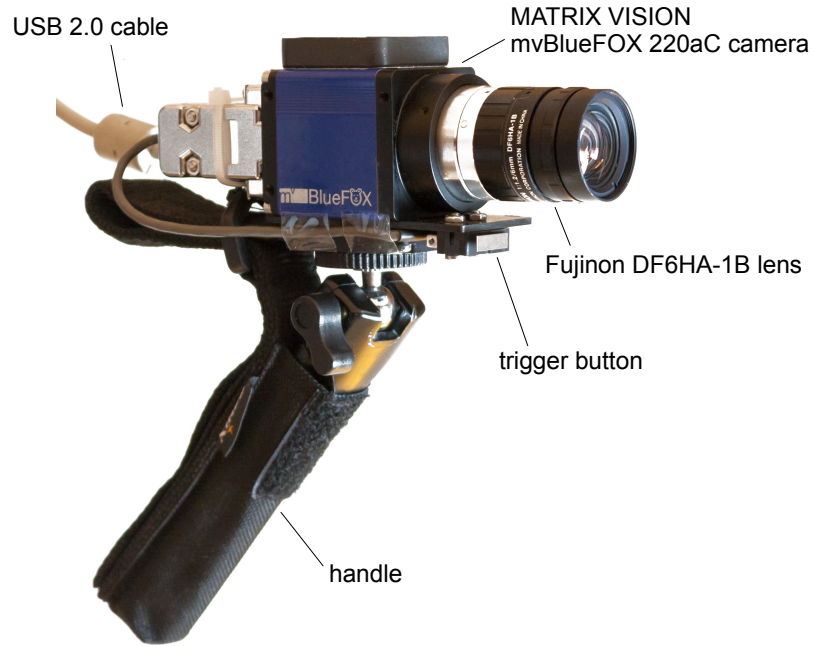


Figure 8: Prototype used for evaluating the DMCT implementation.

## 5 Evaluation and Results

### 5.1 Latency and update rate

Latency  $L$  is defined as the system’s reaction time, i.e. the average timespan between moving the camera and the system reporting the motion. When the pointing location is indicated by an on-screen pointer, users observe a higher end-to-end latency that includes display lag.

$L$  was measured using microphone recordings. The camera was physically bumped repeatedly, which resulted in camera motion and a distinguishable sound (event “A”). Whenever the testing program detected a motion, it generated a sound signal using the computer’s speakers (event “B”).  $L$  was then determined by measuring the timespan between events “A” and “B” in the audio recording and subtracting the system’s audio delay. In an experiment with a 60 Hz display and an Intel Core i7 920 @ 2.66 GHz,  $L = 24$  ms was measured for  $R = 1$  and  $L = 33$  ms for  $R = 2$  (averaged from 60 samples). The 24 ms latency is explained by an average 8 ms of waiting for the v-sync signal (half of  $\frac{1}{60}$  s), 12 ms for image capturing (1 ms exposure) and 4 ms for detecting the marker and computing the camera pose.

Using the same setup, update rates of 60 Hz ( $R = 1$ ) and 30 Hz ( $R = 2$ ) were achieved. However, most displays that were tested required  $R = 2$  for sufficiently clear images.

### 5.2 Pointing precision and 3D position accuracy

First, the 2D pointing precision was measured by placing the camera on a tripod and determining the average Euclidean distance between two consecutive measurements (no ground truth required) over 2000 frames. Among other parameters, camera distance, fiducial count and transparency were varied. Second, the absolute accuracy of the 3D position was mea-

sured by manually moving the camera through the tracking volume and comparing the data to the output of a 6-camera OptiTrack V100:R2 system (see Figure 10).

With 24 fiducials and 100% opacity, a pointing precision below 0.01 mm was measured at a camera distance of 1.3 m using a 46" LCD ( $1366 \times 768$ ). More realistic conditions (12 fiducials with 60% opacity and dynamic background, see below) still resulted in a precision of 0.03 mm, which was 5 times more precise than the OptiTrack system. Figure 9 shows some parameters' influences on pointing precision. As it is to be expected with inside-out systems, the camera's 3D position is measured less accurately. At 1.1 m distance, the average error was 5 mm (90% samples below 9 mm, 99% below 17 mm). This should suffice for the detection of 3D gestures.

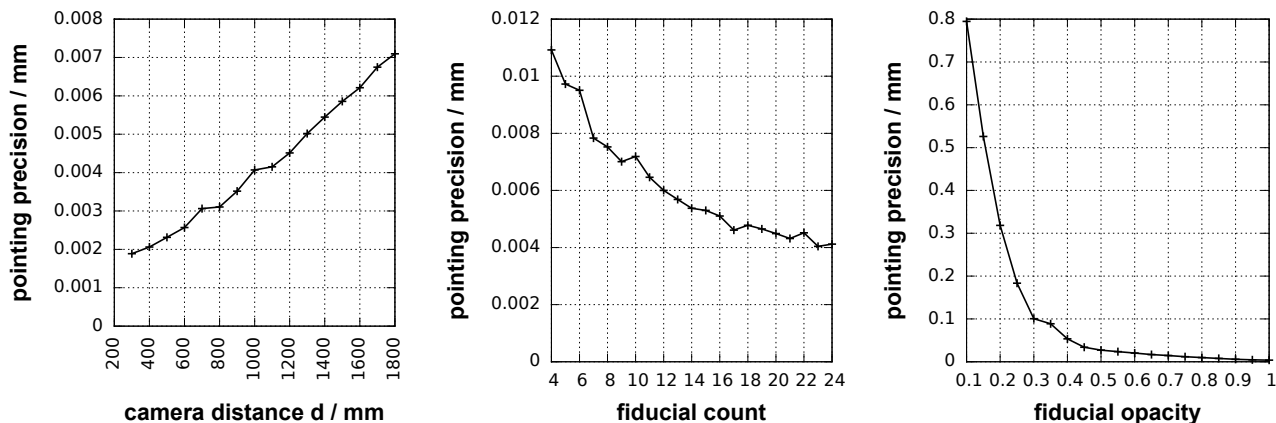


Figure 9: Influence of camera distance (left), fiducial count (middle) and fiducial opacity (right) on the pointing precision. The base settings were: 1 m distance, white background, 24 fiducials, 100% fiducial opacity. In the opacity experiment (right), various screenshots of 2D/3D applications and games were used as background images. The images were randomly selected, translated and scaled on a frame-by-frame basis.

### 5.3 Operational distance

Below some minimum camera distance, the display's resolution is too low to visualize the marker with sufficient detail (in the example marker, the fiducial IDs will become unreadable). Beyond some maximum distance, the size of the projected marker exceeds the display size. Further, the image becomes unsharp outside the camera's focal range. Using the 46" LCD, the system was operational at distances from 0.2 m to 3 m. Beyond 3 m, the marker image projection grew too large in the vertical direction.

### 5.4 Efficiency in pointing tasks

Two user experiments with 30 subjects (12 ♀, 18 ♂, ages 24–56, average age 36) were conducted, alternately using DMCT (12 fiducials) and OptiTrack for comparison. In the first experiment, subjects were asked to point and click at random circular targets as quickly as they could (Figure 10, left). Inter-target distances and diameters were varied systematically.

Subjects knew where the next target would appear, as the experiment was not intended to measure reaction times. Subjects were slightly, but consistently faster using the OptiTrack system by about 5%, regardless of target size and distance. This was attributed to OptiTrack’s lower latency (10 ms), since when artificially increased to match DMCT, a difference was not observed. Subjects rated the marker’s distractiveness 1.93 out of 10,  $\sigma = 1.86$ .

In the second experiment, subjects performed drag-and-drop operations on individual targets that first had to be located among others. The targets were distributed similarly to the marker’s fiducials (Figure 10, right). In this intentionally difficult scenario, subjects rated the same DMCT marker more distracting than previously (5.8 out of 10,  $\sigma = 1.8$ ) and required an average of 7% more time compared to OptiTrack with matched latency.

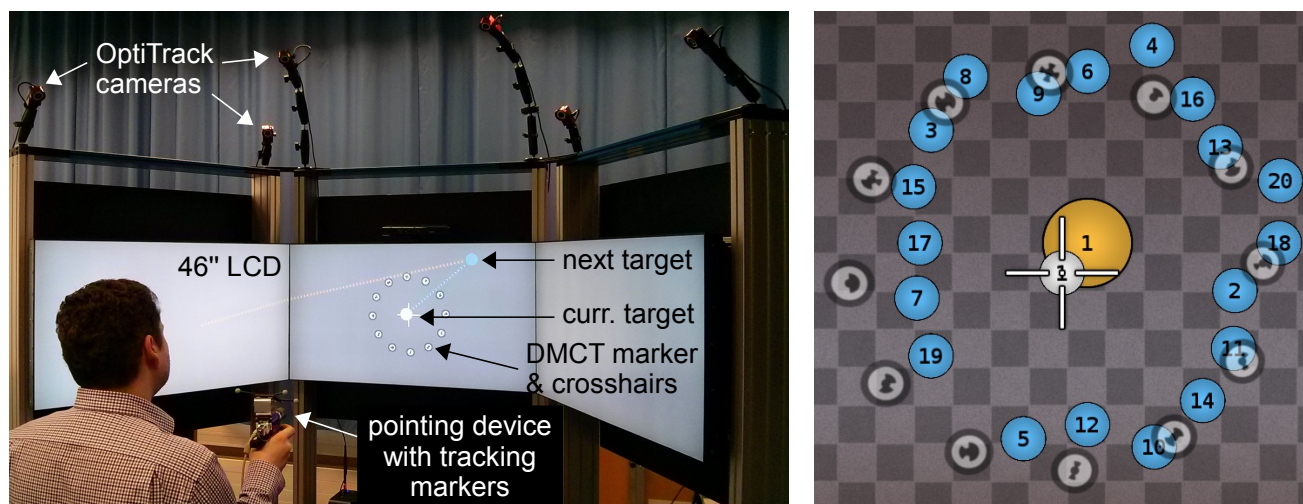


Figure 10: A subject performing the pointing task experiment (left). The OptiTrack system serves as a reference for comparison. In the drag-and-drop experiment (right), the DMCT marker overlaps the targets among which the subject has to find a specific one.

## 6 Conclusion and Future Work

The presented display-based camera tracking approach DMCT is suitable for pointing interaction in multi-display environments. It has been shown to measure pointing locations with sub-millimeter precision. Its dynamic marker concept can be used to achieve larger tracking volumes, increased robustness and reduced marker visibility. When tracking is lost, the recovery concept allows to re-establish it quickly. By carefully timing image captures and taking display characteristics into account, DMCT achieves high update rates and low latencies. In pointing tasks, the prototype performs comparably to an OptiTrack system.

In future, more sophisticated methods of reducing marker visibility should be developed, since the example marker can distract the user. It should be investigated how the user’s perception of the marker is influenced by parameters such as fiducial count, size and pattern. Latency should be reduced further through the use of a CMOS camera. Finally, a wireless

pointing device is highly desirable, for which some technical challenges have to be solved (on-board image processing, power consumption, low-latency wireless communication and synchronization).

## References

- [BBF12] D. Baur, S. Boring, and S. Feiner. Virtual Projection: Exploring Optical Projection as a Metaphor for Multi-Device Interaction. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 1693–1702. ACM Press, 2012.
- [BFL12] M. Baldauf, P. Fröhlich, and K. Lasinger. A Scalable Framework for Markerless Camera-Based Smartphone Interaction With Large Public Displays. In *Proc. of the 2012 Int. Symp. on Pervasive Displays*, pages 1–5. ACM Press, 2012.
- [CPSL10] C. Celozzi, G. Paravati, A. Sanna, and F. Lamberti. A 6-DOF ARTag-Based Tracking System. *IEEE Trans. on Consumer Electronics*, 56(1):203–210, 2010.
- [GSHB07] A. Grundhofer, M. Seeger, F. Hantsch, and O. Bimber. Dynamic Adaptation of Projected Imperceptible Codes. In *Proc. of the 2007 6th IEEE and ACM Int. Symp. on Mixed and AR*, pages 1–10. IEEE, 2007.
- [JHKB06] S. Jeon, J. Hwang, G. J. Kim, and M. Billinghurst. Interaction Techniques in Large Display Environments Using Hand-held Devices. In *Proc. of the ACM Symp. on VR Software and Technology*, pages 100–103. ACM Press, 2006.
- [JOMS06] H. Jiang, E. Ofek, N. Moraveji, and Y. Shi. Direct Pointer: Direct Manipulation for Large-Display Interaction using Handheld Cameras. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 1107–1110, New York, 2006. ACM Press.
- [KBSR07] W. A. König, H.-J. Bieg, T. Schmidt, and H. Reiterer. Position-Independent Interaction for Large High-Resolution Displays. In *Proc. of IADIS Int. Conf. on Interfaces and HCI 2007*, pages 117–125. IADIS Press, 2007.
- [PJO09] N. Pears, D. G. Jackson, and P. Olivier. Smart Phone Interaction with Registered Displays. *IEEE Pervasive Computing*, 8(2):14–21, 2009.
- [PS10] A. Pavlovyh and W. Stuerzlinger. Effects of Latency Jitter and Dropouts in Pointing Tasks. In *Proc. of the Poster Session of the Graph. Interface 2010*, pages 30–32, 2010.
- [RBG01] J. P. Rolland, Y. Baillot, and A. A. Goon. A Survey of Tracking Technology for Virtual Environments. In *Fundamentals of Wearable Computers and AR*, pages 67–112. Lawrence Erlbaum, 2001.



# Enhancing Rendering Performance with View-Direction-Based Rendering Techniques for Large, High Resolution Multi-Display Systems

Martin Weier<sup>†\*</sup>, Jens Maiero\*, Thorsten Roth\*, André Hinkenjann\*, Philipp Slusallek<sup>†</sup>

\*Bonn-Rhein-Sieg University of Applied Sciences    †Saarland University & DFKI GmbH

Institute of Visual Computing

53757 Sankt Augustin

{Martin.Weier|Jens.Maiero}@h-brs.de

{Thorsten.Roth|Andre.Hinkenjann}@h-brs.de

Computer Graphics Lab

66123 Saarbrücken

slusallek@cs.uni-saarland.de

**Abstract:** In contrast to projection-based systems, large, high resolution multi-display systems offer a high pixel density on a large visualization area. This enables users to step up to the displays and see a small but highly detailed area. If the users move back a few steps they don't perceive details at pixel level but will instead get an overview of the whole visualization. Rendering techniques for design evaluation and review or for visualizing large volume data (e.g. Big Data applications) often use computationally expensive ray-based methods. Due to the number of pixels and the amount of data, these methods often do not achieve interactive frame rates.

A view direction based (VDB) rendering technique renders the user's central field of view in high quality whereas the surrounding is rendered with a level-of-detail approach depending on the distance to the user's central field of view. This approach mimics the physiology of the human eye and conserves the advantage of highly detailed information when standing close to the multi-display system as well as the general overview of the whole scene. In this paper we propose a prototype implementation and evaluation of a focus-based rendering technique based on a hybrid ray tracing/sparse voxel octree rendering approach.

**Keywords:** Focus-based Rendering Technique, Multi-Display System, Ray Tracing, Focus-Context



Figure 1: Results of our focus and context strategy using head tracking in front of the large tiled display wall *HORNET*

## 1 Introduction

Focus and context techniques are mainly user-centered because of their purpose of specifically preparing and presenting information to the user in a way that makes it easier and more efficient to gain insights and accomplish tasks. Moreover, focus and context is used for interactive exploration of complex data sets. Focus can be controlled by using the observer's own field of view or, alternatively, through a separate user, controlling the focus in order to draw the observer's attention to a specific position or data segment. In this paper we present a novel context and focus technique to be used in combination with compute-intensive visualization methods like (hybrid) path tracing, volume rendering or sparse voxel octree rendering. As all of the aforementioned techniques are computationally demanding it is challenging to achieve interactive or even real-time frame rates at desirable resolutions. In this publication, we describe an approach to increase rendering performance of such complex visualization techniques by using a focus and context approach.

The presented implementation and evaluation of a view-dependent renderer is based on a hybrid ray tracing/sparse voxel octree approach with the results being displayed on a large  $7 \times 5$  monitor multi-display system. This offers enough interaction space for the user to freely move around. The viewpoint and orientation used by our method are based on a 6-DoF tracking system.

First we give a brief overview on related work in the field of focus and context techniques. Afterwards, we introduce the tiled-display system HORNET that we use as the testing platform. Before stepping into the implementation details of our rendering system, we list some properties of the human visual system that enable us to coarsen the regions in the visual peripheral. To conclude we show the results of some user studies, present the increase in performance that can be achieved by using a hybrid triangle voxel renderer and give an outlook on future work.

## 2 Related Work

Visualization algorithms combined with focus and context techniques have been published a lot over the last decades, most of them with the purpose to support the user in understanding complex data or patterns. One technique is the Document Lens [RM93]. They use a magnifier lens for interactively reading documents in 3D. Using a fisheye focus and context scheme to visualize large 2D hierarchies was published by Lamping et al. [LR94]. The authors also provided an efficient navigation technique to systematically explore the data.

Volume rendering is well-suited for focus and context applications, because volumes often contain a greater amount of information than what can be visualized at once. The focus and context schemes related to volume rendering can be categorized in (a) distortion lenses, which is similar to the Document Lens described above. Different lenses and distortion techniques are already published [ELJ01]. (b) cutaway illustrations, such visualization approaches allow the user a superman-like x-ray vision [KSW06]. And (c) multi-resolution focus and context

approaches. Levoy et al. [LW90] introduced a multi-resolution technique based on an eye tracker.

While the Clearview system [KSW06] and other medical applications like [MCZ13] and [RLH14] successfully integrated focus and context metaphors in their systems, they concentrate on techniques for knowledge transfer, for instance to visualize special features of a volume. The context in those applications is visible to locate and or connect the focused feature to the surrounding, which differs from our approach. Other systems like the foveal inset [SAKH06] use an additional projector to display HiRes content on top of a coarser projection.

Our approach concentrates on (a) large, high resolution multi-display systems and (b) enhancing rendering performance using a hybrid renderer. In contrast to the above mentioned methods, our focus-based rendering technique enables to render the user's point of view in high quality and the peripheral based on a level-of-detail approach, which is not clearly visible for the user. This is similar to [PK13]. However, they propose a system to display and explore 2D gigapixel imagery and not render 3D scenes. Gigapixel images are challenging because of their size, where a system's bandwidth is the major problem. In our system the image generation itself is challenging. Hence, we use focus and context strategies to enhance rendering performance. Furthermore, the proposed system takes the geometry of the multi-display system to compute the focal point of the user into account. This approach is transferable to other systems as well.

### 3 Multi-Display System HORNET

HORNET is an ultra high resolution display together with a corresponding PC cluster that was installed for the purpose of large scale visualization of big data and for 1:1 high quality (global illumination) rendering. The curved display surface measures about  $7 \times 3$  square meters and consists of 35 Full HD (i.e.  $1920 \times 1080$  pixels) monitors with less than three millimeters bezel each. The total resolution sums up to 72 megapixels. The pixel density is high enough to surpass the resolution of the human eye when standing more than two meters away from the displays. The monitors are driven by three *display PCs* that have three up-to-date graphics cards with four outputs each. These graphics cards are capable of producing standard (OpenGL + shader, local illumination) graphics in real-time for moderate scene sizes. For more sophisticated rendering techniques such as physically-based global illumination graphics, twelve *cluster PCs* take over to compute and render the images. These PCs are connected by a 60 Gbit/s fibre ethernet link to the display PCs. For interaction, HORNET is equipped with an optical tracking system consisting of seven tracking cameras with active infrared illumination.

### 4 The human visual system

Looking straight ahead humans have an almost  $180^\circ$  horizontal FOV (Field of View). If the eyeball rotation is included the horizontal FOV is up to  $270^\circ$ . A renderer only needs

to render what can be seen, i.e. what is within the user's FOV. Considering an almost flat or slightly curved immersive environment like HORNET this restriction only occurs if you stand close to the displays or look at them from an angle. However, simply turning of the displays which cannot be seen by the user is problematic since the overall ambient lighting condition in the room will change if everything behind the user is displayed black.

Another important physiological property is the visual acuity. Under an optimal lighting situation the human eye can barely distinguish two points if the outgoing rays of the two points with respect to the eye are in an angle of  $1/60^\circ$ , that is an arcminute (spatial resolution) [Dee98]. However, since the angle of the rays emerging from two points to the eyes gets steeper as the observer moves away, there is a fixed limit on the spatial resolution of an optimal display.

The retina itself consist of  $6 \cdot 10^6$  cones and approximately 20 times as many rods, building the photosensitive layer of the eye [Gol10][p. 28]. The rods are responsible for the brightness sensation in twilight. The cones – divided into multiple types for different wave lengths, namely S-cones (blue), M-cones (green) and L-cones (red) – are responsible for detailed color sensation. The cones are not evenly distributed: The central area of the retina (fovea centralis) consists entirely of cones. Their density drops significantly in the peripheral area. Both rods and cones are not evenly distributed the spatial resolution of the eye drops significantly from within a small area where humans are able to generate a sharp image.

The region of the highest visual accuracy (visus) is only in a  $5^\circ$  angle from the optical axis. This visus quickly drops in extrafoveal direction with respect to the distribution of the cones. Due to this observation we have created a rendering system that only renders a sharp, highly detailed image for a narrow field of view and try to find coarser representations for the parts which are in the visual peripheral area.

However, humans do not realize that they are equipped with such "poor" visual senses. This is mainly due to the fact that the eyes are constantly moving (saccade). Humans perform saccades e.g. to create a sharp image as a scanning saccade exploring the environment and/or to quickly focus on e.g. a moving object or other visual stimulus as a reflexive or predictive saccade [Gol10][p. 128]. Saccades are the biggest challenges that emerge in a rendering system that alters the visual accuracy based on the human visual system (see Sec. 6).

## 5 Focus-based Rendering using a Hybrid Voxel-Triangle Renderer

The renderer combines a coarse voxel representation with polygonal data in a common octree structure. This facilitates rendering coarse representations of a scene for the visual peripheral area and displaying polygonal data within the area with maximal visus. To do so we need to determine what the user sees on the display wall HORNET and which regions have to be rendered sharp. The following section introduces one approach to generate the field of sharp vision (FSV). Afterwards we introduce the hybrid renderer and the metric used to determine what to show on screen and in which resolution. Finally, we propose a post processing step to blur the transitions between multiple levels of our representation.

## 5.1 Finding the Visual Area of the User

In order to render the specific area the user is looking at in a high level-of-detail, we first have to know which area this is (FSV). To this end, the user’s position and orientation in 3D space are determined with an optical tracking system. This data is then used to perform an intersection test with a model of the display wall and a ray describing the user’s position and orientation in front of the wall. This yields the point on the display wall the user is looking at when setting his eyes straight. In our approach the model of the display wall is not described by a 3D model. We perform two 2D intersections to speed up this process. One intersection is performed in the XY-plane (top view) and one in the XZ-plane (side view). This is a valid approach due to the geometry of the wall, being only a straight line from the side view when looking at a specific x-coordinate. The FSV is specified as an angle describing the horizontal and vertical angular extent of the area to be rendered sharp. Five rays are created to describe the user’s FSV. A central ray defined by the direction the user is looking at and four bounding rays described by angular offsets used to describe the FSV. The intersection of these rays with the virtual scene is used to compute a leftmost, rightmost, topmost and bottommost intersection point in normalized device coordinates.

## 5.2 Metric to create the Detail-guide Image

The intersection of the user’s field of sharp vision and the display wall results in an elliptical/oval shape  $e$ . This shape is defined by the position of the central ray  $c \in \mathbb{R}^2$ , the user is looking at, as well as the bounds  $h_{left}, h_{right}, v_{top}, v_{bottom} \in \mathbb{R}$  all in normalized device coordinates. For now we assume that we have an orthonormal coordinate system with the extent of the oval on the  $x$  and  $y$  axis. We want to compute a metric for each ray that hits the view plane in a point  $p \in \mathbb{R}^2$  describing if a region has to be rendered with full detail or if a coarser representation is sufficient. The idea is to use the distance from the point  $p$  to the oval  $e$  to compute a greyscale image serving as a “detail-guide image” (DGI). In our case this means that areas which are black in the DGI will be rendered at full detail, decreasing down to the lowest level-of-detail with increasing brightness (see Fig. 4b).

Everything inside the oval shape is rendered sharply. To determine if the point  $p$  lies within the oval shape we project it using the parameters from the oval shape to a unit circle. First we determine the quadrant of the oval shape  $e$  in which  $p$  falls. Now the distances from  $h_{left}, h_{right}, v_{top}, v_{bottom}$  to  $c$  depending on the quadrant are computed. These distances are denoted  $s = (s_{horiz}, s_{vert}), s \in \mathbb{R}^2$  respectively.

To determine if a point lies within the oval we can compute  $H(p)$

$$H(p) = \begin{cases} 1, & \text{if } \left| (abs(p) - c) \cdot \begin{pmatrix} 1/s_{horiz} \\ 1/s_{vert} \end{pmatrix} \right| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

If the point is in the oval we return zero as distance, if it is not inside the oval we compute the distance  $d$  of the point  $p$  to the oval  $e$ . Finding the distance between a point and an oval is not straightforward. Each point  $p_e$  on a oval/ellipse centered in the coordinate systems

origin can be described by

$$\begin{pmatrix} s_x \cos(\phi) \\ s_y \sin(\phi) \end{pmatrix} \quad 0 \leq \phi \leq 2\pi$$

Finding the distance of a point to a point on the oval/ellipse can thus be computed by the function

$$\text{dist}(\phi) = \sqrt{(p_x - s_x \cos(\phi))^2 + (p_y - s_y \sin(\phi))^2}, \quad (1)$$

We are interested in finding  $\phi : \phi \leq \theta \forall \theta \in [0, \pi/2]$  for our specific quadrant, because we have an oval shape. One solution to minimize this function is to use a nested interval approach to find  $\phi$ . However, such an approach is not robust, inaccurate and computationally demanding. Therefore, we use an approach introduced in [Ebe13]. Since, for the closest point on an ellipse/oval holds, that the normal of this point must point towards  $p$ , we can reformulate fn. 1 and create a new function  $F(t)$  (fn. 2) where the unique root gives the minimal distance.

$$F(t) = \left( \frac{s_x p_x}{t + s_x^2} \right)^2 + \left( \frac{s_y p_y}{t + s_y^2} \right)^2 - 1 = 0 \quad t \geq -s_x^2 \quad (2)$$

To find this unique root a hybrid approach between bisection and newtons method is used. See [Ebe13] for implementation details. Finally, we norm this distance according to our normalized device coordinates  $d_{norm} = \frac{d}{\sqrt{(2)}}$ ,  $d_{norm} \in [0, 1]$ .

$d_{norm}$  can be computed independently for each intersection of our ray with the view plane and is used during the traversal of the octree to determine when to stop traversing and thus to decide to display the coarse voxel or the fine polygonal representation. If the octree has  $n$  levels and we have traversed the tree to level  $k$  we can stop traversing if

$$k \geq \text{round}(n \cdot (1 - d_{norm}))$$

This way all possible LoDs can be visible on the screen. However, since we have a normalized distance on the view plane this additionally allows to artificially alter  $d_{norm}$  to represent a steeper or shallower fall-off of the voxel representation in the rendering of the peripheral area.

### 5.3 Hybrid Rendering System

In this subsection we give a brief overview on our hybrid renderer, where we combine a pre-filtered voxel description with a polygonal description in a hybrid octree acceleration structure. This representation is used in an OpenCL-based ray caster. Fig. 2 gives an overview on the hybrid acceleration structure. The octree contains a representation of the scene with different levels-of-detail. Each leaf node of the pre-filtered voxel description represents a color and a normal. Inner nodes are computed by averaging colors and normals of the node's children. We encode each node in 8 bytes (see. Fig. 3a) and each voxel in 16 bytes (see Fig. 3b). The leaf nodes either contain an index to a single triangle, if this triangle is the only one the voxel intersects, or an index to a triangle index list, storing for each voxel the count (**bold black and underlined**) and subsequent indices of the triangles the voxel cell intersects.

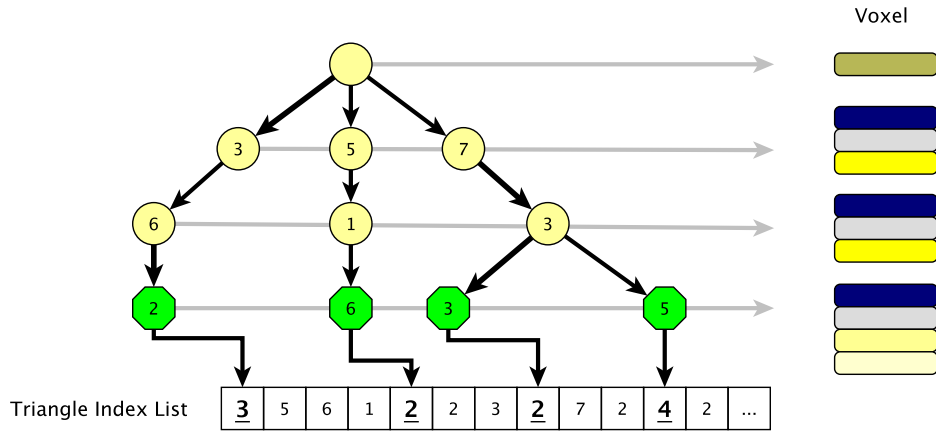


Figure 2: Overview on the hybrid octree structure. Each node contains its child index. Empty nodes are not shown. Inner nodes (light yellow - round) and leaf nodes (green - octahedron) reference voxels, containing colors and normals. Leaf nodes point to either a single triangle index or one entry in a list containing the number of triangles followed by all triangle indices a voxel intersects.

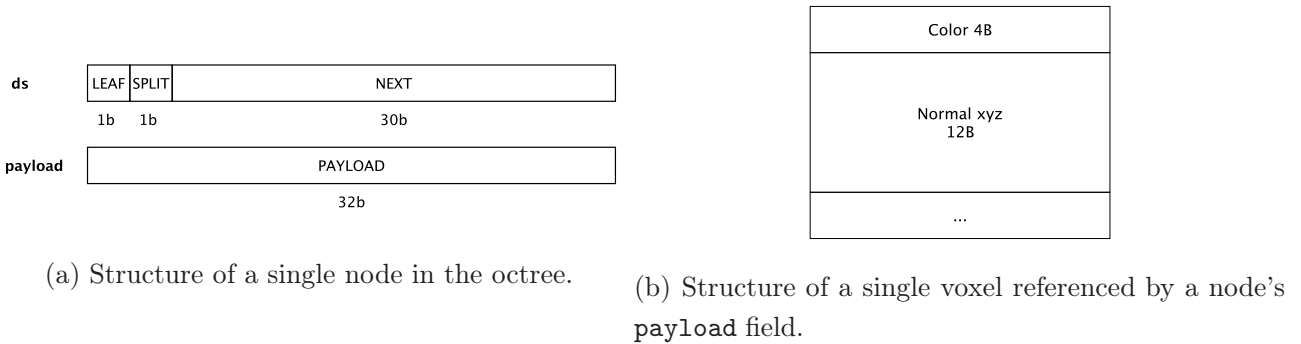
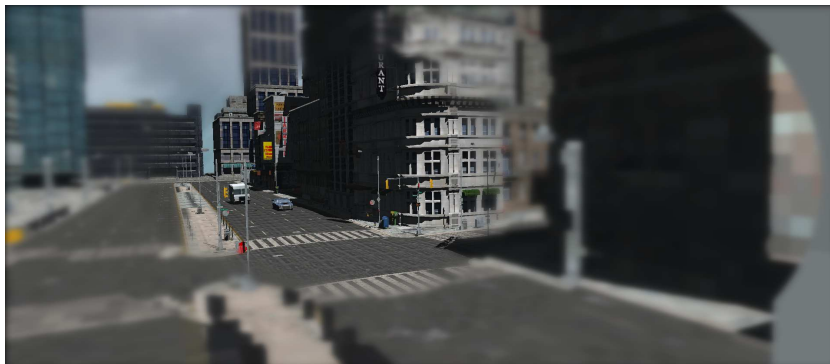


Figure 3: Overview of the acceleration structure node types.

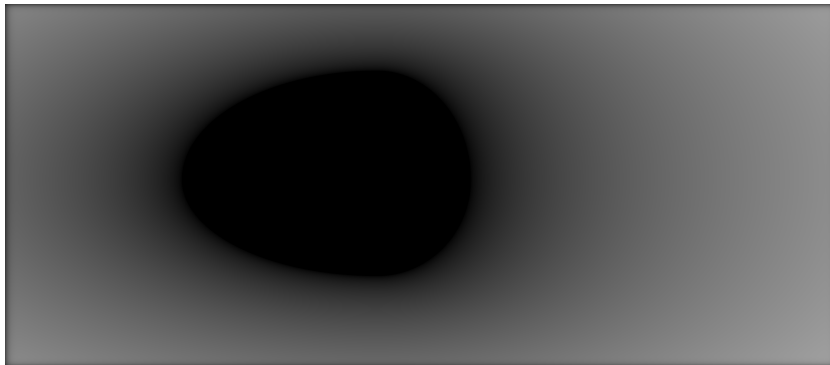
The construction of the acceleration structure starts by voxelizing the whole scene with a fixed resolution. In the voxelization each triangle generates a set of triangle-cell overlaps resulting in the voxel's position, its normal and a color. In addition we keep track of the indices of primitives the voxel cell intersects. These triangle-cell overlaps can either be computed by actually intersecting triangles with voxel cells [Pan11] [SS10] or using a rasterization process using OpenGL and GLSL [CG12]. We have implemented the voxelizer using OpenGL and GLSL because color and normal information are immediately available using the OpenGL rasterization pipeline.

The voxel's position is encoded in a 64 bit morton code [Mor66]. The algorithm starts by sorting the voxel list using the morton codes. Then the leaf nodes are created that store the index to the triangle index list. Afterwards the octree is constructed bottom-up on the CPU.

To start rendering the data structure is uploaded into the GPU memory. The traversal of this data structure is implemented in OpenCL using a small stack on the GPU similar to



(a)



(b)

Figure 4: These images show the rendering of the Urban Sprawl scene with a limited FSV (a) and the detail-guide image to steer the reduction of visual accuracy (b)

the one described by [LK11]. However, once we have reached a leaf node we start intersecting the triangles referenced by the indices in the triangle index list. Due to the coarser representation of the scene using voxels we can stop traversal early, display the voxel or traverse the tree deeper to compute the intersection against the polygonal representation. To individually decide when to stop we use a metric based on the intersected ellipse values computed beforehand.

#### 5.4 Post-processing the Images

As hard borders between the different levels-of-detail can be greatly disturbing to the user, we employ a gaussian blur with a fixed  $\sigma$ . First, this is applied to the rendered image. After doing this, the blurred image and the original image are added together in the following way:

$$I' = w \cdot I_b + (1 - w) \cdot I_s, \quad w = \min\left(1, \frac{d}{f}\right),$$

where  $I_b$  is the blurred image,  $I_s$  is the sharp image,  $d$  is the distance from the ellipse border on the image plane and  $f$  is a user-specified falloff-constant, defining the size of the border area around the ellipse in which the interpolation between the two images occurs.

## 6 Results

Fig. 4a shows a rendering of the hybrid renderer and Fig. 4b the used DGI. Fig. 4a shows that only a small area is rendered sharply. In the visual peripheral area the blurred



coarse voxel approximation is used. Fig. 1 shows the system with a tracked user in front of the display wall. In the next section we show the results of some preliminary user studies measuring the FSV. We use these first user studies to determine an average FSV. In the following section we show the performance impact of using the averaged FSV in a typical scenario with prerecorded tracking data for our hybrid renderer.

### 6.1 Preliminary user study

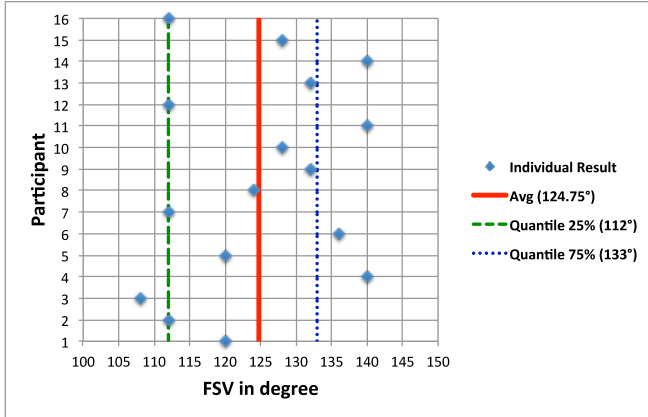


Figure 5: Results of the first user study where the FSV was narrowed

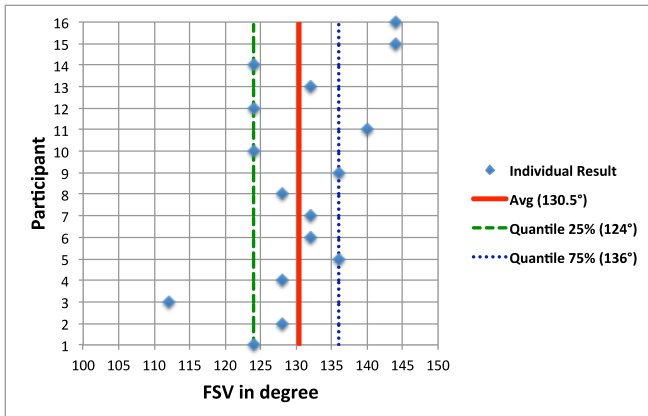


Figure 6: Results of the second user study where the FSV was enlarged

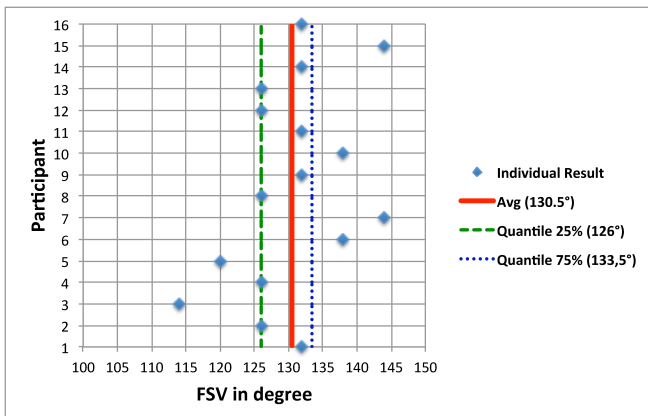


Figure 7: Results of the third user study where testing FSV for randomly chosen angles

We performed three different user studies to measure the user’s field of sharp vision (FSV) in a static setup where the users were not allowed to move and look around. For each test we used a single angle for the FSV both in horizontal and vertical direction. We tested 16 participants, 9 male and 7 female. All were between 18 and 47 years old and had no known strong visual defects (Hyperopia, Myopia, Astigmatism, ...) or wore corrective glasses or contacts.

We haven’t performed any tracked dynamic test yet because humans can hardly focus on a single point when they move around (saccadic eye movement) and we had no eye tracker available. In addition, coping with eye movement would need a higher update frequency from the renderer.

The goal for user studies was to find the FSV where a user can barely distinguish if the image in his visual peripheral area is blurred or sharp. Therefore, we placed the participants in front of the HORNET display wall with a distance of 60cm to the center display row. Afterwards, we displayed a fixation cross in their central visual field. We told the participant to focus on the center of the cross at all times. Using this setup we started a first study where we showed a completely sharp rendering for two seconds, then a grey image for two seconds and finally a rendered image with blurred borders using a modified FSV for another two seconds.

In the first test we started with a wide FSV

and narrowed it down. Each time we asked the participants if they could see any difference in the sharpness of the first or the second image. If the participants told that they noticed the difference we repeated the test with this FSV two more times. In addition, we performed one run showing the completely sharp rendered image twice. We displayed the grey image in between to make sure that the visual perception cannot notice a difference due to changes in the visual peripheral area. Fig. 5 shows the results of the user study presenting the FSV of the participants.

For the second user study we performed the same test the other way around. We started with a narrow FSV and widened it each time. During the first runs the participants could clearly tell the difference between the sharp and the partially blurred image. Each time we asked if they noticed a difference in sharpness between the first and the second image. Fig. 6 shows the results of the second study

In the last study we presented a virtual scene with randomly shuffled FSV (from  $0^\circ$  to  $180^\circ$  degree angle in  $6^\circ$  steps). This time we didn't display a grey image in between. This comes closer to a real dynamic scenario since now the visual difference between to images can be perceived more directly. For each run we asked the users if they consider their peripheral area to be sharp or blurred. Our results show that there is a clear line when a user considers an image sharp or blurred. However, we had a couple of cases where an image with a  $120^\circ$  angle was considered sharp where one with a FSV of  $126^\circ$  was considered blurred. However, these misjudgments always happened in a very narrow region of FSVs. Fig. 7 shows the result of the highest FSV the users considered to be blurred.

After the user studies the results show that on average an FSV of  $130.5^\circ$  for our setup is sufficient to create the sensation of a sharp image. It is important to note that we can likely chose narrower FSV if we spent more effort on blurring the images in the center. In addition, most users don't even seem to bother if they have a narrower FSV right away. We are planning on investigating this in follow up user studies letting the users solve specific task. Furthermore, we want to integrate eye tracking because when we let the participants look around, not by moving their head but only their eyes, using their optimal FSV, everyone was able to see the blurriness they couldn't see focussed on the fixation cross.

## 6.2 Performance impact

We measured the performance impact on our renderer using a Nvidia GTX 680 on an Intel Xeon E5520 system with 16GB RAM. To display the rendered images on HORNET we use SAGE (Scalable Adaptive Graphics Environment) developed at the University of Chicago [DN02]. We grab the DVI output of the render PC using a DVI grabber installed on another machine and display it on SAGE. Therefore, we can only support resolutions up to 4k in this scenario. Since we use a single machine for rendering, we can create fully detailed images with about 8-10 FPS. Right now our absolute performance is not interactive. We compared the run times of the renderer using the averaged FSV of  $130.5^\circ$ , determined in our user studies, to the run times of the renderer when displaying a completely sharp image. For a typical scenario, where we let the users freely walk or interact with the data set, we get an

average speed up of up to 32% for the Sponza scene with 0.28 Mio. triangles voxelized with a resolution of  $512^3$ , down to 25% for the Urban Sprawl Scene with 750k triangles voxelized in a resolution of  $2048^3$ . We render all images in 4k since Full-HD shows similar results in the performance gain by rendering a narrow FSV.

## 7 Summary and Outlook

We introduced a system to enhance the rendering performance for large tiled display walls by using a hybrid approach presenting coarse sparse voxel data for regions that are in the visual peripheral area and high resolution polygonal data in the FSV. Even though the FSV of  $130.5^\circ$  seems wide, we found that it is sufficient for typical scenarios to create a drastic speed up in the application. However,  $130.5^\circ$  means that standing more than two meters away from the wall you have to render the image with the full resolution, i.e. the FSV covers the entire wall. For that distance we can already start reducing the resolution to render on that display because rendering Full-HD on each display exceeds the spatial resolution of the eye. In practice, we observed that people working with display walls to discuss data sets stand quite close to them. We benefit most in these situations. In addition, we can speed up the renderer leading to a higher frequency for dynamic scenes. Furthermore, we are confident that we can further reduce the FSV by applying more elaborate pre-processing schemes like an edge aware blur to hide transitions between the voxels and polygonal representation. In addition, the renderer itself must be altered to support parallel rendering on multiple machines. To benefit from the FSV we are planning on developing different load balancing strategies.

We would also like to investigate how the required field of view changes in dynamic, tracked scenarios. Even though it might be visible in some scenarios it usually should not bother the user to fulfill a specific task. As future work we want to integrate focus and context based strategies in other renderers as well, like our path tracer. There we could use a focus and context strategy to gather more samples for parts that are within the FSV. In addition, we want to investigate how multiple users can work with such a system and if we can still benefit from their combined FSV.

## References

- [CG12] C. Crassin and S. Green. *Octree-Based Sparse Voxelization Using The GPU Hardware Rasterizer*. OpenGL Insights. NVIDIA Research, July 2012.
- [Dee98] M. F. Deering. The limits of human vision. *2nd International Immersive Projection Technology Workshop*, 1998.
- [DN02] M. F. Deering and D. Naegle. The sage graphics architecture. *ACM Trans. Graph.*, 21(3):683–692, July 2002.
- [Ebe13] David Eberly. Distance from a point to an ellipse, an ellipsoid, or a hyperellipsoid. Geometric Tools, LLC, <http://www.geometrictools.com/>, June 28 2013.

- [ELJ01] B. Hamann E. LaMar and K. I. Joy. A magnification lens for interactive volume visualization. In *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on*, pages 223–232, 2001.
- [Gol10] E. Br. Goldstein. *Sensation and Perception*. Wadsworth-Thomson Learning, Pacific Grove, 9th edition, 2010.
- [KSW06] J. Krüger, J. Schneider, and R. Westermann. Clearview: An interactive context preserving hotspot visualization technique. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):941–948, Sept 2006.
- [LK11] S. Laine and T. Karras. Efficient sparse voxel octrees. *IEEE Transactions on Visualization and Computer Graphics*, 17:1048–1059, 2011.
- [LR94] J. Lamping and R. Rao. Laying out and visualizing large trees using a hyperbolic space. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*, UIST '94, pages 13–14, New York, NY, USA, 1994. ACM.
- [LW90] M. Levoy and R. Whitaker. Gaze-directed volume rendering. In *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, I3D '90, pages 217–223, New York, NY, USA, 1990. ACM.
- [MCZ13] K.-L. Ma, C. Correa, and L. Zheng. Visibility guided multimodal volume visualization. *2013 IEEE International Conference on Bioinformatics and Biomedicine*, 0:297–304, 2013.
- [Mor66] G.M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd., 1966.
- [Pan11] J. Pantaleoni. Voxelpipe: A programmable pipeline for 3d voxelization. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, HPG '11, pages 99–106, New York, NY, USA, 2011. ACM.
- [PK13] C. Papadopoulos and A. E. Kaufman. Acuity-driven gigapixel visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2886–2895, December 2013.
- [RLH14] N. Radeva, L. Levy, and J. Hahn. Generalized temporal focus context framework for improved medical data exploration. *Journal of Digital Imaging*, 27(2):207–219, 2014.
- [RM93] G. G. Robertson and J. D. Mackinlay. The document lens. In *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, UIST '93, pages 101–108, New York, NY, USA, 1993. ACM.
- [SAKH06] O. G. Staadt, B. A. Ahlborn, O. Kreylos, and B. Hamann. A foveal inset for large display environments. In *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications*, VRCIA '06, pages 281–288, New York, NY, USA, 2006. ACM.
- [SS10] M. Schwarz and H.-P. Seidel. Fast parallel surface and solid voxelization on gpus. *ACM Trans. Graph.*, 29(6):179:1–179:10, December 2010.

# Black Offset Correction on Multiprojector-Display-Systems

Timon Zietlow, Marcel Heinz, Guido Brunnett

Technische Universität Chemnitz  
Computer Graphics and Visualization  
Straße der Nationen 62  
09107 Chemnitz  
Germany  
E-Mail: ziej@hrz.tu-chemnitz.de

**Abstract:** In order to display a homogeneous image using multiple projectors, differences in the projected intensities must be compensated. In this paper, we apply existing techniques for improving the contrast of multi-displays to the black offset correction. Furthermore we use a simple scheme to involve the displayed context in the correction process. This is done to dynamically improve the contrast in brighter images. In addition we present a metric for the fineness of different methods and their influence on the visual quality.

**Keywords:** black offset correction, multiprojector-display, tiled displays, photometric calibration, context based, blending

## 1 Introduction

Multiprojector-Display-Systems use multiple projectors on one screen to enhance resolution and brightness of the display. Since these projectors cannot be perfectly aligned, the state of the art approach is to calibrate the display in a geometric and photometric way ([RGM<sup>+</sup>03], [MI06], [Hei13]). The geometric calibration used for the techniques covered below was obtained using techniques described by Heinz in [Hei13].

The photometric calibration deals with the determination of projector internal attributes and the compensation of differences in the projected intensities. To achieve specific changes in the projected intensities the reaction of the projector to a given input must be known. This so called projector transfer function [Hei13] is then used to apply the correction needed to create a homogeneous image.

For technical reasons, most projectors are not able to reproduce a perfect black. The residual light projected by a projector on the input of zero is called the black offset. Especially in the overlapping areas of different segments the accumulating black offsets become visible, so the displayed image must be modified to adjust differences in the black offsets throughout the whole display.

The goal is to create a display that appears homogeneous while minimizing the negative effects on the display's quality, such as loss in contrast or shifted colors due to imperfections

in the measured projector transfer function. Previous work on this topic either ignored the correction of black offsets or simplified the correction. The impact of the calibration on the apprehended contrast is as strong as the impact of the other parts of photometric corrections. With growing importance and application of multi-display solutions in a multimedia context, the impact of the often neglected black offset corrections becomes obvious.

## 2 Previous Work

In [RGM<sup>+</sup>03] Raij et al. presented the basic techniques for photometric correction in multi-display systems. Their approach achieves a homogeneous projection result by raising the black offset for each pixel to the maximum value measured throughout the display and marks the baseline for improvements. The effect of loss in contrast was, however, not covered. Majumder proposed in [MI06] a technique using the limitations of human contrast sensitivity to improve the dynamic range. The approach was only applied to the general blending and not to the black offset correction. In order to apply photometric corrections the projector's intensity transfer function and its inverse is needed. Heinz presented in [Hei13] a new technique to measure the projector's internal behavior regarding the transformation of input values to output values. His approach, and software of his work, was used to obtain those measurements for the techniques presented in this work.

## 3 Basic Technique

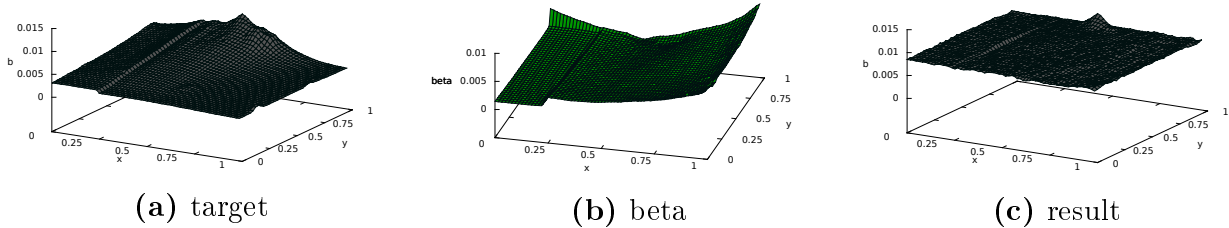
In this section we describe our implementation of the approach proposed by Raij et al. Also the general approach pursued regarding photometric calibration is laid out. To achieve a homogeneous projected image we need to convert the input for each pixel  $i_p(x, y)$  of the projector  $p$  while taking into account the arrangement and the behavior of the projectors:

$$i'_p(x, y) = \alpha_p(x, y) \cdot i_p(x, y) + \beta_p(x, y) \quad (1)$$

where  $\alpha_p(x, y)$  represents the inhibition factor used to blend the segments of each projector and to balance the brightness over the whole display.  $\beta_p(x, y)$  is the offset used to compensate the different black offsets of each projector. In this work we concentrate on  $\beta_p(x, y)$ . Since there will be an  $\alpha_p(x, y)$  and  $\beta_p(x, y)$  for each pixel and projector we refer to alpha- and beta-masks. Even though there are many sophisticated approaches ([RGM<sup>+</sup>03], [Hei13], [MI06]) addressing the creation of alpha-masks, we create a simple alpha-mask based on the approach of Raskar [RBY<sup>+</sup>99]. This way we are able to compare the different approaches to create beta-masks.

Considering that the projectors are very likely to behave in a nonlinear way, the correction is calculated in the linearized space using the projectors transfer function  $f_p$ .

$$i'_p(x, y) = f_p^{-1}(\alpha_p(x, y) \cdot f_p(i_p(x, y)) + \beta_p(x, y)) \quad (2)$$



**Figure 1:** (a) The measured intensities for an uncalibrated segment in a two-segment-setup (b) The computed beta-mask for the same projector. (c) The measured intensities in this segment with applied calibration. The hotspot is ignored due to the selection of the  $b_{max}$  inside the blending zone.

The projector transfer function and its inverse were created after [Hei13] using a DSLR camera and HDR-photography [DM97] to measure the resulting intensities to a given input value. The projector transfer function is then represented by a lookup table mapping each input value to the corresponding output value.

### 3.1 Measuring projected intensities

To capture the projected intensities we used a HDR approach based on the work of Debevec [DM97] and also used by Heinz [Hei13]. With this technique we are able to measure all projected pixels for one input in 30 exposures or less. For the generation of the beta-masks we need the black and white values of the whole display. To measure  $w_p(x, y)$ , the white images, we project a white picture with all projectors and take one HDR photo. With help of the geometric calibration, we are then able to create white images for each projector. With the black images we proceed analogously.

### 3.2 Physically accurate correction

In order to display a homogeneous image we choose  $\alpha_p(x, y)$  and  $\beta_p(x, y)$  depending on  $w_{min} = \min_{\forall p, x, y} w_p(x, y)$  and  $b_{max} = \max_{\forall p, x, y} b_p(x, y)$ , the darkest value for white and the brightest value for black in the uncalibrated display.

Based on [RGM<sup>+</sup>03] the alpha- and beta-masks can be computed with

$$\alpha_p(x, y) = \frac{w_{min} - b_{max}}{w_p(x, y) - b_p(x, y)} \quad (3)$$

$$\beta_p(x, y) = \frac{b_{max} - b_p(x, y)}{w_p(x, y) - b_p(x, y)} \quad (4)$$

With this beta-mask and a correct representation of the transfer function we can create a corrected image (fig. 6b). This calibration has a strong impact on the remaining dynamic range (fig. 5). The projected results show a darker white and a brighter black. To enhance the quality of the calibrated image, we propose two independent approaches.

## 4 Correction using human contrast sensitivity

In [MI06] Majumder and Irani proposed an approach using the human eyes' limited capability to identify differences in contrast to reduce the impact of the calibration. They only apply this technique to the generation of the alpha-masks, not the beta-masks. In the previous approach we used the same target value for each pixel, the  $b_{max}$ , which was achieved by applying the beta-masks. To achieve a calibration aiming for a correction so the human eye is unable to spot any differences we need to compute the target values following three constraints:

- The target value  $b_t(x, y)$  must be greater than  $b_p(x, y)$  since the projector is not able to create a darker value.
- The gradient in the projected intensities must be undetectable by the human eye:

$$\frac{\delta b_t}{\delta x} \leq \lambda b_t(x, y) \quad (5)$$

$$\frac{\delta b_t}{\delta y} \leq \lambda b_t(x, y) \quad (6)$$

We need to choose  $\lambda$  as the minimal change in brightness the human eye can detect. Majumder proposes to choose  $\lambda$  as follows.

$$\lambda = \frac{900 \cdot \tau}{d \cdot \pi \cdot r} \quad (7)$$

with  $\tau$  being the brightness threshold that humans can tolerate per degree of visual angle [MI06],  $d$  is the maximal distance of the user from the screen and  $r$  is the resolution in pixels per squared distance unit. According to Majumder  $\tau = 0.01$  is a well chosen value due to the contrast sensitivity function.

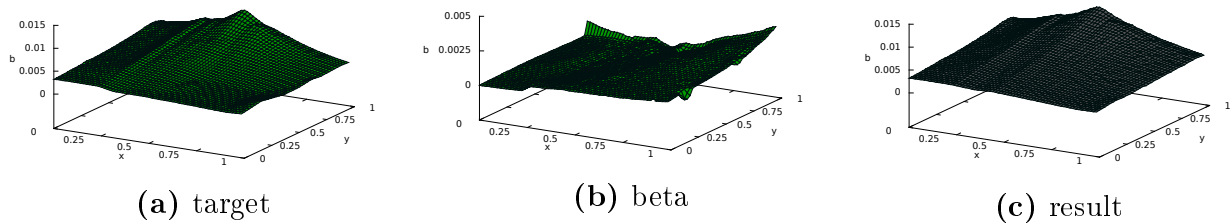
- We need to minimize the target values in order to maximize the contrast:

$$\sum_x \sum_y b_t(x, y) \rightarrow MIN \quad (8)$$

We can compute a mask satisfying all three constraints with a modified version of Majumder's algorithm which generates  $b_t$  in linear time. The beta-mask is then calculated with the new, per pixel, target values:

$$\beta_p(x, y) = \frac{b_t(x, y) - b_p(x, y)}{w_p(x, y) - b_p(x, y)} \quad (9)$$





**Figure 2:** (a) The target values computed for one segment in a two-segment-display. (b) The computed beta-mask for the same projector. (c) The measured intensities in this segment with the correction applied.

## 5 Context-dependent correction

Due to the Weber-Fechner law, the perceptibility of differences in brightness depends on the perceived brightness. The difference in brightness the human eye can perceive grows exponentially with the absolute brightness perceived. We can use this relation to improve the contrast of the projected image. If, for example, a bright image is projected, small differences in smaller dark areas will not be recognized. This makes a correction of the black offset obsolete.

To take the projected content into account we need to analyze it and decide in which areas the projected image is bright enough to go with less offset correction. This analysis of the image must be done for every frame, so a fast approach is necessary.

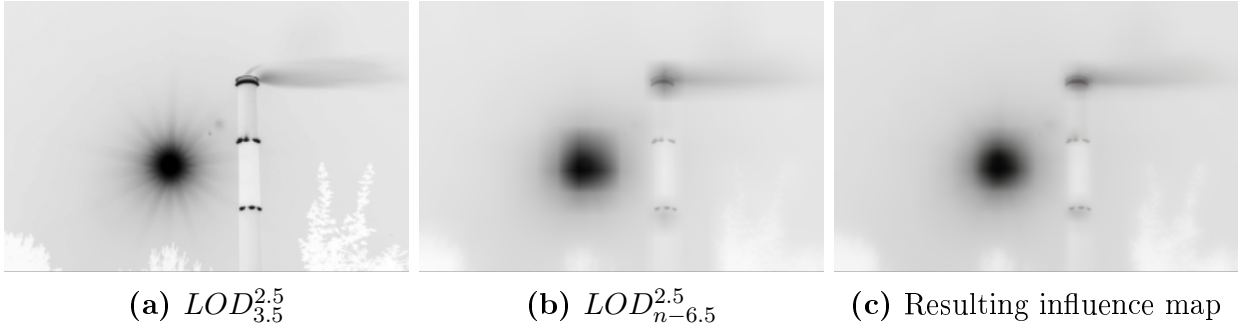
### 5.1 Correction influence map

The correction influence map  $c(x, y)$  contains values controlling the influence of the black offset correction. The correction influence map is reciprocal to the brightness of the projected image.

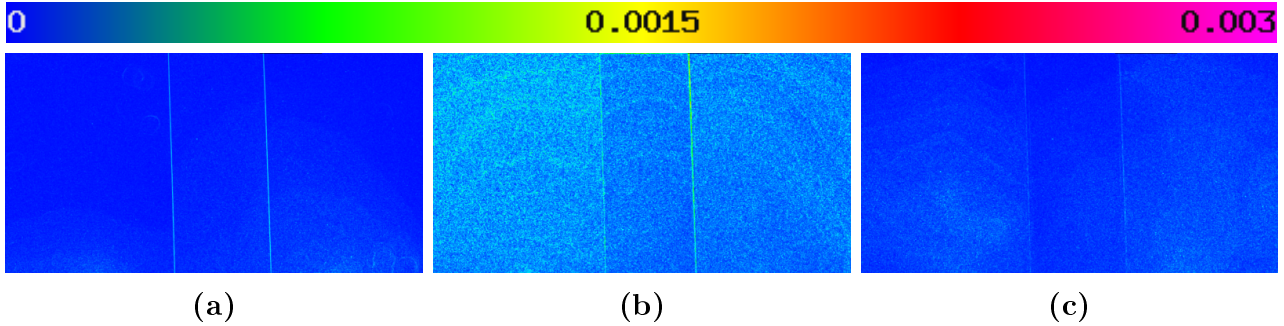
To achieve this we use the inverted greyscale version of the content we are going to display. The second requirement of the context dependent correction is the inhibition of the correction beyond the bright areas, since the saturating effect of the perception also affects areas beyond the bright areas.

The idea is to sample a lower-resolution version of the image. This implements the second requirement to apply the inhibition around the bright areas. We create  $n$  mipmap layers ( $LOD_1, \dots, LOD_n$ ) of our source image. By sampling of these low-resolution images fine details disappear and large areas get blurred (fig. 3b). In order to control the size of the blurred areas, we combine different levels of detail of the same image, resulting in a smooth fading without any perceptible distortions (fig. 3c). Due to the bilinear filtering used to sample the mipmap, the core area of a bright spot is smaller than the original bright area. By applying a gamma correction we compensate this effect. This method sharpens the transition areas between bright and dark areas.

During our experiments we discovered that the use of four different mipmap levels ( $LOD_{n-6.5}, LOD_{3.5}$ ), combined with a gamma correction of 2.5, results in a correction influence map that is not detectable.



**Figure 3:** Different stages in the creation of the influence map: (a) The linear combination of the third and the fourth mipmap levels. (b) The combination of the sixth and the seventh last mipmap levels. (c) The resulting influence map combining both (a) and (b).



**Figure 4:** The measured gradients on a display with two segments projecting black: Without any correction (a), with the correction after [RGM<sup>+</sup>03] (b) and our method of gradient limitation (c). Note, that the strong lines, marking the border of the overlapping area, are fainter in (c) than in (a). Since these borders are well detectable by the human eye this is an improvement. The noise in (b) and (c) is caused by the dithering of the used DLP projectors.

## 6 Results

### 6.1 Measuring the projected image's quality

In order to compare the results of different approaches we decided to use two different characteristics: The gradient  $g(x, y)$  of the image to measure how smooth the resulting projection is and the dynamic range  $d(x, y)$  to determine the negative influence of the correction on the projected image.

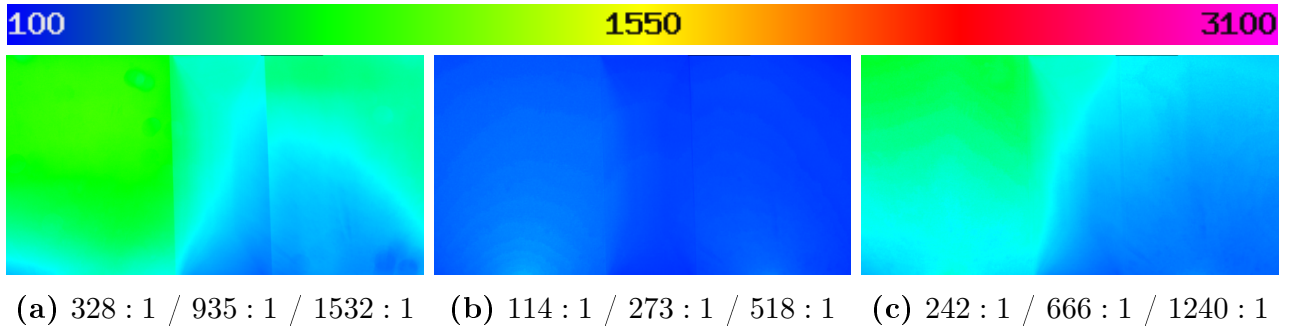
#### 6.1.1 Quality of the correction

In the discrete case we can define the gradient as

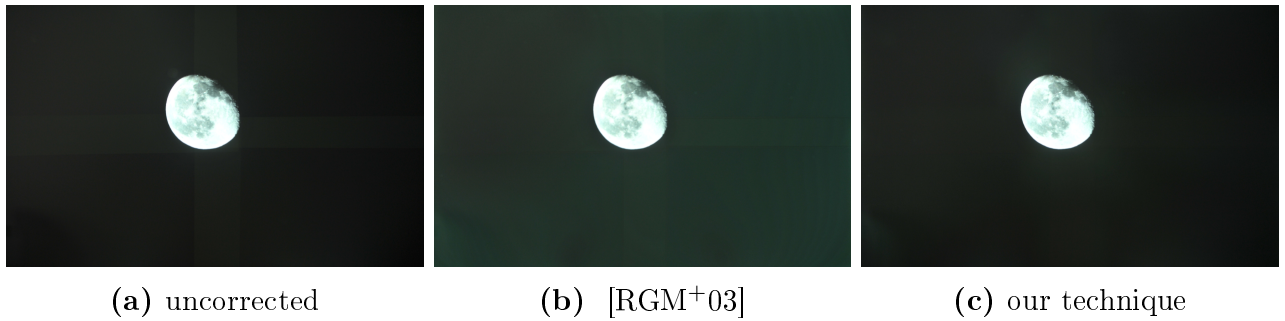
$$D((x, y), (x', y')) = \sqrt{(x - x')^2 + (y - y')^2} \quad (10)$$

$$g(x, y) = \max \left( \frac{|l(x, y) - l(x', y')|}{D((x, y), (x', y'))} \right) \quad \forall x' \in \{x - 1, x + 1\}, y' \in \{y - 1, y + 1\} \quad (11)$$

with  $l(x, y)$  representing the brightness measured at the pixel  $(x, y)$ .



**Figure 5:** The measured dynamic range on a display with two segments. The captions display the minimum/average/maximum values. (a) without any correction, (b) with the correction after [RGM<sup>+</sup>03] and (c) our method of gradient limitation.



**Figure 6:** The photos show three different ways to treat the black offsets. The untreated (a), the physically accurate correction (b) and our approach (c). All three photos were taken with the same camera ( $F/3.5, 1/6s, ISO - 500$ ).

### 6.1.2 Quality of the calibrated result

The dynamic range indicates the range of intensities the display can show. It can also be calculated per pixel:  $d(x, y) = \frac{w(x, y)}{b(x, y)}$

## 6.2 Discussion

The results of the contrast limiting blending are satisfying (fig. 6c). The overlap zones between different segments are not perceptible from the chosen distance. The average dynamic range is improved by 42% (fig. 5) in comparison to the method used by [RGM<sup>+</sup>03].

The dependence of the procedure on the viewer's distance to the screen is the sole restriction. In many situations (especially in VR caves) the viewer's distance is known or limited by the size of the room. So this approach is an adequate solution for those situations.

The use of the correction influence map achieved a better internal contrast ratio but the difference was barely recognizable. This is due to the Weber-Fechner law mentioned above. The improvement in brightness and contrast is achieved by using the capability of the human eye to detect differences in brightness and contrast. This also limits the effect of this approach. The greatest benefit of the procedure is a visible reduction of color disturbance due to the calibration (fig. 7).



**Figure 7:** The photo shows a campfire. In the top right without, in the bottom left with enabled context influence mask. The underlying black offset correction is the physical accurate one. The photo is intentionally overexposed to enhance the visibility of the effect. This also enhances the visibility of the overlap zones. Camera parameters: ( $F/3.5, 1/2s, ISO - 500$ ).

## Literature

- [DM97] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
- [Hei13] Marcel Heinz. *Kalibrierverfahren und optimierte Bildverarbeitung für Multiprojektorsysteme*. dissertation, TU Chemnitz, 2013.
- [MI06] Aditi Majumder and Sandy Irani. Contrast enhancement of images using human contrast sensitivity. In *Proceedings of the 3rd Symposium on Applied Perception in Graphics and Visualization*, APGV '06, pages 69–76, New York, NY, USA, 2006. ACM.
- [RBY<sup>+</sup>99] Ramesh Raskar, Michael S. Brown, Ruigang Yang, Wei-Chao Chen, Greg Welch, Herman Towles, Brent Seales, and Henry Fuchs. Multi-projector displays using camera-based registration. In *Proceedings of the Conference on Visualization '99: Celebrating Ten Years*, VIS '99, pages 161–168, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.
- [RGM<sup>+</sup>03] Andrew Rajj, Gennette Gill, Aditi Majumder, Herman Towles, and Henry Fuchs. Pixelflex2: A comprehensive, automatic, casually-aligned multi-projector display. In *In Proc. IEEE International Workshop on Projector-Camera Systems*, 2003.