Proceedings of SIGRAD 2010

Content aggregation and visualization

November 25–26, 2010

Västerås, Sweden

Edited by

Kai-Mikael Jää-Aro

Thomas Larsson

Linköping Electronic Conference Proceedings, No. 52
Linköping University Electronic Press
Linköping, Sweden, 2010
ISBN 978-91-7393-281-3
ISSN 1650-3686 (print)
http://www.ep.liu.se/ecp_home/index.en.aspx?issue=052
ISSN 1650-3740 (online)
Print: Mälardalen University, 2010

Cover illustrations selected from the papers.

Preface

Welcome to SIGRAD 2010, the annual conference of SIGRAD, the Swedish chapter of Eurographics. We yet again present an interesting crop of papers from all areas of computer graphics.

We wish to thank

the authors, without whom there would not be a conference,
the international programme commmittee, who provided timely and constructive reviews,
the keynote speakers, Sheila Hemami and George Ioannidis, and the invited speakers, Björn Thuresson and Thomas Porathe, for thought-provoking presentations,
the participants, whom we hope will have an enjoyable conference,
Mälardalen University and Mälardalen Real-Time Research Centre for their generous sponsorship.

The editors

Table of Contents

Keynotes

From Single Media to Multimedia — Perception, Coding, and Quality

Sheila Hemami

Cornell University

Abstract

Humans are the ultimate consumers of multimedia information, and effective system design requires a performance metric. While such metrics have been extensively studied for single-media perception for one or more decades, those for multimedia perception and use are still in their relative infancy. In this talk, I will focus on the development of single-media quality metrics for audio and visual information, and contrast it with the development of appropriate metrics for multimedia information. I will describe how humans perceive single-media information, how an understanding of perception has been incorporated into single-media coding and then quality measurement, and I will discuss the current state of understanding of multimedia perception as it has been applied to coding and quality measurement problems

Biography: Sheila S. Hemami (F) received the B.S.E.E. degree from the University of Michigan in 1990, and the M.S.E.E. and Ph.D. degrees from Stanford University in 1992 and 1994, respectively. Her Ph.D. thesis was entitled "Reconstruction of Compressed Images and Video for Lossy Packet Networks" and she was one of the first researchers to work on what we now call "error concealment." She was with Hewlett-Packard Laboratories in Palo Alto, California in 1994 and worked on video-on-demand. She joined the School of Electrical Engineering at Cornell University in 1995, where she holds the title of Professor and directs the Visual Communications Laboratory.

Dr. Hemami's research interests broadly concern communication of visual information, both from a signal processing perspective (signal representation, source coding, and related issues) and from a psychophysical perspective.

Dr. Hemami is an IEEE Fellow and has held various visiting positions, most recently at the University of Nantes, France and at Ecole Polytechnique Federale de Lausanne, Switzerland. She has received numerous college and national teaching awards, including Eta Kappa Nu's C. Holmes MacDonald Award. She is currently Editor-in-Chief, IEEE Transactions on Multimedia (2008-10); Member-at-Large of the IEEE Signal Processing Society Board of Governors (2009-11), and an SPS Distinguished Lecturer (2010-11). She has Chaired the IEEE Image and Multidimensional Signal Processing Technical Committee (2006-07); and served as Associate Editor, IEEE Transactions on Signal Processing (2000-06).

Flexible search and presentation of multimedia

George Ioannidis

IN2

Abstract

Although the amount of image and video files in our hard disks and the web is steadily increasing, we rely on classical retrieval and presentation techniques which can not cope with large collections. Retrieval is still based on textual annotations and the presentation interfaces are mostly simple thumbnail lists. In this talk, I will present search and retrieval beyond and in combination with text, and how this can be used and incorporated in media workflows. Furthermore, based on recent work, I will show how users can aggregate their collections into larger repositories, publish them, and design themselves flexible user interfaces by recomposing UI elements and retrieval models. On that basis, I will also discuss ways of presenting multimedia collections using facets and concepts stemming from information visualization.

Biography: George Ioannidis holds a Master of Science (MSEE) and a doctoral degree (PhD) in Electrical Engineering and Computer Science from the National Technical University of Athens, the leading university in engineering sciences of Greece. He additionally holds a Master of Business Administration (MBA) where the main focus of his work was on innovation management. After several positions in academia among others leading the image and video analysis group at the Center for Computing Technologies in Bremen, he founded 2005 IN2 (http://www.in-two.com) and is since then its director. He has initiated, participated and managed several European research projects in the fields of multimedia information retrieval, content management and content accessibility.

His primary research interests are on dynamically adaptive visualizations, especially from the search and retrieval perspective, which support browsing and navigating multimedia collections and individual multimedia assets. George acts regularly as a programme committee member in international conferences and workshops (e.g. ACM Multimedia Information Retrieval), Journals (e.g. Knowledge Engineering Review) and serves as a reviewer of business plan competitions, and as an evaluator and reviewer of national and international collaborative research projects.

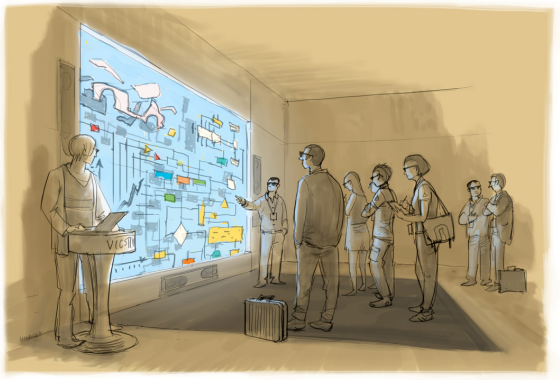
Invited Talks

The VIC Sthlm Arena for Visualization, Interaction and Collaboration

Björn Thuresson

HCI Group, CSC, KTH

Abstract: VIC-Sthlm (http://www.vic-sthlm.se) is one of three Swedish knowledge arenas focusing on visualisation. Our mission is to coordinate and develop new and existing networks and activities within visualisation in the Stockholm region. We focus on business development and knowledge exchange between actors from industry, academia and the public sector. The knowledge arena is financed by KTH and a number of Swedish governmental agencies and research funding institutions.

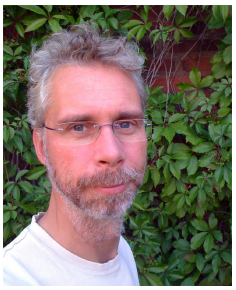


We have about 40 contributing partners from a wide range of domains, including software developers (QlikView, SightLine, Find-Out, Streamshed, Microsoft), hardware (SenseGraphics, Setred, Texas Intruments, Microsoft), data owners (Stockholms Stad, Sustainable Innovation), networks and public spaces (Kista Mobile Multimedia Network, Stiftelsen Elektrum, Dataspelsbranchen, Tekniska Muséet, Digital Arts Center, CineGrid) content developers (Dice, Avalanche, I3dv, Imagination Studios).

The talk will focus on concrete examples of what the collaboration has resulted in, like visualisations of the centre of the Milky Way, a game for Microsoft Surface, analysis of weather TV broadcasts, and more. I will also present the new Visualisation Studio currently being built. KTH and Linköping University recently received a significant grant from the Knut and Alice Wallenberg Foundation for acquiring a state-of-the-art visualisation infrastructure. At KTH, the infrastructure will be in the shape of a visualisation studio, hosted by the CSC School.

The overall goal of the infrastructure project is to strengthen visualisation and collaboration research in Sweden by, (1) supporting a number of selected strategic application areas for which improved visualisation will have a large impact, (2) enabling research focusing on the fundamental technical aspects of high-end visualisation within a wide range of application areas, (3) providing a collaborative test-bed between leading visualisation research sites, (4) support the development of principles for effective collaborative visualisation, and (5) enable research of the human and technical aspects of remote and local collaboration situations for situations with participant groups of various sizes.

The planned equipment for the KTH visualization studio includes an extremely high-resolution (4k) projector, a cinema-quality sound system, high-resolution video communication setups that allow for eye contact, a holographic display, a selection of multi-touch units, a set of haptic workstations, motion and eye trackers, and a supercomputing cluster for advanced calculations.



Biography: Björn Thuresson is a senior researcher at the School of Computer Science and Communication (CSC) at the Royal Institute of Technology (KTH) in Stockholm. He has a BA in Cinema Studies and Journalism (1991) and a MA in Cinema and Communication (1993). Thuresson has acted as research coordinator for the HCI group at KTH and managed several projects, e.g. coordinator for all research activities in European project

INSCAPE (Interactive Storytelling for Creative People), 2004-2008. Thuresson currently devotes all his time coordinating the VIC-Sthlm Knowledge Arena on Visualisation, http://www.vic-sthlm.se. Thuresson has a background in Cinema Studies and in production of film and educational multimedia and in concept development for new media. The research focus on interactive narratives, interaction design, methods for idea generation and prototyping, and overall design processes.

“If this will be the way to drive a ship - just anyone could do it”: 3D Nautical Charts. About creating acceptance and building standards for a VR within the maritime domain

Thomas Porathe

Chalmers Technical University
Sweden

Abstract

This paper is presenting a short overview of a human factors project aiming to improve decision-making in navigation by removing the need to conduct mental rotations. This is done by presenting the map in an egocentric view, here called a 3D map. Laboratory experiments have shown clear advantages for the 3D map compared with traditional exocentric map displays. Finally a short description of ongoing work is presented.

Categories and Subject Descriptors (according to ACM CCS): H.1.2 [Information Systems]: User/Machine Systems—Human factors H.5.2 [Information Systems]: User Interfaces—Standardization

1. Introduction

In 1999 the Norwegian high-speed ferry *Sleipner* crashed against a rock at full speed causing the death of 16 people. In spite of the fact that the vessel was equipped with all modern technical facilities and that the bridge officers knew the route very well they lost their orientation for some 20 fatal seconds. However, the navigation system onboard never lost orientation, all the instruments showed the right digits, but the humans were not able to interpret the information in the short time frame available. The interaction between the human and the system had failed.

During a period from 2001 to 2006 I conducted a Ph.D. research project where I developed and tested a VR based 3D navigation system based on an egocentric view display. The assumption being that this system should be better adapted to the human and allow for faster and more error resistant decision making. I have called this map system *3D nautical chart*. This paper gives a very brief overview of the project so far.

2. A 3D navigation system

Navigation is the aggregated task of wayfinding and motion [DP01]. Not long ago this was a very difficult task. Today, thanks to satellite based positioning systems this is all done automatically. Before the voyage the navigation officer enters the waypoints of the decided route into the navigation system. The track is then displayed on the electronic chart as a line together with a symbol showing the ships present position. The autopilot will then take care of course keeping. All the navigator has to do is to monitor the system. But sometime situations occur when the human is forced to take manual control. It may be incidents involving other ships, or bad weather which the autopilot cannot cope with (like in the *Sleiper* accident). It is then important that the electronic map system has kept the human in the loop to facilitate a switch to manual control.

Whenever an area is larger than one can overlook from a single position some kind of mental support is needed and maps have been around for a long time (see Figure 1).

The view has been that of a bird looking down from the sky. To compare the real world and the map the map reader has to imagine that he is standing on the map surface looking in the right direction and imagine what he would see in front of him. This imagined picture must then be compared with what he really sees in front of him. This is the complicated act of mental rotations. The problem with mental rotations [SM71] is well documented and effectively compli-

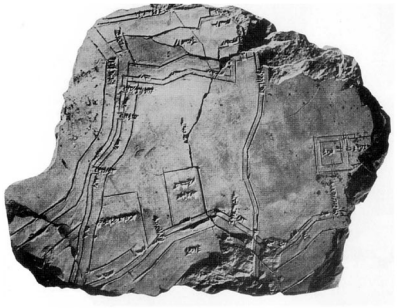


Figure 1: *Early exocentric map. Clay tablet from Mesopotamia from the 16th century B.C. British Museum, London.*

cates map understanding. A talent called spatial ability plays a major role in this process. Furthermore, spatial ability decline with age and favors male [Hal00].

Parallel to exocentric, bird's-eye maps, so called sailing directions have since long been used by mariners. These were verbal descriptions of the coast from a deck perspective and they were soon to be illustrated with egocentric coastal views (see Figure 2).

Figure 2: *Early egocentric map. A coastal view incorporated in a sailing direction over the French west coast: Ushant is the English name for Île d'Ouessant at the western tip of Bretagne. A woodcut from Robert Normans 1590 Safegarde of Saylers.*

The coastal view could be directly compared with what the mariner saw in the real world. No mental rotations were needed. The only problem was that the coastal view only was valid form one particular position. However, today the technical means of creating dynamic coastal views exist. By creating a 3D model of the map and letting the navigation system position the camera, a dynamic egocentric view can be created (see Figure 3).

The question is then of course: does such an egocentric view really facilitate navigation as to become intuitive and direct in a way that could have hindered the Sleipner accident and several similar accidents involving sudden loss of situation awareness?

3. Maze experiment

To answer that question a laboratory experiment were conducted. On a 6 m by 6 m area in a studio a 10 by 10 invisible grid was created. In this grid four mazes were designed, each

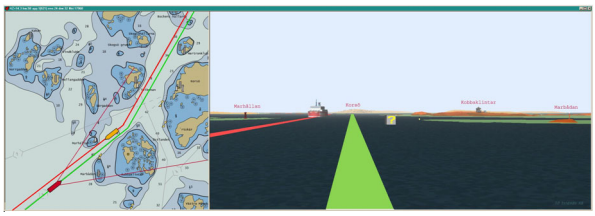


Figure 3: *Two nautical chart displays. To the left the traditional exocentric north-up bird's-eye view and to the right the new egocentric out-of-the-window view. (Screen dump from prototype application).*

with an equally long track with an equal number of turns. Each track lead from a start position to a goal and for each design one traditional exocentric map and one 3-D map was prepared (see Figure 4).

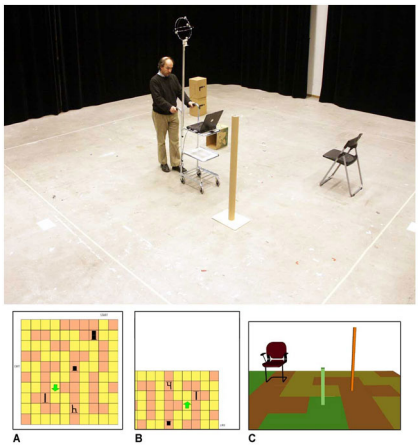


Figure 4: *Top, the studio area acting as the maze with its landmarks (boxes, a tube and a chair). A cart was wheeled manually through the maze. Bottom left (A) the 2-D map in the north-up mode as it was shown on the screen of the lap top computer on the cart for the very position shown in the top picture. The middle (B) map is the 2-D map in course-up mode and right (C) is the 3-D map. The maps depict the cart's position in the large photo.*

In a first block 45 amateur navigators (students and university employees) and in a second block 30 professional bridge officers drove once through each of the four different tracks. Each time using one of four different map types. The order of the map types was randomized. The four types can be seen in Figure 5.

An infrared tracking system mimed the GPS system and sent the position of the cart to a laptop computer fitted on the cart. The subjects were told to drive through the maze as fast as possible with as few "groundings" (entering into

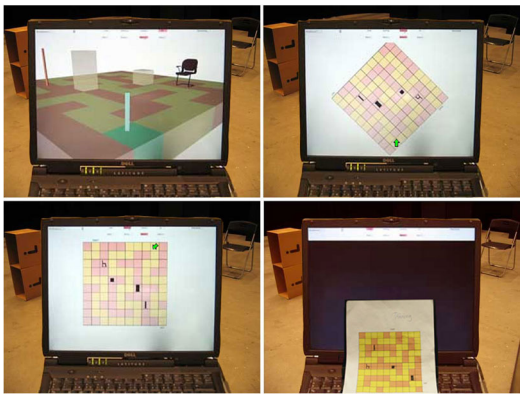


Figure 5: *The four different map types: Top left the egocentric view 3-D map; top right the exocentric head-up map; bottom left the exocentric north-up map and bottom right the traditional exocentric paper map.*

the red squares) as possible. The results were statistically significant and showed that the egocentric 3D view provided faster decision making and fewer errors than the traditional map types (see Figures 6 and 7).

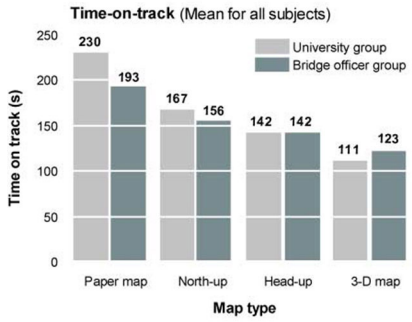


Figure 6: *The time it took for subjects to pass through the maze.*

For more details of these experiments see [Por06] and [PP07]. After having done their four sessions with different maps each subject were asked to rank the user- friendliness of the four different map types on a scale between 1 and 4. The egocentric 3D map was clearly ranked as the most user-friendly (see Figure 8).

In the first block the experiment was conducted on available students and staff at the university. They were all amateur navigators. The study clearly showed that maps supporting cognitive off-loading were more efficient: when the subjects needed to do fewer mental rotations they made faster decisions about the way through the maze and fewer errors. Next we wanted to test if these results would hold for professional navigators schooled in performing precisely these

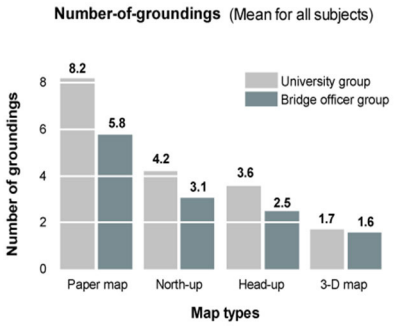


Figure 7: *The number of groundings (cart entering a forbidden red square).*

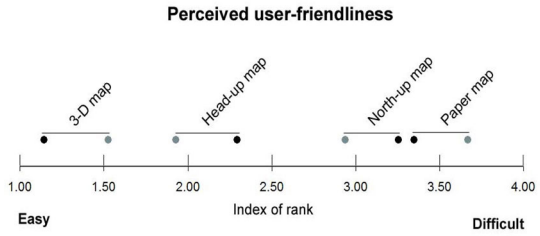


Figure 8: *Results from the ranking of user-friendliness.*

kinds of mental rotations. Much to our surprise they did. They had the same trend of higher efficiency with fewer mental rotations. As expected they did better than amateurs when using exocentric maps in north-up, but surprisingly they did equally well as amateurs using head-up maps (an orientation mode seldom used by mariners), and worse than amateurs on the 3D map. We did not ask the amateur group for computer gaming habits, but we did ask the professional group and found no significant correlation between gaming habits and results in the maze.

The professional group is doing over all better than the amateurs when it comes to errors. We interpret this as a result of the professionals' knowledge of the true consequences of a grounding. In a speed-accuracy trade-off they prioritize accuracy which may not be the case with the amateurs, which also has to be taken into account when looking at the speed results.

Never the less there is a stable trend that the differences in navigation performance between the schooled and the unschooled group is equalized when using displays modes that remove the need of making mental rotations.

By removing the need of performing mental rotations the 3D egocentric map display lessens the cognitive workload of the user. A known problem in automation is that operator tasks changes from manual control to monitoring [End96, WH00]. This in turn leads to what has become

known as out-of-the loop performance problems, meaning that if something goes wrong forcing the human to retake control, she often lacks in situation awareness and valuable time is then lost trying to regain situation awareness and control which might lead to a potentially disastrous situation. When vehicles navigate on autopilot and a monitoring crew members need to retake immediate control, for example to make an evasive maneuver for another ship, the time to realign the map with the real world through a number of mental rotations, might be just too much. A cognitively less demanding display system might save those valuable seconds.

Map applications in cars and mobile phones become more and more frequent. So called "3D modes" are today standard in most applications. Very little research is presented on the human-machine interaction of these map systems; we hope that these findings may add to the knowledge in this field.

4. Gaining acceptance in the maritime domain

The laboratory experiment, the construction of several visualization prototypes and a Ph.D. thesis concluded the first phase of this project in 2006. During this work, and later, I have presented my findings for more than a thousand professional experts from the maritime domain and gained much interest. This has encouraged me to continue working on making the 3D chart become reality. In 2008 I joined the Maritime human factors researcher group at Chalmers Technical University in Gothenburg which gave me a much better platform to continue with the most difficult part of this project: to create acceptance, and finally open for possible standards within the regulating organizations of the maritime domain, IMO (the International Maritime Organization), IHO (the International Hydrographic Organization), IALA (the International Association of Marine Aids to Navigation and Lighthouse Authorities), etc. If the rules that regulate international shipping do not permit the use of 3D nautical charts they will never be, thus much energy needs to be put into letting the findings of this project become known to the international maritime community.

In 2009 two EU financed project started: the *EfficienSea* project headed by the Danish Maritime Safety Administration and the *BLAST* project headed by the Norwegian Hydrographic Service. Both these projects involve the 3D chart. In the BLAST project a 3D chart demonstrator over Zeebrugge in Belgium will be built in the autumn of 2010 and the surveys for this is already underway (see Figure 9).

This chart demonstrator will be evaluated with officers and pilots going in and out of Zeebrugge harbor both on a real ship and in a simulator in Antwerp in the spring of 2011. The results from both these projects will be feed into the so called e-Navigation initiative of the IMO, IHO and IALA.

In October 2010 I am invited to present my research to the Hydrographic Safety and Standards Committee of the IHO. This will be a very important moment.

Figure 9: *Surveying for the 3D chart from the Belgian hydrographic surveying ship Ter Streep in the port of Zeebrugge in September 2010.*

5. Conclusion

This has been an extremely brief summary of an almost ten years long project. The prospects for the 3D charts becoming a reality looks at the moment bright, also because that we can see that in the parallel world of car navigation the same paradigm of exocentric head-up display of maps are the dominant mode used.

During a presentation a number of years ago, a navy captain said: "If this is going to be the way to con a ship - just anyone could do it." I suspect the comment was not entirely meant so, but I took it as a compliment. If a 3D chart can help officers in the same situation as the men on the bridge of *Sleipner* to regain their situation awareness in the split of a second much will be won. It remains to be seen if we will come that far.

References

[DP01] DARKEN R. P., PETERSON B.: Spatial orientation, wayfinding, and representation. In *Handbook of Virtual Environment Technology*, Stanney K., (Ed.). Erlbaum, Hillsdale, N.J., USA, 2001. 1

[End96] ENDSLEY M. R.: Automation and situation awareness. In *Automation and human performance: Theory and applications*, Parasuraman R., Mouloua M., (Eds.). Erlbaum, Mahwah, N.J., USA, 1996, pp. 163–181. 3

[Hal00] HALPEN D. F.: *Sex Differences in Cognitive Abilities*. Erlbaum, Mahwah, N.J., USA, 2000. 2

[Por06] PORATHE T.: Improved situation awareness in navigation using egocentric view 3-D nautical charts. In *IEA 2006* (2006), Elsevier Ltd. 3

[PP07] PRISON J., PORATHE T.: Navigation with 2-D and 3-D Maps: A comparative study with maritime personnel. In *NES 2007* (2007), Nordic Ergonomic Society. 3

[SM71] SHEPARD R. N., METZLER J.: Mental rotation of three-dimensional objects. *Science 171* (1971), 701–703. 1

[WH00] WICKENS C. D., HOLLANDS J. G.: *Engineering psychology and human performance*. Prentice-Hall, Upper Saddle River, USA, 2000. 3

Papers

Tangible Interfaces using Handheld Augmented Reality

P. Rojtberg[1] and A. Olwal[2]

[1]TU Darmstadt, Germany
[2]Kungliga Tekniska Högskolan, Sweden

Abstract

We present a framework for tangible user interfaces on handheld devices through the use of augmented reality, where virtual objects can be manipulated and arranged using physical objects in the real world. Visual feedback is provided in the high-resolution handheld display, where virtual objects are overlaid onto live video from the camera. The virtual objects are registered and tracked in real-time relative to the physical environment using the device's camera. Realistic lighting and rendering of high-resolution virtual models is achieved through hardware-accelerated graphics and shadow mapping. The user can interact with the virtual objects and system parameters both through an overlaid menu interface and through direct touch-screen interaction. We describe our framework, our adaptation of the ARToolKitPlus tracking library for a mobile platform, and a number of interaction techniques that we implemented for a prototype urban planning application.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.3.6 [Computer Graphics]: Methodology and techniques—Interaction techniques.

1. Introduction

An important task in authoring 3D scenes is the navigation of the 3D space. This is necessary to, for example, position objects in the space, but it is also critical to many other tasks. For example, before one can select and change the color of an object, one often has to find a perspective such that the object is visible. Positioning the virtual camera, however, requires similar navigation as for placing an arbitrary object. (See Figure 1.) Placing an object in 3D space not only consists of choosing a 3D vector that represents its position, but also choosing a 3D orientation. Summed up, one has 6 degrees of freedom (DOF) for placing the object. In the best case one would change all 6 DOFs simultaneously for fast placements. (See Figure 2.)

One challenge is that this type of input on desktop systems is typically controlled with a mouse and keyboard. The mouse provides relative 2D input, while the keyboard's modifier keys are used to switch between which 2D dimensions the mouse is controlling at a given time. This, however, means that the user can only manipulate 2 dimensions simultaneously, which in theory could take 3 times longer for getting 6 DOFs due to the implicit temporal sampling.

Figure 1: An augmented reality scene, viewed through the screen of a mobile device that is running our framework. The scene contains virtual buildings that the user can position through the tangible manipulation of multiple camera-tracked markers in the real world.

However, often only either the orientation or the position of an object has to be changed, which requires only 3DOF.

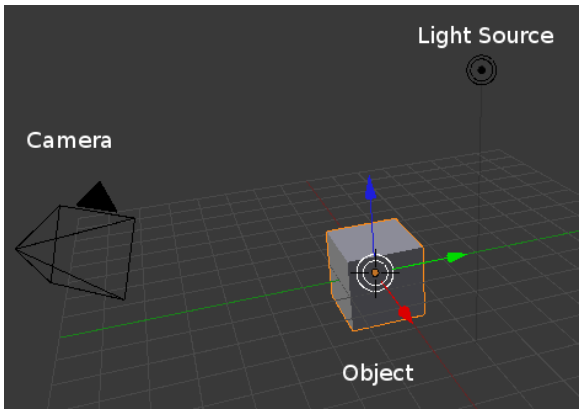


Figure 2: A typical scene for 3D authoring. The camera and the object require 6DOF input and the omnidirectional light requires at least 3DOF to be positioned in space.

In these cases one can try to map the 2D input to 3D to speed up the task. A widely used technique called ArcBall rotation maps 2D movements to 3D rotations [Sho92]. It works by projecting the user's input onto a virtual sphere around the object. The projected 2D movement results in a 3D arc on the sphere. The object is then rotated in 3D along the trajectory of the arc. While this is generally considered a reasonable mapping of 2D movements to 3D rotations, it is not very intuitive, as the user has to know how the mapping works to perform a predictable rotation. But even then, predicting the projection is not trivial. For certain scenarios it might instead be more appropriate to use 6DOF input devices. Tangibles are particularly interesting, as they support physical manipulation and naturally offer 6DOFs. One might consider using a device with embedded sensors (e.g., accelerometers, gyroscopes and magnetometers[†]) to create the tangible objects, but the use of such sensors on a larger scale is limited by issues like performance, resolution, power requirements and cost.

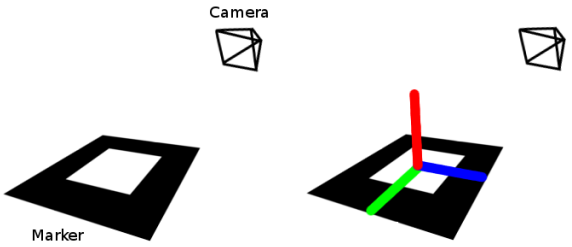


Figure 3: Camera-based marker tracking provides the rotation and translation of the camera relative to marker.

Marker-based augmented reality (AR), on the other hand, uses 6DOF tracking of the camera to register rendered computer graphics overlaid on a live video stream of the real world. This allows the use of marker-based AR as a cheap 6DOF input device, as it only requires a camera and printed markers, as shown in Figure 3. Using the position and orientation of the markers as input allows direct positioning of an object in the 3D scene. This interaction can be extended to multiple objects by tracking multiple markers simultaneously and mapping one marker to each virtual object.

Using this approach we create a tangible interface for 3D authoring which makes most of the necessary operations spatially seamless [BKP08]. Furthermore, the one-to-one mapping gives us spatial references in the real world [HPGK94].

This speeds up the operations for arranging the scene and is generally a natural choice for interacting with AR environments. There are, however, still operations where 6DOF input is counterproductive. Consider the fine arrangement of a scene where only a small rotation along one axis has to be performed. In this situation only one dimension has to be changed and the involuntarily control of free-hand 6DOF input would most likely affect all other dimensions. In these cases one can apply constraints to limit the dimensions that are affected by the input. While marker-based AR interaction has a number of advantages, as mentioned, most real-world examples do not go beyond just positioning the camera inside a non-interactive scene[‡,§].

2. Related Work

Rekimoto and Nagao used the term "augmented interaction" already in 1995, when they presented their NaviCam system [RN95]. Their goals were to move from human-computer interaction to human-real-world interaction as this is where the user's focus actually lies (Figure 4). Furthermore, they wanted to make the interface context-sensitive and narrow the gap between the computer and the real world by making the computer understand real-world events. The NaviCam concept envisioned using handheld devices for interaction and presentation. Due to the limited processing and rendering capabilities on handhelds at the time, the video from the handheld camera was fed to a workstation that did the marker recognition and added the overlays onto the video stream, which was sent back to the handheld display. This still allowed the system to pioneer the proof-of-concept of using a handheld that recognized color codes attached to objects and then displayed a context-sensitive 2D overlay with additional information. It is, however, worth noting that any

[†] http://www.xsens.com/en/general/mti

[‡] http://www.bmw.co.uk/bmwuk/augmented_reality/homepage

[§] http://ge.ecomagination.com/smartgrid/#/video

further interactions happen on the human-computer interaction side, which does not affect the real world.

Our system, in contrast, puts a stronger emphasis on the human-real-world interaction as the primary interactions happen in the real world and the human-computer interaction is mainly used for providing feedback.

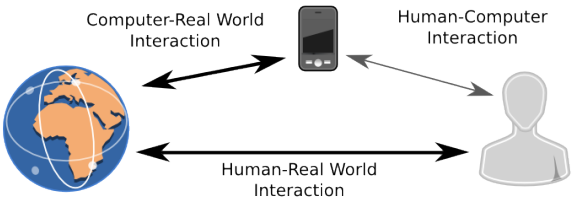


Figure 4: Augmented Reality emphasizes interactions happening in the real-world

Poupyrev et al. [PTW98] presented a virtual notepad as an intuitive method for creating annotations in Virtual Reality (VR) environments. Their application tracked a pen and a paper in the real world and transferred the input to the VR environment. As this is a straightforward and easy-to-implement method for mapping pen-interaction in the real world into a VR environment, we used it to enhance our AR scenario. The touchscreen on the handheld device serves as the notepad and the stylus as the pen.

In recent years, research also started to use mobile phones and PDAs with cameras, as handheld AR devices. As the computational capabilities of these devices improved, it became possible to create self-contained AR applications for large-scale deployment. But the computing power was still limited; although Wagner and Schmalstieg [WS03] presented the first self-contained AR client running on a PDA, the system's 3 frames per second (fps) update rate prohibited interactive applications. They managed to increase the frame rate to 5 fps after outsourcing the tracking computations to a server. The performance was limited in part because they had to use a software pipeline for 3D rendering as the devices did not support hardware-accelerated graphics at the time. The limited hardware also made it necessary to use an optimized fixed-point implementation of the algorithms. Mohring et al. [MLB04] faced the same problem, but achieved 5 fps without server assistance by using a simpler marker detection algorithm where they used special 3D paper markers. While these projects provided 6DOF input by marker tracking, the motivation behind them was to overcome the technical limitations of their platforms. AR was still primarily used for displaying overlays without further interactions with the real world.

Henrysson and Ollila [HO04] worked around the performance problem by using 3DOF tracking and 2D rendering, which was sufficient in their specific scenario of augmenting a 2D paper map. The motivation here was to use context information from carrier cells to localize the phone and display context-aware content, but the augmented interactions were still limited to panning on the augmented map.

Lee et al. [LNBK04] focused on interaction and combined AR with a tangible interface to create an immersive environment for authoring. They modeled behaviors and interactions of a scene rather than modeling geometry. The markers in their system were used to represent:

- Workspaces where objects can be placed
- Virtual paddles that act as a 6DOF cursors
- Buttons that change object properties

Generally, our application follows their approach, as we also focus on the relations between objects and their behavior. Instead of a virtual paddle, however, we use the markers as direct 6DOF input for the virtual objects and the handheld device as a 6DOF cursor and viewport.

Henrysson [Hen07] evaluated the use of mobile phones as 6DOF input devices and implemented a multi-player game as an example application. The two-player AR tennis game has a virtual court that is overlaid on a real table and the players' mobile phones are used as rackets to hit a simulated ball. The work was supplemented by a comparison of different input methods:

- Isometric input: value controlled by buttons
- Isotonic input: relative phone movement controls object
- ArcBall rotation

Henrysson also reports on users' subjective feedback regarding the performance of rotation and translation with the different methods. While ArcBall and isometric input were the fastest methods with only 1DOF, isotonic input was the best for translation and the second best for rotation with 3DOF. Unfortunately, the use of a physical marker for input was excluded as it was occupying too much of the camera's field-of-view. We, however, find it interesting to explore the use of direct manipulation with markers as it enables tangible interaction.

Recent work by Wagner et al. [WLS08] focuses on replacing the obtrusive binary marker patterns with different designs that range from "split markers" (markers that allow any content in their center) to arbitrary images that are tracked using Natural Feature Tracking (NFT) [WRM*08]. They demonstrate that these techniques can be run on handheld devices. NFT makes it possible to use images of the virtual objects as markers, which is not only less obtrusive, but could also be used for a more intuitive representation and indication of the marker that controls the virtual object.

In this work, we focus on general interaction techniques that are independent of the marker tracking used. While the library our work is based on (ARToolKitPlus) only supports binary markers, more meaningful image-based markers would not have changed the general way of interaction. We are, however, interested to explore the use of NFT for more compelling tangibles in the future.

3. Combining Tangible Interfaces and Handheld AR

Using handheld AR for creating a tangible interface has several advantages, where the explicit camera control is perhaps the most important one. While head-worn displays implicitly place the camera at the users' point-of-view, the handheld device can be placed independently. This allows conscious placing of the camera when a specific perspective is desired. The explicit placement also allows using the handheld device as a 6DOF cursor in the scene. Generally speaking, the 6DOF input from the handheld device is always used to place the camera but can optionally also control other parameters. The effect of this could be previewed in real-time as the camera is at the cursor position. Furthermore, there is a one-to-one mapping between the markers and the virtual objects. (See Figures 5 and 6.) This makes it possible to use the marker as a direct and tangible input device for the corresponding virtual object, which avoids the input indirection that is necessary when using, e.g., a virtual paddle.

By choosing the corresponding marker for moving a virtual object, object picking becomes implicit. This is an advantage over isotonic input when working with multiple objects. Additionally, the touchscreen of the handheld device can be used as a freely positionable plane for 2D authoring inside the 3D scene.

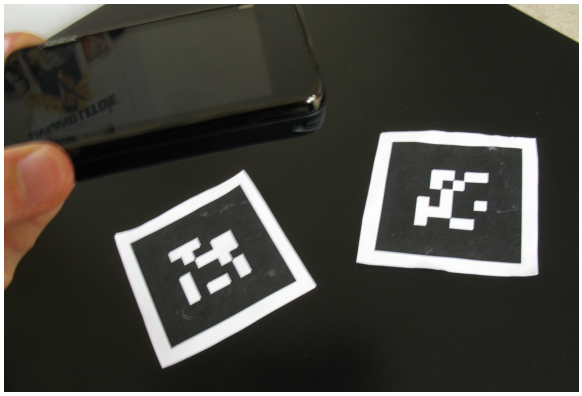


Figure 5: A marker-based handheld AR configuration. The handheld device controls the camera, while printed paper markers control the position and orientation of virtual objects that can be seen through the device's display.

The approach provides 6DOF input for the corresponding virtual object, such that moving a marker in the real world directly corresponds to the movement of the virtual object. We thus shift the focus from human-computer interaction to human-real-world interaction (See Figure 4), which allows an easy-to-use 3D authoring interface. But as the scene is perceived on the screen of the mobile device and not directly from the user's point of view, the output may be redirected.

Using a handheld device has the above-mentioned advantages, while still using widely available and easily deployable techniques. One major drawback of a handheld device

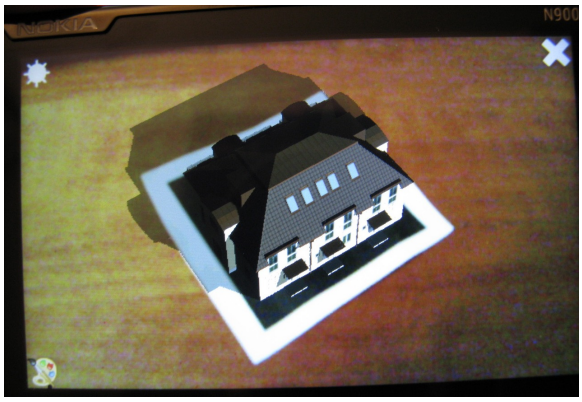


Figure 6: A virtual house model is assigned to a marker. Shadow-mapping helps blending the virtual object with the video of the real-world environment.

is, however, that one hand is occupied by holding the device. This restricts the possible interactions as only the free hand can be used to interact with the scene, while the hand holding the device can only control the camera. A possibility to overcome this limitation is to create the scene without the help of the device, purely relying on manipulating markers. Then one would use the handheld device only for a final visualization and simulation of the scene. This requires meaningful markers in order to be practical. Still, one would obviously loose the AR feedback during this "offline" authoring step. Any operations, which require interactive feedback, like scaling or changing the geometry of the virtual object, would be difficult to perform. Because of this limitation and since the underlying library did not allow sufficiently meaningful markers, this work only focuses on one-handed interaction.

4. Urban Planning Scenario

As we wanted to demonstrate the advantages of tangible interfaces for 6DOF input, we chose a scenario in which the primary task is to arrange rigid objects in 3D space. This constitutes the final step in 3D authoring, where the individual objects are composed to a scene.

An applicable scenario can be found in architecture and urban planning, where a common task is to examine how a new building would fit into an existing environment. During the placement of a new building, one has to, for example, consider the impact on lighting. A new residential building that is added to a group of existing residential buildings should avoid shading the previously lit apartments. (See Figure 1.) This scenario was also explored in the Luminous Planning Table project [UI99], which used augmented reality and tangible interfaces, but using interactive surfaces rather than handheld devices. The particular task would be to move the building in the terrain and examine the lighting

conditions for different times of the day. While this task can still be performed using traditional input methods, the input method will become the limiting factor when the simultaneous manipulation and adjustment of multiple residential buildings has to be made.

In this scenario, a spatially seamless method like the proposed tangible marker-based interface, has the potential to be more intuitive. This motivated our implementation to target this use case.

The use case is, of course, not limited to city planning or the arrangements of static objects. It may also be interesting to include dynamic objects, such as trees, and take their growth over time into into account, as the trees may shade buildings after they exceed a certain size. Such scenarios are supported in our implementation through mechanisms to modify geometry at runtime, e.g., by scaling them. This can of course also be applied to any geometry, for example, when modeling houses of different sizes.

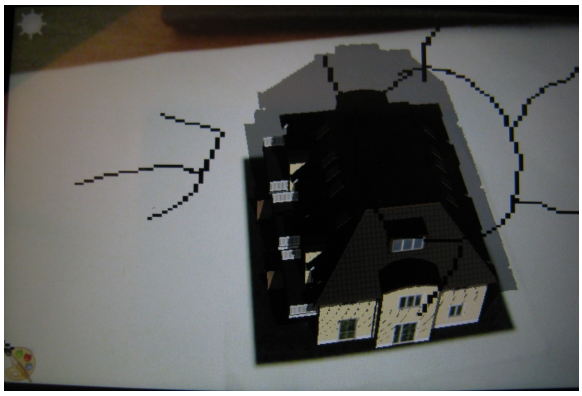


Figure 7: Our framework allows the user to enter an annotation mode, which freezes the video view. The user's annotations on the screen plane are then mapped to the corresponding 2D plane in the 3D scene.

As there might be further information one might want to add to the scene, like wind direction or location of a nearby highway, we allow annotating the scene by drawing on a 2D plane placed in the 3D space, in the spirit of the Virtual Notepad [PTW98], as shown in Figures 7 and 8.

In summary, the application enables the following ways to experience and manipulate the scene:

- Video see-through (6DOF camera positioning and projected overlay).
- Specify light vector using the current direction of the handheld (time of day).
- Draw annotations on the 2D plane defined by the handheld display. (See Figures 7 and 8.)
- Tangible 6DOF manipulation of virtual objects using physical markers. (See Figure 1.)
- Virtual interaction (scaling).

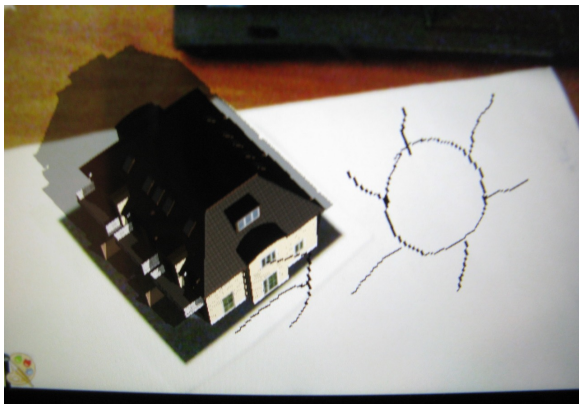


Figure 8: As the user changes the perspective by moving the handheld, the 2D annotation plane stay fixed in the 3D scene.

The lighting simulation adds a behaviorist aspect and thus goes beyond the 3D positioning which most existing AR interfaces focus on. For realistic lighting simulation with real-time shadows, we take advantage of the programmable graphics hardware which has recently become available on mobile phones.

5. Interactions

In the following we will look at adding an annotation in more detail, as it shows two different approaches for handling input. Adding an annotation happens in two stages; first the user puts the device to the desired position and then the tracking is frozen such that the annotation can be drawn.

5.1. Positioning an Annotation

The annotations are drawn on 2D planes that are positioned (3DOFs) and oriented (3DOFs) in 3D space. Using the device as a 6DOF cursor to place this plane is a natural choice as it is a good real-world representation for the virtual plane to be created.

5.2. Drawing an Annotation

To provide a stable 2D plane, we pause the tracking during the drawing action, such that 6DOF input is temporarily disabled. The device's touchscreen is used as a canvas on which the user's input is mapped to annotations in the 2D plane. (See Figure 8.) The resulting scene satisfies the user's expectations of interacting with a 2D plane [HPGK94]. If we would continuously update the position and orientation of the 2D plane based on the 6DOF tracking, the user would effectively be drawing a 3D-curve in the annotation volume. While this might be desirable for authoring a new 3D object, it is beyond the scope of our scenario, where we want to constrain the input to virtual annotations similar to the "virtual notepad" approach by Poupyrev et al. [PTW98].

6. Rendering and Object Selection

An enhanced version of ARToolKitPlus is used for tracking the markers and rendering overlays, where each virtual object is represented by an ARToolKit binary marker.

Real-time shadowing is performed using shadow mapping, utilizing the programmable OpenGL ES 2.0 pipeline[¶].

A permanent on-screen overlay is displayed to provide functionality for controlling light position and toggling the annotation mode.

6.1. Nokia N900 as a Handheld AR platform

Previously, the difference between a workstation and a handheld device required a different programming approach to develop AR applications. This often involving special fixed-point implementations [WS03] and software rendering, but this is not an issue anymore, thanks to today's modern mobile platforms.

The Nokia N900 provides a similar software environment[‖] to that found on a Linux workstation. This makes it possible to use the same APIs for development, testing, and deployment.

Furthermore, the hardware capabilities are similar to a workstation – although obviously not quite as fast. There is no considerable penalty in relying on floating point calculations and there is dedicated programmable hardware for 3D rendering. In fact, the OpenGL ES 2.0 API is very similar to the standard OpenGL 3.2 API, which makes it attractive to use advanced shader-based rendering techniques on handheld devices.

Although the CPU on the N900 is sufficiently fast for real-time tracking, the fill-rate of the graphics chipset is unfortunately not sufficiently high yet to support the display of multiple complex objects at high frame rates at the native resolution (800×480). The video is captured at 800×480, and processed and displayed at 400×240. We found the video texture upload to be the main bottleneck as it limits the performance to maximum 22 fps in our application. In our current application we are rendering two buildings (487 triangles) at 11 fps with shadow mapping and at 19 fps without. Performance could, however, be increased, for example, by reducing the video and display resolution, and through the use of native DEPTH_TEXTURE support.

[¶] http://www.khronos.org/opengles/sdk/docs/man/

[‖] http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/Architecture/Top_Level_Architecture

6.2. Depth Texture Generation

The drivers for the PowerVR SGX530 currently do not support the DEPTH_TEXTURE texture format, therefore we instead store the depth information in an ordinary RGBA texture. Here, it is not sufficient to just store the value in one of the color channels. The resulting conversion from float to byte would lead to a significant loss of precision, since the float is stored using 32 bits and a byte is stored using 8 bits. We, therefore distribute the floating point number across all four channels.

6.3. Packing

Storing the depth value in an RGBA texture works by using the following method in the fragment shader: as the first step, the packing vector $\vec{p}$ is created. It contains the shifting factors to break the 32 bit number into 4 bytes:

$$\vec{p} = \begin{bmatrix} 256^3 & 256^2 & 256^1 & 256^0 \end{bmatrix}$$

The result vector is then computed by the following equation, where $z \in [0,1]$ is the depth value:

$$\vec{y} = mod(\vec{p} \cdot z, 1)$$

As the result is not automatically converted to bytes, we need to manually cut the resulting floats to fit inside 8 bits. The element-wise modulo cuts off the number to the left.

To cut the number to the right, first a vector $\vec{c}$ is created. It contains shifting factors to shift the numbers to the right again:

$$\vec{c} = \begin{bmatrix} 0 & \frac{1}{256} & \frac{1}{256} & \frac{1}{256} \end{bmatrix}$$

A vector $\vec{t}$ is created to pick the components from our intermediate result:

$$\vec{t} = \begin{bmatrix} y_0 & y_0 & y_1 & y_2 \end{bmatrix}$$

Finally, the cutoff mask $\vec{c} * \vec{t}$ is applied, where $*$ denotes element-wise multiplication:

$$\vec{r} = \vec{y} - \vec{c} * \vec{t}$$

As our intermediate result was created by shifting the z value to the left, the displaced shifting back and subtracting leaves exactly the range of numbers which can now be safely converted to a byte.

6.4. Unpacking

In contrast to the packing operation, the unpacking is quite simple. Again we create the shifting vector $\vec{u}$:

$$\vec{u} = \begin{bmatrix} \frac{1}{256^3} & \frac{1}{256^2} & \frac{1}{256^1} & \frac{1}{256^0} \end{bmatrix}$$

The depth z is then simply $z = \vec{r} \cdot \vec{u}^T$. The packing happens for every pixel in the shadow map and can be limited by the size of the shadowmap, but the unpacking has to be performed for every pixel in the resulting image and thus introduces the overhead of a floating point vector multiplication

and 3 additions. The rendering process is thus also affected by the fragment shader's performance.

6.5. Object Selection/Picking

OpenGL ES 2.0 does not support selection-buffers and simulating these by using multiple rendering passes would lower performance even further, as the scenes are already fill-rate limited. Therefore the selection is implemented by storing an object identifier in the alpha channel (which can be recovered through alpha picking).

6.6. Object Format

The wavefront OBJ format[**] was chosen, as it is straightforward to parse and supports static objects with textured geometry. All faces in the OBJ file must be triangles, as this is the only geometry that OpenGL ES 2.0 supports rendering of. Although it would be possible to triangulate the faces after loading the file, doing so as a preprocessing step saves startup time on the device. It is, furthermore, assumed that the models fit inside the unity cube. This simplifies correct placement of the models on the markers and the positioning of the light source such that it results in good depth-map resolution. Model preparation can be done in most 3D editors, such as Blender[††].

7. Improving ARToolKitPlus

As part of this work the library ARToolKitPlus 2.1.1 was refactored and its architecture optimized. This lead to the release of ARToolKitPlus 2.2 which is now usable as a lightweight library for marker-based AR applications. Although the feature set is small compared to modern AR libraries that support NFT[‡‡], it is still powerful for rapid prototyping. Its low computational requirements also make ARToolKitPlus well-suited for handheld devices, where resources are limited. It may be reasonable to at this stage trade NFT tracking for more advanced graphics or simulation, as the computation and memory requirements of NFT are much higher than those for binary marker tracking. In the following, the main changes to the library will be motivated and explained.

7.1. Accessing Multiple Marker Positions

Although the internal marker detection function returned all markers detected in the image, ARToolKitPlus::TrackerSingleMarker only returned the best one by certainty. This class was extended to work in two stages:

- Return the IDs of all detected markers.
- Allow selection of which detected marker to process.

7.2. Building as a Shared Library

The original ARToolKit library used static sized arrays for loading, e.g., additional markers. Hence the size had to be defined at compile time using the preprocessor. ARToolKitPlus was an advancement in this regard as it was a templated library, where the template arguments were used for configuration. This allowed simultaneous usage of several differently configured instances. Yet it has the major drawback that the templated source code has to be available during compilation, which results in longer compilation time, as no compilation units can be cached. As there are no shareable compilation units, one also loses all the advantages of shared libraries, e.g., updates without recompiling. Therefore, templated parametrization at compile time was removed in favor of runtime parametrization and dynamic memory allocation. This should even allow the framework to completely abandon parametrization and manage the memory fully dynamically.

7.3. Architectural Optimization

To reduce the complexity of ARToolKitPlus, its architecture was restructured and reduced to contain a minimal set of interfaces and classes that support all required functionality. This included removing some abstractions, fixed-point implementations (no advantages on modern mobile hardware) and legacy camera calibration files. It was also identified that the library did not expose the full functionality of the underlying ARToolKit library, like the tracking of multiple individual markers.

- **MemoryManager** This class abstracted calls to *malloc*/*free* and *new*/*delete*. Although this might have been useful when running ARToolKit on the first Windows-based PDAs, many devices today have modern Linux kernels with better memory management. Furthermore, there was no working alternative memory manager available and the same functionality can be achieved by overloading *std* :: *new*.
- **CameraFactory**, **CameraImpl**, **arParam** ARToolKitPlus supported both old style binary configuration files, as well as, newer and more precise textfiles from the Camera Calibration toolbox. As loading the binary files involved changing the byte order of floating point numbers, which can not be guaranteed to work correctly, and there was a better alternative available, the old style *CameraImpl* was removed and replaced by *CameraAdvImpl*. Furthermore, all abstracting interfaces were removed so *Camera* could replace both *CameraAdvImpl* and *arParam*.
- **Logger** This class allowed output redirection, which was used for error messages and for status output. Error messages should be printed regardless of a logger being set or not. Status messages on the other hand can be easily redirected using the shell or overriding *std* :: *cout*. Therefore this class was removed and error messages are printed using *std* :: *cerr*.

[**] http://en.wikipedia.org/wiki/Obj
[††] http://www.blender.org
[‡‡] http://studierstube.icg.tu-graz.ac.at/

- **Profiler** This class allowed timing of some selected methods, but was removed as it is not a very accurate method for performance evaluation. Bottlenecks can be found using *callgrind* and timing information can be obtained using *SystemTap*.
- **TrackerImpl**, **TrackerSingleMarkerImpl**, **TrackerMultiMarkerImpl** These were the only available implementations for *Tracker*, *TrackerSingleMarker*, *TrackerMultipleMarker*. Therefore the abstraction was removed and the implementations were renamed to the former interfaces names.

8. Conclusions

We have presented an optimized, lightweight framework for creating tangible user interfaces through AR on commercially available mobile devices. We exploit the capabilities of modern mobile platforms to blend the boundaries between real and virtual worlds through rendering with hardware-accelerated graphics and techniques for realistic shadows.

In our architecture and urban planning scenario, AR is used for intuitive authoring and interaction with a complex scene. We demonstrate how a tangible AR interface can enable direct manipulations of 3D scenes and a more efficient workflow with the interactive lighting simulation in our application. Moreover, we show how the display's viewport can be used as a stable annotation surface in the 3D environment.

9. Future Work

As most of the relevant information is already available to multiple users, as it is simply stored in the topology of the markers, the next logical step would be to implement a communication interface between the handheld devices (similar to what was demonstrated in AR tennis [Hen07]). This would allow collaboration by sharing annotations and lighting position between multiple devices. The best way to achieve this would be to use link-local multicast IP or a peer-to-peer Bluetooth connection between the devices.

For the chosen architecture scenario it may also be interesting to include sound in the simulation. This could, for instance, allow the simulation of the noise level based on the distance to the street and intermediate objects, like anti-noise barriers.

We are also interested in supporting NFT to allow more meaningful markers like photographs or renderings of the virtual objects. This would make an "off-line" editing mode more feasible, where the scene is constructed only using the real-world objects. The augmented view through the device would primarily be used to view the simulation results and for interaction with the virtual content. This separation would allow two-handed interaction in the authoring process, which could increase performance for sorting and arranging physical objects.

On-board sensors, e.g., accelerometers and gyroscopes, could be employed for improved tracking robustness and performance during, e.g., high-speed movement and temporary marker occlusion.

We also plan to look into how the cursor's role on the device could be expanded by allowing direct manipulation of virtual objects. We are, for example, considering supporting the dragging of parts on the virtual model using the touch screen or even with the whole device.

We see our framework being usable for a number of different application scenarios, which we plan to explore further, in future work. Some examples include:

9.1. Interactive Simulation for Museum Exhibitions

The technique can be used in museums to bring exhibitions to life. Consider a ship that sunk because of the wrong weight distribution. Visitors could experience the physics using the tangible interface presented in this work together with the simulation, through a handheld device. One can use one marker on the blueprint of the ship and another marker representing the counterweight. While manipulating the placement of the counterweight on the ship, one could experience water running inside the ship and see a 3D model of the ship and how the simulated physics affects its buoyancy. While the same simulation would also be possible only using VR, the interaction might be more difficult and the usage inside the museum could be limited. By using AR on the other hand, one might also use miniature models of the ship and the weights to provide real physical sources and only run the simulation on the device.

9.2. Cannonball Game

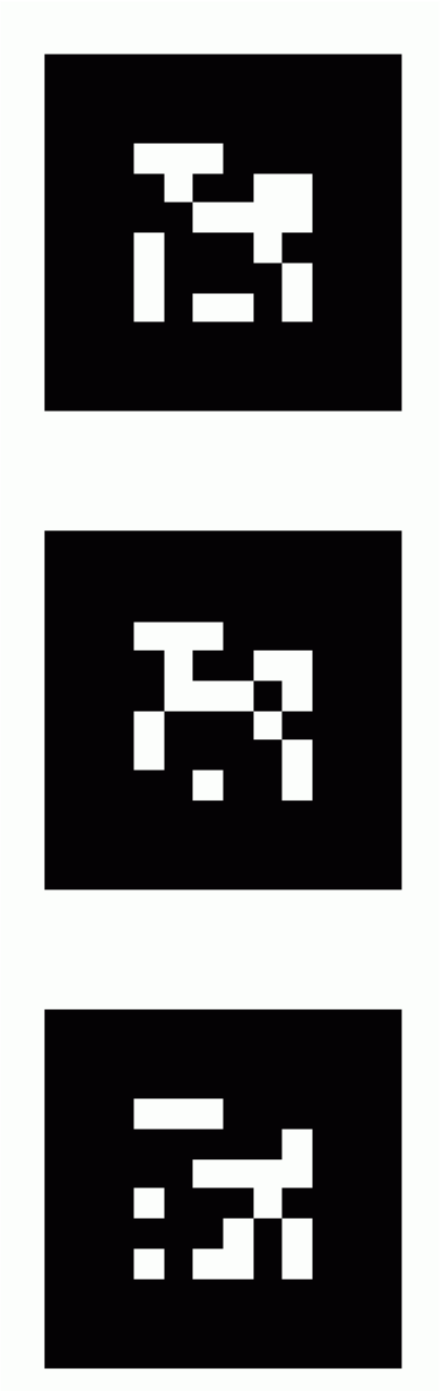
Players place markers, which represent castles with cannons, on a shared surface with the goal to destroy the opponent's castle first. The rotation of the marker controls the cannon, while the handhelds provide feedback of simulated wind and projectile trajectory. The concept is borrowed from "Ballerburg", a popular computer game from the 90s. A tangible AR version could provide an easier-to-use interface for 3DOF input and be used as a teaching tools for the physics of uniform acceleration.

9.3. Sociology simulation

The markers are placed on a city map and represent vaccination centers. The map itself is also tracked using an additional marker. The device runs a simulation and visualizes the amount of people that can be reached using the topology that is controlled by manipulating the position of the different centers. This application can be used for education or for rapid prototyping. It reuses existing data provided by the map and augments it with distance information to perform a simulation.

10. Acknowledgments

We thank Nokia for providing the N900 mobile devices that were used in this research.



References

[BKP08] BILLINGHURST M., KATO H., POUPYREV I.: Tangible augmented reality. In *ACM SIGGRAPH ASIA 2008 courses* (2008), ACM, p. 7. 2

[Hen07] HENRYSSON A.: Bringing Augmented Reality to Mobile Phones. *Linköpings universitet, Norrköping* (2007). 3, 8

[HO04] HENRYSSON A., OLLILA M.: UMAR: Ubiquitous mobile augmented reality. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia* (2004), ACM, p. 45. 3

[HPGK94] HINCKLEY K., PAUSCH R., GOBLE J., KASSELL N.: A survey of design issues in spatial input. In *Proceedings of the 7th annual ACM symposium on User interface software and technology* (1994), ACM, p. 222. 2, 5

[LNBK04] LEE G., NELLES C., BILLINGHURST M., KIM G.: Immersive authoring of tangible augmented reality applications. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality* (2004), IEEE Computer Society, p. 181. 3

[MLB04] MOHRING M., LESSIG C., BIMBER O.: Video see-through AR on consumer cell-phones. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality, 2004. ISMAR 2004* (2004), pp. 252–253. 3

[PTW98] POUPYREV I., TOMOKAZU N., WEGHORST S.: Virtual Notepad: handwriting in immersive VR. In *Proceedings of the Virtual Reality Annual International Symposium* (1998), Citeseer, pp. 126–132. 3, 5

[RN95] REKIMOTO J., NAGAO K.: The world through the computer: Computer augmented interaction with real world environments. In *Proceedings of the 8th annual ACM symposium on User interface and software technology* (1995), ACM, p. 36. 2

[Sho92] SHOEMAKE K.: ARCBALL: a user interface for specifying three-dimensional orientation using a mouse. In *Graphics Interface* (1992), vol. 92, pp. 151–156. 2

[UI99] UNDERKOFFLER J., ISHII H.: Urp: a luminous-tangible workbench for urban planning and design. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit* (1999), ACM, pp. 386–393. 4

[WLS08] WAGNER D., LANGLOTZ T., SCHMALSTIEG D.: Robust and unobtrusive marker tracking on mobile phones. In *Proceedings of the 2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality-Volume 00* (2008), IEEE Computer Society, pp. 121–124. 3

[WRM*08] WAGNER D., REITMAYR G., MULLONI A., DRUMMOND T., SCHMALSTIEG D.: Pose tracking from natural features on mobile phones. In *7th IEEE/ACM International Symposium on Mixed and Augmented Reality, 2008. ISMAR 2008* (2008), pp. 125–134. 3

[WS03] WAGNER D., SCHMALSTIEG D.: First steps towards handheld augmented reality. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers* (2003), Citeseer, p. 127. 3, 6

Appendix A: Preparing geometry, and compiling/running the application

Model preparation in Blender

- Import the model using $File \rightarrow Import$
- Align the model to the center of the coordinate space using $Space \rightarrow Transform \rightarrow ObData$ to Center
- Scale the model using $Space \rightarrow Transform \rightarrow Properties$ so that the size along any axis is at most 2 (so the model fits in the unity cube)
- Export the model in the .obj File Format ($File \rightarrow Export \rightarrow Wavefront$). It is important that you check "Triangulate" and "Normals" in the following dialog.

Compilation Instructions

As default the application associates marker-id 0 with "models/casa/casa.obj" and marker-id 1 with "models/farm/farm.obj". These are hardcoded in *Scene.cpp* and therefore changing the association or adding further markers requires recompilation.

Dependencies The application can be compiled on Ubuntu($\geq$ 10.04) and on Maemo 5. It has the following dependencies:

- Qt ($\geq$ 4.6)
- gstreamer ($\geq$ 0.10.13)
- ARToolKitPlus ($\geq$ 2.2)
- X11 (only for grabbing volume keys under Maemo 5)
- GLEW (for non GLES 2.0 compilation only)

On Ubuntu this requires an additional PPA[§§] and on Maemo 5, the extras-devel repository has to be enabled.

Compiling QMake is used as the build system. To initiate the build process first run *qmake* and then *make*. The according build flags are set automatically depending on the platform. To configure the build options edit "arapp.pro" in the source folder.

Maemo specific notes The compilation for the N900 happens inside a cross compilation environment called Scratchbox. The installation of the Scratchbox environment and the Maemo 5 SDK are described in the Maemo Wiki[¶¶].

To transfer data to/from the device *sshfs*[||||] is recommended, as simply attaching the device over USB does not allow copying files outside the home folder, and files inside the home folder can not be executed. USB networking[***] also gives much higher transfer rates than WLAN.

Manual for running the Application

The easiest way to install the application is to enable the extras-devel repository on the N900 and then install it using the App Manager (ARapp in Section Graphics).

1. Print the two supplied markers SimpleStd_000 and SimpleStd_001 (See previous page).
2. Open the camera to adopt to the ambient light. Close the camera application, but leave the lens cover open.
3. Start the ARapp.

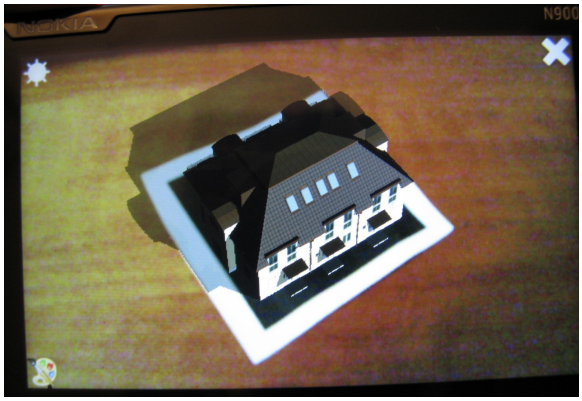


Figure 9: The AR application with on-screen controls in the corners. *Sun* icon sets light to camera position (top-left), *X* icon closes the application (top-right), and the *Palette* icon enters/exits annotation mode (bottom-left).

- The controls overlay allows fixing the light vector to the current camera position (*Sun* icon, top left), closing the application (*X* icon, top right) and entering/exiting annotation mode (*Palette* icon, bottom left). (See Figure 9).
- The display of the annotation plane can be toggled pressing "c" on the keyboard (only works for new annotations).
- Once fixed, the light vector can be manually rotated around the y-axis using the arrow keys.
- Objects can be scaled using the zoom keys of the device. An object can be selected by tapping on it on screen.

Alternatively the application can be started using the console, the binary is located in /opt/arapp/arapp and can be started with the "noshadow" option which disables shadowmapping for better performance.

§§ `https://edge.launchpad.net/~rojtberg/+archive/ppa`

¶¶ `http://wiki.maemo.org/Documentation/Maemo_5_Final_SDK_Installation`

|||| `http://wiki.maemo.org/Documentation/Maemo_PC_Connectivity_Tutorial/File_Sharing#Using_SSHFS_mounts`

*** `http://wiki.maemo.org/USB_networking`

Joint Structural and Inter-frame Skipping for MPEG-2 Video

Damir Isovic

Mälardalen University, Sweden
damir.isovic@mdh.se

Abstract

In this paper, we present a combined structural and inter-frame skipping method for quality aware processing of MPEG-2 video upon overload situations. The method selects video frames both based on the structural properties, such as frame types, sizes and position in the video stream, as well as on the internal, sub-frame characteristics, i.e., the number of relevant macroblocks within a frame.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Information Systems]: Multimedia Information Systems—Video, H.4.3 [Information Systems]: Communications Applications—Videoconferencing

1. Introduction

Today, a wide range of multimedia services has become an integral part of many industries from telecommunications to broadcasting and entertainment to consumer electronics. Distributed multimedia applications and mobile computing systems using wireless networks are becoming increasingly popular. Using such systems provide the end-users the possibility of transparently streaming multimedia content between devices of varying capabilities.

Moreover, the current trends are to move video processing from dedicated hardware to software, for reasons of cost, rapid upgradeability, and configurability. While being more flexible, software solutions are more irregular, since video processing will compete for the CPU with other applications in the system. At the same time, video is not only watched on classic TV sets, but increasingly displayed on smaller devices ranging from mobile phones to web pads, with limited system resources.

To provide smooth video playback in such heterogeneous multimedia streaming applications using software video processing, the system must be able to respond to the varying resource demands, on all the constrained devices, in such a way that the resulting quality (video or audio playback) is acceptable to all the end-users of the system. In other words, the system must be able to *adapt* the multimedia content to match the capabilities of the streaming networks and the sending/receiving devices.

Consequently, we need methods for decreasing the load introduced by media applications. There are basically two ways to do this: quality reduction, and frame skipping. With the quality reduction strategy, the decoder reduces the load by using a downgraded decoding algorithm, while frame skipping means that not all frames in a video stream are decoded and displayed, i.e., some of the frames are skipped.

In this paper, we focus on the frame skipping approach. Frame skipping can be used sparingly to compensate for sporadic high loads, or it can be used frequently if the load is structurally too high. Moreover, frames can be skipped both before sending the stream on the network, i.e., on the sending device, if the network bandwidth is restricted, and on the display device, if the processing power is limited.

However, frame skipping needs appropriate assumptions about the video stream to be effective. Skipping the wrong frame at the wrong time can result in a noticeable disturbance in the played video stream. In extreme cases, the decoding of a large and important frame might just not make it, therefore being lost and impeding quality, while simply skipping to decode a small preceding frame might have freed the resources for completion, with only slight quality reduction. In addition, skipping a frame may affect also other frames due to inter-frame dependencies. In a typical movie, a single frame skip can ruin around 0.5 seconds. Thus, frame skipping needs appropriate assumptions and constraints about streams to be effective [IFS03] .

In our previous work [IF04] we have developed a quality-aware frame skipping approach for MPEG-2 video based

on realistic timing constraints for the decoding of MPEG streams. Given that not all frames can be processed, it selects those which will provide the best picture quality while matching the available resources, starting only such decoding, which is guaranteed to be completed on time. This *structural* approach to frame skipping, i.e., whole frames are skipped, is very effective with respect to making fast skipping decision at run-time, but it is not very fine grained. It does not examine the contents of the frame when selecting frames, which could play an important role. For example, the structural approach cannot provide a good comparison of two consecutive *B* frames of approximately the same size.

In this paper, we propose to use *sub-frame* selection together with the structural selection, i.e., to consider the information contained *within* a frame whenever the structural approach fails to compare two frames. As a first step towards a selection criteria on sub-frame level we have analyzed a number of MPEG-2 video streams with respect to the frame contents. We have looked into the bitstream organization (i.e., slices, macroblocks and blocks), of diverse video streams to identify the most redundant picture elements which are to be skipped first upon overload situations. Then, we proposed a new set of criteria for frame skipping based on the internal frame properties, to be used when comparing two frames on sub-frame level. Finally, we have integrated the new sub-frame skipping algorithm with the existing structural skipping method, providing a frame skipping method that both considers the whole frames and the information within single frames when making decision which frames in a video stream should be kept upon overload situations.

2. Related work

A server based algorithm for integrating multimedia and hard real-time tasks has been presented in [AB98]. It is based on average values for execution times and interarrival intervals. Work on predicting MPEG execution times, which is necessary to know for efficient frame skipping, has been presented in [BMP98, BA00]. A method for real-time scheduling and admission control of MPEG-2 streams that fits the need for adaptive CPU scheduling has been presented in [DA00]. The method is not computationally overloaded, qualifies for continuous re-processing and guarantees Quality-of-Service (QoS). However, no consideration on making priorities on the frame level has been done. A frame skipping pattern that makes distinction between frames has been presented in [NHW00]. However, only one skipping criterion, QoS human [NLW*02], has been applied when selecting frames, taking no consideration about frame sizes, buffer and latency requirements, or compression methods used.

Most standard video processing methods will fail to satisfy the demands of MPEG-2 upon overload situations as they do not consider the specifics of this compression standard. In our work, we consider both the structural properties of a video stream as well as the sub-frame properties of single video frames when making skipping decisions.

3. Structural frame skipping of MPEG-2 video

3.1. MPEG-2 Video Stream

The MPEG-2 standard defines three types of frames, *I*, *P* and *B*, see figure 1-a. The *I* frames or *intra* frames are simply frames coded as still images. They contain absolute picture data and are self-contained, meaning that they require no additional information for decoding. *I* frames have only spatial redundancy providing the least compression among all frame types. Therefore they are not transmitted more frequently than necessary.

The second kind of frames are *P* or *predicted* frames. They are forward predicted from the most recently reconstructed *I* or *P* frame, i.e., they contain a set of instructions to convert the previous picture into the current one. *P* frames are not self-contained, i.e., if the previous reference frame is lost, decoding is impossible.

The third type is *B* or *bi-directionally* predicted frames. They use both forward and backward prediction, i.e., a *B* frame can be decoded from a previous *I* or *P* frame, and from a *later I* or *P* frame. They contain vectors describing where in an earlier or later pictures data should be taken from. They also contain transformation coefficients that provide the correction. *B* frames are never predicted from each other, only from *I* or *P* frames. As a consequence, no other frames depend on *B* frames. *B* frames require resource-intensive compression techniques but they also exhibit the highest compression ratio, on average typically requiring one quarter of the data of an *I* picture.

Predictive coding, i.e., the current frame is predicted from the previous one, cannot be used indefinitely, as it is prone to error propagation. A further problem is that it becomes impossible to decode the transmission if reception begins part-way through. In real video signals, cuts or edits can be present across which there is little redundancy. In the absence of redundancy over a cut, there is nothing to be done but to send from time to time a new reference picture information in absolute form, i.e., an *I* frame. As *I* decoding needs no previous frame, decoding can begin at *I* coded information, for example, allowing the viewer to switch channels. An *I* frame, together with all of the frames before the next *I* frame, form a *Group of Pictures (GOP)*, see 1-b. The GOP length is flexible, but 12 or 15 frames is a common value. Furthermore, it is common industrial practice to have a fixed pattern (e.g., *I BB P BB P BB P BB*). However, more advanced encoders will attempt to optimize the placement of the three frame types according to local sequence characteristics in the context of more global characteristics.

As mentioned above, *B* frames are predicted from two *I*

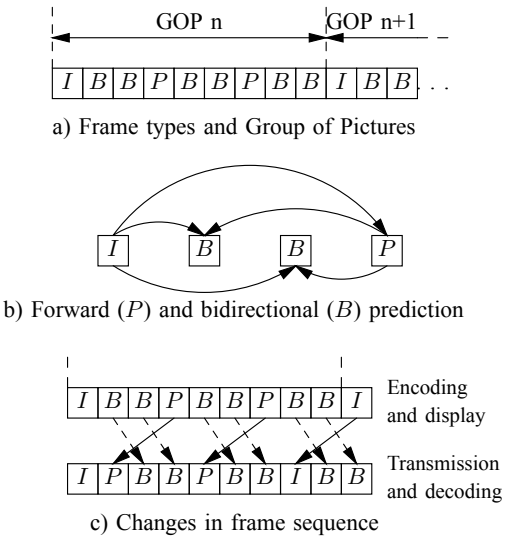


Figure 1: *MPEG-2 video stream*

or *P* frames, one in the past and one in the future. Clearly, information in the future has yet to be transmitted and so is not normally available to the decoder. MPEG gets around the problem by sending frames in the "wrong" order. The frames are sent out of sequence and temporarily stored. Figure 1-c shows that although the original frame sequence is *I BB P* ..., this is transmitted as *I P BB* ..., so that the future frame is already in the decoder before bi-directional decoding begins. Picture reordering requires additional memory at the encoder and decoder and delay in both of them to put the order right again. The number of bi-directionally coded frames between *I* and *P* frames must be restricted to reduce cost and minimize delay, if delay is an issue.

3.2. Structural skipping

In our previous work [IF04] we have identified a number of criteria that are to be applied when making skipping decisions. According to the frame type criterion, the *I* frame is the most important one in a GOP. If we lose the *I* frame in a GOP, then the decoding of all consecutive frames in the GOP will not be possible, since all other frames in the GOP depend directly or indirectly on the *I* frame. *B* frames are the least important ones because they are not reference frames.

Frame position criterion is applied on *P* frames. Skipping a *P* frame will cause the loss of all its subsequent frames, and the two preceding *B* frames within the GOP. For instance, skipping the first *P* frame (P_1) would make it impossible to reconstruct the next *P* frame (P_2), as well as all *B* frames that depends on both P_1 and P_2. And if we skip P_2 then we cannot decode P_3 and so on. Hence, the closer to the start of the GOP the more important *P* frame.

Frame size criterion applies mainly to *B* frames. According to our analysis [IF02], there is a relation between frame size and decoding time, and thus between size and gain in display latency. The purpose of skipping is to increase display latency. So, the bigger the size of the frame we skip, the larger display latency obtained. However, skipping large *B* frames might not always be the best option. Small *B* frames might exploit complex compression techniques which minimize frame size, but are more expensive to decode, in terms of needed processing power. Frame prediction from reference frames is found to be most computationally expensive [MP93]. Hence, if the network bandwidth is limited, then large *B* frames should be skipped first, and if the objective is to decrease the CPU load, small, more compressed frames should be skipped.

Skipping distribution criterion says that with the same number of skipped *B* frames, a GOP with *evenly* skipped *B* frames will be smoother than a GOP with uneven skipped *B* frames, e.g if we have a GOP=*IBBPBBPBBPBB* then even skipping $I-BP-BP-BP-B$ will give smoother video than uneven skipping $I--PBBPBB--$, since the picture information loss will be more spread [NLW*02].

Please refer to our previous work [IF04] for details on all identified structural criteria and the structural frame skipping method.

4. Sub-frame skipping

A picture frame consist of a number of *macroblocks*, which are 16x16 arrays of luminance pixels, or picture data elements. Macroblocks in a frame can be coded as intra and non-intra, where the first type does not need a reference to be decoded, while the second one is either forward-predicted, backward-predicted or forward-and-backward predicted from other macroblocks. The macroblocks within an *I* frame are coded as intra. Most of the macroblocks in *P* and *B* frames are non-intra. However, some of them are coded as intra, which is a way to refresh the video information without introducing an extra *I* frames. This guarantees that after a certain number of frames all macroblocks in a frame have been intra updated. This will stop the error propagation when a part of a video frame is lost during transmission.

4.1. Intra macroblocks

Intra-macroblocks are the most interesting to keep when making skipping decisions because they are used as reference in other frames. The more intra-macroblocks in a frame, the more dependencies on the frame.

We have analyzed diverse MPEG-2 streams for the amount of intra-coded macroblocks per *P* and *B* frames, see

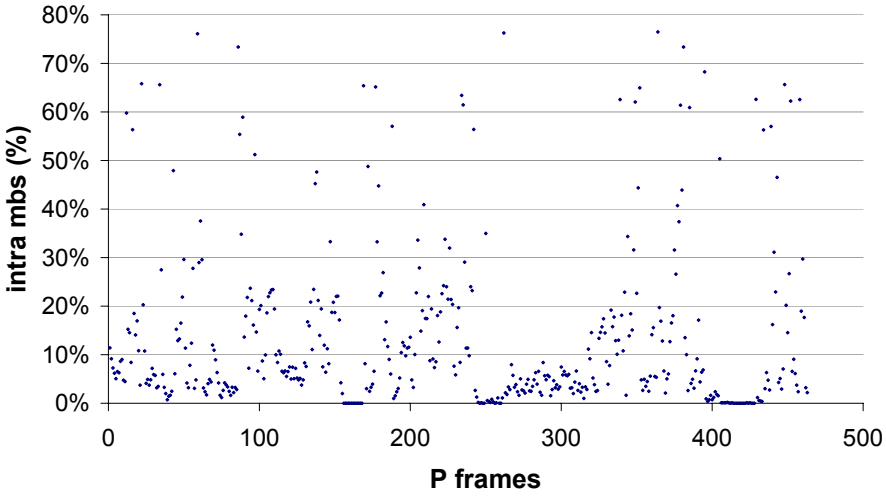


Figure 2: *Intra macroblocks (mbs) per P frames in an example MPEG-2 video stream*

figures 2 and 3for an example. We can see from the figure that the number of intra macroblocks vary a lot between different frames. Hence, we can propose a new criterion for selecting frames on inter-frame level: the more intra macroblocks per a frame, the higher priority will be assigned. Once again, observe that this is valid on the decoder side, i.e., the stream is either stored locally or, in the case of video streaming, the GOP has been transmitted to the decoder. On the other hand, if we perform stream adjustment before sending the stream on the network, we might need to actually do the opposite, i.e., keep the frames with less intra macroblocks, since they have lower bit sizes.

4.2. Skipped macroblocks

Furthermore, we have analyzed the percentage of skipped macroblocks, i.e., macroblocks for which no data is encoded. Skipped macroblocks are used to achieve higher compression ratio. When a macroblock is skipped, it is implicitly defined by the standard in the following way: in a *P* frame, a skipped macroblock is a direct copy of the corresponding macroblock from the previous *I* or *P* frame. In a *B* frame, a skipped macroblock is reconstructed by assuming the motion vectors and motion type (i.e., forward, backward, or bidirectional) are the same as the last encoded macroblock. In this case, skipped macroblocks can not follow intra-coded macroblocks because then there would be not motion type or motion vectors defined.

The average numbers of skipped macroblocks per frame type for some example streams are presented in table 1. We do not present skipped macroblocks for *I* frames simply because we could not identify any. According to the MPEG standard [MPE96], even *I* frames can have skipped macroblocks (that use only spatial redundancy), but we could not find any skipped *I* macroblocks in any of the analyzed streams. We conclude that macroblocks are seldom skipped in *I* frames. Skipped macroblocks per *B* frames for an example video stream are shown in figure 4.

MPEG-2 Video Streams	*P* frames	*B* frames
Drama movie, 720x576	19%	15%
Action movie, 354x240	7%	10%
Philharmonic, 720x576	25%	22%
Cartoon, 354x240	12%	27%

Table 1: *Average skipped macroblocks per P and B frames*

The more skipped macroblocks per frame, the more similar it becomes to some other frame. Hence, frames with a lot of skipped macroblocks should be given lower priority, since some of the adjacent frames contain the same video information. Besides, no other macroblocks are reconstructed from skipped macroblocks.

4.3. Zero-motion macroblocks

By performing the sub-frame analysis we could make an interesting observation: the total number of macroblocks for some *P* frame is not equal to the sum of all intra macroblocks and forward-predicted macroblocks for the frame. *P* frames do not exploit backward prediction, i.e., they do not contain any backward-predicted macroblocks, hence the total sum of all macroblocks per *P* frame should be the sum of all intra and forward-predicted macroblocks. The explanation is that in *P* frames (and only *P* frames), there are some macroblocks which are not intra (i.e., motion compensation is in use) but also do not define any forward motion vectors. By definition, these macroblocks are interpreted as using mo-

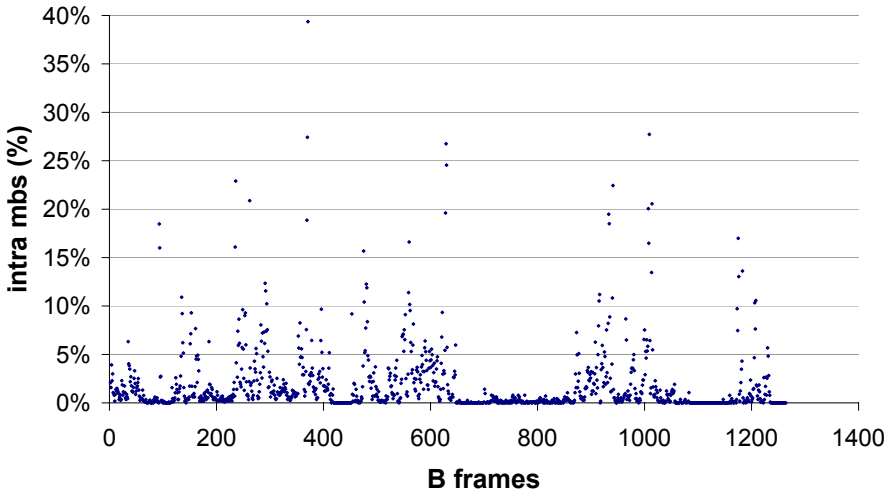


Figure 3: *Intra macroblocks (mbs) per B frames in an example MPEG-2 video stream*

tion compensation with a motion vector defined as (0,0). It is a special case in the standard because it happens so often. That what makes it interesting is those macroblocks are very good candidates for skipping on sub-frame level, since no other macroblocks depend on them. So, an additional inter-frame criterion for *P* frames is: the more (0,0) macroblocks in a *P* frame, the lower priority should be assigned.

5. Combined skipping algorithm

The new skipping algorithm that we propose uses the following set of structural and sub-frame skipping criteria:

1. *Frame type* – Assign highest priority to *I* frames, and lowest priority to *B* frames.
2. *GOP position* – Assign higher priority to *P* frames that are closer to the start of the GOP.
3. *Skipping distribution* – Assign priority to *B* frames such that the skipped frames are distributed more evenly in the GOP.
4. *Frame size* – Assign higher priority to larger *B* frames.
5. *Intra macroblocks* – Assign higher priority to *P* and *B* frames with larger number of intra-coded macroblocks in the frame.
6. *Skipped macroblocks* – Assign lower priority to *P* and *B* frames with larger number of skipped macroblocks per frame.
7. *Zero motion macroblocks* – Assign lower priority to *P* frames with more zero motion macroblocks.

When deciding the relative importance of frames for the entire GOP, we assign priorities to frames according to all criteria collectively applied, rather than applying a single criterion. Since the criterion 1 is the strongest one, the *I* frame will always get the highest priority, i.e., all frames in the GOP depend on it. For *P* frames, in our previous work, we always kept the *P* frames closer to the start of the GOP. Here, we make better, more fine-grained decisions on which frames to keep, based on the internal structure of the frames, i.e., the number of intra, skipped and zero motion macroblocks. Similarly, we used to apply only the frame size criterion for *B* frames, but in the new algorithm we look deeper into the structure of the frame to make better skipping decisions.

Here is the the pseudo-code for the frame selection algorithm that takes a set of frames, e.g., a GOP, as an input and assigns importance values to the frames based on the identified criteria:

Let:

N	= GOP length
M	= distance between reference frames
$\mathcal{P}$	= a set containing all P-frames in the GOP
$\mathcal{B}$	= a set containing all B-frames in the GOP
$v(f)$	= importance value of frame f
ESC_i	= i^{th} even-skip chain of *B* frames

Step 1: Assign the highest value to the I-frame (equal to the number of frames in the GOP).

$$v(I) = N$$

Step 2: The set $\mathcal{P}$ contains all P-frames, sorted according to their position in GOP. The longer the distance from the I-frame, the lower the importance value.

$$\forall P_i \in \mathcal{P}, 1 \leq i \leq |\mathcal{P}|$$
$$v(P_i) = N - i$$

Step 3: Reassign priorities of P-frames according to the intra-macroblock criterion. Apply skipped macroblock and zero motion criteria to break ties.

$$\forall P_i \in \mathcal{P}, 1 \leq i \leq |\mathcal{P}|$$

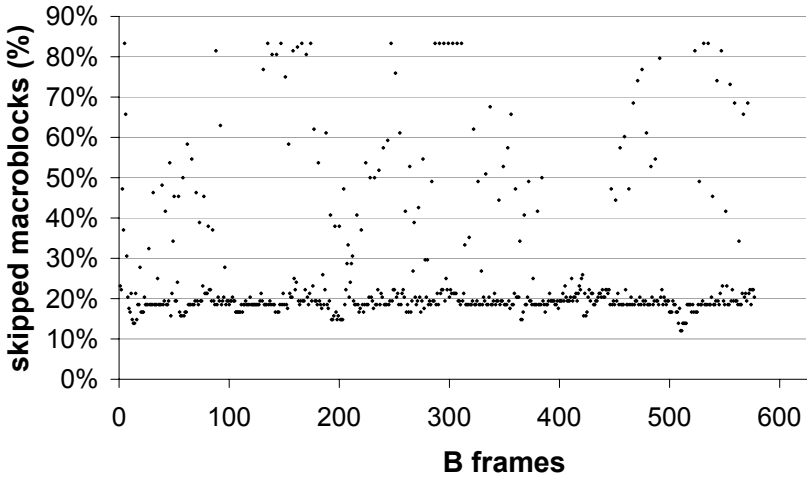


Figure 4: *Skipped macroblocks in B frames*

$$
\begin{aligned}
&if\ (intra(P_i) > intra(P_{i-1}))\\
&\quad swap(P_i, P_{i-1})\\
&elseif(intra(P_i) == intra(P_{i-1}))\\
&\quad if\ (skipped(P_i) < skipped(P_{i-1})\\
&\quad\quad swap(P_i, P_{i-1})\\
&\quad elseif(skipped(P_i) == skipped(P_{i-1}))\\
&\quad\quad if\ (zero(P_i) < zero(P_{i-1}))\\
&\quad\quad\quad swap(P_i, P_{i-1})
\end{aligned}
$$

Step 4: Initially set all values for B-frames to the lowest P-value.

$$
\begin{aligned}
&\forall B_k \in \mathcal{B}, 1 \le k \le |\mathcal{B}|\\
&\quad v(B_k) = min[v(P_i) \mid 1 \le i \le |\mathcal{P}|] - 1
\end{aligned}
$$

Step 5: Identify all "even-skip" chains for B-frames and sort them according to the total byte size. Decrease the importance values of the B-frames, depending on which chain they belong to.

$$
\begin{aligned}
&ESC_1 = \{B_1\} \cup \{B_{1+j*M} \mid 1 \le j \le \tfrac{N}{M-1}\}\\
&\forall i, 2 \le i \le |\mathcal{B}^*|\\
&\quad ESC_i = \{B_i\} \cup \{B_{i+j*M} \mid 1 \le j \le \tfrac{N}{M-1}\}\\
&\quad if\ \ sum(ESC_i) > sum(ESC_{i-1}))\\
&\quad\quad swap(ESC_i, ESC_{i-1})\\
&\quad \forall B_k \in ESC_i\\
&\quad\quad v(B_k) = v(B_k) - |ESC_{i-1}|
\end{aligned}
$$

Step 6: Within each chain, apply sub-frame skipping criteria to assign unique priorities to the B-frames in the chain.

$$
\begin{aligned}
&\forall B_k \in ESC_i\\
&\quad if\ (intra(B_k) > intra(B_{k-1}))\\
&\quad\quad swap(B_k, B_{k-1})\\
&\quad elseif(intra(B_k) == intra(B_{k-1}))\\
&\quad\quad if\ (skipped(B_k) < skipped(B_{k-1})\\
&\quad\quad\quad swap(B_k, B_{k-1})
\end{aligned}
$$

The presented algorithm skips small B-frames first. If the objective is to utilize limited network bandwidth, then the "even-skip" chains above should be sorted in acceding order, i.e., large B-frames should be skipped first. Moreover, in this case, we would also keep the frames with less intra macroblocks, since they have lower bit sizes. The algorithm for opimizing the stream before sending it over network is very similar to the one presented above, and hence, omitted in this paper.

6. Conclusions

In our previous work, we proposed a structural approach for quality-aware frame skipping of MPEG-2 video, which selects frames according to a set of criteria on the frame level, such as frame types, positions and sizes. In this paper, we extended it by considering the internal structure of the frames when making skipping decisions.

First, we analyzed a number of MPEG-2 video streams with respect to the frame contents, to identify the most redundant picture elements within a single frame. Then, we proposed a set of new, sub-frame skipping criteria, such as intra, skipped and zero macroblock, used for for intra-frame skipping. Finally, we integrated the sub-frame skipping with the structural skipping, and proposed a joint structural and sub-frame skipping approach for MPEG-2 video.

We have previously evaluated our frame skipping method based on structural skipping, by using both subjective and objective quality measurements. Currently, we are extending it to the joint structural and inter-frame skipping approach presented in this paper. Furthermore, looking into how we can apply similar methods on MPEG-4 video.

References

[AB98] ABENI L., BUTTAZZO G. C.: Integrating multimedia applications in hard real-time systems. In *Proceedings of the 19th IEEE Real-Time Systems Symposium* (Madrid, Spain, 1998). 2

[BA00] BURCHARD L. O., ALTENBERND P.: Estimating decoding times of mpeg-2 video streams. In *Proceedings of International Conference on Image Processing (ICIP 00)* (Vancouver, Canada, September 2000). 2

[BMP98] BAVIER A., MONTZ A., PETERSON L.: Predicting mpeg execution times. In *Proceedings of ACM International Conference on Surement and Modeling of Computer Systems (SIGMETRICS 98)* (Madison, Wisconsin, USA, June 1998). 2

[DA00] DITZE M., ALTENBERND P.: Method for real-time scheduling and admission control of mpeg-2 streams. In *The 7th Australasian Conference on Parallel and Real-Time Systems (PART2000)* (Sydney, Australia, November 2000). 2

[IF02] ISOVIC D., FOHLER G.: Analysis of mpeg-2 streams. In *Technical Report at Malardalen Real-Time Research Centre,Vasteras, Sweden* (March 2002). 3

[IF04] ISOVIC D., FOHLER G.: Quality aware MPEG-2 stream adaptation in resource constrained systems. In *ECRTS* (Catania, Italy, July 2004). 1, 3

[IFS03] ISOVIC D., FOHLER G., STEFFENS L. F.: Timing constraints of mpeg-2 decoding for high quality video: misconceptions and realistic assumptions. In *Proceedings of the 15th Euromicro Conference on Real-Time Systems* (Porto, Portugal, June 2003). 1

[MP93] MAYER-PATEL K.: Performance of a software MPEG video decoder. In *ACM Multimedia Conference* (1993). 3

[MPE96] Iso/iec 13818-2: Information technology - generic coding of moving pictures and associated audio information, part2: Video. 4

[NHW00] NG J. K.-Y., HUI C. K.-C., WONG W.: A multi-server design for a distributed MPEG video system with streaming support and QoS control. In *Proceedings of the 7th International Conference on Real-Time Systems and Applications* (Cheju Island, South Korea, December 2000). 2

[NLW*02] NG J. K., LEUNG K. R., WONG W., LEE V. C., HUI C. K.: Quality of service for mpeg video in human perspective. In *Proceedings of the 8th Conference on Real-Time Computing Systems and Applications (RTCSA 2002)* (Tokyo, Japan, March 2002). 2, 3

Interactive Image-Space Volume Visualization for Dynamic Particle Simulations

M. Falk and S. Grottel and T. Ertl

VISUS – Visualization Research Center, University of Stuttgart, Germany

Abstract

Particle-based simulation plays an important role in many different fields of science and engineering. Two common visualization approaches for the resulting data are glyph-based rendering and density sampling employing volume rendering. Fine geometric features are inherently captured by glyph-based methods. However, they might suffer from aliasing and the global structure is often poorly conveyed. Volume rendering preserves the global structure but is limited due to the sampling resolution. To avoid aliasing artifacts and large memory footprints, we propose a direct volume rendering technique with on-demand density sampling of the particle data, as combination of splatting, texture slicing, and ray casting. We optimized our system with a novel ray cast termination employing early-z-test culling and hardware occlusion queries utilizing inter-frame coherency.

Our system contains a fully-featured volume renderer and captures all geometric features of the data set representable at the available display resolution. Since no pre-computation is required, the proposed method can be used easily to visualize time-dependent data sets. The effectiveness of our approach is shown with examples from different application fields.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

There are many particle-based simulation methods from different fields of science, such as molecular dynamics, agent-based simulations, discrete element method, and smoothed particle hydrodynamics. These simulations usually comprise a large number of entities which interact with each other, either in a pairwise interaction scheme or by contributing to a common continuous field which in return affects all entities, which can be atoms, molecules, or mesoscopic or macroscopic particles. From a visualization point of view all these entities can be handled rather equally as particles.

Particle-based simulations usually generate large data sets due to their time-dependent nature. Visualizing such data sets interactively is still a challenge. On commodity workstations, the mere data set sizes require highly optimized software to achieve interactivity. Even the mapping from the individual particles to visual primitives might not be clear. Depending on the focus of the simulation different visualization methods are commonly used, from points or sprite-based particles, over density or probability volumes, to highly specialized visualizations like protein secondary structures. While point- or sprite-based rendering directly shows the original data, these images often suffer from high visual complexity due to visual clutter from a high number of discrete graphical primitives.

To gain a more compact representation, which is usually easier to understand, the particle data is sampled into a density volume and an iso-surface is rendered for a significant iso-value. This approach is also related to metaball rendering (also called blobby surfaces). Surface representations are common for protein data sets, surfaces or layers separating specific areas of the data sets in molecular dynamics, and iso-surfaces of electron position probabilities in quantum mechanics simulations. However, this technique has the drawback of additional memory requirements for storing the sampled volume data in addition to the particle data sets.

We therefore propose a volume rendering system with on-demand reconstruction of the volume data to minimize this

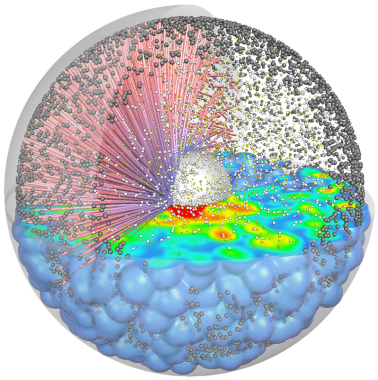


Figure 1: *Combining the volumetric visualization with the rendering of the simulated cell emphasizes the propagation of the signal. See Sec.4.1 for details.*

additional memory consumption. Our contribution is an optimized slice-based volume ray casting interleaved with an on-demand volume data reconstruction. Our method uses hardware occlusion queries to implement a novel termination method for the ray casting. We exploit the frame-to-frame coherency to compensate the additional overhead and latencies which these queries introduce. To show the effectiveness of our approach, we present examples from different fields of application, including propagation of the signaling front inside a cell in the context of systems biology, displaying relevant protein surfaces for biochemistry, as well as continuous material representations for data sets from molecular dynamics simulations of laser ablation processes. Compared to particle visualization and object-space volume ray casting our method results in superior image quality, due to the evaluation in image-space, has less memory requirements, since the volume data is only partially created on-demand, and still reaches comparative rendering performance.

2. Related work

Particle-based simulation Particle-based simulations have been used and studied for a very long time. The smoothed particle hydrodynamics introduced by Gingold and Monaghan [GM77] or molecular dynamics introduced by McCammon et al. [MGK77] are nowadays among the most popular simulation methods. In recent years the calculation power of graphics hardware was harvested for these techniques. Kipfer et al. [KSW04] presented a GPU-based system for particle simulations, which was, however, primarily intended for rendering and animation and not scientific simulation. A similar approach was presented by Kolb et al. [KLR04] in the same year and was extended to simulate coupled particles, i. e. particles reacting with each other and not only reacting to an underlying velocity field [KC05], allowing a Lagrangian particle simulation of fluids. Van Meel et al. [vMAF*07] used the modern general purpose GPU API CUDA for a fully functional molecular dynamics simulation on graphics hardware. A smoothed particle hydrodynamics simulation on the GPU was presented by Zhang et al. [ZSP08] to simulate and render liquids for computer graphics.

In contrast to such physically motivated simulations, which usually simulate a high number of particles with rather simple particle interactions (e. g. pair potentials), there are simulations following an orthogonal approach. E. g. in systems biology the movement of molecules and proteins inside a cell are simulated, where the internal structure of the cell – the cell cytoskeleton – is used to direct the movement of the proteins toward the core of the cell [KLR09], resulting in a simulation of less particles but with rather complex behavior (see Fig. 1).

Particle-based visualization Particle data sets are often directly visualized using glyphs to represent the individual particles. Gumhold [Gum03] presented an efficient way of GPU ray casting for glyphs with polynomial surfaces, ellipsoids in his work, to visualize symmetric tensor fields. A similar approach was used by Klein et al. [KE04] to visualize magnetic fields. Based on the same method Reina et al. [RE05] showed how to ray cast complex glyphs, constructed from quadratic primitives, like spheres and cylinders. This approach was extended by Grottel et al. [GRE09] to arbitrarily composed glyphs with optimized data transfer between CPU and GPU. Tarini et al. [TCM06] used methods from computer graphics, like ambient occlusion and silhouettes to enhance the depth perception. A recent publication by Grottel et al. [GRDE10] highly optimizes the rendering of particle glyphs employing hardware occlusion queries and hierarchical depth buffers, allowing interactive visualization up to 100,000,000 particles on commodity workstations.

For some fields of application, like proteins in biochemistry, specialized visual metaphors exist. Krone et al. [KBE08] showed how to generate the protein secondary structure representation on the graphics hardware. In a recent publication they used GPU ray casting of quadratic and quartic surfaces to render protein surfaces [KBE09]. Falk et al. [FKRE09] used GPU ray casting of spheres and point splats to visualize the movement of proteins inside a cell structure with different visual metaphors.

Metaballs rendering Compact representations of particle data sets, like surfaces or layers between different parts of the data, are most of the time the structures of interest. Such surfaces can be defined as iso-surfaces of volumetric data. Blinn proposed [Bli82] a metaball surface, an iso-surface in a density volume, defined by radial symmetric density kernel functions based at each particle. These metaballs can be rendered by ray casting or by extracting a geometric representation. Marching Cubes [LC87] is the most prominent algorithm to extract the geometry of such a surface.

On modern graphics cards the metaballs can be interactively evaluated and rendered. Kooten et al. [vKvdBT07] distribute points with repulsive forces on the surface and perform point-based rendering after the correct iso-surface is found. Sampling with too few points results in holes and

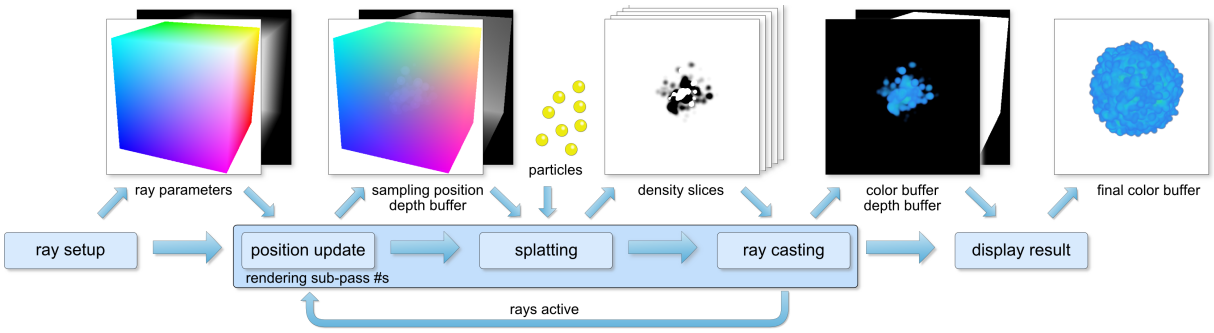


Figure 2: *Flow chart of our sliced ray casting method. In- and output buffers of the various stages are depicted above. After initial ray setup, multiple rendering sub-passes of sampling position update, density reconstruction through splatting and volume ray-casting are performed to generate the final image.*

visual artifacts. Müller et al. [MGE07] propose two possible ways of iterative ray casting metaballs. One approach – "walking depth plane" – uses multiple rendering passes and samples the density volume at specific depths in object space, which is related to our approach. The other approach pre-calculates a neighborhood list and evaluates the density from all neighboring particles in an iterative, single-pass raycasting. However, both methods only hardly achieve interactivity for medium-sized data sets. A method with a similar fundamental idea was presented by Kanamori et al. [KSN08], which however uses Bézier Clipping to find the ray-isosurface intersection. Linsen et al. [LvLRR08] extended this surface approach to multivariate data.

Volume rendering As the metaball surface is an isosurface of a density volume, another approach is the complete reconstruction of this density volume and to perform volume rendering. The system described by Cha et al. [CSI09] follows this approach. In biochemistry conformational changes of proteins can be visualized as flexibility volume in a similar way, as presented by Schmidt-Ehrenberg et al. [SEBH02].

Early implementations of volume rendering including shading, like presented by Westermann et al. [WE98] blend object-space-aligned texture stacks. View-aligned texture stacks provide higher visual quality but require 3D textures. Nowadays, volume ray casting is used, as presented e. g. by Drebin et al. [DCH88]. Krüger et al. [KW03] presented a view-aligned ray casting approach employing programmable graphics hardware. These multi-pass rendering approaches have been superseded by single-pass volume ray casting, e. g. presented by Stegmaier et al. [SSKE05].

Another approach for volume rendering is splatting, as proposed by Westover [Wes90]. This approach has also been optimized to harvest the computational power of modern GPUs, e. g. by Neophytou et al. [NM05], to perform correct image-space aligned splatting of rectangular voxels. Fraedrich et al. [FSW09] presented a system using this technique in a hierarchical fashion to interactively visualize a large astronomy data set.

3. Our Approach

Our method of sliced ray casting with on-demand volume reconstruction combines the ideas of texture slicing, ray casting, and splatting. The reconstruction of the density field from the particles is performed in screen space at viewport resolution and at the sampling depths used by the volume ray casting. The issue of perspective correction is thus moved from volume rendering to data reconstruction. A tight interleaving between the volume rendering and the data reconstruction allows to minimize the required memory footprint. A novel method for termination of the ray casting based on inter-frame occlusion queries is employed to optimize the overall performance of our system.

3.1. Overview

We propose to subdivide the viewing volume in viewing direction in concentric spherical shells centered at the eye **e**. The sampling positions on this shells coincide with the sampling positions of classical ray casting. These shells are equally separated by the volume sampling distance $\Delta\lambda$ and are used to reconstruct the density volume. This is done by perspectively correct splatting of the particles density footprints. The first slice is placed at λ_{min}, the minimal distance between **e** and bounding box of the particles to skip the empty space in front of the camera. This way λ is used as volume ray casting parameter and controls the density volume reconstruction, thus avoiding sampling artifacts. The method, as depicted in Fig. 2, consists of several calculation steps solely performed on the GPU. Three inner steps are performed in a loop for #*s* times and form the iterative ray casting. The whole process is controlled by the CPU.

Ray setup Ray direction **v**, maximum ray length λ_{max} based on the back-side of the particles bounding box, and the start position at the front face of the bounding box are stored

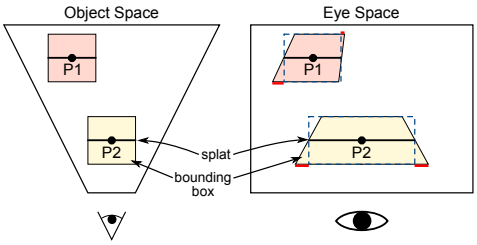


Figure 3: *Perspective correction is necessary when projecting splats from object space into image space. Otherwise the splats are not fully represented in image space.*

in buffers for each ray, which corresponds to a single image fragment. Additionally, rays might be flagged as initially inactive, if their starting positions lie outside the bounding box. Compositing our image with an opaque scene is possible by adjusting λ_{max} based on scenes depth buffer.

Position update In sub-pass number s we first update the sampling positions $\mathbf{p} = \mathbf{e} + \lambda\mathbf{v}$ with $\lambda = \lambda_{min} + s \times \Delta\lambda$. Initially inactive rays might now become active, if their new sampling position lies inside the bounding box. Rays which stepped behind the bounding box are terminated using the depth buffer. The depth test then will kill any fragments for inactive rays. After the update, a hardware occlusion query might be issued to feed-back the number of active rays to the CPU. See Sec. 3.4 for details.

Splatting To evaluate the density volume at the sampling positions $\mathbf{p}$ we splat perspectively correct footprints of the particle kernel functions into the buffer used for the current sub-pass s. Employing multiple render targets, the density is evaluated for several subsequent depths at once, reducing the required overall draw calls for the particles. Details are discussed in Sec. 3.2.

Ray casting We perform direct volume rendering using ray casting for the reconstructed density slices and composite the outcome with the result of previous sub-passes to gain the final image. Additionally, rays may be terminated by setting the depth buffer accordingly, if the accumulated opacity exceeds a threshold. See Sec. 3.3 for details.

Display result After these steps of our sub-passes are performed $s_{max}(f)$ times, a value determined by the occlusion queries of the previous frame $f-1$, the image of frame f can be displayed. The results of the occlusion queries can now be collected to determine $s_{max}(f+1)$ for the following frame $f+1$ (see Sec. 3.4).

3.2. Splatting

We follow the approach of metaballs as definition for the density field, with the density $\bar{\rho}(\mathbf{p})$ at position $\mathbf{p}$ computed from neighboring particles j by

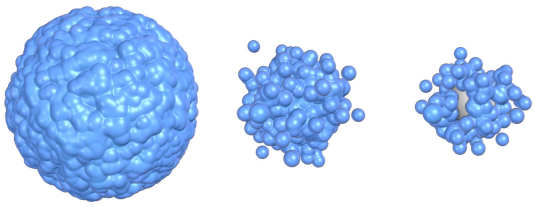
$$\bar{\rho}(\mathbf{p}) = \sum_j m_j W\left(|\mathbf{p} - \mathbf{p}_j|, h\right) , \qquad (1)$$

where $\mathbf{p}_j$ is the particle position and $W \in C^2[0,h]$ is a radial symmetric kernel function with support radius h. m_j is the particle's mass and is used to normalized the overall result to easier evaluate the iso-value (i. e. $\bar{\rho}(\mathbf{p}) = 1$). For each particle j we use a symmetric polynomial density kernel function (the smooth step function) with finite support radius h, but any other function with finite support could be used. Since we evaluate the density volume in image-space, the splatting of these kernel functions must be perspectively correct, as can be seen in Fig. 3. We employ the method of perspectively correct glyph raycasting, presented by Klein et al. [KE04]. The image space size of the point-sprite used to rasterize the splat is determined by h. The overall density $\bar{\rho}(\mathbf{p})$ is only a sum of the contributions of the individual particles $m_j W\left(|\mathbf{p} - \mathbf{p}_j|, h\right)$. Additive blending of these contribution yields the correct value. To compute several density slices at once, multiple render targets are used, with the first slice placed at λ and the remaining slices being $\Delta\lambda$ apart from each other.

Using slices with uniform depths the particles not contributing can be efficiently culled, by moving them outside the viewing frustum. This method would get far more complicated and computational expensive, if the density slices do not have uniform depth values, like in the approach of Müller et al. [MGE07], and the speed-up from culling is completely nullified by this computational load. The culling can be further optimized by sorting the particles along the viewing direction. Then the particles contributing to the current density slice are within an interval. We use a Radix Sort in CUDA to sort the indices of the particles according to their image space depth. Afterward two loops on the CPU can be used to get all indices of the first and last particles for all slices in $\mathcal{O}(n)$ with n particles. Alternatively, a parallel reduction of the indices can be performed on the GPU, requiring $\mathcal{O}(\log(n))$ for each slice, resulting in $\mathcal{O}(m\log(n))$ for m slices. Like Sec. 4.5 shows, sorting is only beneficial for large data sets.

3.3. Volume ray casting

Within a stack of density slices, generated using multiple render targets, we perform ray casting similar to single-pass volume rendering. For a continuous iso-surface, interpolation between the density slices is necessary, which is straightforward within a stack of density slices. To interpolate between stacks we reuse one slice of the previous sub-pass. For illuminating the iso-surface, gradients have to be computed. For a single particle the gradient is given by the vector from the position of the particle $\mathbf{p}_j$ to the sampling position $\mathbf{p}$, due to the radial symmetry of the kernel functions. The overall gradient is computed as summation similar to Eq. 1 by additive blending of the weighted gradients. The results of the sub-passes are composited yielding the final image. The accumulated opacity along an active ray is



$t = 5$ $t = 100$ $t = 140$

Figure 4: *Cellular signal transduction. Three time steps of the simulation are shown with the image-space approach. Signaling proteins are started at the cell membrane and then moved toward the nucleus.*

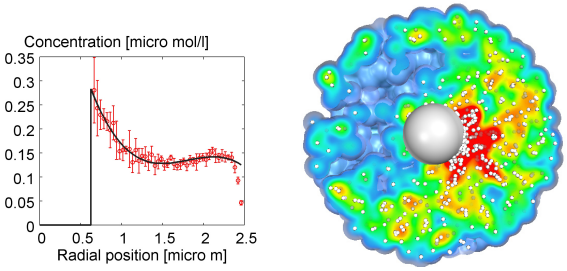


Figure 5: *Spatial effects are not covered by the radial concentration profile (left), but are visible with the volumetric visualization (right).*

also tested against a threshold to terminate the ray using the depth buffer.

3.4. Ray cast termination

Since the sub-passes of our approach are CPU controlled, but the states of the individual rays exist on the GPU, a feed-back is required. Using the depth buffer to mask inactive rays, a single hardware occlusion query can count the number of active rays using the depth test and deliver this number to the CPU. We use this information to control the number of sub-passes $s_{max}(f)$ for frame f. For the initial frame $f = 0$ the number is set to the maximum value required $s_{max}(0) = \max_{\forall \text{rays}} \frac{\lambda_{max} - \lambda_{min}}{\Delta\lambda}$ to sample the whole bounding box. Additionally, fragments for inactive rays are automatically rejected by the built-in early-z-test during position update and splatting.

The hardware occlusion queries however introduce additional computational overhead and latencies, as the results are not immediately available. To reduce the overhead, the number of occlusion queries is minimized. Six queries are issued during the last few sub-passes of the current frames, since we expect high frame-to-frame coherency. We issue them in the sub-passes $s_{max}(f) - \{50, 20, 5, 2, 1, 0\}$, what we found makes our system quite adaptive.

To hide the latencies, we exploit the frame-to-frame coherence and postpone the evaluation of the query result and the adjustment of the maximum number of sub-passes for the next frame $s_{max}(f+1)$ until the current frame f is finished and displayed. We collect the results of the queries in the order they have been issued and test the number of active rays against a threshold. The sub-pass of the first query with too few active rays will be set as $s_{max}(f+1)$. If all queries show too few active rays, the number of sub-passes is strongly reduced to $s_{max}(f+1) = s_{max}(f)/2$. If no query has few enough active rays, the number of sub-passes is increased by a fix value $s_{max}(f+1) = s_{max}(f) + 25$, limited by the overall maximum $s_{max}(f+1) \leq s_{max}(0)$.

4. Results

Our algorithm is implemented in C++ with OpenGL and GLSL shaders. For comparison with our image-space approach, we implemented the object-space method from Kolb et al. [KC05], where the density field is reconstructed as uniform grid and stored as 3D texture. The density field is subsequently visualized with standard ray casting. We conducted tests on a Windows PC with a Intel Core2 with 2.4 GHz, 2 GB RAM, and a NVIDIA GeForce 280 GTX with 1 GB. The viewport size was 512×512 and the resolution of the uniform grid was set to 256^3. The sampling distance $\Delta\lambda$ was set to $1/256$.

The ray parameters, sampling position, direction and length, are stored in floating point textures with 32 bit precision. Density values are stored in 16 bit floats. The ray parameter textures and the density slices have viewport resolution. For a 512×512 window we therefore need 8 MB for ray parameters and 12 MB for our image-based slicing with 8 slices compared to 256 MB for a full 512^3 volume. The particle positions are stored on the GPU in a vertex buffer and are updated once per frame. In the following, we show the effectiveness of our approach and discuss quality and performance results.

4.1. Signal transduction

The signal transduction process inside a cell is modeled with an agent-based Monte Carlo simulation [FKRE09]. The cell model contains signaling proteins, their reaction partners, the cytoskeleton, and the nucleus. The signal is initiated at the cell membrane and transported to the nucleus by signaling proteins.

The initial 25,000 signaling proteins move by diffuse and motorized transport toward the nucleus. Due to reactions along the signaling pathway, the number of active signal proteins decreases. Fig. 4 depicts three time steps visualized with our approach. Compared to the radial protein concentration profiles biologists use (Fig. 5 left), the visualization of the protein densities includes more details. Spatial effects, like a higher concentration on one side of the nucleus, are not visible in the concentration profile, but these effects are clearly visible when visualizing the density field (Fig. 5 right).

In Fig. 1, the volumetric visualization of signaling pro-

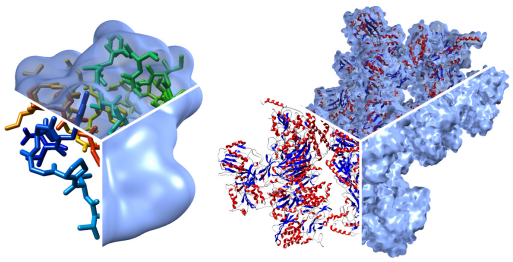


Figure 6: *Visualizations of two proteins. Left: crambin with ∼330 atoms (∼45 amino acids). Right: oxidoreductase with ∼75,000 atoms (∼10,000 amino acids). The lower images show proteins in stick representation and the proteins secondary structure, as well as the SES volume rendering. The upper images show combinations of both.*

teins is combined with the rendering of the proteins, the nucleus, and the cytoskeleton from the simulation. The protein concentration was mapped to colors from blue to red for low and high densities, respectively. The combination of both visualizations exposes the signal distribution clearly.

4.2. Protein surfaces

In biochemistry, the visualization of smooth molecular surfaces is of great importance for many applications such as docking or protein-solvent interaction. Solvent Excluded Surface (SES) and Molecular Skin Surface (MSS) are the most commonly used analytic models. However, they are computationally expensive, which makes them unfeasible for large dynamic data sets. Interactive rendering of large proteins must therefore often rely on approximations like metaballs. Our method can be used as such a fast approximation. The representation can easily be adjusted to resemble SES or MSS. The result is a smooth surface visually equivalent to ray casted MSS presented in [CLM08]. Fig. 6 shows a semitransparent SES approximation in combinations with different molecular models.

4.3. Laser ablation

The data sets shown in Fig. 7 and 8 result from molecular dynamic simulation which was coupled with finite element analysis. A pulsed laser beam heats up a solid metal block and a bulge is build.

The cutaway in Fig. 7, rendered at 1500×900, reveals the density distribution inside the material and contains roughly 250,000 particles. Glyph-based rendering of the particles (top left) hardly reveals any structures in static images.

As the bulge builds up, it breaks open at some point. Droplets of atoms are formed and expelled from the bulge leaving a crater in the metal. In Fig. 8, semi-transparent isosurfaces were visualized from the splash-like structures with

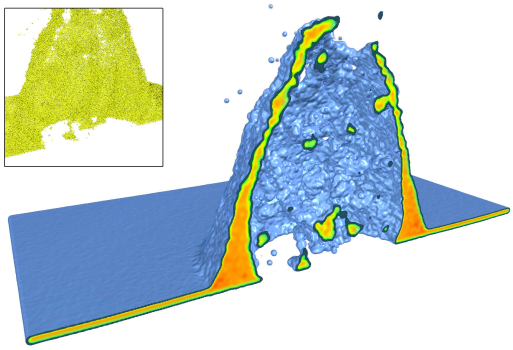


Figure 7: *Laser ablation leads to a bulge on a metal block. Particle rendering with glyphs is not sufficient to grasp the overall shape in still images (top left). Cutaway of the bulge, visualized with the image space approach at* 1500×900 *pixels, revealing the density distribution inside.*

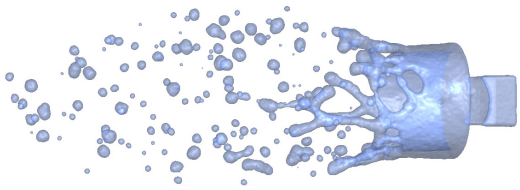


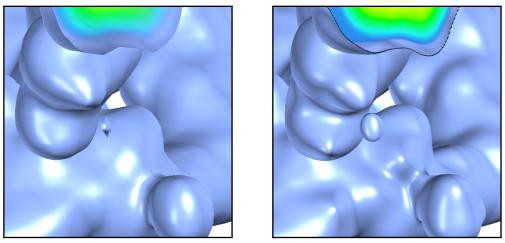
Figure 8: *When the bulge created by laser ablation breaks open, atoms are expelled and droplets form. The data set contains 742,141 particles and the image was rendered at* 1500×450 *at 0.7 fps with our image space method.*

our approach. The data set was rendered at a resolution of 1500×450 pixels and contains about 750,000 particles.

4.4. Qualitative results

In the object-space approach, problems can arise because the uniform grid has a finite resolution as illustrated in Fig. 9(a). Density splats smaller than a voxel cannot be stored completely leading to distorted or cube-like appearances of the original filter kernel. Sharp corners get smoothed out by trilinear interpolation of the uniform grid. With our approach all geometric features are conveyed in image space (Fig. 9(b)). Small features like the ellipsoid in the center of the close-up are fully captured with the image space method, whereas being deformed and barely visible in object space. The image-space technique also yields more distinct highlights. Hence, groves and ridges are depicted more clearly.

The hardware clip planes of OpenGL can be used to cut away parts of the volume. However, when particles are clipped at the intended cut planes, the final volume will show no sharp cuts because the kernel is evaluated for the complete particle footprints (Fig. 10, left). Additionally, the missing contribution of clipped particles leads to wrong results. Therefore, we perform the clipping in the fragment



(a) (b)

Figure 9: *Close-ups (800 × 800 pixels) of a small region of Fig. 7 are depicted: (a) object space, (b) image space. Ridges, grooves, highlights, and small features are more distinct in image space.*

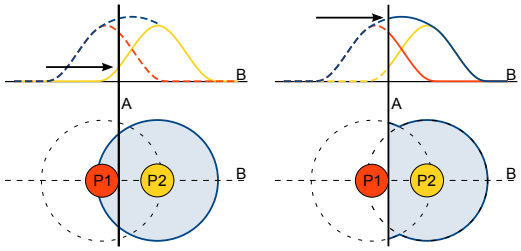


Figure 10: *If P1 is clipped, its density contribution is neglected leading to a wrong density (left). For correct volumetric clipping, P1 has to be considered (right). A denotes the clip plane, the density profiles are shown for the cross-sections B.*

program during density computation. The hardware clip planes are positioned in a way that only non-contributing particles are discarded. The density computed in Eq. 1 is set to zero if the sampling position **p** is outside the cut planes. The final result is a correct volumetric clipping as illustrated in Fig. 10, right.

4.5. Performance

A synthetic data set was used to demonstrate the scalability of our approach to time-dependent data sets with about 100,000 particles. Over 300 time steps, the number of particles increases linearly from zero. The performance was measured for this data set, examples of signal transduction (*Cell*), and two data sets from laser ablation simulation (*Bulge* and *Splat*). Table 1 shows the rendering performance for these data sets. The effect of sorting along the viewing direction becomes apparent for a large number of particles when the benefit outweighs the additional costs.

As time-dependent, real-world data set, we use the signal transduction data depicted in Fig. 4. In Fig. 11, the performance over the number of particles is shown as well as the number of particles over time. Our sliced ray casting in image space benefits from early-ray termination and outperforms the object space techniques when dealing with more than 6,000 particles. The time spent to rasterize the splats

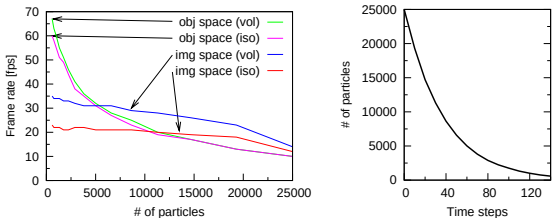


Figure 11: *Rendering performance of object-space and image-space methods for the signal transduction data set (left). Timings for direct volume rendering (vol) and isosurface rendering (iso) were measured. The number of particles is decreasing during the simulation (right).*

Data set	Particle number	Object space vol	Object space iso	Image space vol	Image space iso
Synthetic	12,276	26.51	34.00	12.58	8.57
(sorted)		23.40	–	12.74	–
Synthetic	101,664	7.28	7.71	3.23	2.62
(sorted)		9.57	–	3.98	–
Cell	25,000	10.90	10.75	14.62	12.28
	584	67.36	60.08	35.13	23.05
Bulge	509,423	–	1.16	–	0.83
(cutaway)	∼250,000	–	1.59	–	0.30
Splat	742,141	–	0.97	–	0.7

Table 1: *Rendering performance. Timings for direct volume rendering (vol) and isosurface rendering (iso) are measured in frames per second.*

corresponds with the number of proteins while the time needed to perform the multiple render passes keeps almost constant, which may explain the almost constant frame rate for low particle numbers. The rendering of the isosurface is affected by higher memory bandwidth. Instead of only one value for the density, now four values, have to be written and read for each density slice in our sliced ray casting. In object space, the bad performance might be due to the gradient computation which requires six additional texture lookups in the density texture.

5. Conclusion and future work

We present an interactive image-space ray casting technique with on-demand reconstruction of the volume data from point data, following the metaball approach. A novel method of terminating the ray casting based on hardware occlusion queries and frame-to-frame coherence is used to optimize the overall performance. The possibilities for visual cues and effects of volume rendering are superior to classical particle-based glyph rendering, and our approach does not significantly increase the memory requirements, compared to classical volume rendering. Since our technique does not require

any per-computation it is perfectly suited for time-dependent data. We achieve high image quality and good interactivity for large particle data sets. The effectiveness of our approach has been shown on several examples from simulations in systems biology, biochemistry, and physics.

We plan to further extend our algorithm and to integrate it into a visualization framework for particle data sets. Employing an hybrid image-space object-space subdivision we will implement empty-space-skipping as well as optimization of the ray setup. The reconstruction of the density volume could also be completely implemented with CUDA, however, preliminary experiments showed that the processing power of the OpenGL rasterization engine cannot be achieved by a CUDA-only implementation, due to the requirement of random-access to the memory of the density volume. However, we want to further investigate this approach, especially because of the cache optimizations of the upcoming Fermi graphics cards.

Acknowledgments

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) and the Collaborative Research Centre SFB 716 at the University of Stuttgart. The laser ablation data sets were kindly provided by Steffen Sonntag. The authors also want to thank Michael Krone for his rendering of protein data sets.

References

[Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Transactions on Graphics 1*, 3 (1982), 235–256. 2

[CLM08] CHAVENT M., LEVY B., MAIGRET B.: MetaMol: High-quality visualization of molecular skin surface. *Journal of Molecular Graphics and Modelling 27*, 2 (2008), 209–216. 6

[CSI09] CHA D., SON S., IHM I.: GPU-assisted high quality particle rendering. In *Eurographics Symposium on Rendering* (2009), pp. 1247–1255. 3

[DCH88] DREBIN R. A., CARPENTER L., HANRAHAN P.: Volume rendering. In *ACM SIGGRAPH 1988* (1988), pp. 65–74. 3

[FKRE09] FALK M., KLANN M., REUSS M., ERTL T.: Visualization of signal transduction processes in the crowded environment of the cell. In *IEEE Pacific Visualization Symposium (PacificVis '09)* (2009), pp. 169–176. 2, 5

[FSW09] FRAEDRICH R., SCHNEIDER J., WESTERMANN R.: Exploring the millennium run - scalable rendering of large-scale cosmological datasets. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1251–1258. 3

[GM77] GINGOLD R., MONAGHAN J.: Smoothed particle hydrodynamics: Theory and application to non-spherical stars. *Monthly Notices Royal Astronomical Society 181* (1977), 375–389. 2

[GRDE10] GROTTEL S., REINA G., DACHSBACHER C., ERTL T.: Coherent culling and shading for large molecular dynamics visualization. In *Eurographics/IEEE Symposium on Visualization* (2010). to appear. 2

[GRE09] GROTTEL S., REINA G., ERTL T.: Optimized data transfer for time-dependent, gpu-based glyphs. In *IEEE Pacific Visualization Symposium (PacificVis '09)* (2009), pp. 65–72. 2

[Gum03] GUMHOLD S.: Splatting illuminated ellipsoids with depth correction. In *International Fall Workshop on Vision, Modelling and Visualization* (2003), pp. 245–252. 2

[KBE08] KRONE M., BIDMON K., ERTL T.: Gpu-based visualisation of protein secondary structure. In *EG UK Theory and Practice of Computer Graphics (TPCG)* (2008), pp. 115–122. 2

[KBE09] KRONE M., BIDMON K., ERTL T.: Interactive visualization of molecular surface dynamics. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1391–1398. 2

[KC05] KOLB A., CUNTZ N.: Dynamic particle coupling for GPU-based fluid simulation. In *Symposium on Simulation Technique (ASIM)* (2005), pp. 722–727. 2, 5

[KE04] KLEIN T., ERTL T.: Illustrating magnetic field lines using a discrete particle model. In *Vision, Modelling and Visualization (VMV '04)* (2004), pp. 387–394. 2, 4

[KLR04] KOLB A., LATTA L., REZK-SALAMA C.: Hardware-based simulation and collision detection for large particle systems. In *ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware* (2004), pp. 123–131. 2

[KLR09] KLANN M. T., LAPIN A., REUSS M.: Stochastic simulation of signal transduction: Impact of the cellular architecture on diffusion. *Biophysical Journal 96*, 12 (2009), 5122–5129. 2

[KSN08] KANAMORI Y., SZEGO Z., NISHITA T.: GPU-based fast ray casting for a large number of metaballs. *Computer Graphics Forum 27*, 2 (2008), 351–360. 3

[KSW04] KIPFER P., SEGAL M., WESTERMANN R.: Uberflow: a GPU-based particle engine. In *ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware* (2004), pp. 115–122. 2

[KW03] KRÜGER J., WESTERMANN R.: Acceleration techniques for GPU-based volume rendering. In *IEEE Visualization 2003* (2003), pp. 287–292. 3

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. In *ACM SIGGRAPH Computer Graphics and Interactive Techniques* (1987), pp. 163–169. 2

[LvLRR08] LINSEN L., VAN LONG T., ROSENTHAL P., ROSSWOG S.: Surface extraction from multi-field particle volume data using multi-dimensional cluster visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1483–1490. 3

[MGE07] MÜLLER C., GROTTEL S., ERTL T.: Image-space GPU metaballs for time-dependent particle data sets. In *Vision, Modelling and Visualization (VMV '07)* (2007), pp. 31–40. 3, 4

[MGK77] MCCAMMON J. A., GELIN B. R., KARPLUS M.: Dynamics of folded proteins. *Nature 267*, 5612 (1977), 585–590. 2

[NM05] NEOPHYTOU N., MUELLER K.: GPU accelerated image aligned splatting. In *International Workshop on Volume Graphics* (2005). 3

[RE05] REINA G., ERTL T.: Hardware-accelerated glyphs for mono-and dipoles in molecular dynamics visualization. In *Eurographics/IEEE VGTC Symposium on Visualization* (2005), pp. 177–182. 2

[SEBH02] SCHMIDT-EHRENBERG J., BAUM D., HEGE H. C.: Visualizing dynamic molecular conformations. In *IEEE Visualization 2002* (2002), pp. 235–242. 3

[SSKE05] STEGMAIER S., STRENGERT M., KLEIN T., ERTL T.: A simple and flexible volume rendering framework for graphics-hardware–based raycasting. In *International Workshop on Volume Graphics 2005* (2005), pp. 187–195. 3

[TCM06] TARINI M., CIGNONI P., MONTANI C.: Ambient occlusion and edge cueing to enhance real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1237–884. 2

[vKvdBT07] VAN KOOTEN K., VAN DEN BERGEN G., TELEA A.: Point-based visualization of metaballs on a GPU. In *GPU Gems 3*, Nguyen H., (Ed.). Addison-Wesley Professional, 2007, ch. 7, pp. 123–148. 2

[vMAF*07] VAN MEEL J. A., ARNOLD A., FRENKEL D., ZWART S. F. P., BELLEMAN R. G.: Harvesting graphics power for MD simulations. *Molecular Simulation 34*, 3 (2007), 259–266. 2

[WE98] WESTERMANN R., ERTL T.: Efficiently using graphics hardware in volume rendering applications. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), pp. 169–177. 3

[Wes90] WESTOVER L.: Footprint evaluation for volume rendering. *Computer Graphics (Proceedings of SIGGRAPH 1990) 24*, 4 (1990), 367–376. 3

[ZSP08] ZHANG Y., SOLENTHALER B., PAJAROLA R.: Adaptive sampling and rendering of fluids on the GPU. In *Eurographics/IEEE VGTC Symposium on Volume and Point-Based Graphics* (2008), pp. 137–146. 2

Particle-based Rendering for Porous Media

S. Grottel[1] and G. Reina[1] and T. Zauner[2] and R. Hilfer[2] and T. Ertl[1]

[1]Visualisation Research Center (VISUS), University of Stuttgart, Germany
[2]Institute for Computational Physics, University of Stuttgart, Germany

Abstract

Particle-based modeling and simulation of granular or porous media is a widely-used tool in physics and material science to study behavior like fracture and failure under external force. Classical models use spherical particles. However, up to 10^8 polyhedral-shaped particles are required to achieve realistic results comparable to laboratory experiments. As contact points and exposed surfaces play important roles for the analysis, a meaningful visualization aiding the numeric analysis has to represent the exact particle shapes. For particle-based data sets with spherical particles, ray tracing has been established as the state-of-the-art approach yielding high rendering performance, optimal visual quality and good scalability. However, when rendering polyhedral-shaped particles, there is no issue with visual quality comparing polygon-based rendering approaches and ray casting, whereas the polygon-based approaches cause significantly lower fragment load. The paper at hand investigates the advantages and drawbacks of both approaches by analyzing the performance of state-of-the-art rendering methods employing vertex-buffer objects, hardware-supported instancing, geometry shader, and GPU-based ray casting.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; I.3.8 [Computer Graphics]: Computational Geometry and Object Modeling—Applications.

1. Introduction and Related Work

In many fields of science, including physics and material science, particle-based simulation is a well-established tool for studying material properties and material behavior under external forces. Material samples are modeled by large amounts of particles representing discrete entities from the application domain. In molecular dynamics, these are usually atoms, or mass-center-based representations of molecules, e. g. Lennard-Jones mass centers [GRVE07]. These particles have no specific shape, but a simple effective radius, and are thus visualized as spheres. Rendering these is well understood [Gum03] and high-performance algorithms are available. Ray casting has been established as the state-of-the-art approach for rendering this type of data [GRDE10] allowing for interactive visualization of several millions of spheres on standard desktop computers. For quadratic surfaces, e. g. spheres or cylinders, this approach yields best performance, scalability and visual quality, compared to alternatives like texture-based point sprites or mesh-based approaches. For arbitrarily-shaped, curved surfaces the superior visual quality of ray casting is also favorable [KHK*09].

In materials science one is often confronted with materials exhibiting complex stochastic microstructures and textures. Important examples are porous materials [Hil96]. Some classes of porous materials, such as sandstones, exhibit a granular microstructure resulting from the physicochemical processes that generated the material. Simulations of such media often start from models with spherical grains, due to the simplicity of handling interactions. However, for realistic results non-spherical particles are required. These can be modeled by composing a grain out of several spherical sub-grain particles (e. g. employing the discrete element method [JBPE99]). This approach has scalability issues when trying to achieve simulation system sizes of 10^8 grains. This is the required size to be comparable to laboratory experiments. Therefore, current simulations try to employ polyhedral-shaped particles [LBFH10], which, however, complicates the process of evaluating contact points

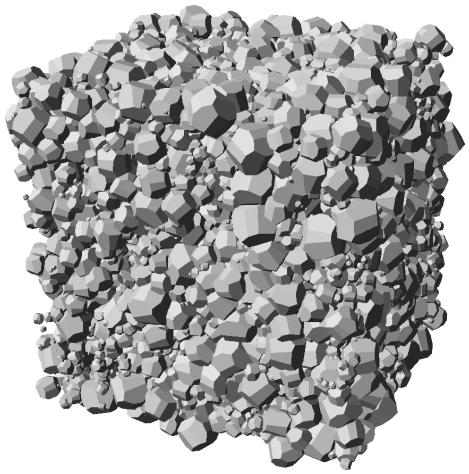


Figure 1: A small porous media sample modeled from 10 000 particles of 100 crystallite template types with 18 faces each.

and forces. To achieve consistency with numerical analysis, visualizing these particles correctly is of high importance.

The scenario of porous media, e. g. sandstone, which the work at hand specifically looks at, is closely related to visualizing granular media, as such media is also modeled from polyhedral-shaped particles, namely quartz crystallites [HZWH09]. An exact visualization of the particles is required as their shapes define the shape and amount of the surface of the media in cavities and tunnel networks exposed to surrounding media like gas or liquid (e. g. consider thick-film gas sensors [MC95]).

Rendering arbitrary polyhedral shapes is traditionally achieved by means of a polygon mesh. Although this is straightforward, it is not clear whether this is the best approach when visualizing very large data sets. On the one hand, a data set of 1 000 000 particles easily requires up to 200 000 000 triangles to be rendered (Tab. 1). On the other hand, usually there are not a million unique particle meshes, but particles are scaled and rotated instances of only several dozen particle templates. This fact can be exploited to optimize the rendering performance, allowing for interactive visualization of data sets with up to several millions of particles. These data set sizes are currently produced by simulations in the application domain. A similar approach was used by Lampe et al. [LVRH07] to visualize large proteins by rendering instances of amino acids instead of individual atoms.

The main contribution of this work is the presentation and comparison of state-of-the-art rendering techniques that yield polyhedra. The detailed performance analysis identifies the best approach based on data set size and particle complexity.

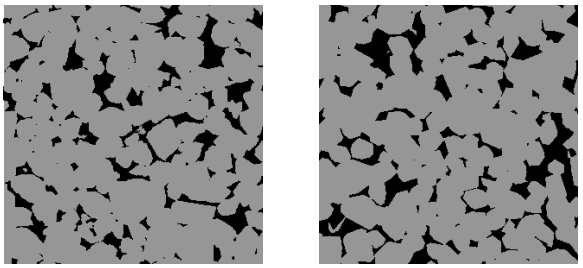


Figure 2: Comparison of 2D sections (2.25 mm × 2.25 mm) experimental data (μ-CT; **left**) with reconstructed model (**right**) of Fontainebleau sandstone. Grains are shown in gray, while space in-between is shown in black. The reconstructed model is stochastically very similar to the experimental data. The degree of stochastic similarity was measured and documented quantitatively using numerous geometric observables described in detail in [LBFH10].

2. Modeling Porous Media

Porous media may be loosely characterized as materials containing a complex system of internal surfaces and phase boundaries [Hil96]. The random appearance of the phase boundaries has lead to a variety of stochastic models [Hil02, Hil00]. Stochastic reconstruction models for porous media have been investigated extensively in [MH99, MTH00, BH99].

A fundamental drawback of stochastic reconstruction models as well as segmentation of 3D X-ray or synchrotron microtomograms obtained from scattering experiments is the representation of the microstructure on a regular (cubic) lattice [BHK*09]. Such a representation at a single, fixed resolution precludes multiscale modeling of porous media, because only a single scale can be represented with currently available data manipulation capacities. Recently this fundamental limitation of lattice-based models was overcome in stochastic continuum models [BHK*09, BØH*09, LBFH10].

Visualization and 3D-imaging of stochastic continuum models is important for identifying and modeling regions of interest in multiscale porous media, such a microporous regions with sub-micron-sized pores or microcrystalline regions with nanometer crystallites. It has never been carried out for multiscale sandstones due to the lack of suitable models on the pore scale. Here we report a first step in this direction. We present fast visualizations of continuum models for sandstones with polyhedral grains obtained from a novel molecular dynamics method of generating the stochastic point process underlying the stochastic continuum models. While the molecular dynamics method will be described elsewhere, we report here details of the rendering technique.

When visualizing media modeled by such crystallites we can exploit several factors. The rather small number of crystallite template types makes this scenario a perfect candidate

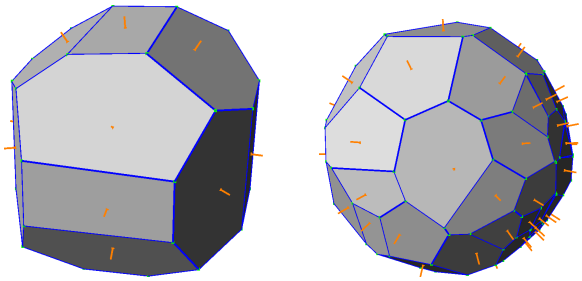


Figure 3: Two crystallites used to model the porous media; **Left:** rather uniform crystallite with 18 faces; **Right:** crystallite with 50 randomly placed faces used for performance measurements only. Orange lines show the face normals defining the tangent planes.

for instancing approaches. The flat faces of the crystallites seem to be well suited for classical mesh-based approaches, as there are no resolution concerns, neither in form of mesh tessellation nor as image-space resolution, e. g. for curved surfaces. The crystallite definition using tangent planes of a single sphere yields always completely convex crystallites as the structure can also be obtained by employing Voronoi Diagrams on spheres [NLC02]. Thus, each face of each particle also is a convex polygon within the tangent plane. This allows for the rather simple generation of a triangle-mesh representation for each particle.

3. Rendering Approaches

Particle-based rendering is originally based on graphical point primitives (i. e. `GL_POINT` in OpenGL). This approach scales very well for very large numbers of particles, but is usually restricted to simple particle types like point sprites. Ray casting with programmable graphics hardware allows for smooth particle shapes, like spheres or cylinders. For increased particle complexity or arrays of mesh instances there are several instancing techniques available. They usually employ vertex-buffer objects (VBOs), hardware-supported instancing, or programmable geometry shaders.

As has been published earlier [GRE09], modern graphics cards can easily process particle-based data sets with up to a million particles without the need of any optimized data structure. Since we are only interested in performance and scalability of the different rendering approaches described in this work, we do not apply any optimized data structure, as such would impair the scaling behavior with respect to data set sizes. For a system capable of visualizing multi-million particle data sets we can apply object-space subdivision and visibility prediction based on occlusion queries as has been previously presented [GRDE10].

All approaches share some common ideas: a particle-space coordinate system is introduced for each particle, similar to object-space, in which the geometry of the used crystallite is placed at the origin with a defined orientation. The transformation from this system to the object-space coordinate system is described by eight scalar values (3 for position, 1 for size, 4 holding an orientation quaternion). When using opaque representations, the order in which the particles are drawn can be optimized to minimize state changes of the rendering engine. This is the case when the particles are sorted based on the crystallite template type they instantiate.

3.1. VBO-based Rendering

The first approach is based on the idea to store all particle template types in graphics memory. For each particle the triangle mesh is computed in particle-space and stored within a VBO. The best-suited VBO access-mode is `GL_STATIC_DRAW` as the particle templates are not altered after creation.

For each particle type, the corresponding VBOs (vertices and normal vectors) are activated. To instantiate the particle the VBOs are drawn once (One call of `glDrawArrays` per particle). The transformation from particle-space into object-space can be implemented either by using the built-in model view matrix, or by using a simple shader program, which results in better performance due to fewer state changes. Because of the high number of OpenGL function calls this rendering approach has the highest CPU load.

3.2. Hardware-supported Instancing

The high number of function calls of the simple VBO-based rendering, which result in high CPU load, can be reduced using hardware-supported instancing. While this instancing rendering approach is very similar to the previous one—both using one VBO per crystallite template type—the placement, orientation, and scaling of the particles has to be transferred differently to the shader program when using instancing, since all particles are drawn with a single function call. The shader program must be able to determine the transformation data, which has to be uploaded to the graphics hardware as well, based on the instancing index.

We store the required transformation data in a single `RGBA` float texture (two texels per particle) for optimized texture upload, since the vertex data upload, which would originally be used for this data, is already used by the particle template VBOs. However, the maximum texture size ($8k \times 8k$) limits the number of particles of one crystallite type to 33.5 millions ($8k \times 8k/2$) to be drawn in a single call. To overcome this limit, we can simply use multiple draw calls.

3.3. Geometry Shader

Programable geometry shaders allow for using the vertex data upload for the particle data again, similar to the clas-

#Faces	#Triangles	#Vertices (unique/drawn)
4	4	4/12
10	∼ 26	∼ 15/∼ 46
20	∼ 68	∼ 36/∼ 108
50	∼ 200	∼ 98/∼ 300

Table 1: Number of triangles and vertices per crystallite for a given number of faces; the numbers vary slightly for the different crystallite templates due to the different plane cutting conditions. The two numbers of vertices show the number of *unique* vertices, required when storing the mesh data in VBOs, and the number of vertices needed to be *drawn* based on the number of triangles (relevant for the geometry shader approach).

sical approach. The idea is to upload point data and perform the crystallite template instancing through the geometry shader. We generate one shader for each crystallite type, which is similar to the idea employed by [LVRH07]. The triangle-meshes of the crystallites are stored not in VBOs but in the corresponding geometry shader's code directly. The calculation of the mesh is done on the CPU. We then generate a geometry shader code which outputs this mesh directly in normalized device coordinates. The output type `GL_TRIANGLE_STRIP` allows to efficiently render each face of a crystallite, similar to the previously described approaches.

Since the output size of each geometry shader is known, load-balancing within the graphics hardware should be possible. However, the maximum number of vertices output from the geometry shader is limited. On current graphics cards, the limit[†] of scalar values output per geometry shader invocation is 1024, resulting in a maximum number of $1024/8 = 128$ vertices (8 components since we need to set `gl_Position` and `gl_FrontColor`). Table 1 shows that crystallites with 20 faces are already very close to that limit (23-24 faces exceed the limit). However, this limit is not crucial, since 18 faces are sufficient to achieve a realistic model, as stated in section 2. Nevertheless, if more complex crystallites are required in future, the geometry shader approach will need to be modified (e. g. splitting up the crystallites into several shaders), which will introduce more state changes and additional data to be uploaded.

3.4. Ray Casting

The original particle-based visualization works with point primitives and GPU-based ray casting of implicit surfaces resulting in perspective-correct object appearance of each particle. This approach is especially beneficial for smooth quadratic surfaces, like spheres, due to the always optimal image-space resolution. However the ray casting gets more and more computationally intense the more intersections between the viewing ray and particle geometry have to be calculated.

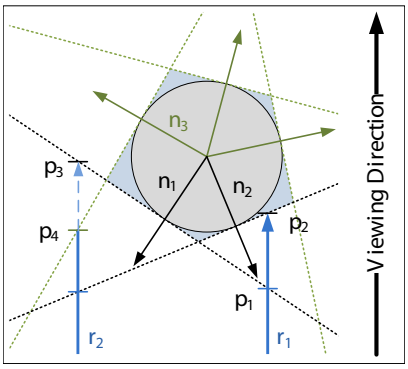


Figure 4: The principle of ray casting convex polyhedra (in 2D convex polygons). Viewing ray $\mathbf{r_1}$ hits tangent plane of normal $\mathbf{n_1}$ at point $\mathbf{p_1}$. Hit point $\mathbf{p_2}$ is the farthest front face hit point and thus the correct point. Viewing ray $\mathbf{r_2}$ would choose $\mathbf{p_3}$ this way. However the front-most back face hit point $\mathbf{p_4}$ with plane of normal $\mathbf{n_3}$ is closer to the viewer. Thus viewing ray $\mathbf{r_2}$ does not hit the polyhedron (polygon) at all.

The central idea of GPU-based glyph ray casting follows the approach of point sprites, for which OpenGL `GL_POINTS` are extended to image-space window-aligned quads. While the classical point-sprite approach places a texture on each quad, e. g. the image of a sphere, GPU-based ray casting performs a single ray casting step, also known as local ray tracing, in the fragment shader for each fragment, calculating the perspective correct rendering of any implicit surface encoded in the ray-surface intersection equation.

For the polyhedral-shaped particles used in this paper, a viewing-ray plane intersection has to be calculated for each tangent-plane of the crystallite. For each intersection, a simple dot product of the viewing ray and the plane normal specifies whether the plane was hit front-side. The correct intersection is the farthest hit of a front face. However, if the nearest hit of a back face is in front of this one, the crystallite is not hit at all (See Fig. 4). Note that no sorting of intersections needs to be performed because simple `min` and `max` operations during the hit tests are sufficient. This approach is basically similar to the work on bounding objects for ray tracing presented by Kay and Kajiya [KK86].

Similar to the idea of the geometry shader-based approach, we can render all particles using a single draw call with one shader for each crystallite template type performing the ray casting and lighting operations.

[†] `GL_MAX_GEOMETRY_TOTAL_OUTPUT_COMPONENTS`, see the geometry shader extension specification [Geo]

4. Results

All performance measurements were conducted on a machine with Intel Core I7 980X 3.33 GHz CPU, 12 GB RAM, NVIDIA GeForce GTX 480 graphics card, running Windows 7 (x64). The viewport resolution was 1024×1024. Table 2 shows all rendering performance values.

In principle, all techniques scale linearly with the number of particles. The two exceptions to this are the performance values for 10 000 particles and the values of the ray casting rendering technique. For small particle numbers the overhead of the individual methods (e. g. even back buffer clearing) becomes the limiting factor. The ray casting approach scales better with the number of particles, because it is based on fragment processing—while all other approaches are based on vertex processing—and the number of fragments per particle decreases with increasing number of particles due to the limited screen-space resolution.

For the VBO-based rendering approach there is no significant difference between rendering of crystallites with 4 to 20 faces. We presume that this is due to the high CPU load of this method as discussed in Sec. 3.1. Thus, the difference between the VBO sizes is insignificant. This is not the case when comparing the values of the 20-faced crystallite with the values of the 50-faced crystallite. As the number of faces roughly double the frame-rates are roughly halved, as would have been expected.

The hardware instancing rendering method, which follows an idea similar to the VBO-based rendering, trades the high CPU load for the additional overhead of requiring the upload of the particle data to the graphics memory as textures. Quite surprisingly, despite of the upload this approach is favorable for almost all cases, except for complex particles (20 or 50 faces) in small data sets (10 000 particles). For the small data sets the overhead of the texture upload limits the overall rendering performance of this approach. However, even in these situations the performance is still comparable to the VBO-based method.

The geometry shader is known to perform quite poorly when the number of output primitives varies strongly or when the number of output primitives is much higher compared to the number of input primitives. Since we create one shader for each crystallite template type with a fixed number of output-primitives the first aspect of varying output is not an issue. However, the second aspect of a disadvantageous ratio of input primitives to output primitives can be seen with growing number of faces per crystallite. While the geometry shader clearly outperforms all other rendering techniques for tetrahedral-shaped crystallites ($\times 2$ compared to hardware instancing and up to one order of magnitude compared to VBO-based rendering; top-most curve in Fig. 6), it quickly becomes slower as the number of output triangles increases. For 10-faced crystallites it is roughly at the same level as VBO-based rendering, but it scales better with higher numbers of particles, possibly due the CPU limitation of the VBO-based approach. For 20-faced crystallites and at least 100 000 particles the ratio of input objects to output objects is disadvantageous to such an extent that the geometry shader approach results in the worst performance. The geometry shader programs fail to compile for 50-faced crystallites as discussed in Sec. 3.3. Although it would be possible to create the whole crystallite's geometry with two or three shaders, this would require increased particle upload ($\times 2$ or $\times 3$) and an increased number of state changes due to the additional shaders. Considering the huge performance drop from 10-faced crystallites to 20-faced crystallites and the additional overheads mentioned above, it is unlikely that this approach would yield better performance than the alternatives.

The ray casting approach is the only approach considered in the work which is based on fragment-processing instead of vertex-processing. It therefore adds a dependency to the screen-space sizes of the rendered particles. For large particles this method is much slower than the vertex-based methods, which is shown by the values of the small data set (10 000 particles). However, even for the 100 000 particles data sets, particles become small enough in screen space (still 20×20 pixels) that this method yields performance comparable to the other rendering methods. For data sets of 1 000 000 particles or more, the ray casting approach scales extremely well, as the particles keep getting smaller in screen space, and the method reaches frame rates $\times 2$ to $\times 5$ faster than the alternative methods (see blue lines in Fig. 6).

5. Conclusion and Future Work

In this paper we presented four different techniques for rendering polyhedral-shaped particles which are used to model porous media. The rather straightforward VBO-based rendering has a simple implementation but does not scale well with the number of particles due to the CPU-controlling scheme. Since the modeling of porous media has similar concepts as instancing, hardware-supported instancing yields better performance than pure VBO-based rendering for medium-sized and large data sets. The geometry shader shows the best rendering performance for simple particle sizes, i. e. tetrahedra. For large data sets and complex shaped particles GPU-based ray casting highly benefits from the small particle sizes in image-space and achieves the best performance of all methods. This may not be the case for large display installations.

As a rule of thumb, for data set sizes below 1 000 000 particles the *instancing* approach yields best performance results. Rendering data sets with several millions of particles the *ray casting* approach is fastest on workstation computers. The *geometry shader* works very well with tetrahedral-shaped particles.

An optimized hybrid implementation could be obtained when choosing the different rendering approaches for different situations, even within a single rendering cycle (e. g.

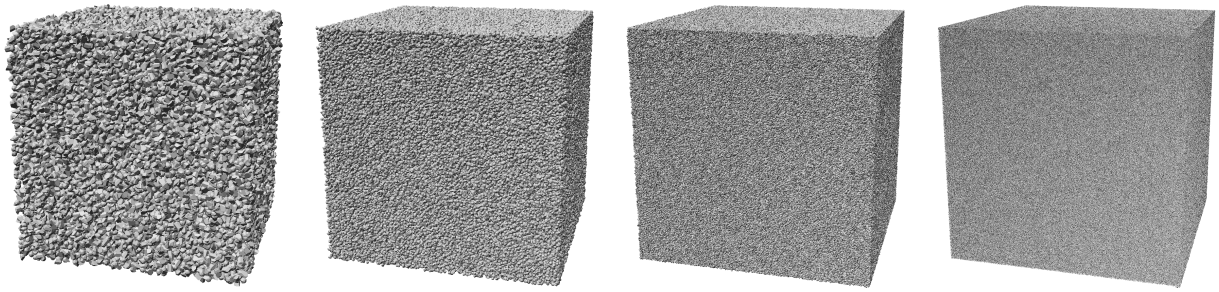


Figure 5: The data sets used for the performance measurements (Tab. 2) with (from **left** to **right**) 100 000, 1 000 000, 10 000 000, and 100 000 000 particles. The used crystallite types have always 20 faces. All rendering methods produce exactly the same images.

		#Particles				
#Faces	Technique	10 000	100 000	1 000 000	10 000 000	100 000 000
4	VBO	644.5	80.0	7.73	0.778	0.061
	Inst	1259.0	295.6	34.3	3.03	0.254
	GPUGeom	1430.1	524.8	69.0	7.51	0.747
	Raycast	181.3	81.7	25.9	4.31	0.566
10	VBO	635.5	80.1	7.7	0.758	0.064
	Inst	962.6	209.4	24.8	1.98	0.16
	GPUGeom	612.8	88.1	9.1	0.911	0.086
	Raycast	160.0	62.9	19.3	3.19	0.39
20	VBO	610.6	76.6	7.64	0.682	0.066
	Inst	592.3	105.8	11.5	1.12	0.094
	GPUGeom	174.1	19.2	1.86	0.183	0.018
	Raycast	133.8	61.0	15.2	2.11	0.25
50	VBO	315.2	38.0	3.87	0.37	0.038
	Inst	306.7	40.9	4.1	0.392	0.039
	GPUGeom	–	–	–	–	–
	Raycast	117.1	45.5	8.85	1.09	0.118

Table 2: The rendering performance values in FPS achieved by the different rendering techniques for different data sets. *#Faces* are the number of crystallite faces (not triangles). Details on the rendering techniques can be found in the corresponding subsections: *VBO* in Sec. 3.1, *Inst* in Sec. 3.2, *GPUGeom* in Sec. 3.3, and *Raycast* in Sec. 3.4. Note that the geometry shader was unable to compile for crystallites with 50 faces (see Sec. 3.3 for discussion).

using ray casting for many small particles in the background, while using instancing for the big particles in the foreground). However, such an implementation would be quite complex. Considering the fact that ray casting is always interactive and for large data sets it is even the fastest method, it is dubitable if this effort is justified.

There are some further improvements to the presented methods and additional methods we would like to investigate as future work. The geometry shader approach could be further optimized by only generating front faces. Similar to the test performed by the ray casting approach the geometry shader could perform a back-face culling which would roughly decrease the number of output primitives by a factor of two. However, assuming there were no overhead, the geometry shader would then e. g. reach the performance values of 10-faced crystallites for 20-faced crystallites, and thus, would still be slower than hardware-supported instancing for crystallites of relevant size. Nevertheless, the geometry shader approach could benefit from future hardware architectures.

Beyond the question of fast rendering the visual evaluation of material properties arises. When rendering additional information, e. g. onto the surface of each crystallite, the evaluation conditions on which rendering method to apply may change. Mesh-based approaches have the advantage of easily usable texture-coordinates, while ray casting is already performed on a per-fragment basis e. g. allowing for on-the-fly evaluation of a 3D scalar field, like a distance field to the particle surfaces. For future work we plan to analyze these aspects, not only for rendering but also for GPU-based

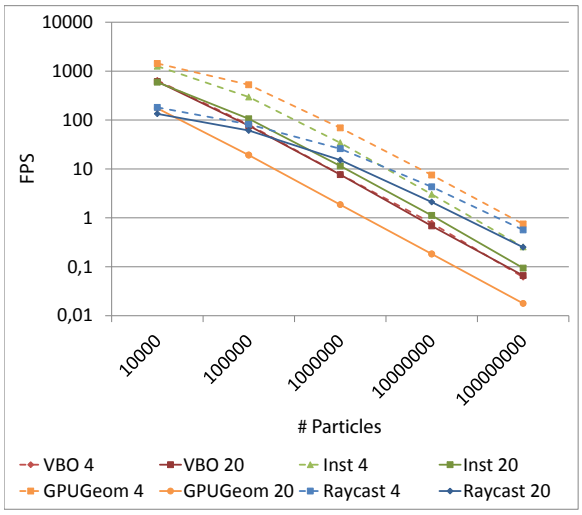


Figure 6: Rendering performance of all methods in FPS for tetrahedral-shaped crystallites (dashed lines) and 20-faced crystallites (solid lines). Geometry shader for 4-faced crystallite is fastest (orange line). Most methods decrease linearly with the number of particles, except for when rendering very few particles. Ray casting results (blue lines) decrease more slowly than linear and thus are beneficial for large data sets.

evaluation, especially when applied not to the particles themselves but to the empty space in between.

Acknowledgements

This work is partially funded by Deutsche Forschungsgemeinschaft (DFG) as part of SFB 716 projects B.3 and D.3.

References

[BH99] BISWAL B., HILFER R.: Microstructure analysis of reconstructed porous media. *Physica A 266* (1999), 307. 2

[BHK*09] BISWAL B., HELD R., KHANNA V., WANG J., HILFER R.: Towards precise prediction of transport properties from synthetic computer tomography of reconstructed porous media. *Physical Review E 80* (2009), 041301. 2

[BØH*09] BISWAL B., ØREN P., HELD R., BAKKE S., HILFER R.: Modeling of multiscale porous media. *Image Analysis and Stereology 28* (2009), 23–34. 2

[Geo] GLSL geometry shader 4 extension specification. http://developer.download.nvidia.com/opengl/specs/GL_EXT_geometry_shader4.txt. 4

[GRDE10] GROTTEL S., REINA G., DACHSBACHER C., ERTL T.: Coherent Culling and Shading for Large Molecular Dynamics Visualization. *Computer Graphics Forum 29*, 3 (2010), 953–962. http://www.visus.uni-stuttgart.de/megamol. 1, 3

[GRE09] GROTTEL S., REINA G., ERTL T.: Optimized Data Transfer for Time-dependent, GPU-based Glyphs. In *Proceedings of IEEE Pacific Visualization Symposium 2009* (2009), pp. 65–72. 3

[GRVE07] GROTTEL S., REINA G., VRABEC J., ERTL T.: Visual Verification and Analysis of Cluster Detection for Molecular Dynamics. vol. 13, pp. 1624–1631. 1

[Gum03] GUMHOLD S.: Splatting Illuminated Ellipsoids with Depth Correction. In *Workshop on Vision, Modelling, and Visualization VMV'03* (2003), pp. 245–252. 1

[Hil96] HILFER R.: Transport and relaxation phenomena in porous media. *Adv. Chem. Phys. XCII* (1996), 299. 1, 2

[Hil00] HILFER R.: Local porosity theory and stochastic reconstruction for porous media. In *Räumliche Statistik und Statistische Physik* (2000), Stoyan D., Mecke K., (Eds.), Lecture Notes in Physics, Vol. 254, Springer, p. 203. 2

[Hil02] HILFER R.: Review on scale dependent characterization of the microstructure of porous media. *Transport in Porous Media 46* (2002), 373. 2

[HZWH09] HARTING J., ZAUNER T., WEEBER R., HILFER R.: *Numerical Modeling of Fluid Flow in Porous Media and in Driven Colloidal Suspensions*. Springer, 2009, p. 349. 2

[JBPE99] JENSEN R. P., BOSSCHER P. J., PLESHA M. E., EDIL T. B.: DEM simulation of granular media-structure interface: effects of surface roughness and particle shape. *International Journal for Numerical and Analytical Methods in Geomechanics 23*, 6 (1999), 531–547. 1

[KHK*09] KNOLL A., HIJAZI Y., KENSLER A., SCHOTT M., HANSEN C., HAGEN H.: Fast Ray Tracing of Arbitrary Implicit Surfaces with Interval and Affine Arithmetic. *Computer Graphics Forum 28*, 1 (2009), 26–40. 1

[KK86] KAY T. L., KAJIYA J. T.: Ray tracing complex scenes. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM, pp. 269–278. 4

[LBFH10] LATIEF F., BISWAL B., FAUZI U., HILFER R.: Continuum reconstruction of the pore scale microstructure for fontainebleau sandstone. *Physica A: Statistical Mechanics and its Applications 389*, 8 (2010), 1607 – 1618. 1, 2

[LVRH07] LAMPE O. D., VIOLA I., REUTER N., HAUSER H.: Two-Level Approach to Efficient Visualization of Protein Dynamics. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1616–1623. 2, 4

[MC95] MARTINELLI G., CAROTTA M. C.: Thick-film gas sensors. *Sensors and Actuators B: Chemical 23*, 2-3 (1995), 157 – 161. 2

[MH99] MANWART C., HILFER R.: Reconstruction of random media using Monte Carlo methods. *Physical Review E 59* (1999), 5596. 2

[MTH00] MANWART C., TORQUATO S., HILFER R.: Stochastic reconstruction of sandstones. *Phys.Rev.E 62* (2000), 893. 2

[NLC02] NA H.-S., LEE C.-N., CHEONG O.: Voronoi diagrams on the sphere. *Computational Geometry 23*, 2 (2002), 183 – 194. 3

Short Papers

Augmented Reality Meets Industry: Interactive Robot Programming

A. Ameri E.[1,2], B. Akan[1,3] and B. Çürüklü[1,4]

[1]Malardalen University, Sweden
[2]aai08001@student.mdh.se
[3]batu.akan@mdh.se
[4]baran.curuklu@mdh.se †

Abstract

Most of the interfaces which are designed to control or program industrial robots are complex and require special training for the user and the programmer. This complexity alongside the changing environment of small medium enterprises (SMEs) has lead to absence of robots from SMEs. The costs of (re)programming the robots and (re)training the robot users exceed initial costs of installation and are not suitable for such businesses. In order to solve this shortcoming and design more user-friendly industrial robots, we propose a new interface which uses augmented reality (AR) and multimodal human-robot interaction. We show that such an approach allows easier manipulation of robots at industrial environments.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Computer Graphics]: Multimedia Information Systems—Artificial, augmented, and virtual realities

1. Introduction

For several decades large companies producing mass-market products have used industrial robots in, e.g. machine tending, welding, and palletizing. Robots are cheaper nowadays and their technology has improved significantly, yet still, in small medium enterprises (SMEs) robots are not commonly found. Even though the hardware costs of industrial robots have decreased, the integration and programming costs make them unfavorable for many SMEs. In order to make industrial robots more common within the SME sector, industrial robots should easily be (re)programmable by engineers that work in the production line at a manufacturing plant. Our goal is to give an industrial robot the ability to communicate with its human colleagues in the way that humans communicate with each other, thus making the programming of industrial robots more intuitive and easy. Consequently, a human-like interaction interface for robots will lead to an easier and richer communication between humans and robots.

Traditional way of programming industrial robots is to use the teach pendant to sequentially move the robots tool center point (TCP) through the desired points. However the traditional programming method suffers in three ways: (i) Jogging an industrial robot with 6 degrees of freedom with a joystick with two degrees of freedom is very time consuming and cumbersome; (ii) the operator doesn't get any visual feedback of the process result before the program has been generated and executed by the robot; (iii) many iterations are needed for even the simplest task [PPS+06].

Developing new methods for operating or programming robots, needs to address these problems. A visual feedback on what the robot is viewing and what it is (or will be) doing, can help the operator to get a better understanding of robot's actions and perform complex actions easier and faster. To implement the visual feedback channel, a view of the working environment is presented to user through a unified system. The system overlays visuals through augmented reality to the user and it also receives inputs and commands through a high level multi modal language. Such

† This project is funded by Robotdalen, VINNOVA, Sparbanksstiftelsen Nya, EU - European Regional Development Fund.

an approach would speed up the programming phase of the industrial robot and utilize the intuitive process knowledge of the operator. Furthermore, it can provide usability of previously learned/taught skills at higher levels.

Augmented reality (AR) is a term used for overlaying computer generated graphics, text and three dimensional (3D) models over real video stream. Virtual information is embedded into the real world, thereby augmenting the real scene with additional information. Augmented reality proved to be useful in several industrial cases, for visualizations. Regenbrecht et al. [HGW05] have applied AR in different automotive and aerospace industrial scenarios successfully. Their field of research included applying AR technology to service/maintenance, design/development and training. They conclude that there are some more steps needed until the end-users accepts this technology in their field of expertise. Olwal et al. [OGL08] used 3D optical visualization techniques to visualize the process of a CNC machine to the operator. AR also provides great opportunities for Human Robot Interaction (HRI), and has been widely used in tele-robotics, because AR allows the operator to work as if he is present at the remote working environment [FON09, MSS02, JCG+05]. However AR can be very beneficial for programming industrial robots as well, whether it is remote or local. Through wearable computers and head mounted displays it is possible to visualize and generate paths through a pointing device [PPS+06]. In their work Chong et al. [CONY09] visually tracked marker to define collision-free paths for the industrial robot to follow. Once the path is generated a virtual robot simulates the behavior of the robot on the screen.

In this paper a context dependent multi modal language, which is backed up by an augmented reality interface, is proposed. The proposed language architecture makes it possible to manipulate, pick or place the objects in the scene. Such a language shifts the focus of industrial robot programming from coordinate based programming paradigm to object based programming scheme. On the other hand the augmented reality interface will support such programming schemes by presenting the user with visual feedback on robot's plans and actions; and also allowing the user to manipulate the scene through the same interface.

The rest of the paper is organized as follows: In Section 2 we give an overview explanation of proposed system architecture and the augmented reality module. In Section 3 we present test cases and Section 4 provides discussion and conclusion.

2. System Design

The Augmented Reality (AR) module is a part of our multimodal interface for controlling and programming the robot. It acts both as a visual interface which represents visual data about the scene and robot actions to the operator. It also acts as one of the input modalities and allows the operator to manipulate the AR scene by mouse and voice commands. The AR module is an extension of the virtual reality (VR) and simulation environment described in [ACSA09]. The augmented reality environment consists of the virtual models of the physical entities that are around the robots operating range, such as other robots, work-objects, tools, work-benches, etc. The objects and the robots are defined in local coordinate systems, therefore allowing the system to represent drop zones and gripping locations in object coordinates. Parent child relations are also kept for every object and their sub-objects. When an object is put into another object or over another object, it becomes its child object, and its coordinates and orientation are represented in its parent's coordinate system, which enables the system to track the objects location. If the parent object is moved then all the child objects follow the parent object.

The AR system consists of a camera which is mounted on the robot's gripper. In order to generate the virtual simulated scene for the same view, the OpenGL camera in the simulation engine will be following the robot's camera and the output from the virtual simulation system is overlaid on the video stream from the camera, thus generating the augmented reality scene for the user. Another benefit of having the camera on the robot's gripper is the fact that robot and operator share the same view and therefore the operator can supervise the working range of the robot through a single camera. It also helps the user to understand the scene better through the robot's view. In this work the camera is used by the user to monitor the workbench. Based on the view from the camera the user gives instructions to the robot. Note that, the main focus of this work is to evaluate the multimodal AR/Speech robot control system and we are not using the camera for object recognition (although it is possible). Figure 1 shows the ABB IRB140 robot with the mounted camera on our test setup.

As mentioned before, the AR frontend acts as a modality for controlling the robot through the multimodal interface. The user is able to select objects or drop-places on the AR scene by a mouse. For example if the operator is planning to move an object from one place to another they can say 'move this' while clicking on the desired object and clicking on the target location after that. It is also possible to just move the object by selecting it and moving it to a drop-zone through drag & drop on the AR view. The operator will then be presented by the set of actions on the AR view which show robot's simulated plan for performing the command. Operator can then decide to execute or cancel the actions.

Augmented reality also acts as a mean for presenting visual feedback to the operator. All the recognized objects in the robot's field of view are presented with a green semi-transparent overlays and the selected object is presented with a red color including information about it. The information includes coordinates, orientation and any meta

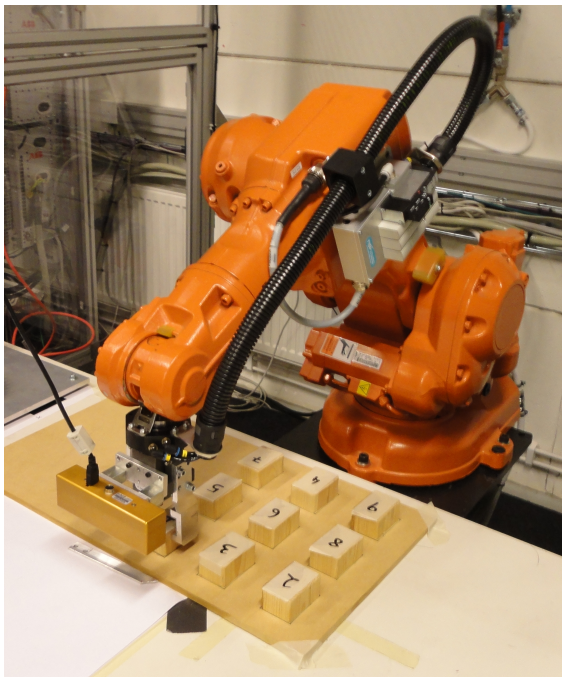


Figure 1: *Robot at work. Note that the camera is mounted on top of the gripper.*

data associated with it (weight, material, etc.). AR view also represents robot actions to the operator. The actions and path to be taken by the robot arm are shown by red wireframe cubes (gripper action) and yellow lines (movements). This allows the user to view the simulated actions and paths and revise or remove it before the actions are performed. Figure 2 shows a screenshot of the augmented reality interface.

Since the camera is mounted on the gripper of the robot, it is not always possible to visualize the paths to be taken by the robot, because the robot is not fully visible in the AR view. In such cases it is easier for the user to switch to virtual reality view, observe the path to be taken by the robot and switch back to AR view. Figure 2 shows two screenshots of the system, one in AR and the other in VR view.

3. Experiments and Results

In order to test efficiency of the augmented reality robot control interface, two tasks were defined and the participants were asked to perform the tasks by using the robot through the AR interface. The first task was simple and aimed at helping the users to get familiar with the interface. The second task was more complex and will be our main point of discussion.

3.1. Test Setup

A wooden palette is placed on the robot's workbench. The palette contains 9 drop locations which are organized in a 3x3 grid. There is also another drop location outside of the grid. There are also 9 wooden blocks which are numbered from 1 to 9. The grip locations on the blocks are pre-defined. The upper sides of the blocks are also defined as drop locations, so stacking the blocks on top of each other will be possible. The whole workbench is accessible to the robot, so it can perform tasks on the blocks and palette.

Four subjects are asked to participate in the experiments. Two of them have no prior experience in operating industrial robots and two of them are professional robot programmers. A brief description of the system is given to all the subjects orally and they all receive the same instructions.

3.2. First Experiment: Stack-up

The goal of this task is to get the subjects familiar with the system, its commands and operation logic. Thus, it consists of the simple task of stacking all the blocks on top of each other. The order of the blocks is not important in this experiment. All the subjects could perform the task in almost 5 minutes and there was no visible difference between the performances of the subjects (professional vs. unprofessional); therefore all the subjects find the instructions enough to operate the robot.

3.3. Second Experiment: Sort

This is a more complex task which challenges the subjects' ability to operate the robot: the blocks are put in a random order on different drop locations in the grid. The only empty location at start of the task is the one outside of the grid. The subjects are then asked to sort the wooden blocks inside the grid with block number one on top-left corner and number 9 on the bottom-right corner. There are of course many different ways to solve the problem, but the goal is to see if the users can use the robot to perform their plans easily. The test subjects used different approaches to solve the problem, but they all succeeded in sorting the wooden blocks in desired locations as they had planned.

During the experiment we noticed that some of the users decided to cancel some of the commands that they had given to the robot before executing them. This is due to the fact that the AR system gives immediate graphical feedback on the user's command; which in turn helps the user to visualize and review the results of that action. If the results are not similar to what the user had in his/her mind, they can simply cancel the command before performing it.

4. Conclusion and Future Work

Humans have the ability to absorb visual information with high levels of complexity in virtual environments and learn

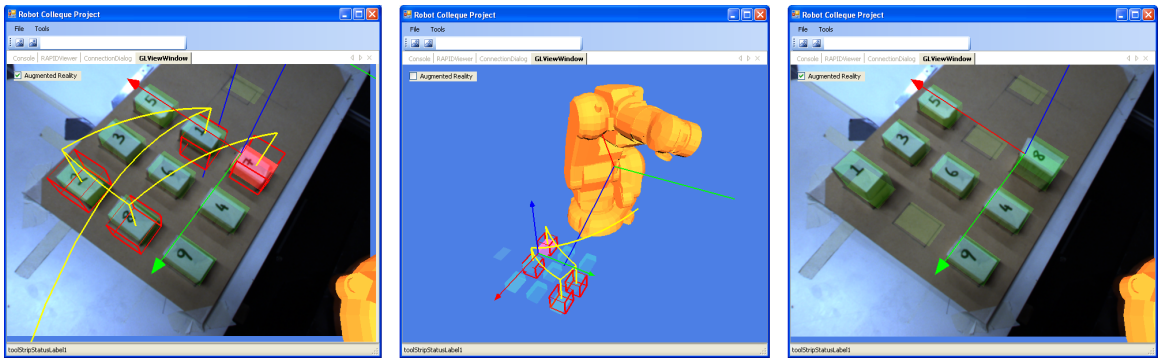


Figure 2: *Left: Screenshot from AR window. Yellow lines indicate paths and red cubes show gripper actions. Middle: The same scene at VR view. Right: AR view after performing the actions.*

from visual medias faster than other ones [IMK09]. In this work we showed that integration of working area information and streamed video from robot's camera helps humans to operate and program a robot easier and faster. We have also shown that augmented reality interfaces can act as a two way medium for interaction between the operator and the robot. In such a setup the AR interface presents graphical information to the operator in the form of live video and informative overlays. On the other hand, the operator can use the same screen to interact with the robot and its working area.

Designing and implementing richer AR interfaces which are capable of presenting and gathering more information through the same channel will be our next step towards a new generation of robot interfaces. Our final goal is to replace touch pendant controls with a more flexible device which allows communication with the robot through the interface described in this paper. We believe that these interfaces alongside multimodal communication systems can change the way we program robots todays.

References

[ACSA09] Batu Akan, Baran Curuklu, Giacomo Spampinato, and Lars Asplund. Object Selection using a Spatial Language for Flexible Assembly. Mallorca, Spain, September 2009. Emerging Technologies on Factory Automation, ETFA09.

[CONY09] J.W.S. Chong, S.K. Ong, A.Y.C. Nee, and K.Y. Youmi. Robot programming using augmented reality: An interactive method for planning collision-free paths. *Robotics and Computer-Integrated Manufacturing*, 25(3):689–701, June 2009.

[FON09] Hongchao Fang, Soh Khim Ong, and Andrew Yeh-Ching Nee. *Robot Programming Using Augmented Reality*. IEEE, September 2009.

[HGW05] Regenbrecht H., Baratoff G., and Wilke W. Augmented reality projects in automotive and aerospace industry. *IEEE Computer Graphics and Applications*, 25:48–56, 2005.

[IMK09] Kartiko I., Kavakli M., and Cheng K. The impacts of animated-virtual actors' visual complexity and simulator sickness in virtual reality applications. *In proc. CIGV*, pages 147–152, 2009.

[JCG+05] Carlos A Jara, Francisco A Candelas, Pablo Gil, Manuel Fernández, and Fernando Torres. An augmented reality interface for training robotics through the web. *Communication*, pages 189–194, 2005.

[MSS02] R. Marin, P.J. Sanz, and J.S. Sanchez. A very high level interface to teleoperate a robot via Web including augmented reality. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, (May):2725–2730, 2002.

[OGL08] Alex Olwal, Jonny Gustafsson, and Christoffer Lindfors. Spatial augmented reality on industrial CNC-machines. *Proceedings of SPIE*, 6804:680409–680409–9, 2008.

[PPS+06] T. Pettersen, J. Pretlove, C. Skourup, T. Engedal, and T. Lokstad. Augmented reality for programming industrial robots. *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 319–320, 2006.

Six-button Click Interface for a Disabled User by an Adjustable Multi-level Sip-and-Puff Switch

C. Gerdtman[1,2] and M. Lindén[2]

[1]School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
[2]Motion Control i Västerås aktiebolag, Västerås, Sweden

Abstract

There are various ways to control a computer, but today the mouse function, to move a cursor and to perform a click function, is essential. For motion-disabled persons, especially persons with high spinal cord injuries, special aids are needed to perform these interactions. In this study, an alternative way to perform a click function has been investigated, the sip-and-puff function. The purpose of the study was to improve an existing equipment used for click function, a sip-and-puff unit. An user-centred design approach was used, which means that the users took an active role in the development process, testing the prototypes in each development step and providing input for improvements. A survey of existing sip-and-puff units were performed, and from this and successive tests with the users, a sip-an-puff system was developed. As a result, a six level sip-and-puff unit to control a computer was developed. It was shown that spinal cord injured with high neck injuries could control a computer by using a multi-level sip-and-puff and that they could control the computer faster when they used a multileveled sip-and-puff instead of using a traditional sip-and-puff that only had two levels.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces—Input Devices and Strategies

1. Introduction

The computer has become an important and much used tool in our every day life. The computer is also used and appreciated tool in school, at work and at leisure. For people who are motor impaired, the computer can be the aid, which help them, handle their everyday life [LZ05]. The computer has as a technical aid given disabled persons greater opportunities to perform meaningful activities, such as writing, reading and communicating [RZ04].

Motor impaired people have with the help of Information Technology been given great opportunities to a more independent life and thereby a higher quality of life [LLS04] [JS96]. This also applies to people with high neck injuries [DHW*04].

For people with these kind of injuries, it is very hard, or sometimes impossible, to use traditional keyboards or mice, to control a computer. Therefore, people who only can use head movements to control a computer need alternative computer interfaces. Since the mouse function is regarded as more important than text input with graphical interfaces [ZM98], alternative ways to achieve a mouse function has been investigated.

For people with high spinal injuries, but who still can move their heads, several alternative steering facilities are available; steering through the face [Bra98], the head [CCKL03], the voice [IH01] and the eyes [SAC*07]. There is also a possibility to control the computer through technologies that doesn't demand any rotation of the head, like steering with your breath [Ori10], brain waves [PKSK03], the jaw [Spi10], the lips [Lif10], the pupils [EIIM10], the tongue [Nor05] and the teeth [SBGP08].

For the head controller, there are a number of different technologies to choose from to control the cursor; camera [BGF02], ultrasound [NE02], infrared [CTC*01], and motion sensors [GL06]. All techniques have their advantages and disadvantages that need to be consider on choosing the solution.

Something that is easily forgotten when choosing technical aids is that the equipment at a long term can cause cumulative trauma disorders. People who often use a traditional mouse can get troubles with their arms and hands, a so-called mouse arm [Vib10]. Such problems from upper limbs and back/neck have been reported on disabled people who used their computer equipment for some years [SS04]. These inconveniences are most likely to be related to the relatively unilateral load some custom controlled computers provides, and also on the unergonomic and static sitting position and posture, with high loads on the neck and back.

It is therefore important that the equipment is gentle for the user and fits the user's needs and abilities. One advantage is if it is possible to switch between different devices.

One way to do this is to combine different technical aids to obtain the best solution for the individual. The functionality of the mouse can be divided into the cursor control and the click part.

Mouse clicks can be achieved; with a sip-and-puff, with software that provides an automatic clicks when the cursor is held still for a certain period of time, with eye blinking or with the eyebrows.

For many disabled people with high spinal cord injuries, a head controlled mouse, moving the cursor, with sip-and-puff as a click function, is a common combination. For example, two short sips can correspond to a double click, whereas sucking and then keeping your breath can correspond to a click-draw function.

The purpose of this paper was to examine how a regular equipment to control a computer can be improved with special emphases on the user-interface. The idea was to improve a working aid based on the special needs and requirements of people with high spinal cord injuries. It was also desirable that the device should be useful for as many as possible.

2. Methods and materials

A sip-and-puff, as in Figure 1, is a well-known technical aid for the target group, which also has proven not to cause any loads for a long-term user.

A sip-and-puff unit as a click function can be combined with various systems to control the computer cursor.

An Internet survey was made over which sip-and-puff units that exists on the market, what they were used for and their functionality. Data for approximately 20 units were compiled. The sip-and-puff units are mainly used to control computers or motorized wheelchairs, but also to control different types of motor vehicles, sailing boats, environmental control, to control page-turners, and more.

Most sip-and-puff units have only the features sipping and puffing, which corresponds to two functions. There are also devices with only sipping or puffing, respectively. The more

Figure 1: *A sip-and-puff device.*

advanced had four functions, two levels of sipping and two levels of puffing [Gew10]. There are also units for lip control, that has a sip-and-puff integrated, but they only have two sip-and-puff functions [Lif10]. What mainly distinguish the units, are their attachment and their appearance. However, their functions are very similar. Some different units from the survey are listed in Table 1. As the majority only had two functions they are presented by "standard sip-and-puff" in Table 1.

Name	Sip	Puff	Other
Standard sip-and-puff	1	1	-
Gewa 5500 SB-4H	1*2	1*2	Two levels
Jouse2	1	1	Cursor control
IntegraMouse	1	1	Cursor control
Permobil 1820361	1	1	Cursor control
Sip & Puff Mouth Joystick Game Controller	3*1	3*1	4 lip buttons

Table 1: *Sip-and-puff overview.*

The investigation showed that there are many users who wished for double or triple sip-and-puff connections in order to better control the computer. Instead of a sip-and-puff with many different levels, the wanted several 2-functional sip-and-puff units parallel. Many times it was used to increase the ability to play games.

Three users of a sip-and-puff and two helping equipment prescribes were interviewed. The users, who all had spinal cord injuries, thought that the sip-and-puff as a clicking unit worked well, but they were sometimes forced to use two units' two control two different things. This meant that the users had

to have two mouthpieces in front of them. A request of only using one mouthpiece with the same functionalities came up. The prescribes recommended to have a variable sensitivity of the sip-and-puff unit, since the capacity of sip and puff can vary enormously between individuals, and especially disabled persons.

After that the actual development work of the sip and puff unit started. A user-centred design approach was used [Bö00], which meant that on the basis of the surveys and the interviews with the users, a first prototype was developed and tested by the users. The users were interviewed and opinions were addressed and after adaptation, a new test period began. This was repeated until there were no more complaints left or the remaining complaints were of minor importance.

The most common way to construct a sip-and-puff unit is to take a pressure sensor and then decide the pressure level for contact changes for sip respectively puff. That means you put the values 0 or 1. Since most pressure sensors are analogue, it corresponds scaling an analogical sensor to a digital two level function. There is no limitation in how many levels the pressure sensor itself can discriminate, but it can be difficult to control too many sip and puff levels as a user.

For maximum freedom and control of the development process, the entire sip-and-puff unit was constructed from scratch, both hardware and firmware. The sip-and-puff unit is relatively simple in its design and consists basically of a pressure sensor (from Analog Devices) and a microcontroller with flash memory and USB support (from Microchip). In this case the USB human interface device (HID) class was used. The USB HID class is used for devices like mice, keyboards and other similar devices. Therefore the sip-and-puff unit acts like a regular mouse and no USB driver needs to be installed on the computer.

The first prototype hade five levels on both the sip and puff function (10 functions). At the first field test, the pre-programmed values of the sip and puff turned out to be completely misaligned for the disabled users. A disabled person with a high neck dysfunction does not have the same muscular strength and control of his body as a person who is fully able to move. Therefore came the conclusion that the users themselves had to be able to tune in the levels.

For the second field test a computer program was developed, where the users themselves could set the values for the sip-and-puff levels. In discussions with the users, it appeared that they wanted a sliding scale that they could tune in after their own mind. It proved though that the user had some difficulties tuning in the levels themselves. A practical test showed that most users did not use all 10 functions and that they also thought it was too hard separating so many levels since the difference in air pressure became pretty small.

From prescribes side came a request to tune in the level both by a sliding scale and with entering a number. This number should be possible to save in order to reset to normal adjustment for a certain user. It is also easier to remember a setting this way. Therefore a saving function was installed so you could keep your old settings for the future, in case you changed computer.

Changes were introduced taking to consideration of the opinions that had been expressed, which resulted in the number of levels being changed to only three, as seen in Figure 2. It turned out that even three levels could be too much, so therefore a function where you could choose between 1, 2 or 3 levels of sip and puff were implemented. The sip and puff settings are independent, so it is possible to choose one sip level but three puff levels. It is also possible to completely shut off one function by simply choosing 0 sip or puff levels.

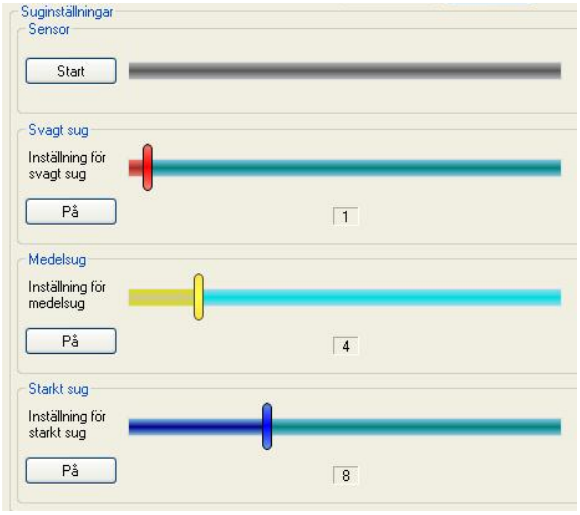


Figure 2: *Sip-and-puff software parameter settings.*

It turned out that the settings were still a problem for the user and that they wished for an easier way to set the levels. Therefore an auto setup function was installed, (the grey slide bar) where the user sips light (red), medium (yellow) or hard (blue) and then the levels are set automatically, and the puff function is set in the same way. The advantage of the auto setup is that the levels are individual adapted to the user.

When the users could adjust the equipment to the right levels on their own, a request came up to also be able to set which clicking function every level should correspond to. Therefore, also this functionality was implemented, as shown in Figure 3.

It turned out pretty quickly that the users also wanted the possibility to choose other functions than those existing on a regular computer mouse. Wishes to be able to generate keyboard strokes emerged, which was also implemented in the software. After that, script-functions were implemented. The more commands the users could make, the quicker the computer can be controlled.

Figure 3: *Click and sound setting menu.*

When more functions to choose between were available, the need to certify which sip or puff level you had chosen became apparent. Therefore sounds were added to each pressure level. Relatively quickly it came clear that you also needed the option to turn the sound off, so that feature was immediately implemented. It turned out that the users themselves wanted to decide which sounds that each level should have, and therefore this feature was introduced. There was also a request to choose your own sound. That feature was also introduced but since it needed a certain handling before you could get the sound, few actually used that function.

3. Results

A six level sip-and-puff unit to control a computer was developed through a user centred developing process. A user centred design process leads to close interaction with the users, which results in a big commitment, and good response during the field tests. The users were a part of the entire developing process that lasted for about 3 months.

The unit is connected through the computers USB port and powered through the USB port. No software needs to be installed on the computer to be able to use the device. However, software needs to be installed if you want to change the settings. The settings are saved in a memory in the unit, which makes it possible to move the unit between different computers without having to reset it.

The users can set the air pressure levels for the sip-and-puff device individually for each level and choose which feature it should have. The settings of the device are made through an installing program so that the user can set the levels easily on their own. In the installing program, the user can plug in different features and sounds to each level and save their own settings.

An analysis of the saved setting files showed (unsurprisingly) that different users had set different settings on the pressure levels. More interesting was that the users had different settings on both the sip and the puff levels. In comparison with non-disabled people, the levels were low.

The users could control the computer faster when they used a multileveled sip-and-puff instead of when they used a traditional sip-and-puff that only had two levels.

The developed sip-and-puff unit could be used with other equipment to control a computer. The sip-and-puff unit could be combined with tools that gave control of the mouse as mouse buttons to give clicking functions, and more. The device could also be used with existing tools as a complement to these. There was no problem in using the device along with traditional mice and keyboards. The subjects in this study were all pleased with their new sip-and-puff device and wanted to keep it after the trial period was over.

4. Discussion

The aim to produce a gentle tool through improving an existing product worked out well. Cumulative trauma disorders is a problem for users who spend long shifts for a long period of time in front of the computer. It is therefore essential that the tools are easy and ergonomic to use and that you can choose between different methods to control the computer. You have to consider both the individual's functional requirements but also to consider the environment. Likewise, it is advantageous if the users have different control methods available, so that they can switch between them.

A good aid helps, but it is also important to consider the user's posture during the computer work, since many of the users are in wheelchairs when the computer is being used. The manual wheelchair is originally developed to be an effective transportation tool and is not always the best seating for computer use.

Something to investigate further is how the sip-and-puff unit would work together with software for cursor control. There is a group of disabled people who just have a clicking feature and control the computer through that along with the software that provides the cursor control. Would they be helped by a multi-level sip-and-puff where they currently only use a two levelled?

5. Conclusion

In this study, a multi-level sip-and-puff unit was developed with a user centered design approach. The unit was especially developed to be used as an aid for spinal cord injured. It was found that the users could control a computer faster when they used a multileveled sip-and-puff instead of when they used a traditional sip-and-puff that only had two levels. It was also found, that the sip-and-puff unit was appreciated by the user group.

6. Acknowledgment

The authors wish to thank all the volunteers who participated this study.

This work is supported by the Swedish Knowledge Foundation (http://www.kks.se) and partially funded by the national Swedish Real-Time Systems research initiative ARTES (http://www.artes.uu.se).

References

[BGF02] BETKE M., GIPS J., FLEMING P.: The camera mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Trans. Neural Syst. Rehabil. Eng. vol. 10*, no. 1 (Mar 2002), pp. 1–10. 1

[Bra98] BRADSKI G. R.: Computer vision face tracking for use in a perceptual user interface. *Intel Technol. J. vol. 2*, no. 2 (1998), pp. 1–15. 1

[Bö00] BÖDKER S.: Scenarios in user-centred design -setting the stage for reflection and action. *Interacting with Computers vol. 13, Issue 1* (Sep 2000), pp. 61–75. 3

[CCKL03] CHEN Y. L., CHEN W. L., KUO T. S., LAI J. S.: A head movement image (hmi)-controlled computer mouse for people with disabilities. *Disabil.Rehabil. vol. 25* (Feb 2003), pp. 163–167. 1

[CTC*01] CHEN Y.L., TANG F. T., CHANG W. H., WONG M. K., SHIH Y. Y., KUO T. S.: The new design of an infrared-controlled human–computer interface for the disabled. *IEEE Trans. Rehabil. Eng. vol. 7*, no. 4 (2001), pp. 478–481. 1

[DHW*04] DRAINONI M. L., HOULIHAN B., WILLIAMS S., VEDRANI M., ESCH D., LEE-HOOD E., WEINER C.: Patterns of internet use by persons with spinalcord injuries and relationship to health-related quality of life. *Arch.Phys. Med. Rehabil. vol. 85* (Nov 2004), pp. 1872–1879. 1

[EIIM10] EBISAWA Y., ISHIMA D., INOUE S., MURAYAMA Y.: Pupilmouse: Cursorcontrol by head rotation using pupil detection technique. *Proceedings of CCCT'04* (2010), pp. 209–214. 1

[Gew10] GEWA: Sip-and-puff 5500 sb-4h. http://www.gewa.se. 2

[GL06] GERDTMAN C., LINDÉN M.: A mems-gyro based computer mouse for disabled. *Micro Structure Workshop '06* (May 2006). 1

[IH01] IGARASHI T., HUGHES J. F.: Voice as sound: using non-verbal voice input for interactive control. *Proc. of the 14th annual ACM symp. on User Interface Software and Technlogy* (2001), pp. 155–156. 1

[JS96] JACKOBSSON E., SKOGLUND K.: När det regnar manna från himlen, har den fattige ingen sked. om it och handikapp. *IT-kommisionens rapport 3/96* (1996). http://www.itkommissionen.se/doc/163.html. 1

[Lif10] LIFETOOL COMPUTER AIDED COMMUNICATION: The integramouse® is a mouse controlled by lip movements and sip and puff clicking. http://www.lifetool.at/show_content.php?sid=218. 1, 2

[LLS04] LUNDÉN G., LUNDMAN G., STRÖMAN M-L.: It för funktionshindrade och äldre. *Vinnova och HI-Hjälpmedelsinstitutet* (2004). http://www.vinnova.se/sv/Publikationer/Produkter/IT-for-funktionshindrade-och-aldre/. 1

[LZ05] LIDSTRÖM H., ZACHRISSON G.: Aktiv med dator – möjligheter för personer med rörelsehinder. *Hjälpmedelsinstitutet* (2005). http://www.hi.se/Global/pdf/2004/04345.pdf. 1

[NE02] NUNOSHITA M., EBISAWA Y.: Head pointer based on ultrasonic position measurement. *EMBS/BMES Conference '02 vol. 2* (2002), pp. 1732–1733. 1

[Nor05] NORBERG A.: Gomplatta – ett fungerande styrsätt. *Hjälpmedelsinstitutet, ITiP – IT I praktiken* (2005). 1

[Ori10] ORIGIN INSTRUMENTS: Sip/puff switch. http://www.orin.com/access/sip_puff/index.htm. 1

[PKSK03] PINO A., KALOGEROS E., SALEMIS E., KOUROUPETROGLOU G.: Brain computer interface cursor measures for motion-imparired and able-bodied users. *Proc. of the 10th Int. Conference on Human-Computer Interaction* (Jun 2003). 1

[RZ04] RYDMAN B., ZACHRISSON G.: Kommunikation genom teknik. *Hjälpmedelsinstitutet* (Aug 2004). http://www.vinnova.se/sv/Publikationer/Produkter/Kommunikation-genom-teknik/. 1

[SAC*07] SENNERSTEN C., ALFREDSON J., CASTOR M., HEDSTRÖM J., LINDAHL B., LINDLEY C., SVENSSON E.: Verification of an experimental platform integrating a tobii eyetracking system with the hifi game engine. *FOI, Swedish Defence Reaserch Agency FOI-R—2227-SE* (Feb 2007). 1

[SBGP08] SIMPSON T., BROUGHTON C., GAUTHIER M. J. A., PROCHAZKA A.: Tooth-click control of a hands-free computer interface. *IEEE Trans Biomed Eng. vol. 55*, no. 8 (Aug 2008), pp. 2050–2056. 1

[Spi10] SPINALISTIPS: Tips av och för personer med ryggmärgsskada. http://www.spinalistips.se/tips-rullstol-med-olika-styrmojligheter-911.html. 1

[SS04] SAMUELSSON K., SAMUELSSON M.: It i praktiken – ergonomi vid användning av it för personer med funktionsnedsättning. *Hjälpmedelsinstitutet* (2004). 2

[Vib10] VIBERG R.: Musarm. *Region Skåne* (2010). http://www.skane.se/templates/HealthCareInfoArticle.aspx?id=264702&catid=36235&showall=1. 2

[ZM98] ZHAI S., MACKENZIE I. S.: Teaching old mice new tricks: Innovations in computer mouse design. *Proceedings of Ergon-Axia '98.* (1998), pp. 80–83. http://www.yorku.ca/mack/axia.html. 1

Parallel Construction of Bounding Volumes

Mattias Karlsson, Olov Winberg and Thomas Larsson

Mälardalen University, Sweden

Abstract

This paper presents techniques for speeding up commonly used algorithms for bounding volume construction using Intel's SIMD SSE instructions. A case study is presented, which shows that speed-ups between 7–9 can be reached in the computation of k-DOPs. For the computation of tight fitting spheres, a speed-up factor of approximately 4 is obtained. In addition, it is shown how multi-core CPUs can be used to speed up the algorithms further.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and techniques—Graphics data structures and data types

1. Introduction

A bounding volume (BV) is a shape that encloses a set of geometric primitives. Usually, simple convex shapes are used as BVs, such as spheres and boxes. Ideally, the computation of the BV dimensions results in a minimum volume (or area) shape. The purpose of the BV is to provide a simple approximation of a more complicated shape, which can be used to speed up geometric queries. In computer graphics, BVs are used extensively to accelerate, e.g., view frustum culling, occlusion queries, selection (picking), ray tracing, path planning, and collision detection.

To speed up the computations, several attractive architectural features of modern processors can be exploited [GBST06]. In particular, single instruction mutliple data (SIMD) computation units offer an efficent mechanism to exploit fine-grained parallelism [Bik04, HOM08]. SIMD computations are well-suited to speed-up various types of multimedia and geometry processing operations [YSL98, MY02, LAML07]. In ray tracing, ray packets and/or groups of BVs can be processed simultaneously using SIMD [WBS07, DHK07].

Here we present our results from turning various sequential BV construction algorithms into vectorized methods by using Intel's Streaming SIMD Extension (SSE). We focus on three of the most widely used BV types: the axis-aligned bounding box (AABB), the discrete orientation polytope (k-DOP), and the sphere. In particular, these types of volumes are suitable for dynamic scenes, since they are fast to recompute [MKE03, LAM06]. As Sections 2–4 show, the SIMD vectorization of the algorithms leads to generous speed-ups, despite that the SSE registers are only four floats wide. In addition, the algorithms can be parallelized further by exploiting multi-core processors, as shown in Section 5.

2. Fast SIMD computation of k-DOPs

A k-DOP is a convex polytope enclosing another object such as a complex polygon mesh [KHM*98]. The polytope is constructed by finding $k/2$ pairs of parallel planes (slabs) that tightly surrounds the object. Implicitly, these planes define a bounding polytope covering the object. If needed, the enclosing polytope can be extracted by computing the intersection of the half-spaces bounded by the planes.

To construct a k-DOP, the minimum and maximum projection values of the n input points are computed for each slab direction, which are given by a pre-determined and fixed normal set N with $k/2$ normals. An efficient choice is to derive the normals from the unit cube. The three face normals are used to define an AABB (6-DOP). Then four "corner" normals can be added to define a 14-DOP, and six more "edge" normals to define a 26-DOP. Since all these normals have components in the set $\{0, \pm 1\}$, it is possible to reduce the cost of the projection operation.

The SSE instructions operate on registers containing four 32-bit precision floating point numbers. To utilize the full width of these registers, the 3D input points are first rearranged, or swizzled, into a set G of $m \approx n/4$ vertex groups

G_0	G_1	...	G_{m-1}
$X_0 = \{x_0, x_1, x_2, x_3\}$	$X_1 = \{x_4, x_5, x_6, x_7\}$	...	$X_{m-1} = \{x_{n-4}, x_{n-3}, x_{n-2}, x_{n-1}\}$
$Y_0 = \{y_0, y_1, y_2, y_3\}$	$Y_1 = \{y_4, y_5, y_6, y_7\}$	...	$Y_{m-1} = \{y_{n-4}, y_{n-3}, y_{n-2}, y_{n-1}\}$
$Z_0 = \{z_0, z_1, z_2, z_3\}$	$Z_1 = \{z_4, z_5, z_6, z_7\}$	...	$Z_{m-1} = \{z_{n-4}, z_{n-3}, z_{n-2}, z_{n-1}\}$

Figure 1: *SoA vertex layout for efficient SIMD processing. The n input vertices are stored in $m = n/4$ vertex groups. Each group G_i holds four vertices stored as 3 arrays X_i, Y_i, and Z_i.*

FIND-k-DOP_SSE
input: $G = \{G_0, G_1, \ldots, G_{m-1}\}$, $N = \{\mathbf{n}_0, \mathbf{n}_1, \ldots, \mathbf{n}_{k/2-1}\}$
output: $D = \{S, L\} = \{\{s_0, s_1, \ldots, s_{k/2-1}\}, \{l_0, l_1, \ldots, l_{k/2-1}\}\}$

1. **for each** $\mathbf{n}_j \in N$
2. $\quad P \leftarrow$ PROJECT$(G_0, \mathbf{n}_j)$
3. $\quad S_j \leftarrow P$
4. $\quad L_j \leftarrow P$
5. **for** $i = 1$ **to** $m-1$
6. $\quad$ **for each** $\mathbf{n}_j \in N$
7. $\quad\quad P \leftarrow$ PROJECT$(G_i, \mathbf{n}_j)$
8. $\quad\quad S_j \leftarrow \text{minps}(S_j, P)$
9. $\quad\quad L_j \leftarrow \text{maxps}(L_j, P)$
10. **for** $j = 0$ **to** $k/2-1$
11. $\quad s_j \leftarrow \min(S_j)$
12. $\quad l_j \leftarrow \max(L_j)$
13. $\quad s_j \leftarrow s_j / \|\mathbf{n}_j\|$
14. $\quad l_j \leftarrow l_j / \|\mathbf{n}_j\|$

Figure 2: *Data-parallel computation of a k-DOP. G is the set of SoA-arranged vertex groups and N is the set of normals or slab directions. The output $D = \{S, L\}$ is the set of intervals defining the size of the k-DOP along the slab directions.*

stored in an array with 16-byte alignment. A single vertex group $G_i = \{X_i, Y_i, Z_i\}$ holds four vertices stored in three arrays, one for each coordinate axis. In Figure 1, this vertex data structure is shown. Note that this representation is often referred to as Structure of Arrays (SoA) [GBST06]. Arrays of four floats are hereafter called 4-tuples.

Pseudocode for the computation of a general k-DOP is given in Figure 2. First all 4-tuples S_j and L_j holding the potential extremal projection values are initialized (Lines 1–4). Then the remaining vertex groups are iterated. For each group and normal, the four projection values P along the normals are computed (Lines 5–7). By using minps and maxps instructions, the current extremal values are updated without introducing branches (Lines 8–9). Finally, the extremal scalar values s_j and l_j are extracted from the resulting 4-tuples S_j and L_j, and they are also scaled to compensate for the use of non-unit length normals in N (Lines 10-14).

Note that when implementing this algorithm, the inner loop (Lines 6–9) is unrolled and each projection calculation is optimized. As an example, given the normal $\mathbf{n} = (1, 0, -1)$, and a vertex group G_i, the computation of the projections reduces to $P = X_i - Z_i$, i.e., four vertices are projected using only one parallel subtraction.

3. Fast SIMD computation of spheres

To compute tight fitting bounding spheres efficiently, the EPOS algorithm has been proposed [Lar08]. EPOS is a two-pass algorithm where, in the first pass, an initial tentative sphere is computed from a few selected extremal points of a model. The selection of these points is made by computing a k-DOP, while also storing the actual points that give rise to the minimum and maximum projection values for each slab direction. A simple extension of the algorithm in Figure 2 accomplishes this. For example, to get the appropriate indices A and B of the points with current minimum and maximum projection values, the following code replaces Lines 8 and 9:

8.1 $M = \text{cmpltps}(P, S_j)$
8.2 **if** $\text{movmsk}(M) > 0$ **then**
8.3 $\quad I = \text{andps}(C_i, M)$
8.4 $\quad A_j = \text{maxps}(I, A_j)$
8.5 $\quad S_j \leftarrow \text{minps}(S_j, P)$
9.1 $M = \text{cmpgtps}(P, S_j)$
9.2 **if** $\text{movmsk}(M) > 0$ **then**
9.3 $\quad I = \text{andps}(C_i, M)$
9.4 $\quad B_j = \text{maxps}(I, B_j)$
9.5 $\quad L_j \leftarrow \text{maxps}(L_j, P)$

With the instruction cmpltps a bit-mask M indicating the smaller projection values in P is created (Line 8.1). C_i contains indices of the vertices in group G_i. A bitwise *and* operation on the mask M and C_i, and the following maxps operation will give the new indices A_j for the current extremal points. In fact, this method eliminates the need for branching completely, since the if-statement on Line 8.2 can be removed without altering the results. However, experimental tests indicated that keeping the if-statement controlling the mask for changes gave faster execution times. The indices of the maximum points are found similarily (Lines 9.1–9-5).

After the retrieval of the k extremal points, the initial tentative sphere is computed using an exact solver which gives the smallest enclosing sphere of the selected points. Then in the second pass of the algorithm, all points of the model are considered again and the sphere is incrementally grown for each point that compromise the current bounding

EXPANDSPHERE_SSE
input: $G = \{G_0, G_1, \ldots, G_{m-1}\}, \mathbf{c}, r$
output: $\mathbf{c}, r$

1. **for each** $G_i \in G$
2. $\quad D \leftarrow$ GETSQUAREDDISTANCES$(G_i, \mathbf{c})$
3. $\quad M \leftarrow \text{cmpgtps}(D, r^2)$
4. $\quad$ **if** $\text{movmsk}(M) > 0$ **then**
5. $\quad\quad \mathbf{c}, r \leftarrow$ UPDATESPHERE$(M, D, G_i, \mathbf{c}, r)$

Figure 3: *Data-parallel algorithm that expands the initially computed sphere when necessary.*

sphere, as shown by the pseudocode in Figure 3. In Line 2, the squared distances D from the current centre to the four current points G_i are calculated. Line 3 returns the mask M indicating points outside the current sphere. The branch in Line 4 handles the case when the sphere has to be updated. The actual updating has some data dependencies, since four points are checked against the current sphere in parallel, and each point in the vertex group that is found to be outside the current sphere leads to an update of both the centre and the radius of the current sphere. This is handled by the function in Line 5.

Triangle Mesh	n_v	n_t
Frog	4010	7964
Chair	7260	14372
Tiger	30892	61766
Bunny	32875	65536
Horse	48485	96966
Golfball	100722	201440
Hand	327323	654666

Table 1: *The number of vertices n_v and triangles n_t for the polygonal meshes used for benchmarking.*

4. Results

A PC with an Intel Core 2 Quad Q8200 CPU, 2.33 GHz with 4GB of RAM running Windows 7, 32-bit version (x86), was used to measure the execution times of the presented methods. The algorithms were implemented in C++ and compiled under Microsoft Visual Studio 2008 using the default release mode. All algorithms discussed in this section were run single-threaded.

For benchmark purposes a test suit executing all algorithms were used. Each algorithm was executed for seven different polygon meshes. These meshes are visualized inside the different types of computed BVs in Figure 4. The number of points for each model is listed in Table 1. Time was measured by calculating the average execution time over 20 repetitions, executed after an initial "warm-up run" to avoid possible start-up artefacts. In this way, each algorithm has the same potential to benefit from the CPU cache and models with fewer points are likely to benefit the most. A real world application will probably not benefit from the cache to the same extent, but as the cache hierarchies in modern computers are complex, it seems not trivial to clear the cache in a comprehensive manner before each algorithm run.

As seen in Table 2, data-parallel computations of k-DOPs at the instruction level improves the computation time of the corresponding sequential algorithms significantly by a factor of 7–9. Clearly, the obtained speedups are better than the ideal speedup of 4 for 4 floating point wide SIMD registers. One probable reason is the branch elimination provided by the minps and maxps instructions in the vectorized solution.

A peculiar exception to the obtained high performance, however, is the AABB computation of the hand model, where a speed-up factor of "only" 4.4 is reached. The degraded speed-up in this case appears to have a connection to L2 cache misses on the used test machine.

The computation times of the spheres given in Table 3 show less improvement and reaches a quite consistent speedup factor of approximately 4. The reasons are due to data dependencies and the extra computation needed to retrieve the index of each extremal point spanning the k-DOP planes. Besides the EPOS algorithm, Ritter's algorithm [Rit90] and a developed SIMD version of it are also included for comparison. Ritter's method is very fast, but it consistently computes larger spheres. Also, note that the EPOS-6 method using SSE seems to reach very close to the high performance of the SSE version of Ritter's method.

The quality of the computed spheres is given in Table 4. The radius differences between the corresponding sequential and SSE methods derive from the fact that several vertices can have identical projections and, therefore, the vertices selected as extremal may vary across different implementations of the algorithms.

5. Multi-core parallelization

By utilizing multi-core processors in addition to SIMD computation another level of parallelism can be exploited. This is straightforward when computing AABBs and k-DOPs, since each thread can compute an optimal sub-volume of a subset of the input points, and then after all points have been processed all the sub-volumes can be merged sequentially into the final optimal BV. The results of using this strategy for computing 26-DOPs are given in Table 5.

Multi-threading was implemented by using OpenMP and the previous mentioned quad-core machine was used. As can be seen, for the five last input models with more than 30000 points, this gives a further speed-up in the range 1.0-1.8 using 2 cores and between 2.2–3.5 using 4 cores. However, since there is a start-up cost for multi-threading, the computation of the 26-DOP becomes slower for the two simplest

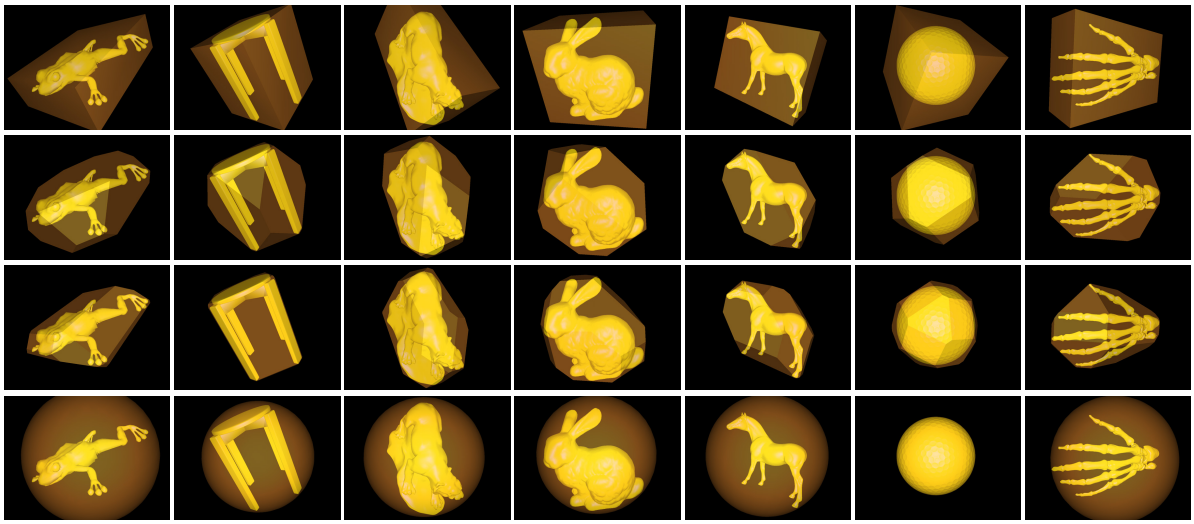


Figure 4: *Visualizations of the polygon meshes used as data sets and the resulting AABBs (row 1), 14-DOPs (row 2), 26-DOPs (row 3), and spheres (row 4). The spheres were computed using EPOS-26.*

	FindAABB			**Find-14-DOP**			**Find-26-DOP**		
Point set	Seq	SSE	*S*	Seq	SSE	*S*	Seq	SSE	*S*
Frog	0.028	0.003	9.11	0.088	0.010	9.23	0.160	0.018	8.78
Chair	0.048	0.006	8.68	0.142	0.017	8.25	0.257	0.032	7.97
Tiger	0.205	0.024	8.64	0.595	0.073	8.20	1.079	0.138	7.84
Bunny	0.217	0.025	8.62	0.634	0.077	8.27	1.152	0.145	7.95
Horse	0.330	0.038	8.72	0.965	0.114	8.49	1.755	0.214	8.20
Golfball	0.670	0.080	8.34	1.956	0.240	8.17	3.548	0.448	7.93
Hand	2.260	0.514	4.40	6.482	0.910	7.12	11.721	1.588	7.38

Table 2: *AABB and k-DOP execution times in ms and speed-ups S.*

	EPOS-6			**EPOS-14**			**EPOS-26**			**Ritter**		
Point set	Seq	SSE	*S*	Seq	SSE	*S*	Seq	SSE	*S*	Seq	SSE	*S*
Frog	0.071	0.021	3.32	0.132	0.036	3.68	0.207	0.056	3.69	0.053	0.019	2.78
Chair	0.120	0.033	3.67	0.223	0.056	4.00	0.338	0.091	3.73	0.088	0.029	3.02
Tiger	0.491	0.114	4.31	0.889	0.190	4.67	1.384	0.299	4.63	0.365	0.111	3.30
Bunny	0.524	0.123	4.25	0.949	0.203	4.67	1.474	0.317	4.65	0.389	0.119	3.27
Horse	0.786	0.190	4.13	1.428	0.324	4.40	2.227	0.509	4.37	0.595	0.198	3.00
Golfball	1.607	0.367	4.38	2.915	0.616	4.73	4.538	0.958	4.74	1.208	0.365	3.30
Hand	5.383	1.444	3.73	9.668	2.245	4.31	15.047	3.339	4.51	4.055	1.442	2.81

Table 3: *Sphere execution times in ms and speed-ups S.*

		EPOS-6		**EPOS-14**		**EPOS-26**		**Ritter**	
Point set	Optimal	Seq	SSE	Seq	SSE	Seq	SSE	Seq	SSE
Frog	0.59903	0.61349	0.61349	0.60019	0.60019	0.59903	0.59903	0.65965	0.65040
Chair	0.63776	0.69474	0.68974	0.64359	0.64359	0.63792	0.63793	0.73014	0.72789
Tiger	0.51397	0.52531	0.52531	0.51507	0.51507	0.51507	0.51507	0.53835	0.53835
Bunny	0.64321	0.65017	0.65017	0.64423	0.64423	0.64415	0.64415	0.67694	0.67694
Horse	0.62897	0.63023	0.63023	0.62899	0.62899	0.62897	0.62897	0.63476	0.63476
Golfball	0.50110	0.50155	0.50154	0.50145	0.50145	0.50114	0.50114	0.51531	0.50350
Hand	0.52948	0.52949	0.52951	0.52949	0.52951	0.52949	0.52950	0.52949	0.52951

Table 4: *The optimal radius and the radii computed by the tested bounding sphere algorithms for each point set.*

Point set	1 core	2 cores	4 cores
Frog	0.018	0.079	0.030
Chair	0.032	0.086	0.037
Tiger	0.138	0.129	0.060
Bunny	0.145	0.143	0.065
Horse	0.214	0.178	0.081
Golfball	0.448	0.297	0.140
Hand	1.588	0.897	0.452

Table 5: *Parallel computation of 26-DOPs using both multiple cores and SSE. Execution times are in ms.*

models. Similar results were obtained when multi-threading was used in the computation of AABBs and 14-DOPs, but since these algorithms involves less work per iteration in the main loop, the start-up cost affected the algorithms slightly more. Therefore, for simple points sets, a scheduling mechanism is needed that can divide the work of computing several BVs at a time among the available cores.

Creating a multi-threaded version of the EPOS-algorithm requires some additional thought. The vertices spanning the slabs of a k-DOP are easily found by a simple extension of the multi-threaded k-DOP computation algorithm. But it is not clear how to best grow the sphere in the last phase of the algorithm efficiently, since it involves updating both the center point and radius of the sphere which introduces data dependecies among the threads. The simplest approach seems to be to keep the initially determined centre point of the sphere stationary, thereby changing the quality of the computed spheres slightly. In this way, the multi-threaded computation of the required radius in the last pass of the algorithm becomes efficient and straightforward. Another simple solution would be to search for all points violating the initially computed sphere using multi-threaded loop parallelization, but instead of updating the center and radius of the sphere repeatedly during the search, the points are only stored in a list. Then the necessary updates of the sphere can proceed afterwards by processing this list sequentially.

6. Conclusions and future work

Clearly, the presented algorithms speed up the BV computation substantially. Since our BV computation scheme generalizes trivially to wider SIMD registers, further significant speed-ups can be expected for future CPUs. For example, the Intel Advanced Vector Extensions (AVX) is supported in forthcoming Intel 64 processors. AVX provides support for 256-bit wide SIMD registers, which means eight 32-bit precision floating point numbers can be processed in parallel. Similarly, the number of cores per CPU will continue to increase and this will also benefit the presented algorithms.

There are several areas worth investigating further. For example, it is expected that parallel BV computation methods can be utilized to make bounding volume hierarchy (BVH) construction faster. Many top-down BVH construction methods repeatedly call a BV computation algorithm for various subsets of the input points. Interestingly, Wald et al presents a parallel BVH building method targeting modern multi-core architectures [WIP08]. As an alternative to CPU-based approaches, it would also be interesting to develop highly parallel GPU-based algorithms for BV and BVH computation, since GPUs offer more concurrency and higher bandwith [LGS*09].

References

[Bik04] BIK A. J. C.: *The Software Vectorization Handbook*. Intel Press, 2004. 1

[DHK07] DAMMERTZ H., HANIKA J., KELLER A.: Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays. *Computer Graphics Forum 27*, 4 (2007), 1225–1233. 1

[GBST06] GERBER R., BIK A. J. C., SMITH K. B., TIAN X.: *The Software Optimization Cookbook, 2nd Ed.* Intel Press, 2006. 1, 2

[HOM08] HASSABALLAH M., OMRAN S., MAHDY Y. B.: A review of SIMD multimedia extensions and their usage in scientific and engineering applications. *The Computer Journal 51*, 6 (2008), 630–649. 1

[KHM*98] KLOSOWSKI J. T., HELD M., MITCHELL J. S. B., SOWIZRAL H., ZIKAN K.: Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (1998), 21–36. 1

[LAM06] LARSSON T., AKENINE-MÖLLER T.: A dynamic bounding volume hierarchy for generalized collision detection. *Computers & Graphics 30*, 3 (2006), 450–459. 1

[LAML07] LARSSON T., AKENINE-MÖLLER T., LENGYEL E.: On faster sphere-box overlap testing. *journal of graphics tools 12*, 1 (2007), 3–8. 1

[Lar08] LARSSON T.: Fast and tight fitting bounding spheres. In *Proceedings of The Annual SIGRAD Conference* (November 2008), Linköping University Electronic Press, pp. 27–30. 2

[LGS*09] LAUTERBACH C., GARLAND M., SENGUPTA S., LUEBKE D., MANOCHA D.: Fast BVH construction on GPUs. *Computer Graphics Forum 28*, 2 (2009). 5

[MKE03] MEZGER J., KIMMERLE S., ETZMUSS O.: Hierarchical techniques in collision detection for cloth animation. *Journal of WSCG 11* (2003), 322–329. 1

[MY02] MA W.-C., YANG C.-L.: Using intel streaming SIMD extensions for 3D geometry processing. In *PCM '02: Proceedings of the Third IEEE Pacific Rim Conference on Multimedia* (2002), Springer-Verlag, pp. 1080–1087. 1

[Rit90] RITTER J.: An efficient bounding sphere. In *Graphics Gems*, Glassner A., (Ed.). Academic Press, 1990, pp. 301–303. 3

[WBS07] WALD I., BOULOS S., SHIRLEY P.: Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Transactions on Graphics 26*, 1 (2007). 1

[WIP08] WALD I., IZE T., PARKER S. G.: Fast, parallel, and asynchronous construction of BVHs for ray tracing animated scenes. *Computers & Graphics 32*, 1 (2008), 3–13. 5

[YSL98] YANG C.-L., SANO B., LEBECK A. R.: Exploiting instruction level parallelism in geometry processing for three dimensional graphics applications. In *Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture* (1998), pp. 14–24. 1

On the Quality of Point Set Triangulations based on Convex Hulls

Peter Jenke, Anders Hast, Stefan Seipel

Abstract

In this paper we describe a method for directly generating triangle strips from unstructured point clouds based on onion peeling triangulation (OPT). It is an iterative reconstruction of the convex hulls of point clouds in the 2D plane, and it uses pairs of subsequent layers to establish triangle strips. We compare the obtained triangulations with the results of Delaunay triangulations in terms of the distribution of the symmetry of obtained triangles and in regard to the number of polygons/vertices emitted. Our initial results show that onion peeling is a straightforward method to directly obtain large triangle strips of point clouds. As expected, the triangulation is not as well behaved as in Delaunay-triangulation [VK07]. In terms of triangle complexity and average strip length OPT is a very favorable triangulation alternative which also lends suitable for the triangulation of 3D point clouds.

Keywords: Surface reconstruction, triangle strip, convex layers, rotating calipers

1. Introduction

The triangulation of large unstructured point clouds is a relevant issue in many applications that use e.g. range imaging data. The result of any such point triangulation should be a well behaved tessellation of the object surface, and ideally it should minimize vertex data flow in the rendering pipeline. This is usually accomplished by constructing triangle strips [AHMS94]. Several methods for constructing triangle strips from a point cloud are known: Some of them operate directly on the point set; others work indirectly, by first computing a triangulation of the point set (usually a mesh) and then applying a stripification algorithm on the result. A method which creates triangle strips from a mesh is presented by Gopi and Eppstein [GE04]: This algorithm searches for a perfect matching in the dual graph of the given triangulation. By the matching, every triangle in the mesh is paired with exactly one of the adjacent triangles. From these pairs, a cycle of connected triangles through the mesh is constructed; the result is a triangle strip. Boubekeur et al. propose a method called surfel strips [BRS05], which aims at producing a number of small triangle strips. This method processes the point set directly, whereby it is partitioned into a number of subsets, using an Octree. For a subset of points, a closest distance plane is fitted upon which points are projected. The actual triangulation is performed by Delaunay triangulation, followed by subsequent stripification.

Arkin presents a method, which operates directly on the point set [AHMS94]. An initial triangulation is created by connecting some interior point with points on the convex hull. Any non-empty triangle of this triangulation is then subdivided by choosing a point inside this triangle and connecting it with the vertices of the current triangle. This process is repeated until all triangles are empty.

2. Triangle strips from Convex Hulls

The convex layers of a planar point set [Cha85] are constructed by incrementally computing the convex hull of the set, calculating the set difference between the set and the hull and computing the hull for the remaining points. This process (also known as onion peeling) is repeated until the set difference is empty.

For creating triangle strips from a set of convex layers, a method based on the rotating calipers [Tou83] is used: Given a convex polygon P. Two parallel lines of support l_1, l_2 through two points p_1, p_2 of P are rotated until both touch another point: p_1 touches p_1' and p_2 touches p_2' (figure 1). Then the angles by which the lines were rotated are compared: The smaller one determines the new direction of the calipers. The lines are moved to the next point pair and the process is repeated until p_1, p_2 are reached again. The algorithm for triangulating an annulus of a set of convex layers works as follows [Tou86]: An annulus has an outer hull A and an inner hull B. The algorithm creates a sequence of

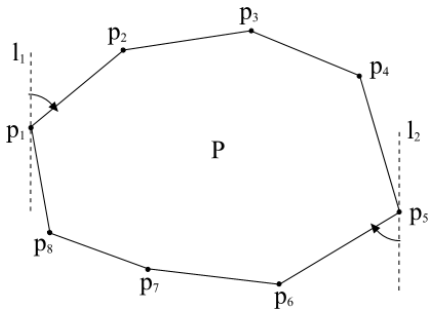


Figure 1: *Rotating calipers.*

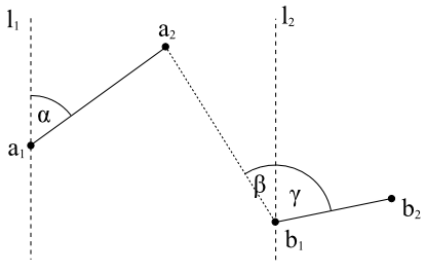


Figure 3: *Triangulating annuli:* $\beta \leq 0$ *and* $\alpha < \gamma$.

points, describing the edges of the triangulation where one point is member of A and the other belongs to B. This set of edges is denoted as R.

As a first step, the leftmost points in A and in $B - a_1$ and b_1 – are found. Two vertical lines of support are defined: l_1 goes through a_1, l_2 through b_1. Three angles are together the criterion, to determine between which points the next edge of the triangulation is placed (figure 2):

- α - enclosed by the line a_1 - a_2 and l_1
- β - enclosed by the line b_1 - a_2 and l_2
- γ - enclosed by the line b_1 - b_2 and l_2

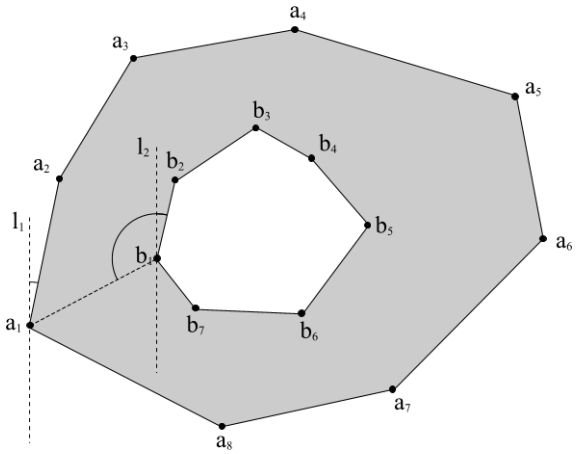


Figure 2: *Triangulating annuli: Startconfiguration.*

The next edge of the triangulation is chosen depending on the angles α, β and γ: If $\beta \leq 0$, the point which can be 'hit' from a_1 is a_2 and from $b_1 - b_2$. The new edge depends on the first touched point: If $\alpha < \gamma$, l_1 hits a_2 before l_2 hits b_2 and the next edge is $a_2 - b_1$ (figure 3). In this case, the rotation center in A is moved to a_2, i.e. l_1 is in the next step rotated around a_2. The line l_2 is still rotated around b_1. The slope of both lines is now the same as the slope of line $a_1 - a_2$. If $\alpha > \gamma$, b_2 is hit first by l_1. Now, the triangulation takes the opposite direction – the new edge is $a_1 - b_2$ and the center of rotation for l_2 moves from b_1 to b_2. In the third case is $\alpha = \gamma$; both a_2 and b_2 are hit at the same time by l_1 and l_2, respectively. Two new edges are added ($a_2 - b_2$ and $b_1 - a_2$) and the center of rotation is moved for both lines.

If $0 < \beta$, the triangulation depends on the angles β and γ. The line l_2 can in this constellation reach a_2 and b_2, and because l_1 is left from l_2, it cannot hit any point at all. As first case, consider $\beta < \gamma$ (figure 4): The first point, which l_2 reaches, is a_2. Here, the rotation is stopped and the new edge $a_2 - b_1$ is added to the triangulation. The rotation center in A is moved to a_2. If $\beta > \gamma$, the first reached point is b_2. The new

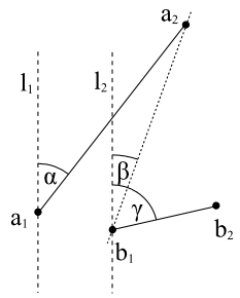


Figure 4: *Triangulating annuli: First case -* $\beta > 0$.

edge $a_1 - b_2$ is added and the rotation center in B is moved to b_2. The last possibility here is $\beta = \gamma$; l_2 hits a_2 and b_2 at the same time. As new edges are $a_1 - b_2$ and $a_2 - b_2$ added and the center of rotation is moved for both lines.

This process is repeated until both startpoints are reached again. In figure 5, a summary of the algorithm is given.

3. Evaluation approach

To assess the quality of the onion peeling triangulation algorithm (OPT) we generated two types of randomly distributed point sets using GNU Octave. The first is a uniform distribution of points within a rectangular area. In the second data set the point density follows a normal distribution from the center to the periphery. We created point sets with varying

Algorithm `TriangulateAnnulus`:
Let A and B be the points of the outer and the inner hull, respectively and T the triangulation of the annulus.

1. Choose a point $a_1 \in A$ and its neighbor $b_1 \in B$ as pair of start-points.
2. Add $a_1 - b_1$ to T.
3. Create two lines of support as vertical lines through $a - l_1$ and through $b - l_2$.
4. Compute the angles α, β and γ.
5. Add vertices and edges to the triangulation and move the center of rotation for the support lines, depending on the angles:
   - If $\beta \leq 0$:
     - If $\alpha < \gamma$: Add edge (a_{i+1}, b_i), move l_1 to a_{i+1}.
     - If $\alpha > \gamma$: Add edge (a_i, b_{i+1}), move l_2 to b_{i+1}.
     - Else: Add edges (a_{i+1}, b_{i+1}) and (a_{i+1}, b_i), move l_1 to a_{i+1} and l_2 to b_{i+1}.
   - Else:
     - If $\beta < \gamma$: Add edge (a_{i+1}, b_i), move l_1 to a_{i+1}.
     - If $\beta > \gamma$: Add edge (a_i, b_{i+1}), move l_2 to b_{i+1}.
     - Else: Add edges (a_i, b_{i+1}) and (a_{i+1}, b_{i+1}), move l_1 to a_{i+1} and l_2 to b_{i+1}.
6. Repeat from 4., until a_1 and a_2 are reached.

Figure 5: *Algorithm for triangulating annuli.*

number of points for both types of distributions and triangulated the point-set using the above described triangulation algorithm. For comparison we use a Delaunay triangulation of the same dataset using GNU Octave. For quality assessment the Delaunay triangulation (DT) can be considered an ideal standard as it maximizes the minimum angle of all angles of a triangle and tends to preserve a well behaved triangulation.

To quantify the quality of a triangulation, we define a symmetry metric as the ratio between the length of the shortest edge and the length of longest edge in a triangle:

$$SM = \frac{E_{min}}{E_{max}}$$

In a perfect tessellation of a surface the symmetry metric SM for all triangles is 1. However, given a point set representing a particular surface, SM will in practice vary across a range from close to zero (extremely skinny triangles) to one (equilateral triangle). To characterize the properties of a specific triangulation algorithm, we determine the distribution function of SM as approximated by the histogram $H(SM)$ (figure 6) for all triangles in a triangulation of a given set of points.

4. Results and discussion

We performed evaluations for a number of randomized point sets ranging from 100 to 3200 points. There were no significant differences in the graphical results nor in the statistics

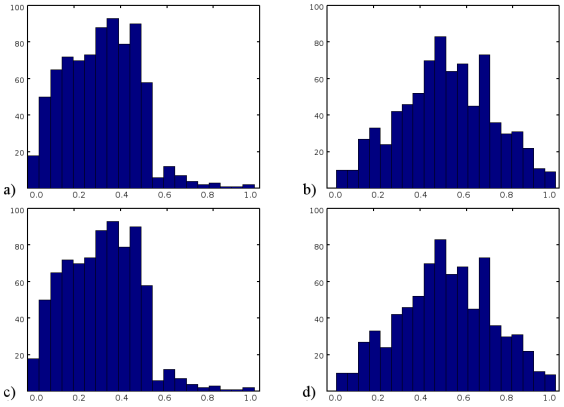


Figure 6: *Distributions of triangle symmetry scores. Upper row: Uniform point distribution, (a) OPT, (b) Delaunay. Lower row: Normal point distribution, (c) OPT, (d) Delaunay.*

for various point set sizes. Therefore we present here the results for the 400-point data set as a representative. Figures 7 and 8 show graphically the results of the triangulations using OPT and DT. DT yields, as expected, to a much more regular triangulation; whereas the onion peeling algorithm tends to produce more skinny triangles with a pronounced circular topology. The distribution of triangle symmetry score, shown in figure 6 suggests a normal distribution for the ideal case of the Delaunay triangulation with a mean triangle symmetry of 0,517 for the uniformly distributed point sets and 0,513 for the normal distributed points, respectively. The symmetry distribution for the onion peeling method is clearly skewed with a bias towards more elongated triangles. The corresponding mean triangle symmetry scores for OPT were 0,310 and 0,314. We had expected this result; it also corresponds to what has been suggested earlier [VK07].

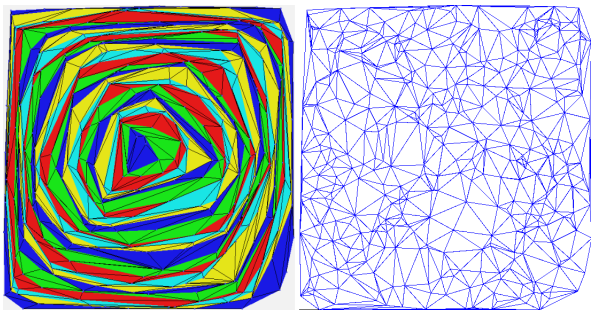


Figure 7: *Uniform point distribution: Left OPT, right DT.*

An interesting finding of our analysis comes from the descriptive statistics of the resulting triangulations. Table 1 summarizes the numbers of triangles and number of strips generated by OPT as well as the number triangles generated

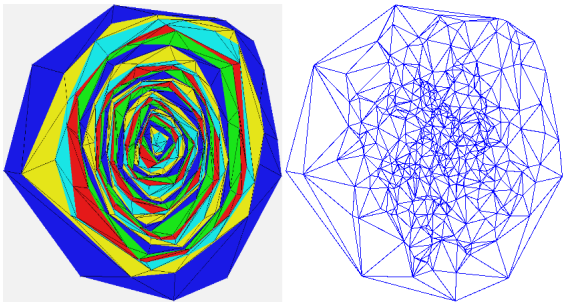


Figure 8: *Normal point distribution: Left OPT, right DT.*

by DT for increasing point set sizes. Although there are significant differences in the shape of the generated triangles, the absolute number of generated triangles is almost identical for both methods regardless of the number of points in the point set (compare row 1 and 4 in table 1). The average number of triangles per triangle strip in OPT (table 1, row 5) ranges from 21 for the smallest point set to 61 in the largest point set. In other words: The average number of vertices to be processed per triangle is as low as 1,09 for the smallest point set and decreases to only 1,03 for the largest point set. We conclude that OPT is a straightforward triangulation method which is certainly favorable as long as triangle shape symmetry is not an issue, considering that the method in itself generates triangle strips with a low average vertex count per triangle.

Number of points	100	200	400	800
#Triangles OPT	189	380	786	1590
#Strips	9	17	26	40
#Triangles Delaunay	187	380	783	1583
#Triangles / strip	21	22.35	30.23	39.75
#Vertices OPT	207	414	838	1670
#Vertices Delaunay	561	1140	2349	4749
#Vertices / triangle	1.095	1.089	1.066	1.050

Number of points	1600	3200
#Triangles OPT	3187	6389
#Strips	9	17
#Triangles Delaunay	3179	6377
#Triangles / strip	47.57	61.43
#Vertices OPT	3321	6597
#Vertices Delaunay	9537	19131
#Vertices / triangle	1.042	1.033

Table 1: *Descriptive statistics for triangulations.*

Although the OPT method as described and evaluated here is a 2D algorithm, it appears as a promising triangulation method for point set surfaces in 3D, as well. Figure 9 shows a triangulation of the Bunny point set. To that end, surface points have been partitioned in 3D using an octree-based subdivision scheme. Point sets in each octree leaf-node have then been triangulated by projection upon a fitting plane, computed with the help of a PCA. The subsequent triangulation in 2D results in a sequence of point indices, later used for ordering the original points into triangle strips. Our initial experiences with this approach indicate that well-behaved triangulation results can be obtained also for 3D point sets when the number of octree-levels is adapted to the convexity of the point surface in 3D. Our current efforts are directed towards exploring this relationship as well as extending onion peeling to handle non-convex hulls.

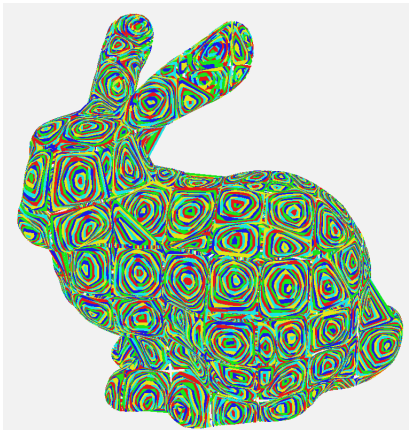


Figure 9: *Bunny data set triangulated with OPT using 4-level octree partitioning.*

References

[AHMS94] ARKIN E., HELD M., MITCHELL J., SKIENA S.: Hamiltonian triangulations for fast rendering. In *Algorithms – ESA '94*, van Leeuwen J., (Ed.), vol. 855 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1994, pp. 36–47. 10.1007/BFb0049395. 1

[BRS05] BOUBEKEUR T., REUTER P., SCHLICK C.: Surfel stripping. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (New York, NY, USA, 2005), ACM, pp. 177–186. 1

[Cha85] CHAZELLE B.: On the convex layers of a planar set. *IEEE Transactions on Information Theory IT-31*, 4 (1985), 509–517. 1

[GE04] GOPI M., EPPSTEIN D.: Single-strip triangulation of manifolds with arbitrary topology. *Computer Graphics Forum (EUROGRAPHICS) 23*, 3 (2004), 371–379. 1

[Tou83] TOUSSAINT G.: Solving geometric problems with the rotating calipers. In *Proceedings of IEEE MELECON '83* (1983), pp. 10–17. 1

[Tou86] TOUSSAINT G.: New results in computational geometry relevant to pattern recognition in practice. In *Pattern Recognition in Practice II* (1986), Gelsema E., Kanal L., (Eds.), pp. 135–146. 1

[VK07] VANĚČEK P., KOLINGEROVÁ I.: Comparison of triangle strips algorithms. *Computers & Graphics 31*, 1 (2007), 100 – 118. 1, 3

Work in Progress

Content Aggregation, Visualization and Emergent Properties in Computer Simulations

Gordana Dodig Crnkovic[1], Juan M. Duran[2] and Davor Slutej[3]

[1]School of Innovation, Design and Engineering, Computer Science Laboratory, Mälardalen University, Sweden. *gordana.dodig-crnkovic@mdh.se*
[2]SimTech, Universität Stuttgart, Deutschland and University of Córdoba, Argentina. *juan.duran@philo.uni-stuttgart.de*
[3]ABB and Mälardalen University, Sweden, *davor.slutej@se.abb.com*

Abstract

With the rapidly growing amounts of information, visualization is becoming increasingly important, as it allows users to easily explore and understand large amounts of information. However the field of information visualization currently lacks sufficient theoretical foundations. This article addresses foundational questions connecting information visualization with computing and philosophy studies. The idea of multiscale information granulation is described based on two fundamental concepts: information (structure) and computation (process). A new information processing paradigm of Granular Computing enables stepwise increase of granulation/aggregation of information on different levels of resolution, which makes possible dynamical viewing of data. Information produced by Google Earth is an illustration of visualization based on clustering (granulation) of information on a succession of layers. Depending on level, specific emergent properties become visible as a result of different ways of aggregation of data/information. As information visualization ultimately aims at amplifying cognition, we discuss the process of simulation and emulation in relation to cognition, and in particular visual cognition.

Introduction

Chen remarks in a newly published article on information visualization:

"The general consensus, as reported by a recent workshop and a few other public presentations, was that information visualization currently lacks adequate theoretical foundations. [Nor07] As a result, many approaches are ad hoc in nature. [...] The search for theoretical foundations increasingly introduces and adopts theories and conceptual frameworks from other fields and disciplines. For example, distributed cognition in human-computer interaction is seen as a potential candidate for a theoretical framework for information visualization. [Sta08] Norman's Seven Stages of Action, also in human-computer interaction, provides a new insight into interacting with information visualizations, specifically on the gulf of execution and the gulf of evaluation. [Lam08]" `http:\\onlinelibrary.wiley.com/doi/10.1002/wics.89/full`, [Che10].

This article offers a contribution to the theoretical foundations of information visualization from the point of view of computing and philosophy studies, providing broader context in which information visualization is related to cognition and emergent properties in physical and simulated worlds.

Info-computational View of Cognition

All of our knowledge is based on information we get from the world. Physicists [Zei05] [Llo06] and [Ved10] suggest possibility of seeing information and reality as one. This is in accord with Informational Structural Realism which says that reality is made of informational structures [Flo08a]. What Floridi assumes to be mind-independent data corresponds to world as information (proto-information). By interactions with an agent it reveals as consisting of structural objects (chunks of information with defined mutual relationships).

Building on Floridi's Informational Structural Realism with information as the fabric of the universe the process of dynamical changes of the universe makes the universe a huge information processing mechanism. It is important to understand that computation performed by the Universe is not the computation we have in our machines, it is com-

putation most generally defined as information processing, [Bur05]. Here is how Chaitin describes the idea of computing Universe (Natural computationalism):

"And how about the entire universe, can it be considered to be a computer? Yes, it certainly can, it is constantly computing its future state from its current state, it's constantly computing its own time-evolution." [Cha07]

The synthesis of Informational Structural Realism with the Computing Universe leads to Info-Computationalism [DC06] [DC10] [Mue10]. In short: information is the structure, the fabric of reality. The world exists independently from us (realist position of structural realism) in the form of proto-information. That proto-information becomes information for an agent in a process of interaction. Formalization of Info-Computational approach within Category Theory may be found in [Bur10].

Information and Computation in Biological and Intelligent Artificial Systems

Within the info-computational framework process of cognition is seen as a successive structuring (organizing) of data, where data are understood as simplest information units, signals acquired by an agent through the senses/sensors/instruments. Information as meaningful data, is turned into knowledge by a computational process going on in an agent. An agent is a physical system (living organism or an intelligent machine) or software possessing adaptive learning behaviors. Living agents have evolved from pre-biotic inorganic forms into increasingly complex systems able to self-organize, adapt to the environment and reproduce -all based on information processing.

Understanding information processing as computation [Bur05] implies that knowledge generation in biological agents involves natural computation, defined by [Mac04] as computation occurring in nature or inspired by that in nature. Part of natural computation characterized by the use of inexact solutions to computationally-hard tasks is called soft computing -many of bio-computing methods are of this soft character. It includes evolutionary computing, neurocomputing, fuzzy logic and probabilistic computing. [Zad98] emphasizes that an "essential aspect of soft computing is that its constituent methodologies are, for the most part, complementary and symbiotic rather than competitive and exclusive." Natural computation is necessarily both symbolic and sub-symbolic information processing, which can be both discrete and continuous, as it corresponds to the dynamics of natural processes.

1. Informational Structures of Reality: Entities and Levels

In order to get a more specific understanding of info-computational thinking in multiscale information, granulation and aggregation, some definitions are in order.

Entity is used to mean anything that is considered to be discretely classifiable by some system. An entity can be composed of other entities (parts) -but it has its own identity. An entity could be a physical object or an abstract idea; within Informational Structural Realism both are different sorts of informational structures, which on the bottom level are data.

Granules are collections of information entities that usually originate at the numeric level (data-level) and are arranged together due to identity, similarity, functional or physical proximity, or alike.

Granular Computing is a new paradigm in which computing is understood as structured information processing in a hierarchy with multiple levels [Bar02]. Three different computational levels are distinguished which involve processing of information with increasing granularity: with the lowest level being numeric processing, the intermediate level processing larger information granules, and the highest level involving symbol-based processing. Within the info-computational framework we can continue to ever more extensive chunks of information that can be processed on increasingly higher levels: variable granulation (clustering/aggregation); system granulation (aggregation), concept granulation (component analysis) and so on. According to [Yao09], Granular Computing is a human-inspired approach, with important applications in the design and implementation of intelligent information systems. [Zad98] sees Granular Computing as the fundamental contribution to Fuzzy Logic which serves as a basis for the methodology of Computing with Words in which data sets have the form of propositions expressed in a natural language and words are interpreted as labels of granules. Zadeh goes as long as to suggesting Granular Logic as a hybrid of Fuzzy Logic with Granular Computing.

"In combination, the methodologies of soft computing, fuzzy logic, granular computing and computing with words are essential to the conception, design and utilization of information/intelligent systems because such systems reflect the pervasiveness of imprecision, uncertainty and partial truth in the world we live in." [Zad98]

Information organization in levels: Conger [Con25] defined level as "a class of structures or processes which are distinguishable from others as being either higher or lower." Levels are the consequence of two complementary processes of *differentiation* (separation, division, the result of analysis) and *integration* (the result of synthesis, which means they form as a result of grouping of some entities on a common "integrative levels").

Levels reflect the inherent nature of reality (objective interpretation like levels of organization, levels of structure) but are always chosen among several possibilities, and in that way reflect our conceptualization of reality (subjective interpretation: levels of description, levels of representation, levels of analysis, levels of abstraction, [San04] and

[Flo08b]. Each level in a hierarchy depends on, and organizes/governs/controls the levels below, while contributing to the qualitatively different (emergent) properties of the level above, [Yao09]. The parts change their identities; new principles apply at the higher level, [Cor95].

Needham, in [Nee43][234], argues that for those successive forms of order in complexity and organization "A sharp change in organizational level often means that *what were wholes on the lower level become parts on the new*, e.g., protein crystals, cells in metazoan organisms, and metazoan organisms in social units." (Emphasis added)

Levels are used to organize informational granule for the reasons of simplicity and clarity. Level thinking is fundamental for representation, organization and synthesis of data, information, and knowledge. A structured organization of information seems to be one important way to outwit the limited capacity of human information processing, as shown by [Mil56].

Emergence is a phenomenon observed when entities on a higher level exhibit novel properties that are not observable on the lower levels. Levels correspond to aggregation of objects (informational granule) which are forming new wholes on a different scale of resolution. Life is an example of emergent phenomenon in objects (organisms) that have signaling and self-sustaining processes. Even though living organisms consist of inorganic parts, those parts show a qualitatively new behavior as a result of mutual interactions. Biology emerges from chemistry. Emmeche et al. [Stj97] identifies the following four primary levels of organization (integrative levels): physical, biological, psychological and sociological.

Emergence occurs in *both space and time* as "the arising of novel and coherent structures, patterns and properties during the process of self-organization in complex systems." [Gol99].

Visual Cognition as Information Processing

Computational Visual Cognition is a study of human visual perceptual and cognitive abilities in a natural setting, including abilities at real world scene understanding, perception, recognition and memory alongside with the role of attention and learning. It uses tools and theories from image processing, cognitive science, computational vision, computer graphics and cognitive neuroscience. [Ull06] has developed computational models of vision that explain how humans recognize objects, perceive motion, use their visual world for task-relevant information, and create coherent representations of their environments. The main strategy here is learning about human visual cognition through construction of artifacts, using model-based reasoning.

Recent studies in biology, ethology and neuroscience have increased our knowledge of biological cognitive functions, and led to the understanding that the most important feature of cognition is its *ability to deal efficiently with complexity*, [GM95]. Insights into natural intelligence, together with the increase in power of computing machinery have brought us closer to the modeling of intelligent behavior. [DC08] argues that Info-Computationalism presents the most appropriate theoretical framework for understanding of the phenomena of cognition and intelligence in both biological and artificial systems. Cognition is seen as based on several levels of data processing [Min10] [Goe93] in a cognizing agent. Information processed from-sensory data processed into perception by an agent can be understood as an *interface between the data* (the world) and an agent's perception of the world, [Hof09]. *Patterns of information should thus be attributed both to the world and to the functions and structures of the brain*. In an analogous way, knowledge can be understood as an interface between perception and cognition. Meaning and agency are the results of the information processing, and its refinement by relating to already existing (memorized) information. The meaning of an object in the external world is recognized through the process of perception of sensory signals, their processing through nervous system, comparison with memorized objects, and anticipation from memorized experiences.

Data, information, perceptual images and knowledge are organized in a multi-resolutional (multi-granular, multi-scale) model of the brain and nervous system, [Min10]. Multiresolutional representation has proven to be a good way of dealing with complexity in biological systems, and they are also being implemented in AI, [Goe93]. This is in accordance with Levels of Processing Models of memory [Cra72]. It should be noted that this one-directional sequential progression from shallow to deep processing, is only a first approximation, while in a more realistic model a combination of bottom-up processing (stimulus-driven) and top-down processing (concept driven) must be applied.

Computational Practices

"Artificial neural networks designed to implement even the most rudimentary forms of memory and knowledge extraction and adaptive behavior incorporate massively and symmetrically interconnected nodes; yet, in the cerebral cortex, the probability of a synaptic connection between any two arbitrarily chosen cells is on the order of 10^{-6}, i.e., so close to zero that a naive modeler might neglect this parameter altogether. The probability of a symmetric connection is even smaller (10^{-12}). How then, are thought and memory even possible? *The solution appears to have been in the evolution of a modular, hierarchical cortical architecture, in which the modules are internally highly connected but only weakly interconnected with other modules*. Appropriate inter-modular linkages are mediated indirectly via common linkages with higher level modules collectively known as association cortex." (Emphasis added) [McN10].

Self-organization and self-description are fundamental intrinsic properties of natural intelligent systems. Learning is an essential part of those capabilities and it requires among others the development of a symbolic representations easy to maintain and use. In intelligent biological systems based upon a hierarchy of functional loops, each of these loops can be treated as a control system *per se* [Min10]. Generation of structures resulting from sensory processes and existing information in a nervous system are built in a multiresolutional way, with many pattern recognition and control mechanisms hardwired.

Describing existing computing practices, [New82] introduces the concept of *levels of analysis for a computer system*, suggesting that there is a higher level than is usually addressed by computer scientists and calls that level *the knowledge level*, distinct from the symbolic level in which representation (logic) lies. Newell defines the knowledge level as embodying all the knowledge obtained or inferable from given information.

Similarly, in case of biological systems, Marr proposed that cognitive processes can be thought of as having three levels of description [Mar82] -an approach that can be generalized to any information processing system:

The top (computational) level describes the characteristics of the inputs and outputs of the computational problem solved by the process.

The middle (algorithmic) level provides the steps that transform the input into the output.

The bottom (implementational) level describes the physical 'machinery' (hardware) that carries out the computation defined by algorithms.

Observe the difference between algorithmic level and computational level in Marr. In Church-Turing model of computation, those two coincide, and computation is identical with algorithm execution. The knowledge layer in Marr is a consequence of his thinking of computing as a real world process while Turing-Church computing is an idealization of computing as an abstract mathematical procedure.

In a similar vein, and focusing on Visual Cognition, [Dia07] finds that images contain only one sort of information - the perceptual (physical) information. Semantics comes from a human observer that perceives and interprets the image. Defining information based on Kolmogorov's complexity and Chaitin's algorithmic information, Diamant proposes a framework for visual information processing, which explicitly accounts for perceptual and cognitive image processing.

Understanding simulations and their underlying mechanisms is informative for several reasons, among others in order to understand our own mechanisms of grasping the reality, as human cognition in many ways resembles process of simulation and emulation. From the cognitive point of view, in order to conceptualize enormous flow of input data that bombards us through our sensory apparatus, we have to focus on limited amount of data organized in structures. Partly, as a result of evolution, this clusterization and aggregation of data happens in our nervous system automatically on the hardware level without our being aware of it (sub-symbolic information processing). From the results of neuroscience and cognitive science we are learning about the implementation of those computational mechanisms in living beings, which appear in a hierarchy of levels.

Simulation, Emulation, Aggregation

Over the last years, there have been several attempts to philosophically define computer simulation [Dur10]. Next we will briefly address two of such attempts that we consider the most prominent of the latest literature. We have picked them in the first place because they set the agenda for the contemporary philosophical discussion on computer simulations. The first one is due to Hartmann:

"Simulations are closely related to dynamic models. More concretely, a simulation results when the equations of the underlying dynamic model are solved. This model is designed to imitate the time-evolution of a real system. To put it another way, a simulation imitates one process by another process. In this definition, the term 'process' refers solely to some object or system whose state changes in time. If the simulation is run on a computer, it is called a computer simulation." [Har96]

The mayor drawback in Hartmann's definition is that it is too broad since it does not differentiate simulations from other similar activities such as imitation or emulation. As a matter of fact, he bases his definition of simulation on the concept of imitation, which he does not define. However simulation should be differentiated from imitation, since in the former case we expect to have knowledge of the target system whereas in the latter we do not. Something similar happens for the case of emulation.

The second definition of computer simulation is due to Humphreys. In this case, the author strengthens the definition of computer simulation by claiming that:

"A computer simulation is any computer-implemented method for exploring the properties of mathematical models where analytical methods are unavailable." [Hum90]

Although Humphreys' definition is more precise than Hartmann's, it is still too general. It is not clear what he means by exploration of properties of mathematical models. Let us try to unfold this.

A major deficiency in Humphreys' concept is that it restricts computer simulations to mathematical models, which is clearly not accurate enough for characterization. Computer simulations are based on algorithms, but an algorithm not necessarily has to be based on a mathematical model, for

it could be based on logical models as well as specifications of a system lacking of any mathematical or logical structure.

Several other definitions have been discussed in recent literature, but Hartmann and Humphreys have set the agenda for philosophical analysis of computer simulations.

The dilemma of computer simulations centers on the question of their value as tools for exploration of the real world. Many philosophers see the deficiency of simulation related to the materiality of experiments, which is not present in simulations. This claim is now known as the 'materiality problem' of computer simulations and has its standard conceptualization: "in genuine experiments, the same 'material' causes are at work in the experimental and target systems, while in simulations there is merely formal correspondence between the simulating and target systems [...] inferences about target systems are more justified when experimental and target systems are made of the 'same stuff' than when they are made of different materials (as is the case in computer experiments)" [Par09].

Three solutions are in place: the lack of materiality of computer simulations make them defective as tools for production of relevant and reliable information (for example [Gua02]; [Mor05]; [Gie09]); the presence of materiality in experiments is ultimately, unimportant for purposes of investigations of real world phenomena (for instance [Mor09]; [Par09]; [Win09]), or some kind of middle point where the physicality of the computer, that is, the actual physical states in which the computer is switched when computing, serves as the basis for making the case of causality within the computer. There is a forth solution to this problem presented by Barberousse, Franceschelli and Imbert [Bar09] who claim that the representational and predictive power of computer simulations does not depend upon physical capacities, but upon a detailed analysis of their semantic levels.

In the following, definitions will be given in order to clarify ideas of simulation, emulation and aggregation.

Emulation vs. Simulation: "One system is said to emulate another when it performs in exactly the same way, though perhaps not at the same speed. A typical example would be emulation of one computer by (a program running on) another. You might use an emulation as a replacement for a system whereas you would use a simulation if you just wanted to analyze it and make predictions about it." -The Free Online Dictionary of Computing.© 1993-2000 1995-05-12

Aggregation is the process of collecting together (aggregating) information from multiple sources. In order to increase semantic value of aggregation, a process of curation is often applied, which is done by humans who are making sense by synthesis and contextualization of aggregated information. The next step we may hope for is semantic web performing automatic intelligent aggregation and curation of information.

Information Visualization

Information visualization is the interdisciplinary study of "the visual representation of large-scale collections of non-numerical information, such as files and lines of code in software systems, library and bibliographic databases, networks of relations on the internet, and so forth", [Fri08]. [Che10] gives the following more general definition: "Information visualization is concerned with the design, development, and application of computer generated interactive graphical representations of information." In the more narrow sense, visualization deals with abstract, nonspatial or qualitative data which are transformed and represented so as to best impart meaning, using information design methods. Scientific Visualization on the other hand is more straight-forward in a sense of interpretation as the data usually are numerical.

As [Che10] points out, the main goal of information visualization is for users to gain insights, so its role is to provide cognitive tools contributing to extended cognition.

The graphical representation of information is particularly suitable due to the massive hardware support for processing visual information which we have in the nervous system and especially in the brain. Once the simulation output has been constructed, the visual format is most easily deconstructed/ decoded/ translated by the brain into its internal representations because of the substantial hardware support for the visual information processing. In simulation models visualization is usually an important part, often accompanied by interactivity and animation (behavior of the system in time, and that is where it connects to visual cognition.

Google Earth as an Example of Visualized Content Aggregation on Different Levels of Resolution/Organization of Content

Google Earth is a geographic information system presenting virtual globe which maps the Earth by the superimposition of satellite images, GIS 3D globe and aerial photography. It displays satellite images of varying resolution of the Earth's surface, but most land is covered in at least 15 meters of resolution. One can overlay different layers of information like crime statistics, or information about culture, history and similar. Google Earth is using mashups with other web applications such as Wikipedia or Google News, which provides automatic aggregation of information using algorithms which carry out contextual analysis.

Google Earth is a good example of Granular Computing framework in which different levels are visually represented. On a low resolution satellite image one can observe large-scale phenomena such as weather patterns, and land relief of a globe, and in highest resolution image one views small-scale phenomena, such as the streets of a town and even particular houses.

The same principle of hierarchical information structur-

ing is used of all data representation in information processing systems, both artificial and biological ones. Biological agents, having necessarily limited resources, rely on layered data representation by grouping of chunks of information of different granularity. At different resolutions (granularities, levels of aggregation) different features and relationships emerge. Granular computing uses this property of cognition in designing more effective machine learning and reasoning systems that smoothly connect to human cognitive systems.

Conclusions

Being a young research field, (The first specialized international journal, Information Visualization, appeared in 2002), Information Visualization currently lacks adequate theoretical foundations. In search for theoretical foundations, theories and conceptual frameworks from other fields and disciplines are adopted, notably frameworks from human-computer interaction such as distributed cognition, [Sta08].

As defined by Shneiderman, Card and Mackinlay, "Information Visualization is the use of computer-supported, interactive, visual representations of abstract data to amplify cognition". Consequently, Information Visualization designs can be seen as tools - cognitive extensions based on active vision which uses graphic designs as cognitive amplifiers.

We present an info-computational theoretical approach unifying the principles governing content aggregation, information visualization and emergent properties in computer simulations and cognition. It puts into a common context information granulation/aggregation, granular computing paradigm, level method of information organization, emergent properties and info-computational character of cognition as an information processing mechanism that in many respects resembles simulation and emulation processes.

References

[Bar02] BARGIELA A. PEDRYCZ W.: *Granular Computing: An Introduction*. Kluwer, 2002. 2

[Bar09] BARBEROUSSE A. FRANCESCHELLI S. I. C.: Computer simulations as experiments. *Synthese 169*, 3 (2009), 557–574. 5

[Bur05] BURGIN M.: *Super-Recursive Algorithms*. Springer Monographs in Computer Science, 2005. 2

[Bur10] BURGIN M.: *Information and Computation*. World Scientific Publishing Co. Series in Information Studies, 2010, ch. Information Dynamics in a Categorical Setting. 2

[Cha07] CHAITIN G.: *Computation, Information, Cognition. The Nexus and The Liminal*. Cambridge Scholars Publishing, 2007, ch. Epistemology as Information Theory,, pp. 2–18. 2

[Che10] CHEN C.: Information visualization. *Wiley Interdisciplinary Review: Computational Statistics 2*, 4 (2010), 387–403. 1, 5

[Con25] CONGER G.: The doctrine of levels. *The Journal of Philosophy 22* (1925), 309–321. 2

[Cor95] CORNING P.: Synergy and self-organization in the evolution of complex systems. *Systems Research 12*, 2 (1995), 89–121. 3

[Cra72] CRAIK F.I.M. LOCKHART R.: Levels of processing: a framework for memory research. *Journal of Verbal Learning and Verbal Behavior 11* (1972), 671–684. 3

[DC06] DODIG-CRNKOVIC G.: Investigations into information semantics and ethics of computing, 2006. 2

[DC08] DODIG-CRNKOVIC G.: Knowledge generation as natural computation. *Journal of Systemics, Cybernetics and Informatics 6* (2008). 3

[DC10] DODIG-CRNKOVIC G.: The cybersemiotics and info-computationalist research programmes as platforms for knowledge production in organisms and machines. *Entropy 12* (2010), 878–901. 2

[Dia07] DIAMANT E.: Modeling visual information processing in brain: A computer vision point of view and approach. *Advances in Brain, Vision, and Artificial Intelligence, Lecture Notes in Computer Science 4729* (2007), 62–71. 4

[Dur10] DURÁN J. M.: *Thinking machines and the philosophy of computer science: Concepts and principles, Information Science Reference*. IGI Global, 2010, ch. Computer simulations and traditional experimentation: from a material point of view, pp. 294–310. 4

[Flo08a] FLORIDI L.: A defence of informational structural realism. *Synthese 161*, 2 (2008), 219–253. 1

[Flo08b] FLORIDI L.: The method of levels of abstraction. *Minds and Machines 18*, 3 (2008), 303–329. 3

[Fri08] FRIENDLY M.: Milestones in the history of thematic cartography, statistical graphics, and data visualization, 2008. 5

[Gie09] GIERE R.: Is computer simulation changing the face of experimentation? *Philosophical Studies 143* (2009), 59–62. 5

[GM95] GELL-MANN M.: *The Quark and the Jaguar: Adventures in the Simple and the Complex*. Owl Books, 1995. 3

[Goe93] GOERTZEL B.: *The Evolving Mind*. 1993. 3

[Gol99] GOLDSTEIN J.: Emergence as a construct: History and issues. *Emergence: Complexity and Organization 1*, 1 (1999), 49–72. 3

[Gua02] GUALA F.: *Model-Based Reasoning: Science, Technology*. Kluwer, 2002, ch. Models, simulations, and experiments, pp. 59–74. 5

[Har96] HARTMANN S.: *Simulation and Modelling in the Social Sciences from the Philosophy of Science Point of View*. Kluwer, 1996, ch. The world as a Process: Simulations in the natural and social science, pp. 77–100. 4

[Hof09] HOFFMAN D.: *Object Categorization: Computer and Human Perspectives*. Cambridge University Press, 2009, ch. The Interface Theory of Perception: Natural Selection Drives True Perception to Swift Extinction. 3

[Hum90] HUMPHREYS P.: Computer simulations. *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association* (1990), 497–506. 4

[Lam08] LAM H.: A framework of interaction costs in information visualization. *IEEE Trans Vis Comput Graph 14* (2008), 1149–1156. 1

[Llo06] LLOYD S.: *Programming the Universe: A Quantum Computer Scientist Takes On the Cosmos*. Knopf, 2006. 1

[Mac04] MACLENNAN B.: Natural computation and non-turing models of computation. *Theoretical Computer Science 317* (2004), 115–145. 2

[Mar82] MARR D.: *Vision: a computational investigation into the human representation and processing of visual information*. W. H. Freeman, 1982. 4

[McN10] MCNAUGHTONA B. L.: Cortical hierarchies, sleep, and the extraction of knowledge from memory. *Artificial Intelligence 174*, 2 (2010), 205–214. 3

[Mil56] MILLER G. A.: The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review 63* (1956), 81–97. 3

[Min10] MINSKY M.: *Information and Computation*. World Scientific Publishing Co. Series in Information Studies, 2010, ch. Interior Grounding, Reflection, and Self-Consciousness. 3, 4

[Mor05] MORGAN M.: Experiments versus models: New phenomena, inference and surprise. *Journal of Economic Methodology 12*, 2 (2005), 317–329. 5

[Mor09] MORRISON M.: Models, measurement and computer simulation: the changing face of experimentation. *Philosophical Studies 143* (2009), 33–47. 5

[Mue10] MUELLER G. D.-C. V.: *Information and Computation*. World Scientific Publishing Co. Series in Information Studies2, 2010, ch. A Dialogue Concerning Two World Systems: Info-Computational vs. Mechanistic. 2

[Nee43] NEEDHAM J.: *Herbert Spencer Lecture. Time, the refreshing river*. 1943. 3

[New82] NEWELL A.: The knowledge level. *Artificial Intelligence 18*, 1 (1982). 4

[Nor07] NORTH A. K. J. S. J. F. C.: Workshop report: information visualization—human-centered issues in visual representation, interaction, and evaluation. *Inf Vis 6* (2007), 189–196. 1

[Par09] PARKER W.: Does matter really matter? computer simulations, experiments, and materiality. *Synthese 169*, 3 (2009), 483–496. 5

[San04] SANDERS L. F. J. W.: *Yearbook of the artificial—nature, culture and technology, models in contemporary sciences*. Peter Lang, 2004, ch. The method of abstraction, pp. 177–220. 2

[Sta08] STASKO Z. L. N. N. J.: Distributed cognition as a theoretical framework for information visualization. *IEEE Trans Vis Comput Graph 14* (2008), 1173–1180. 1, 6

[Stj97] STJERNFELT E. C. S. K. F.: Explaining emergence: Towards and ontology of levels. *Journal for General Philosophy of Science 28* (1997), 83–119. 3

[Ull06] ULLMAN S.: Object recognition and segmentation by a fragment-based hierarchy. *Trends in Cognitive Sciences 11* (2006), 58–64. 3

[Ved10] VEDRAL V.: *Decoding Reality: The Universe as Quantum Information*. Oxford University Press, 2010. 1

[Win09] WINSBERG E.: A tale of two methods. *Synthese 169*, 3 (2009), 575–590. 5

[Yao09] YAO Y.: *Human-Centric Information Processing Through Granular Modelling*. Springer-Verlang, 2009, ch. Integrative Levels of Granularity, pp. 31–47. 2, 3

[Zad98] ZADEH L. A.: Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems. *Soft Computing 2* (1998), 23–25. 2

[Zei05] ZEILINGER A.: The message of the quantum. *Nature* (2005), 438–743. 1