Proceedings of the

27th Central European Seminar on Computer Graphics

April 28 - May 2, 2023
Smolenice, Slovakia

Institute of Visual Computing & Human-Centered Technology
TU Wien

Faculty of Mathematics, Physics and Informatics
Comenius University Bratislava

Institute of Mathematics
Slovak Academy of Sciences

Faculty of Informatics and Informaion Technologies
Slovak Technical University in Bratislava

Partners

Edited by Martin Ilčík, Annalena Ulschmid and Michael Wimmer © 2023
ISBN: 978-3-9504701-4-7

Impressum

Vienna University of Technology
Institute of Visual Computing & Human-Centered Technology
Favoritenstraße 9-11/193
1040 Vienna

ISBN 978-3-9504701-4-7

Welcome to CESCG 2023!

This book contains the proceedings of the 27th Central European Seminar on Computer Graphics, short CESCG, which continues a history of very successful seminars.

The long history of CESCG has started in 1997 in a medium-sized lecture room in Bratislava, bringing together students from Bratislava, Brno, Budapest, Graz, Prague, and Vienna. The idea found wide appraisal and the seminar moved to the beautiful castle of Budmerice, where it was held for eight consecutive years, constantly growing in size and attraction. It was just in the 10th anniversary year 2006 that CESCG had to take a detour to move to Častá-Papiernička Centre, while it was back in Budmerice castle since 2007. Budmerice castle ultimately closed down for public in 2011. After spending that year in Viničné, in 2012 we moved to the beautiful castle in Smolenice. During the COVID pandemics lock-downs in 2020 and 2021, CESCG switched to a virtual mode at Discord and Youtube.

Who are the CESCG heroes who made this year's seminar happen? In no particular order – because many people were involved equally – we would like to thank the organizers from Vienna: **Michael Wimmer**, Max Höfferer, Annalena Ulschmid and Diana Marin. Special thanks goes to **Martin Ilčík** for extensive editorial work and event management, keeping the seminar alive for already 15 years. We are very thankful to the CESCG organizers from Bratislava, Zuzana Berger-Haladová, Lukáš Hudec and **Andrej Ferko**, always an inspiration to CESCG. Stanislav Griguš again produced professional promotion videos.

The main idea of CESCG is to bring students of computer graphics together across boundaries of universities and countries. We focus on sustainable academic and research development in the field of Computer Graphics in Visegrad Countries and Austria. Our mission is to support undergraduate talents in their future careers. We have 10 participating institutions and a tight time schedule of 12 student papers and 6 posters. We welcome groups from Bratislava (Comenius and STU), Slovakia; Prague (Charles and ČVÚT), Czech Republic; Budapest (BME), Hungary; TU Graz and Vienna (Uni, TU and VRVis), Austria; University of Ljubljana, Slovenia; and University of Sarajevo, Bosnia and Herzegovina. Martin Ilčík from TU Wien led the virtual workshop on "Scientific Storytelling" right after the students received the first feedback to their papers. The keynote talk "(Privacy-Preserving) Visual Localization" will be held by Torsten Sattler from the Czech Technical University in Prague.

We assembled an International Program Committee of 11 members, allowing us to have each paper reviewed by two IPC members during the informal reviewing process. Students also cross-reviewed their papers. We would like to thank the members of the IPC for their contribution to the reviewing process:

Jiří Bittner	Jiří Hladůvka	Selma Rizvić
Ciril Bohak	Lukáš Hudec	László Szécsi
Daniel Cornel	Martin Ilčík	Michael Wimmer
Jan Egger	Lukas Lipp	

The reviewing process was further supported by: Andreas Buttinger-Kreuzhuber, Ladislav Čmolík, Petr Felkel, David Hahn, Vlastimil Havran, Jakub Hendrich, Fabian Hörst, Martin Kacerik, Peter Kapec, Stefan Ohrhallinger, Kostiantyn Rudenko, László Szirmay-Kalos, Annalena Ulschmid, Marton Vaitkus, Juraj Vincúr, Maria Wimmer.

With the 20th anniversary of the seminar in 2016, Martin initiated the CESCG EXPO project. This year we restarted the EXPO with companies and research institutions specialized on visual computing presenting their innovative products in an interactive exhibition. The exhibitors are:

Additive Appearance, Prague	Prusa Research, Prague
Canon, Bratislava	Vectary, Bratislava
Escape Motions, Piešťany	VR Group, Brno
Nanographics, Vienna	WildRealm, Bratislava
Procedural Design, Vienna	

For 2020 Martin initiated the ACADEMY project to offer tutorials and lectures by international experts to stimulate knowledge exchange in a way similar to a summer schoool. The COVID pandemics interrupted these plans, so the zero-year offered just a reduced set of workshops virtually. The first real ACADEMY took place 2022, offering a day with 5 tutorials right after the seminar. This year the ACADEMY finally got its intended shape with 12 tutorials and lectures given in 4 parallel tracks over the course of 2 full days:

Real-Time Rendering : Introduction to Vulkan (Johannes Unterguggenberger, TU Wien), Terrain Rendering (Adam Celarek, TU Wien), WebGPU (Žiga Lesar[1], Lucas Melo[2] and Lukas Herzberger[2], University of Ljubljana[1] and TU Wien[2])

Games : Designing Thinky Games (Ján Tóth)

Reconstruction : Photogrammetry (Peter Mydliar, Canon), Airborne Optical Sectioning (Oliver Bimber, JKU Linz)

Music : Math and Music – How Synthesizers work (Peter Mindek, Nanographics), D♭ – A Music Programming Language (Martin Ilčík, Procedural Design)

Neural Graphics : A Brief Introduction to NeRFs (Torsten Sattler, ČVÚT Prague)

Geometry : Combinatorial Maps and Rapid prototyping using ThreeJS (Paul Viville, Strasbourg University)

Creative Production : VR Production (Selma Rizvić and Bojan Mijatović), YouTube – From One to a Million (Stanislav Griguš, SG-production), Creative Photo Walk (Tobias Rittig, Charles University)

The organization of such a large event requires additional funding. We are very thankful to the partners of CESCG 2023 for supporting us financially and by donating prizes for awarding the best student results:

– **KAUST**, King Abdullah University of Science and Technology,
– **VRVis**, Research Center for Virtual Reality and Visualization,
– **Canon**, digital imaging solutions,
– **Prusa Research**, making more than just amazing 3D printers,
– **SISp**, Slovak Society for Computer Science,
– **Epic Games**, cutting-edge games and cross-platform game engine technology,
– **Escape Motions**, developer of innovative visual tools,
– **IEEE Women in Engineering**, promoting women engineers and scientists,
– **Procedural Design**, adaptive content creation,
– **SG-production**, promoting science.

Please note that the electronic version of these proceedings is also available at https://cescg.org/library/.

April 2023,

Martin Ilčík
Annalena Ulschmid
Michael Wimmer

Table of Contents

Partners of CESCG 2023

Extended Reality

AR postcards as a learning tool in Computer Graphics

Aya Ali Al Zayat*
Lejla Becirevic†
Bojan Mijatovic‡
Supervisor Selma Rizvic§

University of Sarajevo
Faculty of Electrical Engineering
Sarajevo, Bosnia and Herzegovina
Sarajevo School of Science and Technology
Sarajevo, Bosnia and Herzegovina

Abstract

Preserving cultural heritage is a vital part of preserving the history of a country. Computer Graphics students created Augmented Reality (AR) postcards with 3D models of Bosnia and Herzegovina (BH) cultural heritage objects. They added various Points of Interest (POIs) related to the objects with different multimedia content. The content in the app is just a small glimpse into the vast history of every single object picked, purposely designed to be easily digested by users of all ages. These postcards were integrated into a single app with the goal to present information about cultural heritage buildings in a new, interesting and easily accessible way.

In this paper, we describe the teaching methodology and resulting AR application. Through the user eXperience evaluation study, we checked if this approach improves the overall experience of learning about cultural heritage objects. The results of this evaluation will contribute to the development of future AR applications for cultural heritage.

Keywords: Computer Graphics education, Augmented Reality, digital heritage

1 Introduction

Computer Graphics (CG) is a very demanding field for education as it consists of many very important and diverse sub-fields. Starting from so-called "hardcore" CG with algorithms for conversion of coordinate systems, matrix transformations, clipping, shading, illumination, and rendering, it expanded to computer vision, computer animation, multimedia, and mixed reality. It is very difficult to present all these notions and important applications of CG within the basic undergraduate course. Therefore teachers need to make certain compromises. At the Faculty of Electrical Engineering, University of Sarajevo, computer graphics is studied in the Computer Science department. Students are offered a basic CG course at the Bachelor level and two elective courses at the Masters level. A common feature of all these courses is that they are more oriented to CG applications and less to graphics algorithms and programming. The reason behind such a decision is that students of this Department have plenty of programming courses to attend and not much opportunity to learn skills such as 3D modeling and animation and Virtual/Augmented Reality applications development. Another particularity of teaching CG at the University of Sarajevo is offering students the basics of graphic design, so their visual presentations and user interfaces (UI) would fulfill the expectations of their users without the necessity to include graphics designers in the software development team.

The main contribution of this paper is the novel approach to teaching computer graphics, which involves adding the artistic aspect of the topic and offering students to obtain application development skills instead of sole graphics programming. In that terms, we present the teaching methodology of a basic Computer Graphics course at the undergraduate level with particular emphasis on an Augmented Reality lab project that was later turned into a digital cultural heritage application. In that project, students were requested to design and implement AR postcards with the most interesting BH cultural monuments. The paper structure is as follows: in the Related work Section we describe the relationship between CG education and AR applications for cultural heritage, Section 3 presents the structure and teaching methodology of the course we describe, and Section 4 shows how the students are being marked. Section 5 describes the AR lab project workflow, while Section 6 shows the novelty of our teaching and explains how it can be used for creating a digital cultural heritage application. We also present results of preliminary user experience evaluation to show the appre-

*aalialzaya1@etf.unsa.ba
†lbecirevic1@etf.unsa.ba
‡bojan.mijatovic@ssst.edu.ba
§srizvic@etf.unsa.ba

ciation of created application by different users. In the last Section, we offer our conclusions and lessons learned.

2 Related work

While the use of novel technologies serves as a catalyst for student engagement in education [6], it is equally important to evaluate the perspective of educators. The survey [13] reveals that teachers are open to the idea of incorporating AR tools into the classroom and are motivated to expand their skills through continued professional development and training sessions. Despite limited prior experience with 3D modeling, the teachers expressed a strong desire to learn and even to contribute to the creation of new AR-based learning tools.

This is not present only in the AR field but many others in computer graphics, as teaching methods can be very diverse. As authors [11] point out: "Computer graphics is often regarded an exciting and enjoyable subject due to it combining technology, art and creativity". This is why many teachers have different approaches to teaching methods and even some smaller changes can be quite refreshing [14]. However, it is also important to follow what others are learning in computer graphics and fill the gaps of so large area as authors show in [2]. Further research shows that thematic AR and VR courses are still rare and fragmented [8] showing us that more good practice is needed in this area.

Since the past decade, AR applications were utilized for cultural heritage preservation. The most active countries in the field of AR use in cultural heritage are Italy, Greece, Spain and the UK [3].

Authors in [12] present an application that demonstrates the value of digital heritage in tourism. Here, the AR application exploits the concept of gamification in order to aid the process of learning history.

Author in [4] presents an AR application in an educational context for a design course. The app is found to aid students in concentrating, self-learning, and also to raise students' confidence.

Additionally, AR can provide more interactive and cooperative communication ways between students [9].

The authors compared mobile learning by means of an AR application on a tablet used in real fieldwork learning to traditional e-learning that takes place in a classroom with a regular desktop computer in using heritage elements of the city Santiago of Chile in [7]. The study found that the mobile learning process significantly enhanced educational outcomes.

In [1] the focus of the mobile AR multi-user game application is historical knowledge gain, and the application was used both in an indoor and outdoor setting.

3 Computer Graphics Course at the University of Sarajevo

Every student in their third year of the Bachelor program at the Faculty of Electrical Engineering, University of Sarajevo, developed an AR application as part of a Computer Graphics Course supervised by graduate teaching assistants. The goal of the course is for students to acquire theoretical and practical knowledge of the basics of computer graphics, such as raster and vector graphics, coordinate systems, geometric transformations, perspective projection, clipping, color theory, color systems, 3D modeling, scene illumination, texture mapping, shading and rendering, UI design and evaluation, and augmented reality.

The course was delivered over one semester and consisted of lectures (2 hours per week) and tutorials (45 minutes per week). Lectures have covered the following topics: raster and vector graphics, color theory and graphics design basics, 3D modeling techniques, basics of Human-Computer Interaction and usability, shading, clipping, anti-aliasing, basics of Unity 3D scripting, 3D scene illumination algorithms, and techniques, Virtual and Augmented Reality and Virtual Reality video.

In order to be able to create the requested lab projects students were offered the following tutorials presented by a graduate teaching assistant and had corresponding written materials for them:

1. Introduction to Blender (getting started with blender: user interface, creating simple objects)
2. Object modeling (using modifiers, proportional editing, tools for mesh editing)
3. Materials and texture mapping (creating new materials, using various built-in materials, shading editor, using an image as texture and UV editing, adding multiple materials to one object)
4. Lighting and rendering (using different built-in light sources, three-point lighting rule, emission shader, adding ambiance with High-Resolution Light (HDR) probes, camera settings, and using cycles as render engine)
5. Preparation of objects for exporting (simplification of objects using collapse, unsubdivide and planar modifiers)
6. Introduction to Unity (getting started with Unity: importing .fbx object to Unity, user interface, game, and scene view)
7. AR development (setting up Vuforia engine, image recognition)
8. Scripting in Unity (using colliders, adding multimedia content on object click such as 3D text, images and image gallery and video)

9. UI in Unity (creating canvas: adding buttons, panels, images)

10. Particle systems

4 Exams and marking methods

Students were marked during the semester on incremental work on their projects. Each mark for 5 separate assignments represented 2% of their final grade and was given for: a document describing the plan of project development, renders of the 3D object, successful image recognition, POIs and scripting, and UI layout and controls.

The final version of every student project went through a detailed revision that consisted of reviewing: the renders of the main object (10%), the model appearance after importing to Unity (5%), POIs and scripting (10%), UI (5%), application layout, ease of use, readability, color harmony, and creativity (10%). This final revision represented 40% of the final grade.

The exams consisted of multiple-choice questions about theoretical and practical knowledge obtained during the course. We organized two partial exams (20% each) and a final exam (40%). Students could pass the course if they obtained at least half of the maximum points at both partial exams or the final exam. They got two opportunities for remedial exams.

Marking creativity is particular for CG courses in our University, while other courses are marked through the standard computer science marking schemes. We introduced it to differentiate extremely talented and dedicated students from those who just learned the requested techniques and skills without new ideas, just to get a good mark. Creative students showed that such thinking yields better and more visually appealing results, which, in the end, sell the software product on the market and make it different from other similar products. Students had the freedom of choosing which POIs to use and the multimedia content that was linked to the object, as well as choosing to add elements to the object surroundings which were not a requirement. Additionally, they had complete creative liberty to choose the color, style, scale, layout, and function of every UI element. There were no predefined visual elements for the student applications. After the best projects were chosen to be integrated into the BH Heritage AR application the UI layout of the individual student projects was made uniform throughout the application.

5 AR lab project

At the beginning of the course, students chose an object which represents a cultural heritage site in Bosnia and Herzegovina. Following the selection process, students wrote detailed documents about their projects which included:

- basic information and photo documentation about the chosen object
- the description of points of interest (POI) objects that are thematically associated with the main object, and the multimedia content that is to be displayed when POI is clicked
- application design plan which includes a layout mockup

The students began the project by creating the 3D model of the chosen object, after which they added materials and textures, lighting, and finally made rendering images of the object from different angles in Blender. After the modeling phase, the object was exported in the .fbx format and imported to Unity. The next phase was adding image recognition using the Vuforia engine which included creating a postcard of the object, and adding it as an image target that would display the 3D object model after being scanned. The phase that followed consisted of adding POI objects and scripting the behavior of displaying the defined multimedia contents on object click. The final phase was creating the application UI, consisting of the basic information about the cultural heritage object, help and exit buttons.

The students were supervised by graduate teaching assistants each week in the form of laboratory sessions during which they showed their progress and presented problems and difficulties they encountered.

The goal of the students' projects was to create an AR application about a cultural heritage object that presents information about the object in a new, interesting and easily accessible way.

6 BH heritage AR application

Interaction methods in AR play a crucial role in making this experience possible, and there are several ways in which AR can be made interactive.

Touch and Gesture-Based Interaction

One of the most straightforward interaction methods in AR is touch and gesture-based interaction. This type of interaction involves using touch and gestures on a device, such as a tablet or smartphone, to control and interact with digital objects in the physical world. It can be a single-touch or multi-touch interaction [10]. This type of interaction is intuitive and can be especially useful for users who are not familiar with other types of input methods, such as a keyboard or mouse.

Voice-Based Interaction

Another popular interaction method in AR is voice-based interaction. This method allows users to interact

with digital objects using voice commands. Voice-based interaction can be especially useful for hands-free scenarios or when users are unable to physically touch their devices. Voice-based interaction can also be more natural and intuitive than touch or gesture-based interaction, as users can simply use their voice to control the AR experience.

Image and Object Recognition-Based Interaction

Image and object recognition-based interaction is another method oftentimes used in AR applications. This type of interaction involves using a device's camera to recognize images or objects in the physical world. Once an image or object is recognized, the device can display digital information or objects associated with it, allowing for an interactive experience [5]. This type of interaction is useful for applications such as product labeling, where digital information about a product can be displayed when the camera recognizes the product's label.

Spatial Mapping and Tracking

This type of interaction involves creating a digital representation of the physical environment, which can then be used to place and track digital objects in the real world. Spatial mapping and tracking allow for a more seamless AR experience, as digital objects can be placed in the real world and remain in place even as the user moves.

Motion Control Interaction

This type of interaction involves using the motion of the device, such as tilting or shaking, to control and interact with digital objects in the physical world. Motion control interaction can be especially useful for games and other interactive experiences that require physical movement.

These interaction models can also be combined to create a more seamless and engaging experience. This approach is much more common as it allows for more creativity and a better user experience. Our application was created in a similar way combining some of the known models into a fully structured application.

6.1 Application description

The application is available in two languages, Bosnian and English so that it can be used by locals and foreigners alike. After choosing the language, the user is presented with an interactive map of Bosnia and Herzegovina as shown in Figure 1, which showcases various objects of interest. Each object is represented by a photo and its name, allowing the user to obtain a visual and written overview of the object. The map also visually presents the various locations of the cultural heritage objects throughout Bosnia and Herzegovina.

Figure 1: Interactive map of Bosnia and Herzegovina

The user can then interact with the map by clicking on the individual objects, providing them with a more in-depth experience. By clicking on the object, the user is taken to a tracking screen, where they can use AR technology to scan an image called AR postcard related to the object (Figure 2) that the user obtained by downloading the postcard from the free link online and printing it.

Figure 2: AR postcard related to the chosen object (free to download)

This allows the user to experience an immersive and interactive view of the object, as shown in Figure 3, enhancing their overall understanding and appreciation of the object by seeing it in full, from every angle, which is sometimes hard to do on-site.

The touch and gesture-based interaction method is utilized with the POI objects by showing related multimedia content after the user clicks on the POI. POIs that are associated with objects, along with their accompanying multimedia content, provide additional and detailed information about them by utilizing the touch and gesture-based interaction method. When the users click on a specific POI, they are presented with the corresponding multimedia content, such as images, videos, or audio content (Figure 4), that provides further information and insight about the object. This enhances the overall understanding and

Figure 3: User view of the object after scanning the corresponding AR postcard

appreciation of the object, as well as offers a more comprehensive experience.

The application features a Help button, which upon acti-

Figure 4: Display of image as the linked multimedia content to the specific POI

vation, displays an image of POIs related to the object and the description of which multimedia content will be shown on click (Figure 5). This allows the user to easily navigate and understand the information available within the application.

Additionally, the application has an Information button

Figure 5: Display of POI images and description of multimedia content shown after clicking the help button

that, when selected, presents the user with basic information regarding the object and the students who developed the application. This allows the user to quickly and easily access a general overview and background of the object, thereby enhancing their overall understanding and appreciation of the object, as well as learning about the individuals behind the creation of the application.

6.2 User experience evaluation

To check if this approach improves the overall experience of learning about cultural heritage objects, we conducted a User eXperience evaluation study. The results of this evaluation will contribute to the development of AR applications for cultural heritage preservation by determining which introduced elements are beneficial and which can be discarded or changed. Interaction methods we used in this application were a mix of Touch and Gesture-Based interaction since users are navigating the application on their mobile phones touch screen and Image and Object Recognition with Motion Control interaction. Users are able to see the object on the screen and even walk around it while it is being tracked with the camera so we could track the reaction of the users and how do they navigate the application itself. The research question of our user experience evaluation was the following:
Does the BH Heritage AR application increase the overall experience of learning about cultural heritage buildings?
The experiment included 17 participants engaged by invitation, all of which were BH citizens. Subjects were invited with balanced demographic features in mind, regarding gender, IT background, and hardware possession. Immediately after evaluating the application, participants were invited to fill out the questionnaire. The questionnaire was organized in three sections:

- Introduction, containing questions related to demographic data
- Section dedicated to measuring (1) the increase in cultural heritage knowledge, (2) the benefits of added content, and (3) application intuitivity.
- Additional comments and critiques regarding their evaluation.

The measurement of the variables of interest was performed using a 5-point Likert scale with the following structure: 2 items for the increase in cultural heritage knowledge, 3 items for experience, and one item for application intuitivity.

The results of the evaluation are presented in Table 1, with an average value and standard deviation per each question: Regarding the measurement of our ultimate goal, user experience, we hypothesized that users would have an overall more immersive experience with the application and would have an easy time learning information about various cultural heritage sites, with the intuitive UI in the application. Therefore, the expected outcomes were high values for Increase in cultural heritage Knowledge, Benefits of Added content, and Application Intuitivity.

		Avg.	St. dev.
K1	On a scale of 1-5, rate your knowledge about the chosen cultural heritage object before using the app	2.88	0.99
K2	On a scale of 1-5, rate your knowledge about the chosen cultural heritage object after using the app	4.41	0.79
A1	I think that POIs contribute to the interest of the application and individual objects	4.71	0.59
A2	I think that the info button helped in learning new information about the object	4.71	0.59
A3	I think that the help button helped in easier determination of POIs and their related multimedia content	4.88	0.33
I1	I think that working in the application was intuitive	4.82	0.39

Table 1: Questionnaire structure and overview of statistical measures. Questions related to the Increase in CH Knowledge are marked by Kx, questions related to the benefits of Added content are marked by Ax and questions related to app Intuitivity are marked with Ix. The responses were conveyed based on a 5-point Likert scale ranging from strongly negative to strongly positive.

The results strongly indicate that the use of the BH Heritage AR application increases the overall experience of learning about cultural heritage objects by immersing users with added content and an easy-to-use interface.

The results of this experiment indicate that the utilization of an AR application with supplementary multimedia content and a user-friendly interface enhances the learning experience of its users. Additionally, it is evident that all the introduced elements to the BH Heritage AR application enhance and are beneficial to the overall user experience. The POIs contributed to the interest of the application and individual objects by providing users with a variety of ways in which to interact with the object, be it by scrolling through a gallery of pictures, reading a text about objects, prominent figures, and historical events linked to the main object, seeing videos about the object or hearing related audio content. The information button further aided in the learning of new information about the objects by providing them in concise, easily understood segments. And finally, the help button helped in easier determining POIs and their related multimedia content by providing a clear visual aid seen in Figure 5.

The participants have not expressed any dissatisfaction or noted any faults in the application. Future evaluations should include more participants, both local and foreign to provide for a more strict statistical analysis of the results.

7 Conclusions

In this paper, we presented a novel methodology for teaching computer graphics, consisting of including basic artistic knowledge such as color theory and graphics design basics and developing specific skills in 3D modeling and AR applications development. The student surveys and marks, as well as the results of the BH Heritage AR application's initial user evaluation, show good results of this methodology. Out of 83 students attending the course, in the end, we had sixteen 10s, thirty-eight 9s, twenty-five 8s, four 7s, and no 6s or fails (passing marks are from 6-10). Students appreciated very much that their lab projects did not only serve for passing the course, but they remained as useful parts of a digital cultural heritage application that informs about BH cultural heritage and attracts visitors to cultural monuments in Bosnia and Herzegovina.

This is one of the very few courses in our Computer Science Department that encourages and develops the creativity of students, instead of teaching them dry algorithms and marking them according to a standard computer science pattern. This way of teaching CG encouraged many students to choose CG elective courses at the Masters level and expand their knowledge with computer animation, game development, and Virtual Reality.

In the future, we will expand the topic of the lab projects to other AR technology applications in education. For example, our students can create AR applications for learning biology, physics, mathematics, and geography that can be used in elementary and high schools to make the learning process more interesting.

References

[1] Anastassia Angelopoulou, Daphne Economou, Vassiliki Bouki, Alexandra Psarrou, Li Jin, Chris Pritchard, and Frantzeska Kolyda. Mobile augmented reality for cultural heritage. In *Mobile Wireless Middleware, Operating Systems, and Applications: 4th International ICST Conference, Mobilware 2011, London, UK, June 22-24, 2011, Revised Selected Papers 4*, pages 15–22. Springer, 2012.

[2] Dennis G. Balreira, Marcelo Walter, and Dieter W. Fellner. What we are teaching in Introduction to Computer Graphics. In Jean-Jacques Bourdin and Amit Shesh, editors, *EG 2017 - Education Papers*. The Eurographics Association, 2017.

[3] Răzvan Gabriel Boboc, Elena Băutu, Florin Gîrbacia, Norina Popovici, and Dorin-Mircea Popovici. Augmented Reality in Cultural Heritage:

An Overview of the Last Decade of Applications. *Applied Sciences*, 12(19):9859, 2022.

[4] Yuh-Shihng Chang. Applying the arcs motivation theory for the assessment of ar digital media design learning effectiveness. *Sustainability*, 13(21):12296, 2021.

[5] Juergen Gausemeier, Juergen Fruend, Carsten Matysczok, Beat Bruederlin, and David Beier. Development of a real time image based object recognition method for mobile AR-devices. In *Proceedings of the 2nd international conference on Computer graphics, Virtual Reality, visualisation and interaction in Africa*, pages 133–139, 2003.

[6] Juan Camilo Gonzalez Vargas, Ramon Fabregat, Angela Carrillo-Ramos, and Teodor Jové. Survey: Using augmented reality to improve learning motivation in cultural heritage studies. *Applied Sciences*, 10(3):897, 2020.

[7] Jorge Joo-Nagata, Fernando Martinez Abad, José García-Bermejo Giner, and Francisco J García-Peñalvo. Augmented reality and pedestrian navigation through its implementation in m-learning and e-learning: Evaluation of an educational program in Chile. *Computers & Education*, 111:1–17, 2017.

[8] Alexandra Klimova, Anna Bilyatdinova, and Andrey Karsakov. Existing teaching practices in augmented reality. *Procedia Computer Science*, 136:5–15, 2018. 7th International Young Scientists Conference on Computational Science, YSC2018, 02-06 July2018, Heraklion, Greece.

[9] Enrui Liu, Changhao Liu, Yang Yang, Shanshan Guo, and Su Cai. Design and implementation of an augmented reality application with an English Learning Lesson. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 494–499. IEEE, 2018.

[10] Theofilos Papadopoulos, Konstantinos Evangelidis, Theodore H Kaskalis, Georgios Evangelidis, and Stella Sylaiou. Interactions in Augmented and Mixed Reality: An Overview. *Applied Sciences*, 11(18):8752, 2021.

[11] Thomas Suselo, Burkhard Wünsche, and Andrew Luxton-Reilly. The journey to improve teaching computer graphics: A systematic review. 12 2017.

[12] Kian Lam Tan and Chen Kim Lim. Digital heritage gamification: An augmented-virtual walkthrough to learn and explore historical places. In *AIP conference proceedings*, volume 1891, page 020139. AIP Publishing LLC, 2017.

[13] Stavroula Tzima, Georgios Styliaras, and Athanasios Bassounas. Augmented reality applications in education: Teachers point of view. *Education Sciences*, 9(2):99, 2019.

[14] Jun Zhou, Ming Ye, and Chun-Lun Huang. Reform of computer graphics teaching method. volume 3, pages 222 – 225, 08 2010.

Improving the VR Experience in a Densely Populated Molecular Environment

Eva Boneš*
Supervised by: Ciril Bohak†

Faculty of Computer and Information Science
University of Ljubljana
Ljubljana / Slovenia

Abstract

Virtual Reality (VR) technology has the potential to provide a highly immersive experience when navigating through densely populated environments, such as macromolecular models. However, current solutions lack the ability to provide both a sense of immersion and a clear overview of the environment. In this study, we propose improvements to the existing systems for an automatically generated guided tour through a molecular model that address these limitations and enhance the overall experience. We propose a sparsification technique based on implicit quadric equations, providing seamless transitions between a closed, enveloping experience and an open, spacious one for improved spatial awareness. Additionally, we give users more control over the tour to alleviate feelings of being overwhelmed in a crowded environment. Our subjective, qualitative results indicate that these methods can improve the overall VR experience compared to existing solutions and provide a more enjoyable tour. However, more research is needed to further enhance the experience and provide an even more engaging and informative guided tour through densely populated molecular environments within VR.

Keywords: sparsification, virtual reality, dense environment, molecular model

1 Introduction

The COVID-19 pandemic has increased public awareness and interest in molecular assemblies. Scientists now have better ways to depict biological structures using 3D modeling approaches, but the use of scientifically accurate models in educating the general public remains limited. One of the reasons being that without proper explanation, the resulting images or videos are often confusing for those without scientific knowledge and can only be fully grasped with some sort of guidance from an expert. This highlights the need for a new approach to science communication, one in which the user can visualize, investigate, and ultimately understand a complex scene, such as a molecular model, in a non-expert setting. To address this issue, researchers have recently developed new systems [6, 1] that allow the creation of interactive scientific documentaries using multi-scale, multi-instance, and dense 3D molecular models.

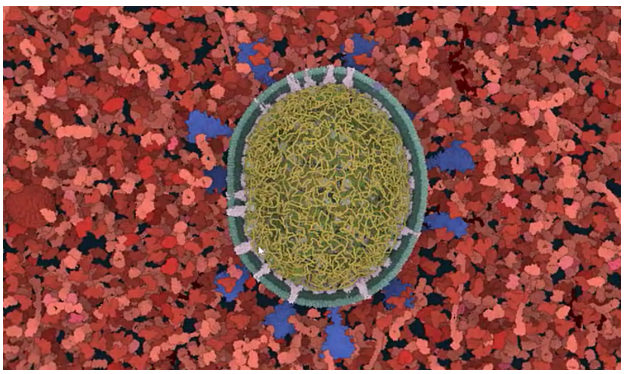
These systems provide a journey through a very crowded environment filled with proteins, lipids, fibers, genetic material, and other structures of various colors, shapes, and sizes. For instance, one of the models shows a SARS-CoV-2 virus with all its parts, such as the spike proteins, the envelope, the membrane, and the RNA, floating in a ”box” filled with blood plasma and all its components. As the camera guides us to key locations within the model, a voice-over explains what these structures are and what functions they perform.

While both systems provide a unique, interesting, and educational experience of a guided tour through a molecular model, there are some issues regarding both of them that were pointed out during a user study conducted by the authors of Nanotilus [1]. One key distinction between the two, from a user experience perspective, is the way each system handles a crowded environment, specifically the sparsification of it. Molecumentary [6] uses a vertical clipping plane set at a particular distance from the viewer and thus avoiding collisions with any objects. Nanotilus on the other hand uses a camera-centric sparsification technique that removes instances in an ellipsoid shape around the user. The different sparsification techniques can to some extent be seen in Figure 1.

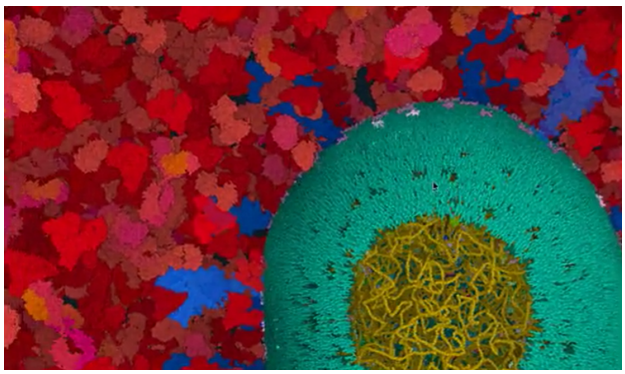
Users reported that with Molecumentary, they did not see a huge added value in being inside a VR as opposed to seeing a video play on a screen. The clipping plane sparsification made the scene seem flat. In contrast, Nanotilus provided a more immersive experience within a 3D model, but at the cost of compromised spatial awareness. Besides that, users also reported feeling overwhelmed with all the information presented to them at the same time and not having enough control to look at and explore the space around them.

In this paper, we present our efforts towards improving

*eb1690@student.uni-lj.si
†ciril.bohak@kaust.edu.sa



(a) Molecumentary



(b) Nanotilus

Figure 1: Different sparsification techniques from Molecumentary (a) and Nanotilus (b) on a SARS-CoV-2 model surrounded by blood plasma. We can see the vertical clipping plane through the whole scene in Molecumentary and the more up-close, full scene in Nanotilus. The difference is much more apparent inside the VR.

the VR experience inside a guided tour through a crowded molecular model, namely:

- A novel sparsification approach based on quadric surfaces that combines the immersive experience with good spatial awareness.
- An innovative ”landing plane” sparsification technique to better connect the real-world standing environment with the VR flying experience.
- Limited user interaction to mitigate the overwhelming effects of excessive information.

In Sections 2 and 3, we introduce the field, outline the foundational systems on which our research is based, and present an analysis of a user study that highlights their limitations. In Section 4, we elaborate on the new techniques we have created to overcome these limitations. We then evaluate the effectiveness of these methods in Section 5, by analyzing their impact on the perception of crowded environments and the user experience of navigating through them. Finally, in Section 7, we provide a summary of our work and propose potential areas for future research and improvement.

2 Related Work

Visualization in scientific communication

Visualization plays a critical role in scientific communication, used for experiment validation and data exploration. Many people believe that its significance will increase even further in the future, with the potential to lead to new breakthroughs in research [2]. For many years, visualization has been recognized as an important tool for promoting understanding of science [5], particularly in presentations of the cosmo- or micro-scale world and associated processes, as people can rely only on illustrations, interactive simulations, and animations as visual aids. Given that, virtual reality (VR) has emerged as a powerful tool that is more than simply a better version of 3D visualization [16] as it enables us to form a conception of and understand things that 2D or even desktop 3D graphics cannot. As a result, VR has been applied in various fields such as biology [18, 19, 12], medicine [9, 15], and other areas [13]. Although classical VR applications offer a valuable learning experience by enabling users to examine objects from a different perspective, they often lack the necessary guidance to enhance their understanding of complex structures. Tools such as the one described in [14] have been developed to allow the creation of VR applications that incorporate storytelling elements, either through written or spoken narration. However, creating a narrated story manually requires substantial time and expertise, leading to the development of new methods like the ones our work is based on and are further discussed in Section 3, which aim to automate the creation process for narrated VR tours.

Occlusion management in VR

As we delve into the microscopic world through molecular models, we discover that the environment is densely populated with elements. To realistically depict this information in our models and VR experiences, while also giving the user some breathing room and preventing collisions, we need some sort of occlusion management. While there are numerous different approaches to this, falling into five design patterns as identified by Elmqvist and Tsigas [4], researchers most often deal with it using sparsification, which involves removing some of the instances inside the model. One way to manage occlusion is to use a cutting plane that divides the scene into visible and hidden parts. However, to keep important objects visible, an extension that exempts certain objects from being cut away was used by authors of Molecumentary in this and previous [7] research. Le Muzic et al. [10] proposed a technique called the visibility equalizer, which also considers

the type of instance being removed, enabling the removal of more abundant objects. A downside to using clipping planes is that they can remove large portions of visualization which reduces the sense of crowdedness in one part of the environment. To avoid this, smart sparsification methods that remove individual instances using different importance measures have been proposed, such as the one presented in [17] by Lesar et al. Another technique for managing occlusion was proposed by Elmqvist [3], who suggested distorting space with a spherical force field that repels objects around the 3D cursor. Similarly to this, Nanotilus uses a centralized sparsification technique that removes instances in concentric zones around the user. In addition, Nanotilus utilizes an automatic visibility equalizer that guides the sparsification process using heuristics.

No single method is inherently superior to the other. The method we choose depends entirely on our use case. For certain visualizations, a clear cutting plane may provide a better model overview, while in other cases, we may prefer to experience the sense of crowdedness. As a guided tour is dynamic, we encounter various scenarios within it, which may call for different sparsification techniques depending on what we aim to see. To address this, we propose a technique that combines both a selective clipping plane and local centralized sparsification.

3 Background

We based our work on two recently published systems, both providing a VR experience of narrated documentaries of molecular models.

Kouřil et al. introduced the concept of adaptable documentaries in their work on **Molecumentary** [6]. They presented a system comprising of real-time visualization, automated exploration, and synthetic commentary to create an adaptable and automated narrated tour. The system consists of two steps: story graph foraging and narrative synthesis.

The first step involves automatically compiling information about the biological model into a story graph that holds model elements, their relationships, and verbal descriptions. The second step utilizes the story graph to generate a sequence of story elements with corresponding commentary using text-to-speech technology. Subcomponents of the model are brought to life using camera animations and occlusion management. The order of model elements shown is determined by either an algorithmic approach that produces a self-guided documentary or by following a storyline supplied as a written text input, making the molecumentary more responsive to user choices. The authors' approach automates the entire process without the need for a domain expert's involvement.

Alharbi et al. published **Nanotilus** [1] as a follow-up to previous research, where they aimed to address the limitations of existing visibility and occlusion management strategies for navigating dense 3D structures. They introduced a new sparsification method that offers an endoscopic inside-out perspective, maintaining the immersive quality of virtual reality, instead of the previously used outside-in view. Since this is the aspect that we focused more on in our paper, the mechanics of it are described in more detail in Section 4.1.

The authors also changed the journey planning part of the pipeline, which roughly replaces the story graph foraging in Molecumentary. They included the use of void spaces as the navigation through them minimizes the number of instances to sparsify, preserving the model realism and increasing the participants' immersion. The process of journey planning starts with identifying the instances that the journey should visit and then generates a path that connects them while traversing through the void spaces.

A user study comparing the two systems was conducted with 29 participants who viewed guided tours of mesoscopic biological models of SARS-CoV-2 and HIV using both of them. The primary aim of the study was to evaluate the user experience regarding various sparsification and navigation techniques. User feedback was collected on engagement and overall user experience, spatial understanding of the displayed structures, traversal of the virtual environment, and the ability to orient oneself in the environment and follow the story.

Participants in the user study preferred the inside-out animation of Nanotilus for its immersive quality, while also noting that it could be difficult to see the whole structure at times by commenting "*[...] its more difficult to understand how all the viral components are positioned.*" and "*Sometimes in the inside view it is too close to see the whole structure.*". On the other hand, Molecumentary was criticized for lacking immersiveness with comments such as "*I honestly feel that this experience doesn't add much to what you would get just looking at a video on a good screen.*" and "*[...] it was not that an immersive experience as the inside-out animation, [...]*". However, users appreciated Molecumentary's outside-in approach, which provides an overview of the virus and displays each component clearly. To balance these strengths, we aimed to create a sparsification method that incorporates both approaches at appropriate points during the tour.

4 Improving the VR experience

The primary area of possible improvement that we focused on was sparsification, as it was one of the key distinctions between the two systems and was a significant point of feedback in the user study. Additionally, we tried to add a bit more control over the tour to the user in order to mitigate feelings of being overwhelmed, which was another issue mentioned in the study.

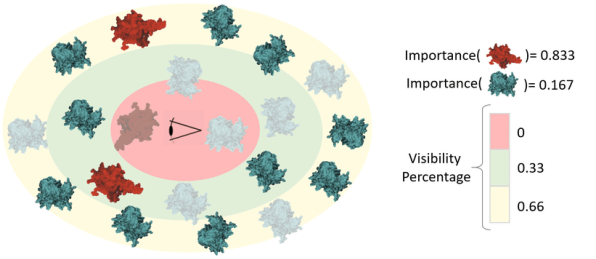
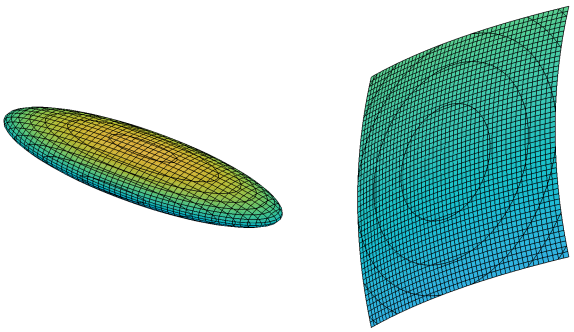


Figure 2: The illustration shows the sparsification procedure, where instances are sparsified based on both importance of the type (cyan or red) and their shell membership (source: [1]).



(a) $10x^2+10y^2+z^2-1=0$ (b) $x^2+y^2-10z=0$

Figure 3: Ellipsoid (a) and paraboloid (b) drawn using implicit quadric equations.

4.1 Sparsification of crowded environments

The authors of Nanotilus focused on developing a local, camera-centric sparsification procedure that allows the viewer to go through a densely packed scene without colliding with the instances. It reduces structural occlusion and provides the user with endoscopic views that convey scene crowdedness.

The sparsification is controlled with three nested and concentric shells surrounding the camera, each with its own visibility percentage. The innermost shell is responsible for hiding instances that may collide with the camera and is set with a visibility rate of 0.0. The middle and outer shells have their default visibility rates set to 0.33 and 0.66 respectively, making some instances visible and some removed. A mathematical description of the shell geometry is used to determine whether an instance in a scene is a member of a shell or not.

The sparsification is then done in two phases. First, the visibility values of instances that should be hidden are updated based on the visibility percentage of the shells. The algorithm then checks for the number of invisible instances inside the shells and assigns a weight to each visible instance, which represents its priority to be hidden. The weight is affected by the distance from the instance to the camera and the importance of the type. The instances with the smallest weight are hidden until a certain threshold is achieved. In Figure 2, we can see an example of selecting instances for sparsifying.

In our implementation, we took the main logic from Nanotilus that is described above, however, the shell shape and implementation of it were modified to facilitate utilization for both capsule-type sparsification and the clipping plane. This was achieved using quadric surfaces, which will be described in detail next.

4.1.1 Quadric surfaces

A quadric surface is a type of surface in three-dimensional space that can be defined by a quadratic equation. Quadratic equations are a type of algebraic equation that can be written in the form:

$$Ax^2+By^2+Cz^2+Dxy+Exz+Fyz+Gx+Hy+Iz+J=0.$$

where x, y, and z are the coordinates of a point on the surface and A, B, C, D, E, F, G, H, I and J are constants. The constants in a quadric equation are real numbers and at least one of them has to be non-zero. These surfaces can be classified into different types depending on the values of the constants in the equation, such as spheres, cylinders, cones, etc.

Representing a capsule shape that was already used in Nanotilus was straightforward, as we just used the implicit equation for an ellipsoid:

$$\frac{x^2}{A^2}+\frac{y^2}{B^2}+\frac{z^2}{C^2}=1 \Rightarrow Ax^2+By^2+Cz^2-J=0.$$

For representing the flat clipping plane, we decided on a paraboloid, because the parameters in the equation can easily be set up such that the resulting surface resembles a plane:

$$z=x^2+y^2 \Rightarrow Ax^2+By^2-Iz=0.$$

Both surface representations can be seen in Figure 3.

The logic behind choosing instances that had to be removed remained unchanged from Nanotilus. Individual instance memberships were determined through a simple calculation: if the surface equation with the instance's position as x, y, and z resulted in a value equal to or less than 0, the instance was considered part of the shell; otherwise, it was not. In Figure 4, we show the instances inside a simple molecular model, colored based on their corresponding shell membership. The innermost shell is colored green, the middle shell is blue, and the outermost shell is red.

When using the ellipsoid shape, what we call the *capsule mode*, we kept the three shells with sparsification ratios of 0, 0.33, and 0.66. During the *plane mode* when we used the paraboloid shape, we only used one shell with a sparsification ratio of 0, meaning everything inside was sparsified.

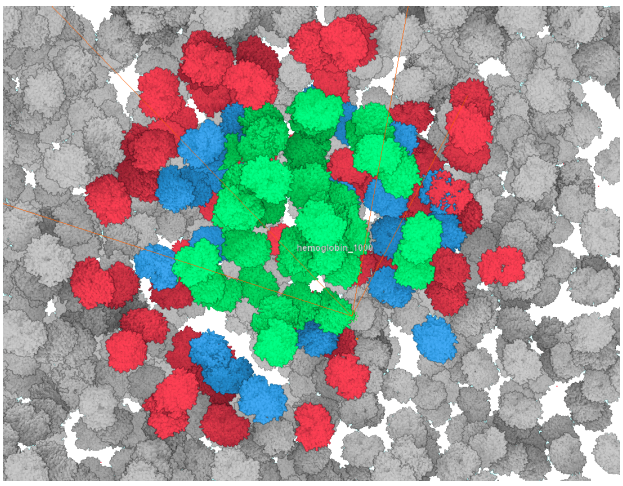


Figure 4: Hemoglobin instances in a simple box molecular model colored according to shell membership.

4.1.2 Transitioning

The smoothness of the visual experience in VR is essential to provide a comfortable and natural-feeling environment. This is because changes in the real world happen gradually, not abruptly, and sudden changes in the surroundings can result in discomfort. To recreate this natural progression in VR, we implemented smooth transitions between the two sparsification modes. Reflecting on how the transitions feel to the user, we refer to the transition from capsule to plane mode as *opening up* and the reverse transition from plane to capsule as *closing down*. Our implementation of the surface shapes with parametric implicit equations made calculating the steps between them trivial. For each of the parameters, we divide the range between the start and end values into the desired number of steps. Combined with a gradual fade-out of instances as their membership changes, this produces a smooth and visually appealing transition between the shapes.

In reviewing the results of the user study and conducting our own testing of the systems, we tried to determine the optimal points for transitions and when certain sparsification modes were most appropriate within the documentary. We concluded that while the user is moving through the scene from one target to another, using the capsule mode makes for a more immersive experience. When stopping at the target and entering a so-called *focus scene*, the plane mode provides a better overview of the scene and gives the user a good perception of the environment. We, therefore, implemented opening up the sparsification immediately after stopping at the target of the focus scene. The sparsification then closes down when the user starts moving toward the next target. Another transition happens at the beginning of the tour when we want to introduce the model that we're showing to the user, so we start with the plane mode and transition to the capsule mode before moving forward. Although we found that the capsule mode is preferable while moving in most cases, we observed that when the camera moves backward, the close proximity of the instances on the left and right and new instances popping into the scene in front of the user can cause some discomfort. To address this, we experimented with opening up the sparsification after we detect that the camera started moving in reverse.

4.1.3 Landing plane

Another feature that we wanted to test for increasing the enjoyment inside VR is to add a horizontal plane as a floor, providing the user with a sense of stability and grounding. Our hypothesis is that adding a floor plane when the user is standing still would help bridge the gap between virtual reality and real-world experiences, as most of the time in reality we are standing on a two-dimensional plane. By creating a more realistic virtual environment, we aim to increase the user's enjoyment and make the virtual experience feel less alien.

In order to maintain the density of the scene and avoid adding unnecessary, false or misleading elements to the model, we came up with a solution that utilizes existing instances in the scene to create the floor. This way, the floor landing plane is just another form of sparsification, achieved by removing instances that are located above a horizontal plane and were previously already selected to be removed through earlier phases of sparsification.

The floor under your feet makes sense only when you are standing still, not when you are moving around in the capsule. Therefore, we integrated the floor sparsification with the plane mode as can be seen in Figure 5, meaning that it toggles on and off with a fading effect whenever there is a transition between the sparsification modes. Since the transition takes place whenever we arrive to the target instance inside a capsule mode, the floor sparsification then has a feeling of landing that capsule. That's why we refer to it as a landing plane.

4.2 Bringing more control to the user

Being in a VR environment, where everything is constantly moving and the user has no control over their direction or speed, can be quite disorienting. To enhance the user's comfort and overall experience, it's crucial to give them some degree of control.

As a simple but effective solution, we added the option to pause the guided tour at any point during the experience. This allows the user to take a break, look around, gather more information about what interests them, or simply pause for a moment. They can also switch to plane mode if they feel lost, claustrophobic, or overwhelmed while in capsule mode. Once they're ready to continue the tour, the sparsification automatically returns to the previous mode and the guided tour resumes.

This feature provides a much-needed sense of control and comfort, making the overall experience more enjoyable and less overwhelming.

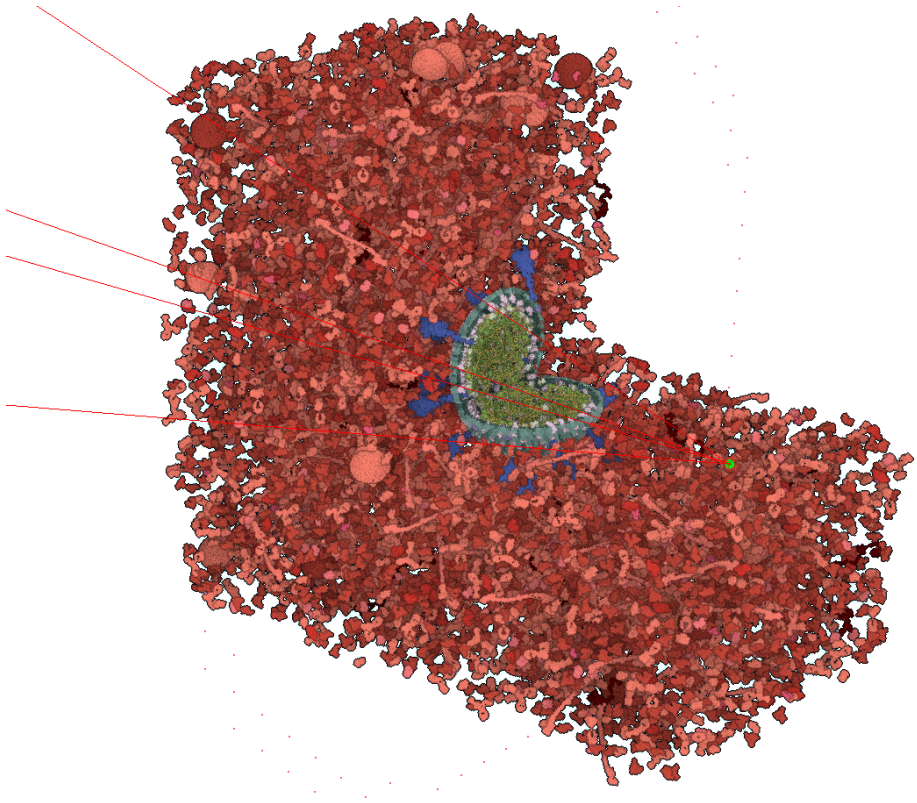


Figure 5: Floor and plane sparsification on a SARS-CoV-2 model as seen from a third-person perspective. The green sphere presents the location of the camera and the red lines the viewing frustum.

5 Results

Our implementation of the system is built upon the Nanotilus platform, which utilizes the Marion library [8]. The application is implemented in C++, OpenGL, and GLSL, leveraging the Qt library, and employs GLSL compute shaders. For the presentation of the resulting guided tours, we utilized both the HTC Vive Cosmos and Oculus Quest Pro headsets. The support for OpenVR was integrated into Nanotilus for use with the HTC headset, but we encountered limitations when using the Oculus. This, along with the deprecation of OpenVR, has motivated us to explore the implementation of OpenXR for improved performance and compatibility in the future.

For developing and testing, we used the SARS-CoV-2 model that was created using a statistical and rule-based modeling approach [11]. The application was run on an Nvidia GeForce RTX 4090 graphics card, AMD Ryzen Threadripper PRO 3995WX 64-core processor, Windows 10, and Qt 5.15.2. with an average framerate of about 25 FPS per eye. Despite being below the typical FPS recommendations for VR, which call for a minimum of about 45 FPS to provide a reasonable experience, and at least 90 FPS for optimal performance, the average framerate was sufficient for us to test the design and evaluate the improvements made to the system. However, we are committed to improving the framerate and overall performance in the near future.

6 Discussion

We tested our implementation and qualitatively evaluated the improvements made to the VR experience.

Our testing revealed that the combination of sparsification modes enhances the overall experience by offering the best of both worlds. The user can fully immerse themselves in the model, feeling connected with the scene as they move from one instance to the next, while still maintaining a sense of their position within the model in the focus scene. We found the transitions between modes visually pleasing, with closing down feeling like you get hugged back into the scene and opening up feeling like taking a deep breath.

We still have to experiment to determine the optimal moments for switching between sparsification modes and when each mode would be most effective during the tour. Additionally, we may want to test out different surface parameters to find the best option. Furthermore, we need to conduct additional tests on the use of the landing plane within the tour to assess whether it evokes the hypothesized feelings when users have the ability to walk around. We found that when simply being in the scene having a plane underfoot doesn't necessarily enhance the experience, but it also doesn't worsen it. The density of the

model is another crucial factor that affects the success of the experience, as less dense models may not provide enough instances to make up the floor, so we may need to investigate this further.

With the additional option of pausing and resuming the tour at any time, as well as the ability to toggle between sparsification modes, the guided tour moves a step closer to becoming a self-exploratory type of documentary. The user has slightly greater control and freedom to personalize their experience, which leads to a more enjoyable journey. Moreover, the pause feature gives the user the ability to take a break, explore the environment, and gather information at their own pace. This creates a more relaxed experience, allowing the user to fully immerse themselves in the VR world.

The control that we give to the user is fairly limited at this point, but it is an important aspect that we will focus on in the future.

Overall, after testing out the system, we have observed that the implementation of the combination of sparsification modes and the added pause feature has indeed had a positive impact on the overall VR experience. However, to verify and validate these findings, we plan to carry out a comprehensive user study that will be conducted on a larger scale. This will allow us to obtain a more accurate and representative evaluation of the impact of our work on the VR experience.

7 Conclusions

In this paper, we proposed improvements to the VR experience for guided tours through crowded molecular models. Our improvements aimed to address the limitations of the existing systems, including the sparsification of crowded environments and the overwhelming effects of excessive information. We proposed a novel sparsification approach based on quadric surfaces, a landing plane sparsification technique, and gave more control to the user during the tour. Our evaluation showed that our methods improved the perception of crowded environments and the user experience of navigating through them.

In the future, we plan to conduct a comprehensive user study to better understand the strengths and limitations of our system. Based on the results of this study, we will focus on improving the technical aspects of our tour implementation to enhance the user experience. This includes optimizing our code to increase the framerate and transitioning from OpenVR to OpenXR to reduce overhead.

Additionally, we want to focus more on adding the exploration aspect to the now completely guided tour as these features have the potential to greatly increase the usability and impact of this type of science communication. To achieve this, we plan to add new and exciting features such as the ability for users to change the story while they are on the tour, explore structures up close by walking around, and select and learn more about the structures that interest them. These enhancements will allow users to have a more interactive and personalized experience while exploring the wonders of science.

References

[1] Ruwayda Alharbi, Ondrej Střnad, Laura R. Luidolt, Manuela Waldner, David Kouřil, Ciril Bohak, Tobias Klein, Eduard Gröller, and Ivan Viola. Nanotilus: Generator of immersive guided-tours in crowded 3d environments. *IEEE Transactions on Visualization and Computer Graphics*, 2021.

[2] Mohamed El Beheiry, Sébastien Doutreligne, Clément Caporal, Cécilia Ostertag, Maxime Dahan, and Jean-Baptiste Masson. Virtual reality: Beyond visualization. *Journal of Molecular Biology*, 431(7):1315–1321, 2019.

[3] Niklas Elmqvist. Balloonprobe: Reducing occlusion in 3d using interactive space distortion. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, volume 2006, pages 134–137, 11 2005.

[4] Niklas Elmqvist and Philippas Tsigas. A taxonomy of 3d occlusion management for visualization. *IEEE transactions on visualization and computer graphics*, 14:1095–109, 09 2008.

[5] Kakonge John Gilbert, Miriam Reiner, and Mary Nakhleh. *Visualization: Theory and Practice in Science Education*. Springer Dordrecht, 01 2008.

[6] David Kouřil, Ondrej Střnad, Peter Mindek, Sarkis Halladjian, Tobias Isenberg, Eduard Gröller, and Ivan Viola. Molecumentary: Adaptable narrated documentaries using molecular visualization. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2021.

[7] David Kouřil, Tobias Isenberg, Barbora Kozlíková, Miriah Meyer, M. Eduard Gröller, and Ivan Viola. Hyperlabels: Browsing of dense and hierarchical molecular 3d models. *IEEE Transactions on Visualization and Computer Graphics*, 27(8):3493–3504, 2021.

[8] Peter Mindek, David Kouřil, Johannes Sorger, Daniel Toloudis, Blair Lyons, Graham Johnson, M. Eduard Gröller, and Ivan Viola. Visualization multi-pipeline for communicating biology. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):883–892, 2018.

[9] Rakesh Mishra, Krishna M.D, Giuseppe Emmanuele Umana, Nicola Montemurro, Bipin Chaurasia, and

Harsh Deora. Virtual reality in neurosurgery: Beyond neurosurgical planning. *International Journal of Environmental Research and Public Health*, 19:1719, 02 2022.

[10] Mathieu Le Muzic, Peter Mindek, Johannes Sorger, Ludovic Autin, David S. Goodsell, and Ivan Viola. Visibility equalizer cutaway visualization of mesoscopic biological models. *Computer Graphics Forum*, 35, 2016.

[11] Ngan Nguyen, Ondřej Strnad, Tobias Klein, Deng Luo, Ruwayda Alharbi, Peter Wonka, Martina Maritan, Peter Mindek, Ludovic Autin, David S. Goodsell, and Ivan Viola. Modeling in the time of covid-19: Statistical and rule-based mesoscale models. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):722–732, 2021.

[12] Eric Pettersen, Thomas Goddard, Conrad Huang, Elaine Meng, Greg Couch, Tristan Croll, John Morris, and Thomas Ferrin. Ucsf chimerax: Structure visualization for researchers, educators, and developers. *Protein Science*, 30, 09 2020.

[13] Jaziar Radianti, Tim A. Majchrzak, Jennifer Fromm, and Isabell Wohlgenannt. A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda. *Computers & Education*, 147:103778, 2020.

[14] Almir Santiago, Paulo Sampaio, and Luis Fernandes. Mogre-storytelling: Interactive creation of 3d stories. In *Proceedings of 2014 XVI Symposium on Virtual and Augmented Reality*, pages 190–199, 05 2014.

[15] Morimoto Tadatsugu, Takaomi Kobayashi, Hirohito Hirata, Koji Otani, Maki Sugimoto, Masatsugu Tsukamoto, Tomohito Yoshihara, Masaya Ueno, and Masaaki Mawatari. Xr (extended reality: Virtual reality, augmented reality, mixed reality) technology in spine medicine: Status quo and quo vadis. *Journal of Clinical Medicine*, 11:470, 01 2022.

[16] Andries van Dam, Andrew Forsberg, David Laidlaw, Joseph Jr, and Rosemary Michelle Simpson. Immersive vr for scientific visualization: A progress report. *IEEE Computer Graphics and Applications*, 20:26–52, 12 2000.

[17] Žiga Lesar, Ruwayda Alharbi, Ciril Bohak, Ondřej Strnad, Christoph Heinzl, Matija Marolt, and Ivan Viola. Volume conductor: Interactive visibility management for crowded volumes. *The visual computer*, pages 1–16, 2023.

[18] Jimmy Zhang, Alex Paciorkowski, Paul Craig, and Feng Cui. Biovr: a platform for virtual reality assisted biological data integration and visualization. *BMC Bioinformatics*, 20, 02 2019.

[19] Xiang Zhou, Liyu Tang, Ding Lin, and Wei Han. Virtual & augmented reality for biological microscope in experiment education. *Virtual Reality & Intelligent Hardware*, 2(4):316–329, 2020.

Computer Vision in Medicine

GrowCut under StudierFenster

Alessandra Masur*
Supervised by: Jan Egger†

Institute of Computer Graphics and Vision (ICG)
Graz University of Technology
Graz / Austria
Institute for AI in Medicine (IKIM)
University Hospital Essen
Essen / Germany

Abstract

Segmentation is a crucial procedure in medical image analysis. The usage of automatic algorithms in this field is an attractive alternative to manual segmentation. One promising semi-automatic segmentation tool is the GrowCut algorithm, which allows n-dimensional image segmentation, providing interactive and dynamic features. Currently, using the GrowCut algorithm for medical image segmentation with a user interface is only possible via medical image analysis software, making it device- and platform-dependent. The GrowCut algorithm without a user interface is available via various implementations but requires a lot of technical knowledge of the user.

The aim of this contribution is to provide a user interface for the GrowCut algorithm on the basis of a web application. This is achieved by implementing an adapted version of the GrowCut algorithm, the Fast GrowCut algorithm, into a client/server based, web-hosted 3-dimensional medical image viewer, called StudierFenster. As a result, the Fast GrowCut algorithm can be used directly inside the online environment without installing software and without technical knowledge of the user. It is now possible to use the segmentation tool on any 2-dimensional transverse slice of a 3-dimensional image. The workflow was made user-friendly, allowing input to be drawn with a brush onto the image and loading the output automatically, making it immediately visible.

Keywords: GrowCut, Fast GrowCut, Segmentation, Medical Image Analysis, Web Application, StudierFenster

1 Introduction

Image segmentation is one of the most important and common practices for medical images analysis [1, 2, 3]. Extracting the region of interest can be done in an automatic or semi-automatic process. The GrowCut algorithm is a widely used semi-automatic segmentation algorithm, working on n-dimensional images, and uses cellular automata to calculate the segmentation. As input, seeds are selected manually inside and outside the region to be segmented, marking foreground and background, respectively. During execution of the GrowCut algorithm, all remaining image pixels get sorted into either foreground or background [4]. It is an attractive segmentation tool because of its interactive and dynamic features. The GrowCut algorithm has been further developed into the Fast GrowCut algorithm, by reformulating it as a clustering problem and approximating the solution, making it significantly faster [5]. Different versions of the GrowCut algorithm are available on medical image computing platforms, such as the 3D Slicer software [6], however, not on a web-based tool where no installation of software is needed. Aside from that, a lot of open source implementations of the GrowCut algorithm are available, which do not provide a user interface and therefore require a lot of technical knowledge of the user.

This contribution aims to provide a user interface for segmentation with the GrowCut algorithm on a device- and platform-independent basis. To achieve this, the Fast GrowCut algorithm is implemented in the Medical 3D Viewer of the "StudierFenster" website (`http://studierfenster.at/`, `http://studierfenster.icg.tugraz.at/`), hereafter referred to as StudierFenster. It provides visualization and segmentation tools for medical images and is built as a client-server model. Its main component is the Medical 3D Viewer, which offers various annotation and segmentation tools [7].

As a result of this contribution, the Fast GrowCut algorithm was successfully implemented into the StudierFenster website as a segmentation feature within the Medical 3D Viewer. It is now possible to use the Fast GrowCut algorithm with a user interface without additional software. Its implementation was achieved by creating the input and viewing the output on the client side and running a Python script with the algorithm on the server side. In the Medical 3D Viewer, the Fast GrowCut algorithm can be used to

*alessandra.masur@student.tugraz.at
†egger@icg.tugraz.at

segment 2-dimensional regions of slices in the horizontal plane.

Giving an overview of this paper, Section 2 gives background on the topic of StudierFenster. The related work is presented in Section 3. In Section 4, the methods of implementing the Fast GrowCut algorithm into StudierFenster are elaborated. Lastly, the results of this contribution are given in Section 5, followed by a concluding discussion and an outlook in Section 6.

2 Background

StudierFenster StudierFenster [1] is a website that was developed in previous work at the Graz University of Technology in association with the Medical University of Graz. It is an online environment providing visualization tools for medical data, manual segmentation tools for medical images, and tools for calculations. The tools can be used directly within a web browser which makes them available to more people and platform- and device-independent. Furthermore, adaptions or updates to the software can be deployed directly on the website, without having to distribute changes to each user as it would be the case with software. StudierFenster is built as a client–server model, where some calculations of segmentations are done by the server part. This makes it unique from other existing web-based tools that only use a client-oriented approach. Adding a server back-end provides more complex functionality. A user study conducted by Weber et al. in 2019 generated positive feedback and StudierFenster has been adapted and worked on since its release [8]. The current functionalities, as of November 2022, include a Digital Imaging and Communications in Medicine (DICOM) browser and converter, the Medical 3D Viewer with 2D and 3D data visualization and various manual annotation tools, automatic aortic landmark detection, aortic dissection inpainting [9], centerline tracking [10], 3D skull reconstruction [11], 3D face reconstruction and registration [12], medical virtual reality viewer, and finally the calculation of the Dice coefficient and Hausdorff distance [8]. The client side is written with Hypertext Markup Language (HTML) and JavaScript, also using the Web Graphics Library (WebGL). The server side is written in C, C++, and Python, using libraries like Insight Toolkit (ITK), Visualization Toolkit (VTK), X Toolkit (XTK), and Slice:Drop. Server requests are processed by a Python Flask server [7].

GrowCut Algorithm The GrowCut algorithm is a semi-automatic image segmentation algorithm. It works on n-dimensional images and the segmentation is an iterative process, in which the user has the possibility to give additional input after each iteration. It uses cellular automata which allows fast and parallel computation. Every pixel in the GrowCut algorithm has a corresponding three-tuple $(l_p, \theta_p, \overrightarrow{C}_p)$, where l_p is the label of a pixel, θ_p is the strength of a pixel and $\overrightarrow{C}_p$ stands for the color value [4]. For the initialization of the seed pixels, some pixels are initialized as foreground $l_p = 1$ and some as background $l_p = 0$. Additionally, all initialized seed pixels are assigned the strength value $\theta_p = 1$. After initialization, the algorithm begins with the iteration process of sorting all other pixels into either foreground or background [13]. Simplified, at each iteration step t, each cell tries to "attack" its neighbors with the intention of spreading the foreground or background labels. The force of the cells is dependent on the strength values θ_p and θ_q and the distance between the vectors $\overrightarrow{C}_p$ and $\overrightarrow{C}_q$ of the attacking and defending cells. If an attacker has the greater attack force, its labels are spread onto its weaker neighbor cells [4]. The computation is finished when every pixel in the region of interest is assigned one of two labels [13].

Fast GrowCut Algorithm The Fast GrowCut was developed by reformulating the GrowCut algorithm as a clustering problem, to which the Fast GrowCut computes a fast, approximate solution. The clustering problem which is based on finding the shortest path, can be solved by applying an adapted version of the Dijkstra algorithm. The adaptation is introduced because the Dijkstra algorithm is static, not allowing any user input after it was initiated. To keep the Fast GrowCut algorithm dynamic and allow editing, only local regions that are affected by a new input are updated with the adapted version of the Dijkstra algorithm [5].

3 Related Work

Different versions of the GrowCut algorithm are implemented in the 3D Slicer platform, where they can be used for 2D and 3D segmentation. 3D Slicer is an open-source medical image computing platform, offering various segmentation tools [13, 14].

GrowCut in 3D Slicer The GrowCut algorithm is implemented in 3D Slicer as a module called "GrowCutSegmentation". It is an editor effect that is based directly on the original GrowCut algorithm first introduced by Vezhnevets and Konouchine [4]. The user can color foreground and background, which are used to compute the segmentation. After the computation is finished, the user can further adapt the segmentation by adding additional edits to the segmentation [6].

One application example is the use of GrowCut in 3D Slicer in research about vertebral body segmentation. There it was found that, depending on the use case, the GrowCut algorithm in 3D Slicer can be a more efficient way to segment medical images than manual analysis. In a study conducted by Egger et al. from 2017 [13], results

[1] http://studierfenster.at/

were compared for segmentation of vertebral bodies in T2-weighted magnetic resonance images (MRI). The GrowCut in 3D Slicer was shown to only require significantly less segmentation time and effort than a manual assessment with similar accuracy [13].

Fast GrowCut in 3D Slicer The Fast GrowCut is implemented in 3D Slicer as a module called "FastGrowCutEffect". It is based on the Fast GrowCut algorithm that was introduced by Zhu et al. in 2014 [5]. The user can segment an image by selecting foreground and background seeds in 3D. After initial segmentation, there is the possibility to refine the segmentation repeatedly until the user is satisfied with the result. Furthermore, the module allows multi-label segmentation [15].

The latest version of 3D Slicer, as of October 2022, uses the Fast GrowCut Segmentation under the name "Grow from Seeds", which is an editor that can be found among other segmentation tools inside the "Segment Editor" module of 3D Slicer [16].

4 Methods

4.1 Workflow

The workflow of using the Fast GrowCut in StudierFenster can be summarised in three steps "Mask Creation", "Executing Fast GrowCut" and "Result Handling". The flowchart in Figure 1 displays the usage of the Fast GrowCut in StudierFenster and gives an overview how the three mentioned steps can be divided further. In Figure 1, blue fields represent user input, and green fields represent steps that are executed automatically.

The workflow starts with loading a nearly raw raster data (NRRD) file in the Medical 3D Viewer in StudierFenster, as it can be seen in Figure 1. The selected file is loaded in the viewer and shown in four different views: a 3D view, a sagittal, a coronal, and a horizontal view in which the Fast GrowCut must be executed. In the horizontal view, the user can start with the creation of the input by coloring the desired foreground area, which is tinted red, and the background area, which is tinted green, with the underlying image still visible. Colored areas can be erased with an erase brush or the whole segmentation can be reset. Both brush modes and the eraser mode feature a round brush with variable size, which can be adjusted by dragging the "Brush Size" slider.

To start the segmentation, both foreground and background need to be specified, which is checked by the implementation. After successful calculation of the Fast GrowCut, its output is automatically shown to the user. The user-specified coloration disappears and instead, the output mask is loaded, coloring the calculated foreground and background areas in red and green, respectively.

The user has the option to delete the colored background, leaving only the segmented foreground on display.

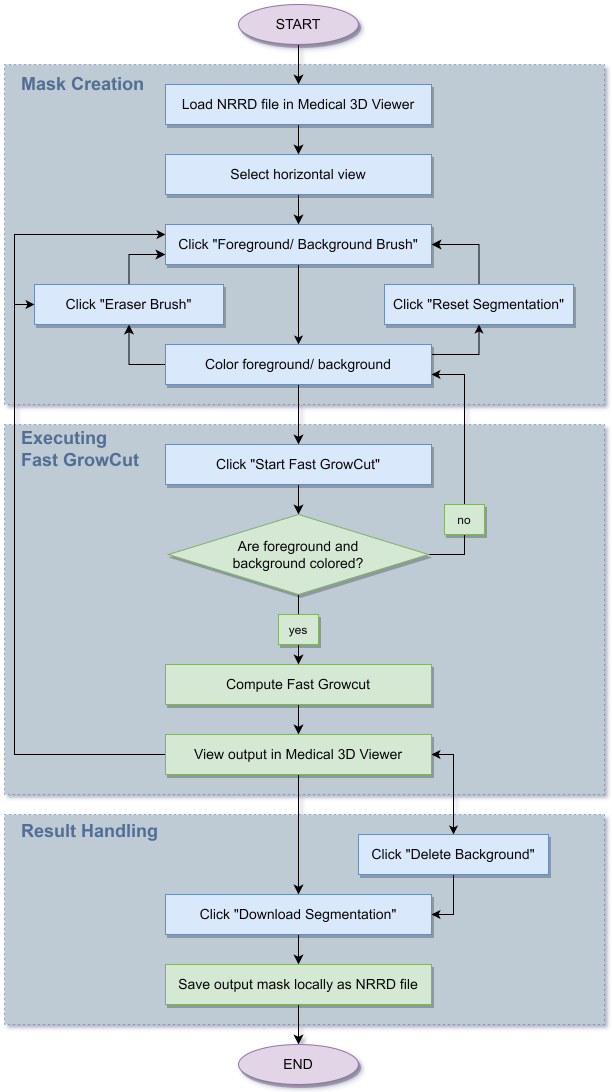


Figure 1: Flowchart demonstrating the Fast GrowCut workflow in StudierFenster. Blue fields represent user input, green fields represent steps executed by the implementation.

The user can further edit the output mask and potentially calculate the Fast GrowCut again if the first output was not satisfactory. Finally, the output mask can be saved locally as an NRRD file where the segmented foreground is colored white and everything else is black.

The Fast GrowCut in StudierFenster can be used through the GrowCut side menu, which contains eight user interface elements, seven buttons, and one slider. A screenshot of the menu is included in Figure 2. It is located with the other segmentation tools in the Medical 3D Viewer side menu and its style, and the implementation of some functions was inspired by other segmentation tools [17].

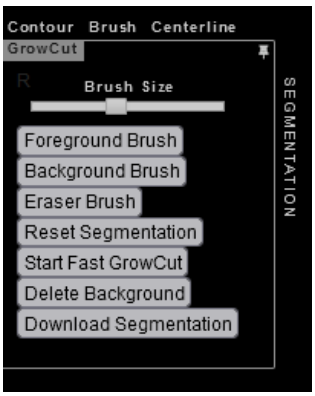


Figure 2: The Fast GrowCut side menu in the Medical 3D Viewer.

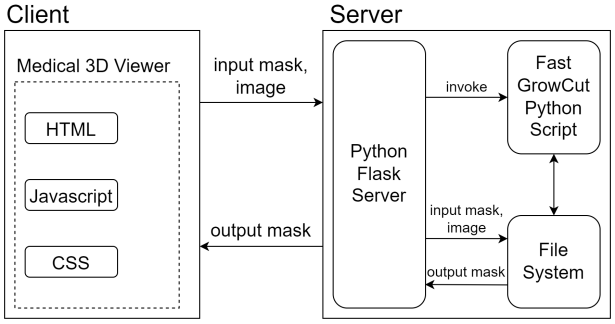


Figure 3: Client/server architecture of the Fast GrowCut implementation in StudierFenster.

4.2 Implementation

4.2.1 Client/Server Architecture

The Fast GrowCut algorithm is implemented in StudierFenster in two parts: a front-end part that is executed on the client side, and a back-end part that is executed on the server side. The client/server architecture is displayed in Figure 3. The client side consists of its interface in the Medical 3D Viewer, which was implemented as HyperText Markup Language (HTML), JavaScript, and Cascading Style Sheets (CSS). The server side consists of the Fast GrowCut implementation and the Flask Server, which is used to communicate between the two sides. The Fast GrowCut is implemented on the server side, because its computation is very intensive and therefore faster and more reliable when done on the server. Furthermore, the algorithm is implemented in a Python script, which cannot be executed in a browser.

The communication between client and server works as followed: The input image and the mask, which are generated on the client side are converted into one JavaScript Object Notation (JSON) object that is sent to the server via an Asynchronous JavaScript and XML (AJAX) request. On the server side, the JSON object is parsed and each of the files is saved individually as an NRRD file in a temporary folder in the file system. The Fast GrowCut Python script then loads the NRRD files from the file system, computes the segmentation, and saves the output mask as an NRRD file in the same folder as the input files. The Python Flask Server waits for the Fast GrowCut to finish, loads the output mask from the file system, and converts it into a JSON object, which is sent back to the client side. Finally, on the client side, the segmentation mask is automatically loaded in the Medical 3D Viewer.

4.2.2 Fast GrowCut Python Implementation

The Python implementation of the Fast GrowCut was adapted from the Python script "growcut_cpu.py" by Shen [2] [18]. It is a Python implementation of Fast GrowCut based on the algorithm by Zhu et al. [5]. The script requires the image data and a seed label array as input [18]. As the Fast GrowCut in StudierFenster is computing two-dimensional segmentations, only the current layer of the image is used for input. The seed label array is of the same size as the two-dimensional input image and contains different labels "1" for foreground and "2" for background. It is generated from the foreground and background information that is drawn by the user that is stored in an HTML canvas element [19]. The output of the Python script is an array of the same size as the seed label array, containing a corresponding "foreground" or "background" label for each array element [18]. This information is mapped to an HTML canvas element, again, for viewing and editing in the Medical 3D Viewer.

5 Results

Figure 4 shows the Medical 3D Viewer with the GrowCut menu opened on the left-hand side, showing the user interface. The loaded image shows an MRI scan of a human brain, the horizontal plane being visible in a big window and the three other views being visible on the right side in small windows.

Results of the Fast GrowCut algorithm used in StudierFenster to segment an image can be seen in Figures 5 and 6. The input masks are visible in Figures 5(a), 6(a) and 6(c) and the corresponding Fast GrowCut results are shown in Figures 5(b), 6(b) and 6(d). In all sub-figures, the foreground is tinted red and the area surrounding the region of interest is tinted green. Figure 5 displays the segmentation of a brain tumor in an MRI head scan and Figure 6 shows the segmentation of a vertebral body in a CT chest scan.

In Figure 6 it is shown that the Fast GrowCut output can be edited before running the Fast GrowCut again with the additional input. Figure 6(a) shows the initial user input and Figure 6(b) shows the corresponding Fast GrowCut

[2] https://github.com/Sherry-SR/fastgc_python/blob/master/modules/growcut_cpu.py

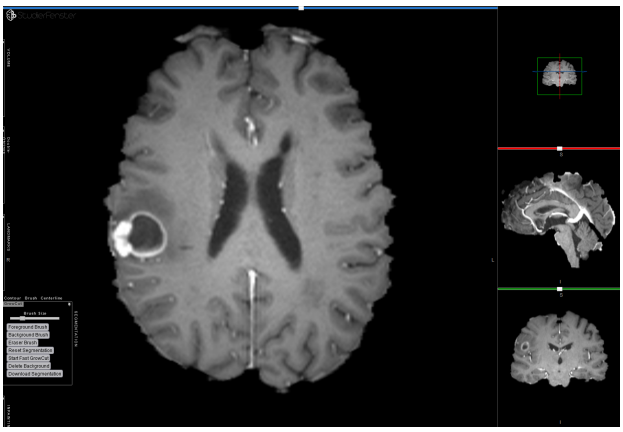


Figure 4: Screenshot of the Medical 3D Viewer with the GrowCut side menu on the left.

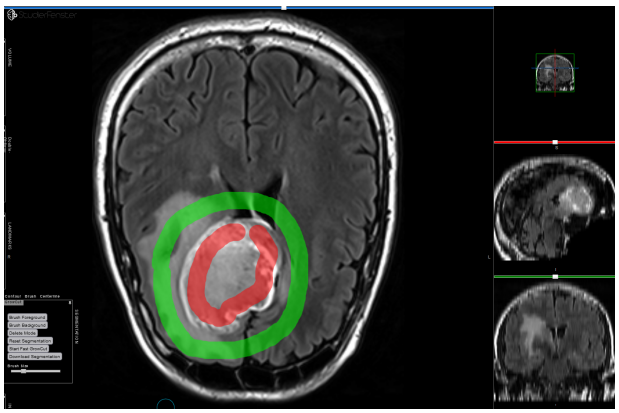
output. The output was edited with the "Delete Mode" and foreground and background brushes, as seen in Figure 6(c) and the Fast GrowCut was run again. Its output is visible in Figure 6(d).

6 Conclusion and Outlook

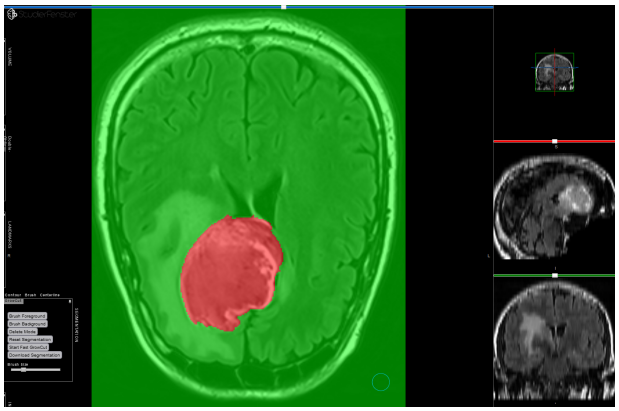
As a result of this contribution, the Fast GrowCut segmentation algorithm was added to StudierFenster, extending the existing segmentation tools, thereby extending the possible use cases.

The Python implementation of the Fast GrowCut by Shen [18] was successfully implemented on the server side of StudierFenster. It was decided to use this implementation of the GrowCut because it could be adapted to consist only of one Python script, which made its applicability simple. Adding this functionality to the architecture of StudierFenster was possible without having to change much of the original Fast GrowCut code. Its straightforward implementation made it stand out against other open-source GrowCut implementations, like the 3D Slicer's "Grow from Seeds" algorithm, for example. The "Grow from Seeds" code is depending heavily on other 3D Slicer modules and functions, which would have made it nearly impossible to extract only the "Grow from Seeds" code, without having to change most of the implementation [16].

After loading an image into the Medical 3D Viewer of StudierFenster, Fast GrowCut can be used to segment any region in the 2-dimensional transverse plane, that has a different color than its surrounding area. For example, a specific use case could be using the Fast GrowCut to segment vertebral bodies, as it was found in a study conducted by Egger et al. in 2017 [13], in which the GrowCut in 3D Slicer was used. Some benefits of using the Fast GrowCut in StudierFenster are its platform and device independency, compared to other implementations like the "Grow from Seeds" algorithm in 3D Slicer [16]. The Fast GrowCut in StudierFenster can be used on various devices, from various browsers. This Fast GrowCut implementation in StudierFenster allows a user to use a graphical interface to select the mask and the output is automatically visible in the original image. This method does not require the user to have any knowledge of Python programming, as opposed to directly using the Fast GrowCut Python script by Shen [18].



(a) User input for tumor segmentation.

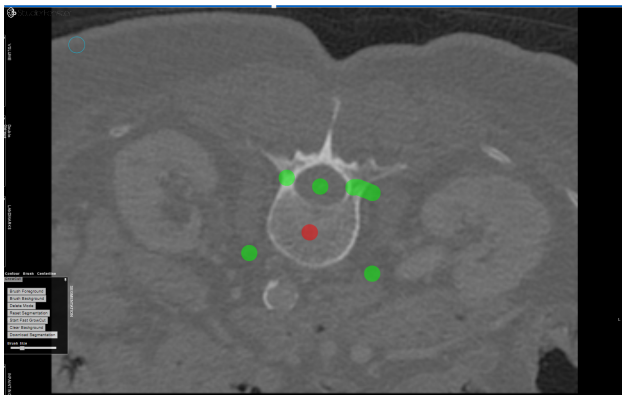


(b) Fast GrowCut output of tumor segmentation.

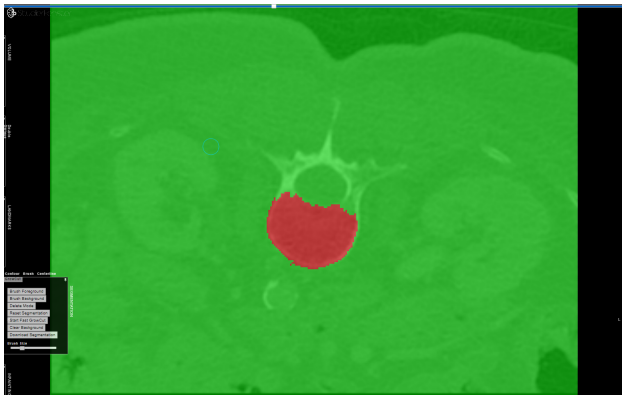
Figure 5: Segmentation of a brain tumor of an MRI head scan with Fast GrowCut in StudierFenster.

Figures 5 and 6 show that the algorithm successfully segmented the colored regions, based on the user input. However, it is visible that the algorithm did not always find the correct borderline between object and background. This was especially the case when the two had similar colors, or when the border was more of a transition, rather than a line. This can be seen in Sub-figure 6(b), where the red color is spilling outside the region of the vertebral body. In Figure 6 it is visible that the segmentation output is more satisfactory, after iterative refinement by the user. Running the algorithm again is significantly faster than running it for the first time, as the algorithm only iterates through pixels that are not colored.

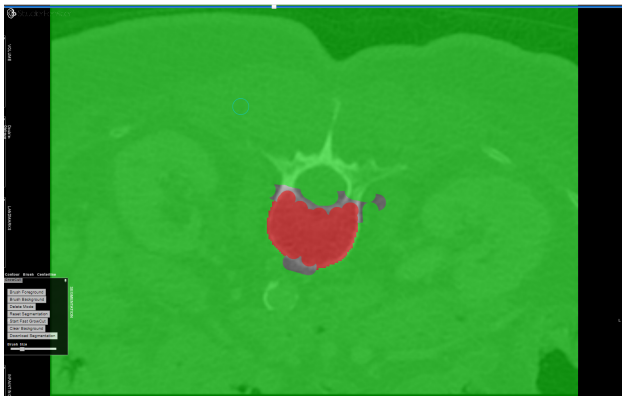
Future research could examine if allowing user input during the segmentation would accelerate the process of refining the segmentation. This could lead to more sat-



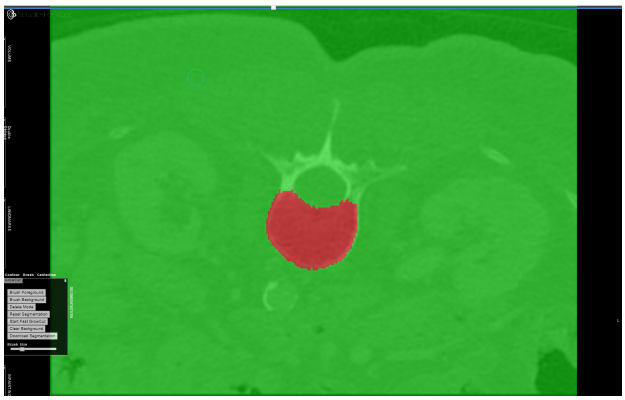
(a) First input for vertebral body segmentation.



(b) First Fast GrowCut output of vertebral body segmentation.



(c) Second input for vertebral body segmentation.



(d) Second Fast GrowCut output of vertebral body segmentation.

Figure 6: Vertebral body segmentation of a CT chest scan with Fast GrowCut in StudierFenster, executing the Fast GrowCut twice.

isfying results after running the Fast GrowCut only once. Allowing user input during the execution of the algorithm would enable users to correct wrongly placed seeds early on.

Furthermore, the Fast GrowCut in StudierFenster could be extended to be used for three-dimensional segmentations, as it is currently only used for two-dimensional segmentations. The implemented Python script of the Fast GrowCut algorithm already allows n-dimensional segmentation [18], however, the process of the input mask creation would need to be extended.

There are open questions, regarding the usage of GrowCut in StudierFenster to segment patient data. The current workflow requires the image data to be stored in the file system on the server side. As this might be undesirable when handling confidential data alternatives should be explored in the future.

Finally, conducting a user study would help to identify flaws in the proposed workflow and could be of assistance in improving the current graphical user interface.

References

[1] Alireza Norouzi, Mohd Shafry Mohd Rahim, Ayman Altameem, Tanzila Saba, Abdolvahab Ehsani Rad, Amjad Rehman, and Mueen Uddin. Medical Image Segmentation Methods, Algorithms, and Applications. *IETE Technical Review*, 31(3):199–213, 2014.

[2] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

[3] Jan Egger, Christina Gsaxner, Antonio Pepe, Kelsey L Pomykala, Frederic Jonske, Manuel Kurz, Jianning Li, and Jens Kleesiek. Medical deep learning–a systematic meta-review. *Computer methods and programs in biomedicine*, page 106874, 2022.

[4] V. Vezhnevets and V. Konouchine. "GrowCut" - Interactive Multi-Label ND Image Segmentation By Cellular Automata. *Graphicon*, 1(4):150 – 156, 2005.

[5] L. Zhu, I. Kolesov, Y. Gao, R. Kikinis, and A. Tannenbaum. An Effective Interactive Medical Image Segmentation Method using Fast GrowCut. In *Int Conf Med Image Comput Comput Assist Interv. Workshop on Interactive Methods.*, volume 17, 2014.

[6] H. Veeraraghavan and J. Miller. Modules:GrowCutSegmentation-Documentation-3.6 - Slicer Wiki. Technical report, 2010. `https://www.slicer.org/wiki/Modules:GrowCutSegmentation-Documentation-3.6`, accessed: 2022-10-23.

[7] J. Egger, D. Wild, M. Weber, C. Bedoya, F. Karner, A. Prutsch, M. Schmied, C. Dionysio, D. Krobath, J. Yuan, C. Gsaxner, J. Li, and A. Pepe. Studierfenster: an Open Science Cloud-Based Medical Imaging Analysis Platform. *Journal of Digital Imaging*, 35, 01 2022.

[8] M. Weber, D. Wild, J. Wallner, and J. Egger. A Client/Server based Online Environment for the Calculation of Medical Segmentation Scores. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3463–3467. IEEE, 2019.

[9] Alexander Prutsch, Antonio Pepe, and Jan Egger. Design and development of a web-based tool for inpainting of dissected aortae in angiography images. In *24th Central European Seminar on Computer Graphics: CESCG 2020*, 2020.

[10] Christina Dionysio, Daniel Wild, Antonio Pepe, Christina Gsaxner, Jianning Li, Luis Alvarez, and Jan Egger. A cloud-based centerline algorithm for studierfenster. In *Medical Imaging 2021: Imaging Informatics for Healthcare, Research, and Applications*, volume 11601, pages 201–206. SPIE, 2021.

[11] Jianning Li, Antonio Pepe, Christina Schwarz-Gsaxner, and Jan Egger. An online platform for automatic skull defect restoration and cranial implant design. In *SPIE Medical Imaging Conference 2021*, page 115981Q, 2021.

[12] Florian Karner, Christina Gsaxner, Antonio Pepe, Jianning Li, Philipp Fleck, Clemens Arth, Jürgen Wallner, and Jan Egger. Single-shot deep volumetric regression for mobile medical augmented reality. In *Clinical Image-Based Procedures, 9th International Workshop, CLIP 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4–8, 2020, Proceedings 9*, pages 64–74. Springer, 2020.

[13] J. Egger, C. Nimsky, and X. Chen. Vertebral body segmentation with GrowCut: Initial experience, workflow and practical application. *CoRR*, abs/1711.04592, 2017.

[14] Jan Egger, Tina Kapur, Andriy Fedorov, Steve Pieper, James V Miller, Harini Veeraraghavan, Bernd Freisleben, Alexandra J Golby, Christopher Nimsky, and Ron Kikinis. Gbm volumetry using the 3d slicer medical image computing platform. *Scientific reports*, 3(1):1–7, 2013.

[15] L. Zhu. Documentation/4.8/Modules/FastGrowCut - Slicer Wiki. Technical report, 2017. `https://www.slicer.org/wiki/Documentation/4.8/Modules/FastGrowCut`, accessed: 2022-10-23.

[16] C. Pinter, A. Lasso, K. Sunderland, S. Pieper, W. Plesniak, R. Kikinis, and J. Miller. Segment editor - 3D Slicer documentation. Technical report, 2022. `https://slicer.readthedocs.io/en/latest/user_guide/modules/segmenteditor.html`, accessed: 2022-10-23.

[17] D. Wild, M. Weber, and J. Egger. A Client/Server Based Online Environment for Manual Segmentation of Medical Images. In *The 23rd Central European Seminar on Computer Graphics (CESCG)*, pages 1–8, 2019.

[18] R. Shen. Fast Growcut alogorithm with shortest path. Technical report, 2019. `https://github.com/Sherry-SR/fastgc_python`, accessed: 2022-12-27.

[19] Mozilla and individual contributors. ImageData - Web APIs — MDN. Technical report, 2022. `https://developer.mozilla.org/en-US/docs/Web/API/ImageData`, accessed: 2022-10-19.

Weakly Supervised Semantic Cell Segmentation Using Knowledge Distillation

Ivana Háberová*
Ivan Vykopal†
Supervised by: Dr. Lukáš Hudec‡

Faculty of Informatics and Information Technologies
Slovak University of Technology
Bratislava / Slovak republic

Abstract

This study proposes a new approach for semantic cell segmentation that combines the use of neural networks and involving humans in the loop with the aim of improving the current state of digital pathology. The goal is to obtain cell segmentation and classification from heart biopsy images based on inaccurate data and simultaneously to reduce the demands on domain experts - doctors. In the first step, the approach utilizes a segmentation model and a combination of different datasets to detect the nuclei of cells in the patches of whole slide images, which are used to increase the amount of data. The proposed approach employs knowledge distillation, a technique that involves training a smaller "student" model to mimic the output of a larger "teacher" model and their chaining. This is done to overcome the limitations of having a small amount of accurate data and a high proportion of inaccurate annotations and to remove inaccuracies through chaining. The proposed approach is evaluated against traditional methods and shows that it achieves improved performance in terms of semantic cell segmentation. This demonstrates the potential for the approach to be applied in biomedical image analysis, where accurate and precise segmentation is essential for downstream analysis.

Keywords: Segmentation, Classification, Knowledge Distillation, Human-in-the-Loop, Weakly Annotated Data, Digital Pathology

1 Introduction

The analysis of whole slide images is one of the important components of pathologists' diagnosis of Cardiovascular diseases (CVDs). Research in this area is also progressing because $\sim$ 17.9M people die each year from CVDs, according to WHO[1], which is approximately one-third of all deaths worldwide. CVDs are heart or blood vessel diseases, such as coronary heart disease, cerebrovascular disease, and rheumatic heart disease. A heart biopsy is an effective way to detect changes in the heart muscle. On the other hand, this procedure is invasive, difficult for the patient - especially if heart problems occur - and requires sufficient time for sample collection, tissue processing and following evaluation by a doctor. Analyzing images after a heart biopsy can be a challenging task, as the tissue samples are often small and may be difficult to interpret.

Over the past 20 years, the field of pathology has made significant advancements in digital imaging through the development and improvement of whole-slide imaging. Digital pathology is a technology that can benefit from high-resolution digital images to aid in diagnosis and treatment planning. It is becoming increasingly popular in pathology departments, offering advantages over traditional, microscope-based methods of analyzing tissue samples. A combination of machine learning and digital pathology can automate image analysis and hence has the potential to revolutionise the field of pathology by improving diagnostic accuracy, increasing efficiency, and reducing costs. Currently, there are several automated tools providing a biomedical image or biomarker analysis like QuPath [4], MONAI [8], CellProfiler [14], or ImageJ [2].

This work presents a novel training strategy for weakly annotated data applied in semantic cell segmentation from histopathological heart biopsy images based on imprecise annotations. The motivation is mainly to reduce the demands on doctors, who can more easily detect problematic areas based on accurate classification or, in conjunction with rigorous quantitative analysis, detect small deviations earlier and thus bring new knowledge to the given area. The research is carried out in cooperation with experts from the Institute for Clinical and Experimental Medicine in Prague (IKEM).

To summarize, our main contributions are: (1) a robust model for nuclei segmentation in different organs and resolutions; (2) an approach that classifies cells in histopathological data using knowledge distillation; (3) a teacher model usable for training other students for different types of data and different types of annotations; (4) a relatively

*ivankahaberova@gmail.com
†ivan.vykopal@gmail.com
‡lukas.hudec@stuba.sk

small network that is well adapted to a specific task (cell classification in H&E images).

2 Related work

Identifying individual tissue types or small cells in a histopathological image is time-consuming and requires experienced doctors. The solution to this pathology image analysis challenge can be using deep learning algorithms that can process and evaluate the images quickly and segment effectively the needed areas - tumours, tissues or cells. The main task of (binary) segmentation is to distinguish the searched tissue from its surroundings, while there is a semantic segmentation, where individual segmented tissues have a different meaning. The encoder-decoder architectures with interconnections known as U-Net [17] proved effective, outperforming previously used methods - a combination of sliding window and convolutional networks. U-Net achieves quantitatively and qualitatively good results, even on a small amount of biomedical data with extensive augmentation. Olaf Ronneberger et al. [17] applied U-Net to a cell segmentation task in light microscopic images as part of the ISBI cell tracking challenge 2015, where they achieved on different partially annotated datasets average IoU (Intersection over Union) 77.5% and 92% and significantly outperformed other algorithms. The extension and improvement of the performance come with the deeply-supervised architecture of U-Net++ [20], which added more connections between the encoder and the decoder along with the intermediate outputs.

Classifying cells in histology images is challenging due to the high intra-class variability and inter-class similarity. Many papers deal with this problem using various modifications of convolutional neural networks (CNN). The first significant improvement in CNN results came with VGGNet [19], which demonstrated not only the positive influence of model depth on classification success but also the advantages of using relatively small reception fields (convolutional filters with size 3×3). VGG-16 and VGG-19 versions differ in the number of VGG blocks (16 vs 19), where one VGG block consists of several convolution layers followed by a max-pooling layer. In 2016 He et al. [12] proposed using residual blocks in the neural network. Applying skip connections or shortcuts made it possible to go deeper with the architecture and increased the network's learning ability. Similar to VGG, there are several versions of the Deep residual network architecture or ResNet, such as ResNet-18, ResNet-50 or ResNet-152. Xception [7] model outperforms on ImageNet classification dataset many state-of-the-art models such as VGG-16, ResNet-152 or Inception V3. This architecture is based entirely on depthwise separable convolution layers, which provide great computational efficiency. With the goal of application in diagnostics, where you cannot rely on high-performance computers, some authors try to design models "as simple as possible". This is the case with RCCNet [5], which was created with the aim of colon cancer nuclei classification and has 1.5M learnable parameters compared to VGG-16 with 138M parameters.

To deal with weakly annotated data, there is the human-in-the-loop method, based on domain experts' involvement in interacting with artificial intelligence to obtain more accurate annotations[16]. Most of the research involving experts in the process consists of three main phases: training the preliminary model, predicting unseen data with the preliminary model, and correcting predicted annotations using domain experts. Predictions and corrections are performed in the loop until certain conditions are met. Annotations can be corrected not only by experts but also through crowdsourcing, either by manual correction from a domain expert or by marking them as correct or incorrect [11, 3].

With weakly annotated data or small amounts of data, training a robust model that achieves the expected results is challenging. For this reason, different approaches are used to utilize the currently available resources optimally. A knowledge distillation approach works with weakly annotated data, transferring knowledge between two or more models. The principle of this approach, called Teacher-Student architecture, is to train a Teacher on a small amount of data or weakly annotated data and then train the student using the trained Teacher. In this case, the annotations obtained by the Teacher are used in training the student. Several methods are based on Teacher-Student architectures, including Teacher-Student chaining[18] or substituting Teacher and Student in the training process[6]. Both methods aim to use weak or insufficient annotations to train the best possible model well generalized to the desired task.

Pathologists in IKEM use QuPath for analyzing data - nuclei and higher morphological structures. QuPath can segment cells using parametric methods like color thresholding based on H&E staining for segmentation, whereas, for classification, there are three methods: K-nearest neighbors, Random Forest, and Artificial Neural Network (ANN). The disadvantage of this tool is the excessive dependence of the results on the initial setting by the doctor, which may differ each time based on different concentrations of staining color and therefore result insufficient.

3 Dataset

In our study, we work with two publicly available histological datasets Lizard[9], MoNuSeg[13] and custom dataset based on IKEM data. All three datasets contain histological images stained with hematoxylin and eosin staining. Images from each dataset are shown in Figure 1. By combining them, we created the largest dataset for nuclei segmentation in multiple organs with different magnifications to create a robust model for segmentation.

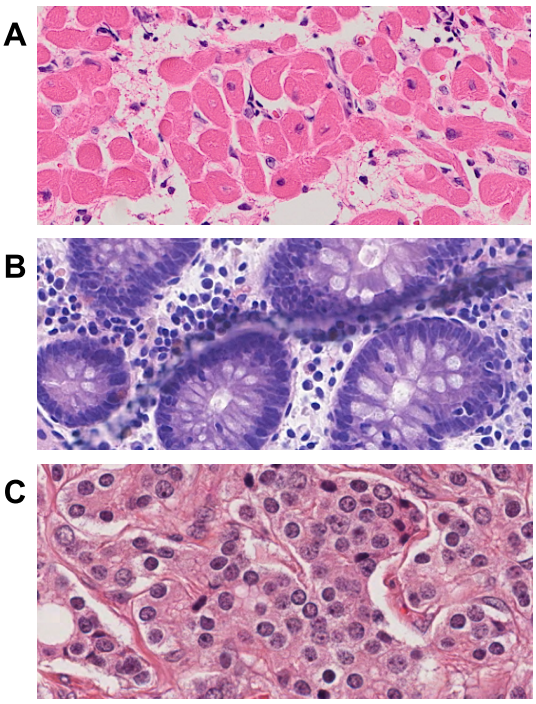


Figure 1: Comparison of datasets: A - IKEM, B - Lizard, C - MoNuSeg after normalization using Macenko method.

3.1 Lizard

Lizard dataset [9] consists of regions of interest (ROI) from Whole Slide Images (WSI) scans of the colon region. This dataset is designed to segment and classify nuclei and consists of six different datasets. Lizard is the largest histological dataset, with 238 ROI images and approximately 495K nuclei. The nuclei annotations were generated based on a multi-step approach consisting of segmentation and classification by the HoverNet [10] neural network to refine automatic and semi-automatic predictions, with the pathologist involved throughout the workflow to refine the segmentations and classifications. Table 1 shows the number of nuclei in the images.

3.2 MoNuSeg

MoNuSeg is a multi-organ dataset and contains 37 histological images together with their annotations. Similar to the Lizard dataset, we work with binary annotations to segment the nuclei. Table 1 shows the total number of nuclei in this dataset.

3.3 IKEM dataset

The IKEM dataset contains 25 WSI scans, each comprising three to five tissue sections (called fragments) from the heart region. The WSI format can store information about a given tissue in several resolutions with relatively small memory requirements, where the highest reaches dimensions up to 48724×17910.

Dataset	Nuclei count	Immune cells	Muscle cells	Other cells
Lizard	495 179	-	-	-
MoNuSeg	21 623	-	-	-
IKEM SSA	470 563	118 950	130 022	221 591
IKEM WSA	469 591	127 521	127 259	214 811
IKEM EA	6 834	1 947	2 794	2 093

Table 1: The number of nuclei in each dataset along with their classifications.

Cooperating pathologists provide us:

- QuPath project with trained object detectors and classifiers
- cell annotations as GeoJSON-s obtained by QuPath automatically
- several Artificial Neural Networks (ANN) classifiers
- 4 335 manual annotations on 6 WSI scans with 6 834 nuclei. We call these annotations expert annotations (EA).

The pathologists pre-trained the ANN classifier by iterative manual correction of cell classifications. We used this classifier to generate strong synthetic annotations (SSA). Then, we randomly selected one significantly weaker classifier from the previous iterative improvements and generated weak synthetic annotations (WSA). The resulting distributions of immune, muscle, and other cells are shown in Table 1.

4 Proposed method

Our study aimed to create a comprehensive approach for analyzing nuclei in histological images, from segmentation to classification and applicable to various tissue and organ types.

We focus on the problem of weak annotations that are generated by the QuPath tool using an Artificial Neural Network that has been trained by doctors. Our method's objective is to leverage weak annotations with minimal demands on doctor input effectively.

4.1 Data preprocessing

All data provided by IKEM, whether obtained automatically by QuPath or expert annotations, have the first processing step in common.

Preprocessing 1 (Fig. 2) consists of several steps to ensure efficiency and fast data processing by neural networks. The data are stored as multidimensional matrices, which greatly increases the memory requirements. For this reason, we chose to save in three resolutions - original, ½, and ¼. At the same time, with the aim of reducing memory requirements, images are divided into fragments, where

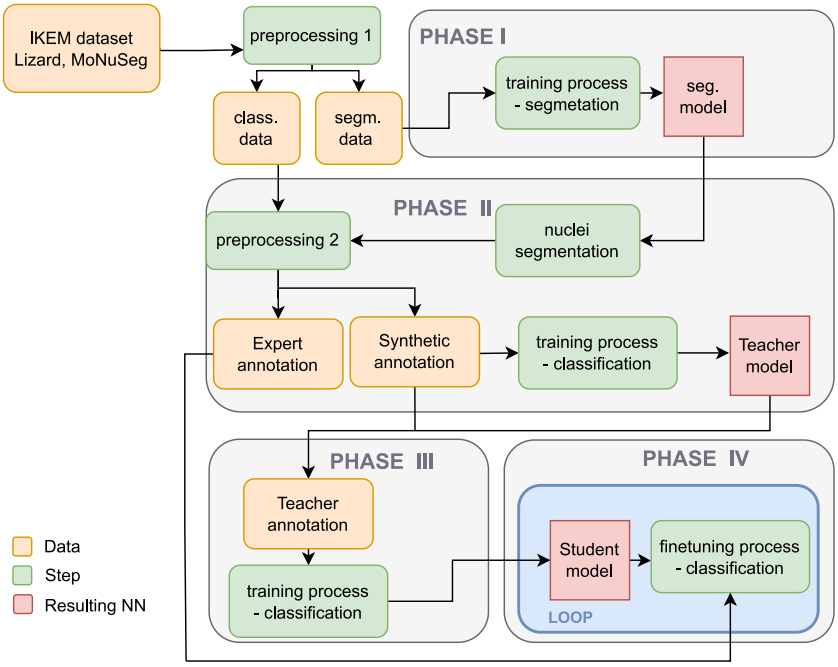


Figure 2: Overview of the proposed approach showing individual training phases, together with inputs, data flow and outputs. The result offers three trained models - one cell segmentation and two cell classification models.

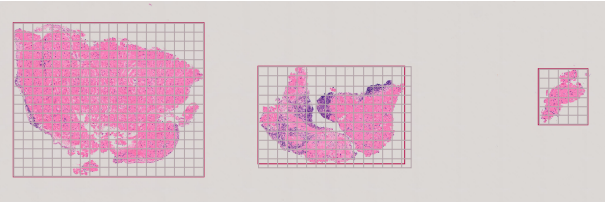


Figure 3: WSI scan with 3 fragments (red marked) and patch subdivision (grey marked). In preprocessing, only individual tissue fragments are stored based on their bounding boxes, so an unnecessary white area is omitted.

the position is chosen according to the smallest possible bounding boxes of individual parts of the tissue.

After saving the fragments as separate entities in different resolutions, optional image data normalization follows. We chose Macenko normalization [15] to reduce bias between datasets, which arose as a result of staining in another laboratory.

The proposed approach needs annotations in the format of multidimensional masks, so the original GeoJSONs are converted to multichannel images. Each channel contains information about one type of desired nuclei.

The images and corresponding annotations (of all datasets) are divided into patches of the selected size, in this case, 512×512, where we use the sliding-window approach without an overlay as shown in Fig. 3. Lizard and MoNuSeg images are stored as binary masks (1-nucleus, 0-background), while images of incompatible dimensions were zero-padded. IKEM data is converted to binary form when loading images during segmentation training with regard to their further use in classifications.

The second part of preprocessing (preprocessing 2 in Fig. 2) uses nuclei segmentation from Phase I. (Fig. 2). For each nucleus segmented, we identified a class using generated synthetic annotations and expert annotations. We assigned each nucleus to one of the classes: other (0), immune (1), muscle cells (2) and background (3). The background class was assigned to nuclei identified by our model but not segmented and classified by QuPath.

To identify the class of each nucleus, a patch is generated around it and SSA and EA are utilized to determine its classification. Patch size 16×16 is used for multiple and 32×32 for the original image size. For the original size, we used a larger patch size to contain only one nucleus, as opposed to multiscale data, where in some cases, a 16×16 patch contains more than one nucleus in the smallest magnification and only part of it in the highest.

When training models in all steps, the data is divided with a random distribution in the ratio of 70:15:15 into training, testing and validation parts.

4.2 Nuclei segmentation

In the initial stage of our comprehensive approach, we aim to segment nuclei in histological images.

In Phase I (Fig 2), we experimented with two traditional architectures, U-Net and U-Net++, commonly used in medical data segmentation. We modified both architectures by replacing the Upsampling layer with the inverse convolution layer, ConvTranspose. The benefit of using

ConvTranspose, a form of learning upsampling, is that it results in a larger number of trainable parameters, leading to a more robust model with a larger capacity.

We train models with all three datasets and a total of 1.93M nuclei in 512×512 patches. We trained our models using the Dice Loss function and Adam optimizer. The training process lasted for 30 epochs using 0.3 as dropout and 0.0001 as the learning rate. The best model was selected based on the validation loss function.

4.3 Classification

The following steps in this approach focus on obtaining the classification of the cells found in the images in the IKEM dataset using the Knowledge distillation approach. The task of the resulting model is to classify cells into three classes - muscle cells, immune cells and others. We chose teacher-student chaining with three models, where first, the weak ANN trains the complex ResNet-18 model in phase II. Subsequently, in phase III, the teacher transfers the information to the RCCNet model, fine-tuned in the phase IV by involving an expert.

The chosen loss function in all classification-trainings is CrossEntropy, optimized by Adam over 30 epochs.

4.3.1 Classification - Teacher

In Phase II, a network called a teacher is trained, which is necessary for the next process. After training on weakly annotated data (WSA and SSA), the goal is to obtain a robust classification model that can extract the essential information from weak annotations. The task consists in assigning each of the cells (based on our nucleus segmentation) to one of the classes required by the experts.

The specificity of this step also lies in the created dataset (described in 4.1 Data Preprocessing), where based on its different versions (various patch-size, normalization, various resolutions) more experiments were performed.

The selection of architecture focuses on state-of-the-art classification models with a large learning capacity, such as ResNet-16, ResNet-18, Xception or VGG. The ResNet architecture was changed for the needs of the chosen patch size, and the VGG architecture was modified (using only one VGG block with two linear layers) to process small images and preserve information efficiently. All these architectures are trained on multiscale data only, which leads to using the best architecture to classify data in the original magnifications to perform all the following steps in experiments.

In the training process, the demands on doctors are significantly reduced, especially by the processing of a weakly annotated dataset created by QuPath, but also by the fact that we do not require designing or setting parameters as QuPath does. The result is a classification model that can obtain enough important information and features from weakly annotated data and can thus be used for creating annotations in the next step.

4.3.2 Classification - Student

Our proposed method is training a network called Student to obtain a relatively small classification model specified for the given task. The goal is the same as when training the teacher (described above) - classifying cells based on small patches of images of a heart biopsy. However, the access to the provided data and the training process - aimed at transferring relevant knowledge from the teacher to the student, followed by specification by adding an expert to the loop - is different. We can divide this step into two phases: training (phase III) and fine-tuning (phase IV).

During the training process, the same patches of images are used for training as in the initial teacher training. The difference is that the teacher determines the label for each segmented nuclei. When training a student, we work with the highest resolution and use a teacher who has been trained on images with the highest resolution.

We chose RCCNet as the student architecture. The training duration depends on the network size, which is important to optimize as much as possible. The evaluation is done after each epoch against the SSA, where the model achieving the best results is saved.

In the last Phase IV, an already partially trained student model is trained again on expert annotations with the aim of precise specification for a given task - refining predictions by applying the human-in-the-loop approach.

Fine-tuning differs from training mainly in the data and its processing, where expert annotations are used in this step. When training, instead of using all the data, we look for a suitable lower limit of the count of cells (the same count from each cell class), which is necessary for a sufficient improvement of the results. To prevent overfitting, which could occur with a small amount of data, some experiments with different hyperparameters settings were performed with values for learning rate from interval $\leq 0.00005, 0.001 \geq$ and dropout rate from $\leq 0, 0.5 \geq$.

The student obtains more accurate information from reliable annotations, which should be reflected in better semantic segmentation and reducing or eliminating the error from the original weakly annotated data. The result is a relatively small network that is well adapted to a specific task, and at the same time, it can be quickly and efficiently modified by the next round of fine-tuning.

5 Evaluation

We ran all our experiments on a computer with a graphic card NVIDIA RTX 2080 Ti with 11GB of GPU RAM and 32GB of total memory RAM.

5.1 Nuclei segmentation

For segmentation, we trained U-Net and U-Net++, where the final nuclei segmentation model is chosen based on the metrics achieved on the test set. Based on the value of

Model	Accuracy [%]	Precision [%]	Recall [%]	Dice [%]
U-Net	97.75	72.19	90.56	79.66
U-Net++	97.44	67.92	92.71	78.04

Table 2: Evaluation of segmentation models U-Net and U-Net++ on test data.

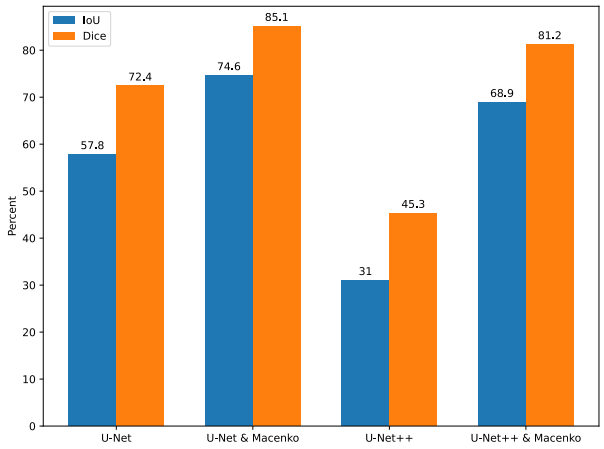


Figure 4: Segmentation results on WSI scans with and without normalization from the IKEM dataset against QuPath segmentation.

Dice over the test set, which was 79.66% for U-Net and 78.04% for U-Net++, we selected the trained U-Net model for further processing. Results are shown in Table 2.

The proposed method for nuclei segmentation was qualitatively evaluated and compared to the results obtained through automatic segmentation using QuPath. The results showed that the proposed method performed slightly better than QuPath. QuPath's approach for segmentation relies on defining threshold values for each staining and following statistical methods.

To further evaluate the segmentations, we quantitatively compared the segmentations produced by our models on WSI scans with the QuPath tool segmentations using the metrics Intersection over Union and Dice score. The results of U-Net and U-Net++ on data without and with normalization using Macenko's method are presented in Figure 4. The evaluation of WSI scans revealed that in both cases, the U-Net architecture performed better than U-Net++, both without image normalization and with normalization using Macenko's method.

5.2 Classification - Teacher

We evaluated strong and weak synthetic annotations against expert annotations, using metrics such as F1 Score, Accuracy, Precision and Recall. The evaluation was based on the cell segmented by the U-Net model. During this process, 574 segmented nuclei were identified as background. The results presented in Table 3 showed that weak synthetic annotations perform better than strong ones.

Data	F1 [%]	Accuracy [%]	Precision [%]	Recall [%]
SSA	82.01	81.82	82.44	81.82
WSA	82.34	82.23	82.56	82.23

Table 3: Comparison of classifications obtained using strong and weak synthetic annotations compared to expert annotations.

	Test set		Doctor set	
IKEM data	F1 [%]	Accuracy [%]	F1 [%]	Accuracy [%]
SSA	86.20	86.24	81.89	81.55
SSA & Macenko	82.90	82.66	81.54	81.24
WSA	85.64	85.34	82.04	81.75
WSA & Macenko	82.57	82.49	81.98	82.00

Table 4: Evaluation of Teacher architecture ResNet-18 on the test set and doctor annotations.

After evaluating the performance of ResNet-16, VGG-4 and Xception architectures on multiscale data using 16×16 patch, we identified that the ResNet-16 architecture achieved the highest F1 Scores. Therefore, we used the residual block-based architecture for our experiments on original-size data using 32×32 patch size. Due to the increase in patch size, we moved to architecture with a larger learning capacity - ResNet-18 as the Teacher model in all the following analysis steps.

We trained the selected ResNet-18 architecture on data without and with normalization using Macenko with strong and weak synthetic annotations. The results of training ResNet-18 as a Teacher model on these different data combinations can be found in Table 4.

When comparing the results of training the ResNet-18 model on data with and without Macenko normalization, the model performs better on data without applying normalization for both test data and expert annotations. In our analysis, we also compared the performance of the ResNet-18 model when training on SSA and WSA from QuPath. As shown in Table 4, the results indicate that training on SSA performs better on the test set. However, in the case of expert annotations, the results are better in the case of training using WSA. This may be caused by the fact that the doctors selected the ANN as the best based on a qualitative evaluation and visual comparison.

5.3 Classification - Student

Our evaluation of the trained models, called students, focused on the RCCNet architecture. We trained these models using the previously trained teachers from the previous step, utilizing different versions of the data, including both strong and weak synthetic annotations and data with and without normalization. After evaluating each model on the test set, we further evaluate the trained model using expert annotations. The results of the different data versions can be found in Table 5.

Based on the results presented in Table 5, the RCCNet

IKEM data	Test set F1 [%]	Test set Accuracy [%]	Doctor set F1 [%]	Doctor set Accuracy [%]
SSA	85.06	84.16	81.76	81.57
SSA & Macenko	81.84	81.03	82.02	82.02
WSA	85.13	84.96	81.71	81.42
WSA & Macenko	81.88	81.14	82.00	81.88

Table 5: Evaluation of Student architecture RCCNet on the test set and doctor annotations after training using Teacher architecture.

Labels count (per class)	F1 [%] Before fine-tuning	F1 [%] After fine-tuning	Accuracy [%] Before fine-tuning	Accuracy [%] After fine-tuning
100	81.88	80.47	81.90	80.80
250	81.92	80.73	81.95	81.30
400	82.17	79.85	82.25	80.56
550	82.64	81.21	82.79	82.29
750	83.17	80.15	83.43	81.19
850	83.73	78.40	84.13	79.13
1000	84.24	78.89	84.74	80.26
1250	83.55	82.47	84.24	82.62
1500	**83.30**	**84.93**	**84.23**	**85.58**
1750	82.57	83.99	83.88	84.50
1900	82.73	84.81	85.17	87.23

Table 6: Evaluating the performance of the student architecture on strong annotations with normalization before and after fine-tuning.

model performed better on data without normalization and using SSA within the test set. However, the normalized data performed better for expert annotations. Using WSA performed better for the test set and for expert annotations, the results were better using SSA. The results suggest that by training the student with the teacher, we achieved a higher level of generalization in the trained student model, leading to the improved classification of medical data using normalization.

Our last evaluation is focused on students fine-tuning based on gaining information from expert annotations. Fine-tuning aims to find the smallest number of annotations needed to improve RCCNet students trained by ResNet-18 teachers. For these experiments, we perform a grid search over the chosen nuclei counts for each class concerning the class that contained the smallest number of annotations. We experimented with nuclei counts of 100, 250, 400, 550, 750, 850, 1000, 1250, 1500, 1750 and 1900. Each of the above values represents the per-class count, which we divided into training and validation sets. The test set, over which we computed the metrics for fine-tuned models, is created from the remaining number of nuclei in the dataset.

As presented in Table 6, the results indicate that fine-tuning the pre-trained RCCNet model using SSA and normalized data led to improved performance compared to the initial training. Specifically, using 1500 nuclei per class during the fine-tuning process resulted in a higher F1 Score and Accuracy than before fine-tuning or using lower nuclei counts. However, utilizing a small sample of medical data to fine-tune the RCCNet model led to worse results.

6 Conclusion & Discussion

This study presents a novel robust approach for cell segmentation and classification, evaluated on WSI scans of heart biopsy. This approach can generally be applied to any histological images from different organs and different types of cells. Our method consists of traditional methods used in medicine combined with novel methods for working with weakly annotated data.

Using the Knowledge distillation and the Teacher-Student architecture for nuclei classification, we have identified that it is possible to improve results by using this approach under certain conditions. In this work, we specifically use Teacher-Student chaining. According to our results, the best use of this technique appears to be in the case of normalized data. We identified that there might be an improvement in the results after fine-tuning the student model.

An improvement in the results for normalized data was observed when the number of manually annotated cells per class reached a threshold of 1500, suggesting it may be a suitable cut-off point for expert annotations.

A potential limitation is a total number of cells manually annotated by pathologists. If more cells were annotated, further experiments could be performed to more accurately evaluate the impact of fine-tuning the student model on the medical annotations.

Future research may explore the potential of the Teacher-Student architecture without relying on ANN annotations. This could involve training an initial teacher model on a set of medical annotations and using it to train a student model on previously unseen data, allowing for further analysis and investigation of the approach.

References

[1] Cardiovascular diseases (cvds). https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds). Accessed: 2023-03-23.

[2] Michael D Abràmoff, Paulo J Magalhães, and Sunanda J Ram. Image processing with imagej. *Biophotonics international*, 11(7):36–42, 2004.

[3] Saeed Alahmari, Dmitry Goldgof, Lawrence Hall, Palak Dave, Hady Ahmady Phoulady, and Peter Mouton. Iterative deep learning based unbiased stereology with human-in-the-loop. In *2018 17th*

IEEE International Conference on Machine Learning and Applications (ICMLA), pages 665–670, 2018.

[4] Peter Bankhead, Maurice B Loughrey, José A Fernández, Yvonne Dombrowski, Darragh G McArt, Philip D Dunne, Stephen McQuaid, Ronan T Gray, Liam J Murray, Helen G Coleman, et al. Qupath: Open source software for digital pathology image analysis. *Scientific reports*, 7(1):1–7, 2017.

[5] SH Shabbeer Basha, Soumen Ghosh, Kancharagunta Kishan Babu, Shiv Ram Dubey, Viswanath Pulabaigari, and Snehasis Mukherjee. Rccnet: An efficient convolutional neural network for histological routine colon cancer nuclei classification. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1222–1227. IEEE, 2018.

[6] Sushovan Chaudhury, Nilesh Shelke, Kartik Sau, B Prasanalakshmi, and Mohammad Shabaz. A novel approach to classifying breast cancer histopathology biopsy images using bilateral knowledge distillation and label smoothing regularization. *Computational and Mathematical Methods in Medicine*, 2021, 2021.

[7] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[8] Andres Diaz-Pinto, Sachidanand Alle, Alvin Ihsani, Muhammad Asad, Vishwesh Nath, Fernando Pérez-García, Pritesh Mehta, Wenqi Li, Holger R Roth, Tom Vercauteren, et al. Monai label: A framework for ai-assisted interactive labeling of 3d medical images. *arXiv preprint arXiv:2203.12362*, 2022.

[9] Simon Graham, Mostafa Jahanifar, Ayesha Azam, Mohammed Nimir, Yee-Wah Tsang, Katherine Dodd, Emily Hero, Harvir Sahota, Atisha Tank, Ksenija Benes, et al. Lizard: A large-scale dataset for colonic nuclear instance segmentation and classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 684–693, 2021.

[10] Simon Graham, Quoc Dang Vu, Shan E Ahmed Raza, Ayesha Azam, Yee Wah Tsang, Jin Tae Kwak, and Nasir Rajpoot. Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. *Medical Image Analysis*, 58:101563, 2019.

[11] Noah F Greenwald, Geneva Miller, Erick Moen, Alex Kong, Adam Kagel, Thomas Dougherty, Christine Camacho Fullaway, Brianna J McIntosh, Ke Xuan Leow, Morgan Sarah Schwartz, et al. Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. *Nature biotechnology*, 40(4):555–565, 2022.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Neeraj Kumar, Ruchika Verma, Deepak Anand, Yanning Zhou, Omer Fahri Onder, Efstratios Tsougenis, Hao Chen, Pheng-Ann Heng, Jiahui Li, Zhiqiang Hu, et al. A multi-organ nucleus segmentation challenge. *IEEE transactions on medical imaging*, 39(5):1380–1391, 2019.

[14] Michael R Lamprecht, David M Sabatini, and Anne E Carpenter. Cellprofiler™: free, versatile software for automated biological image analysis. *Biotechniques*, 42(1):71–75, 2007.

[15] Marc Macenko, Marc Niethammer, James S Marron, David Borland, John T Woosley, Xiaojun Guan, Charles Schmitt, and Nancy E Thomas. A method for normalizing histology slides for quantitative analysis. In *2009 IEEE international symposium on biomedical imaging: from nano to macro*, pages 1107–1110. IEEE, 2009.

[16] Eduardo Mosqueira-Rey, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal. Human-in-the-loop machine learning: A state of the art. *Artificial Intelligence Review*, pages 1–50, 2022.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[18] Shayne Shaw, Maciej Pajak, Aneta Lisowska, Sotirios A Tsaftaris, and Alison Q O'Neil. Teacher-student chain for efficient semi-supervised histology image classification. *arXiv preprint arXiv:2003.08797*, 2020.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 3–11. Springer, 2018.

Computer Vision and 3D Reconstruction

Distributed Surface Reconstruction

Patrick Komon, BSc*
Supervised by: Projektass.in Diana Marin, BSc MEng †

Institute of Computer Graphics
Vienna University of Technology
Vienna / Austria

Abstract

As 3D scanning technology is advancing, both quality and quantity of available point cloud data is increasing. Many applications require the reconstruction of the surface of a scanned object as a 3D model. As scans become exceedingly detailed, point clouds become larger and surface reconstruction more computationally challenging. A fast and scalable solution for the reconstruction problem is needed. We constructed and implemented a scalable distributed surface reconstruction algorithm called DISTRIBUTEDBALLFILTER based on the recently developed BALLFILTER algorithm. We executed our implementation on the VSC3+ high performance computing cluster and empirically analysed its speedup as well as its parallel scaling behaviour. In our tests for DISTRIBUTEDBALLFILTER we achieved running times around five times faster than with BALLFILTER.

Keywords: Surface reconstruction, Point cloud, Distributed memory, Parallel computing

1 Introduction

In recent years, 3D scanning technology has become easier to obtain and use. As a consequence, more scanned data is made available as part of private or public projects. One such project is "Wien gibt Raum" [19] in which more than 100 terabytes of point cloud data was collected. It is often favourable or necessary to work with 3D models consisting of points, edges and faces that represent real-world objects rather than point clouds. Thus the challenge is to create a model from the point cloud that most accurately represents the scanned object. In computer graphics, this problem is known as *surface reconstruction*. Introducing noise and outliers, as they are present in most real-life scans, makes solving this problem even more challenging. Large point clouds require fast and scalable surface reconstruction algorithms so that a 3D model can be calculated in reasonable time.

*e11808210@student.tuwien.ac.at
†dmarin@cg.tuwien.ac.at

1.1 Related work

There are several approaches for surface reconstruction. You et al. [21] categorize the existing surface reconstruction algorithms based on their methodology into *interpolation*, *approximation* and *learning-based* approaches. While *soft-computing* approaches are mentioned separately, they will not be further explained as they are not relevant to this work.

Interpolation approaches (also called *combinatorial* approaches) try to reconstruct the surface that (exactly) goes through all sampled points. Usually they use the Delaunay complex or the Voronoi diagram of the point cloud. The CRUST algorithm introduced by Amenta et al. [1, 2] is a well known Delaunay-based combinatorial approach. In their work, they also introduced the notion of ε-sampling, relating surface features with the sampling density. It inspired multiple variants, each improving reconstruction quality for specific cases, for example POWERCRUST, which improves in noisy and under-sampled regions [3].

Approximation approaches try to find the surface by finding a function that best agrees with all sampled points, similar to curve fitting. Widely used in practice, SCREENEDPOISSON surface reconstruction [7] applies Poisson's equation to solve the reconstruction problem. However, it requires knowledge of the surface normal for each sampled point.

Learning-based approaches facilitate some form of machine learning. One recent example is POINTS2SURF [6], which managed to reduce the reconstruction error by 30% compared to SCREENEDPOISSON.

1.2 BALLFILTER algorithm and limitations

Recently, Ohrhallinger [12] developed a new Delaunay-based reconstruction algorithm called BALLFILTER. As the corresponding paper is not published yet, we cannot explain its inner workings. In general, it calculates the Delaunay complex and filters its triangles. By design, BALLFILTER is a serial algorithm and therefore is limited by the physical capabilities of a single machine. This effectively limits the size of data sets it can process in reasonable time.

1.3 Distributed approach

In this paper, we address these limitations from a parallel computing perspective by introducing a distributed-memory parallel version of BALLFILTER. DISTRIBUTEDBALLFILTER first subdivides the input into a three-dimensional grid with overlapping cells. Then BALLFILTER is run on each grid cell separately. Finally, all results are collected and merged together to create the final result.

This removes the memory restriction imposed by using a single machine. Moreover, it enables scaling the input size while still maintaining acceptable run times. We aim at utilizing parallel infrastructure, as is present in state-of-the-art high-performance computing (HPC) clusters, to execute BALLFILTER on much larger data sets than previously possible. We evaluate its performance and scaling behaviour from a parallel computing perspective and compare it against the original algorithm. Specifically, DISTRIBUTEDBALLFILTER will has been tested and run on the VSC-3+ cluster [5].

2 Background

In this chapter, we will be revisiting some theoretical basics. First, we will cover the Delaunay complex. Then, we will briefly mention important aspects of the BALLFILTER algorithm. Lastly, we will go over typical assumptions of the distributed-memory parallel computing model.

2.1 Delaunay complex

Many surface reconstruction algorithms are based on the Delaunay complex (commonly referred to as Delaunay triangulation). In particular, they operate by examining the tetrahedrons obtained from calculating the Delaunay complex for the input point cloud. The challenge then becomes finding the subset of tetrahedrons that most closely reconstructs the original object.

The Delaunay complex of a three-dimensional point set S consists of (non-overlapping) tetrahedrons as well as all of their vertices, edges, triangles. These tetrahedrons must be constructed from points of S and cover its entire convex hull. Furthermore, they must fulfil the *empty-sphere property*, that is, the circumsphere of each tetrahedron must not contain any (other) vertices. For every set of points the Delaunay complex can be calculated in $O(n \log n)$ time [9].

2.2 BALLFILTER reconstruction method

Because the paper presenting the BALLFILTER algorithm is not published yet, we cannot provide details about its function. The most important steps are calculating the Delaunay complex and filtering its triangles. It can reconstruct open surfaces and runs in $O(n \log n)$ time for n points.

2.3 Distributed-memory parallel computing

In the distributed-memory parallel computing model there is a number of p processes that are connected via a communication network. Each process can only access its own local memory. Processes can only interact with each other by communicating over the communication network. There are several paradigms, standards and frameworks providing means for communication. For high-performance computing, the most used is the Message Passing Interface (*MPI*) [10]. It utilizes the paradigm of message passing. Processes communicate with each other by explicitly sending and receiving messages [17]. Costs for such operations are determined by the concrete topology of the communication network.

We will be using the notion of speedup. The *(absolute) speedup* S_{abs} of a parallel algorithm is its improvement over a baseline sequential version, in our case BALLFILTER [15]. The *relative speedup* S_{rel} measures the improvement of multiple processes over using a single one.

3 Method

In the following, we will explain in detail how the parallelization of the algorithm works. First, we will explore how the input is subdivided. Next, we will discuss how those parts are distributed among multiple processes. Then, we will examine the parallel reconstruction step and derive its parallel runtime complexity. Lastly, we will discuss how the resulting partial meshes are merged back together.

3.1 Splitting into overlapping tiles

The approach for input subdivision was developed by Brunner [4] and is thoroughly explained in their dedicated work. We will only explain the most important aspects.

Splitting the input into independent parts is necessary to enable parallelism. We will be splitting the point set along a regular 3D grid into cells. The grid is axis-aligned, covers the entire point cloud and the number of grid cells along each axis is given as input parameters x, y and z, creating a total of $s = x \times y \times z$ cells. The grid cell dimensions are calculated based on the minimal and maximal coordinates present in the point cloud.

However, the union of the Delaunay triangulations of all grid cells is not equal to the Delaunay triangulations of the entire point clouds, as triangles constructed from vertices in different, neighbouring cells will be missing. Visually, this leads to cuts along the 3D grid in the resulting model. Put differently, we have to make sure not to miss triangles, which have vertices in different grid cells.

We solve this problem by considering points in an area around every grid cell in addition to the points within it. This area is given by the so-called *padding*. We call the set of all points within a grid cell and its surrounding padding

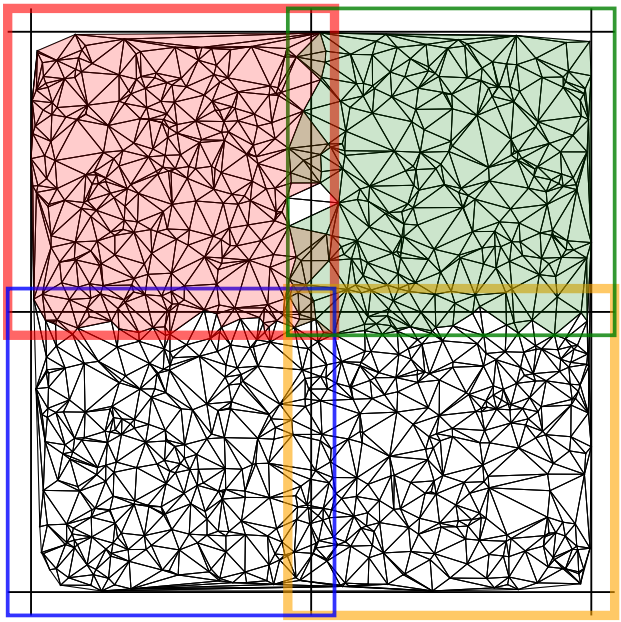


Figure 1: Delaunay-triangulated 2D point cloud, subdivided into a regular grid. Tiles are highlighted by coloured borders with varied thickness for readability.

a *tile*. The set of tiles is not a partition of the point cloud but a set of overlapping subsets. The number of tiles is equal to the number of grid cells. The padding is added to the grid cell dimensions, extending the cell along each axis in positive and negative directions.

The input subdivision is illustrated in Figure 1. Tiles are regions with coloured borders. The vertices of red-coloured triangles belong to the upper left tile. The vertices of green-coloured triangles belong to the upper right tile. In the overlap region, there are triangles with all vertices belonging to both tiles. These will be part of the Delaunay triangulations in each tile separately and thus be considered for reconstruction by BALLFILTER. While this cannot lead to holes in the resulting mesh, it may lead to redundant triangles. This may be fixed as part of a post-processing step, see 3.4.

Two vertices of a triangle can only be in two different tiles exclusively, if the edge between these vertices is longer than double the padding. This case is illustrated in Figure 1, where between green and red tile, there are two white triangles. These will be missing in the separate Delaunay triangulations of each tile. However, by choosing a padding relative to the parameters used for BALLFILTER, we ensure that all triangles that BALLFILTER considers are also present in the Delaunay triangulation of at least one tile.

Checking a single point for tile membership requires checking its coordinates against the grid as well as the padding. This can be done in $O(1)$, therefore calculating the memberships of all points can be done in $O(n)$.

The overlap between tiles leads to the duplication of some points of the input set. In particular, the maximal number of tiles that may be overlapping each other at any given location dictates the (maximal) factor of duplication. Choosing a padding smaller than half the tile dimensions in each direction would imply a maximum of four overlapping tiles in the 2D case (see Figure 1) and eight overlapping tiles in the 3D case.

Therefore, for DISTRIBUTEDBALLFILTER, any point of the input set may be part of up to eight tiles and thus be duplicated up to eight times. The total number of points n' after splitting is $n' \leq 8n$ and is in $O(n)$. From here on we assume $n' = cn$ with some constant factor $c < 8$.

3.2 Work distribution

The next step is to assign the tiles to a fixed number of p *processes* (also called *machines* or *nodes*). We assume all processes are capable of performing the same amount of work in the same time (*assumption of identical machines*). The running time of the entire program is determined by the running time of the slowest process. There is no assumption about the distribution of the points. There may be significant differences in the number of points between tiles. We call an execution of BALLFILTER for a single tile a *job*. The challenge is to assign jobs to processes in such a way that the total running time is minimized. This problem is well known as *load balancing* [8].

There are various ways of approaching this problem and generating optimal solutions can be computationally complex. Instead of trying to find optimal solutions, we will focus on efficiently finding a solution, that is reasonably close to the optimum, thus *approximating* it.

Formally, we want to assign jobs in such a way, that the maximum running time within all processes is minimized. The running time for processing a tile is dependent on the number of points it contains. Thus, the goal is to minimize the total number of points (that is, the sum of the tile sizes) any single process gets assigned.

To minimize this sum, we will be using *list-scheduling* with the *longest-processing-time-first (LPT) rule*. It is a simple and fast approximation algorithm for the scheduling problem, that guarantees a $4/3$-approximation [20], i.e. the calculated schedule is slower than the optimal by factor $4/3$ at the most. In each iteration, a job is assigned to the process with the least number of points to process yet. The jobs are assigned in order from highest number of points to lowest. Sorting the jobs can be done in $O(s \log s)$, with s being the number of tiles. The process assignment is done in $O(s \log p)$. As it only makes sense for p processes to process at least s tiles, it holds that $p \leq s$. Therefore, calculating the assignment is in $O(s \log s)$.

3.3 Processing tiles

Each process executes BALLFILTER on every tile it was assigned. The result of BALLFILTER is a set of triangles, represented by the indices of the vertices in the original point cloud. The run time of a single process is the sum

of the run times of BALLFILTER for the set of all tiles LT assigned to this process, thus

$$O\left(\sum_{t_i \in LT} |t_i| \log |t_i|\right).$$

The term with the largest tile size dominates this sum, so the largest tile of each process dictates its asymptotic run time. The overall run time is determined by the slowest process, which is the one with the largest tile. Therefore, the overall bound can be expressed as

$$O(|t_i| \log |t_i|) \quad \text{with} \quad \forall t_j \in T : |t_i| \geq |t_j|.$$

Hence, the running time of DISTRIBUTEDBALLFILTER is dependent on the distribution of points within the tiles, which itself depends on the distribution in the point cloud, the chosen grid and the padding.

In the best case, every process is assigned exactly an equal part of all points (after duplication), i.e. a number of $\frac{cn}{p}$ points. Therefore, the largest tile t_i can contain at most this many points, formally $|t_i| \leq \frac{cn}{p}$. Since *LPT list-scheduling* guarantees a $4/3$-approximation and BALLFILTER can be done in $O(n \log n)$, this implies a bound for the parallel running time in

$$\begin{aligned} O\left(\frac{4}{3}|t_i| \log |t_i|\right) &= O\left(\frac{4}{3}\frac{cn}{p} \log \frac{cn}{p}\right) \\ &= O\left(\frac{4c}{3}\frac{n}{p}(\log c + \log n - \log p)\right) \\ &= O\left(\frac{n \log n}{p}\right). \end{aligned}$$

In the worst case, almost all points are assigned to a single process. This may happen when the tile sizes are extremely unbalanced, e.g. if two tiles with sizes $cn-1$ and 1 should be assigned to two processes. As the largest tile may be of size cn, the run time of the slowest process and thus the parallel running time is in

$$\begin{aligned} O(|t_i| \log |t_i|) &= O(cn \log cn) \\ &= O(n(\log c + \log n)) \\ &= O(n \log n) \end{aligned}$$

which is (asymptotically) equal to running BALLFILTER on the original point cloud. Due to duplicated points and communication overhead, the actual run times of DISTRIBUTEDBALLFILTER are expected to be higher than BALLFILTER in this case. However, such cases may be mitigated most of the time by carefully choosing the parameters for splitting.

3.4 Merging results and output

Upon finishing executing BALLFILTER on all assigned tiles, each process sends the resulting triangles to a single process, which joins all received sets together and outputs the final model. The padding causes all triangles that are considered by BALLFILTER to be entirely contained within at least one tile. Therefore, there is no need to explicitly connect the meshes of neighbouring tiles, the result is simply the union of all triangle sets.

However, as mentioned in 3.1, reconstructed triangles may be part of more than one tile. In this case, merging the results generates redundant geometry in the resulting mesh. Removal of those triangles is possible as a post-processing step. We will not take redundant triangle removal into account for running time considerations and because it does not interfere with the visual quality of the reconstructed model.

3.5 Summary

To summarize, DISTRIBUTEDBALLFILTER processes a set of n points using the following steps. Note that the point duplication caused by tile overlap is assumed to be a constant factor as discussed in 3.1.

1. Split the input point cloud into tiles in $O(n)$ (on a single node).
2. Calculate schedule in $O(s \log s)$ (on a single node).
3. Perform BALLFILTER in parallel worst case $O(n \log n)$, best case $O(\frac{n \log n}{p})$ (on p nodes).
4. Merge results from nodes in $O(n)$ (on a single node).

The overall asymptotic run time complexity is $O(n \log n)$ in the worst and $O(n + \frac{n \log n}{p})$ in the best case.

4 Implementation

In this chapter, we will briefly mention the technology used as well as discuss the implementation structure.

4.1 Technology

For the implementation, C++ has been used as it offers both, performance and high-level abstractions. As input splitting involves a large number of simple operations, has been use *CUDA* to utilize the large scale shared-memory parallelism possible on GPUs [11]. For distribution of the tiles to the processes, we use the distributed file system of the VSC-3+ cluster, *BeeGFS*. For the reconstruction step performed on each tile, the original implementation of BALLFILTER has been. For the merging step, each process' outputs are sent back to a single node using *Open-MPI* [14].

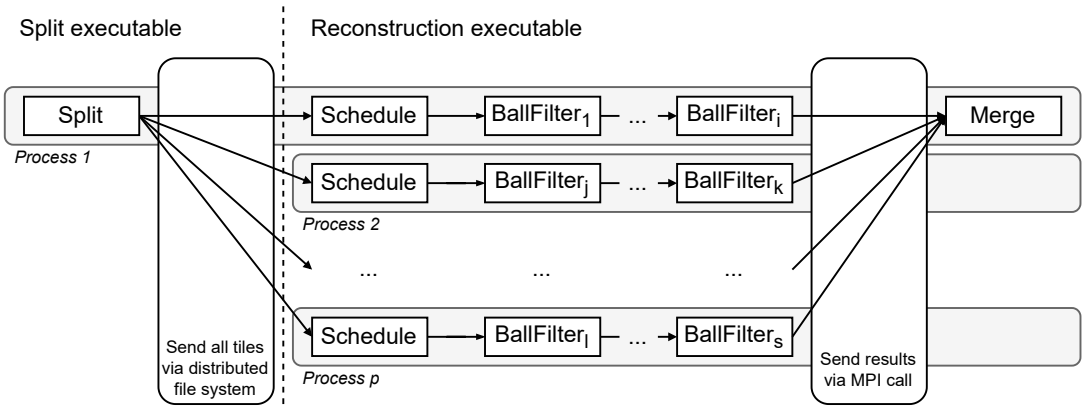


Figure 2: Implementation of DISTRIBUTEDBALLFILTER

4.2 Structure

Splitting the input point cloud and performing surface reconstruction are implemented in separate executables. The output of the splitting step, which is also the input of the reconstruction step, consists of a set of files, each representing a tile (*tile file*). This promotes decoupling of input splitting and reconstruction and allows exchanging implementations as well as introducing additional processing steps. On the VSC3+, tile files are written to the distributed file system, making all tiles available to all nodes after the splitting step.

Then the reconstruction executable is started on all nodes in parallel. Each process calculates the job schedule (tile file assignment) independently. Compared to calculating it on a single node and communicating it to all others, it is still faster because no communication is required. Then each process runs BALLFILTER on each of its assigned tiles. When finished, all nodes send their results to the node with the least load. This node then merges the results and outputs the model. The entire process is visualized in Figure 2. Notice, the structure differs slightly from the one proposed in 3.5. We deliberately chose this approach due to the availability of a distributed file system.

5 Results and Evaluation

In this chapter, we will discuss the results of our implementation on specific data sets and parameter combinations. First, we will give an overview of the hardware environment and explain the data sets and parameter combinations selected for testing. Moreover, we will visualize the running times and reason about the effect of scaling the number of processes p or the input size n. Finally, we compare these times to the original implementation of BALLFILTER.

5.1 Hardware environment and Datasets

The algorithm has been run on the VSC-3+ cluster. It consists of various types of nodes with different hardware specifications [5]. We used two different node types. For the splitting step, we used a single node with a NVIDIA Pascal GeForce GTX 1080 GPU. For the reconstruction step, we used a number of identical nodes, containing two Intel Xeon E5-2660v2 2.2GHz processors with 10 cores each (so 20 cores in total) and 64GiB of RAM. The reconstruction step has been run multiple times while varying the number of nodes used in order to analyse the scaling behaviour.

As input point clouds for testing, two data sets have been selected, both obtained via photogrammetry provided by *Pix4D* [13]. They were chosen based on their large size and real world relevance. The point clouds were truncated to create inputs of various sizes n. In order to observe the running times of DISTRIBUTEDBALLFILTER on scaled input, n is varied while keeping the number of nodes p fixed.

5.2 Parameters

The parameters for BALLFILTER only influence the quality of the resulting 3D model and generally do not significantly impact performance. For this reason, we keep them fixed for all runs at the values recommended in [12].

Splitting the input requires three parameters, x, y and z, denoting the number of tiles along each axis, respectively. Correctly choosing these parameters is vital for work distribution and in turn has a great impact on the overall run time. Although not strictly needed, information about the distribution of points in the input point cloud is helpful for picking concrete values. During testing, these values will be picked based on the number of nodes to employ.

The number of processes p is closely related to the number of tiles. Generating less tiles than there are nodes allocated would leave some nodes idle, waiting for the others to finish, so $p \leq s$. Having more tiles allows for better work distribution and may enable faster overall running

times. This also depends on how balanced the tiles are to begin with. However, using more tiles than nodes leads to overhead by point duplication. At some point, the benefit of better work distribution is exceeded by the additional work of processing duplicated points. In our tests, we will use $s = p$ and $s = 2p$.

5.3 Measured running times

For a closer examination, we chose the `eclepens` data set. The exact running times for various runs with different parameter combinations are listed in Table 1 as well as the absolute and relative speedup. T_{split} and $T_{reconstruct}$ are the times for running the split and reconstruct executables respectively as shown in Figure 2.

The running time of the split step are consistently very low. In comparison, the running times of the reconstruction step are dominating the total running time. As this paper's main focus is the reconstruction step, we will neglect T_{split} and refrain from analysing its impact on the total running time.

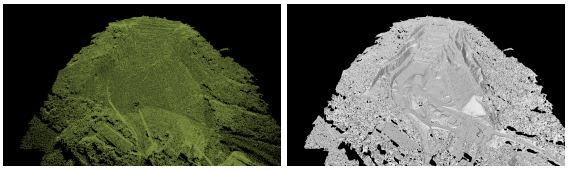


Figure 3: Input point cloud (left) and reconstruction (right) of the `eclepens` (8 million points) data set calculated using DISTRIBUTEDBALLFILTER with 16 processes and 32 tiles in 11.43s.

5.4 Scaling behaviour and speed-up

Figure 4 shows the running times of the original (shared-memory) implementation of BALLFILTER and our implementation of DISTRIBUTEDBALLFILTER ran with 16 nodes and 32 tiles for different versions of `eclepens`. The data set has been truncated to specific sizes (from $2^0 = 1$ to $2^5 = 32$ million points) to simulate growing input size.

BALLFILTER has an asymptotic running time bound of $O(n \log n)$. The asymptotic running time bound of DISTRIBUTEDBALLFILTER is $O(n \log n)$ for the worst and $O(n + \frac{n \log n}{p})$ for the best case. The distributed version is in the worst case (asymptotically) as fast as the original algorithm and faster in the best case. The factor by which it is faster is called absolute speedup S_{abs}. In reality, the absolute speedup will always be somewhere between 1 and p, depending on the balance of the tile set which in turn is based upon the distribution of points in the input point cloud. If p is considered constant, the asymptotic run time complexity for DISTRIBUTEDBALLFILTER is $O(n \log n)$ for all cases and matches the one of BALLFILTER. Therefore, the only difference between both running times lies

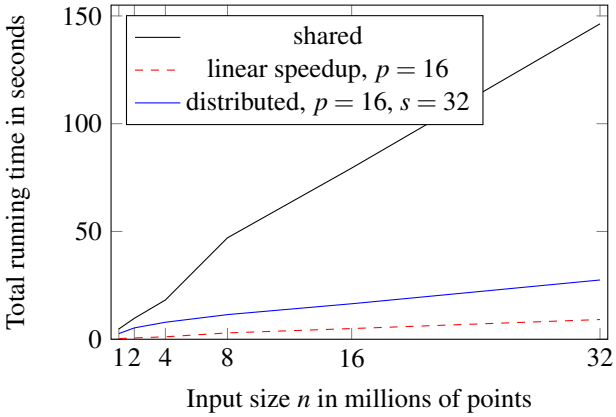


Figure 4: Running times of the original BALLFILTER vs DISTRIBUTEDBALLFILTER with increasing input size for `eclepens`

in the coefficient (which is ignored by asymptotic complexities).

In our example, the distributed version with 16 nodes was faster than the shared version for all input sets. Both curves look similar, as their asymptotic running times suggests. The absolute speedup increased with the size of the input up to a factor of 5.32. The best possible absolute speedup, i.e. linear speedup, would be 16, which would lead to running times that are $\frac{1}{16}$th of the original shared running times.

Now we will investigate how the running times change when the number of processes p is scaled up. We executed DISTRIBUTEDBALLFILTER on the `eclepens` data sets with 16 and 32 million points multiple times, each time doubling the number of nodes used. The number of tiles was set to twice the number of nodes ($s = 2p$), so that in cases of imbalance, the scheduling algorithm can balance out the work between the processes. The running times are listed in Table 1 and visualized in Figure 5.

For one node, DISTRIBUTEDBALLFILTER was slower than the original implementation. That is to be expected because of the overhead required by the distributed implementation. Using two or more nodes, however, significantly decreased the running times compared to the original implementation.

As the absolute and relative speedup approach a value of around 6, the running times stay the same even when the number of nodes is increased. For larger numbers of tiles, the additional work caused by duplicate points eventually cancels out the performance gained by parallel execution.

To summarize, DISTRIBUTEDBALLFILTER performed well on scaling up input size n as well as scaling the number of nodes p. Specifically, on the original `eclepens` data set, when using 16 nodes, it was shown that DISTRIBUTEDBALLFILTER outperformed BALLFILTER by a factor of 5.66. We also observed that having a higher number of tiles can, and in many cases will, increase the overall running time because it enables better work distribution,

	p	$s = x \times y \times z$	T_{split}	$T_{reconstruct}$	T_{total}	S_{abs}	S_{rel}
`eclepens` 16 million points	1	$2 = 2 \times 1 \times 1$	1.35175	88.8745	90.22625	0.88	1.00
	2	$4 = 2 \times 2 \times 1$	1.59808	53.7387	55.33678	1.43	1.63
	4	$8 = 2 \times 2 \times 2$	1.34450	31.4786	32.82310	2.42	2.75
	8	$16 = 4 \times 2 \times 2$	1.25440	22.4666	23.72100	3.35	3.80
	16	$32 = 4 \times 4 \times 2$	1.56540	14.8993	16.46470	4.82	5.48
	32	$64 = 4 \times 4 \times 4$	1.66721	11.8065	13.47371	5.89	6.70
`eclepens` 32 million points	1	$2 = 2 \times 1 \times 1$	1.56998	168.3390	169.90898	0.86	1.00
	2	$4 = 2 \times 2 \times 1$	1.72504	96.3045	98.02954	1.49	1.73
	4	$8 = 2 \times 2 \times 2$	1.62856	61.7569	63.38546	2.31	2.68
	8	$16 = 4 \times 2 \times 2$	1.75069	36.3197	38.07039	3.84	4.46
	16	$32 = 4 \times 4 \times 2$	1.84338	25.6505	27.49388	5.32	6.18
	32	$64 = 4 \times 4 \times 4$	2.24963	23.7843	26.03393	5.62	6.53

Table 1: Parameter combinations and running times for `eclepens`

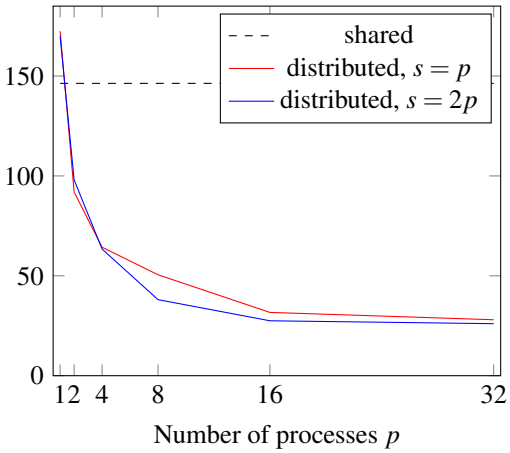


Figure 5: Absolute running times of the original BALLFILTER vs DISTRIBUTEDBALLFILTER with one tile per process and two tiles per process respectively, run on `eclepens` with 32 million points.

despite requiring more duplicated points. Figure 4 shows that DISTRIBUTEDBALLFILTER is a large improvement towards linear speedup.

6 Conclusion and Future Work

Finally, in this chapter we will briefly summarize this paper's results and give a short outlook on future work that could be done on DISTRIBUTEDBALLFILTER.

6.1 Summary

In this paper, we presented a distributed-memory parallel algorithm for 3D surface reconstruction. It works by splitting the input into overlapping chunks, reconstructing a mesh from each chunk in parallel using BALLFILTER and merging the chunk results back together.

We have shown the asymptotic run time complexity to be $O(n \log n)$ in the worst and $O(n + \frac{n \log n}{p})$ in the best case, depending on the distribution of points within the input point cloud. We implemented the algorithm in C++ and tested it on the VSC3+-cluster. In our test runs, we observed that DISTRIBUTEDBALLFILTER improves the running times considerably compared to the original BALLFILTER.

6.2 Future Work

While we analysed mainly from an empirical perspective, DISTRIBUTEDBALLFILTER can be analysed in a more formal setting. Speedup and scaling properties can be argued and proven formally in order to evaluate our approach in a more theoretical sense. Also, the best- and worst-case asymptotic running time complexities could be expressed in more concrete ways, as coefficients oftentimes do matter in the practical comparison of algorithms. Formally taking into account the distribution or balance of the points within the input point cloud may yield further insights into the properties of DISTRIBUTEDBALLFILTER algorithm.

References

[1] Nina Amenta, Marshall Bern, and David Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2):125–135, 1998.

[2] Nina Amenta, Sunghee Choi, Tamal Dey, and Naveen Leekha. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry & Applications*, 12, September 2000.

[3] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, SMA '01, page 249–266, New York, NY, USA, 2001. Association for Computing Machinery.

[4] Lukas Brunner. Technical report - work in progress. TU Wien, 2022.

[5] VSC Vienna Scientific Cluster. Vienna Scienfic Cluster (VSC) - website. https://vsc.ac.at/systems/vsc-3/, 2022. [Online; accessed 13-September-2022].

[6] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J. Mitra, and Michael Wimmer. Points2Surf: Learning Implicit Surfaces from Point Clouds. In *Computer Vision – ECCV 2020*, pages 108–124. Springer International Publishing, 2020.

[7] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), July 2013.

[8] Jon Kleinberg and Éva Tardos. *Algorithm design*, chapter 11.1 Greedy Algorithms and Bounds on the Optimum: A Load Balancing Problem. Pearson Addison Wesley, Boston, Mass. [u.a.], internat. ed.. edition, 2006.

[9] Geoff Leach. Improving worst-case optimal delaunay triangulation algorithms. In *In 4th Canadian Conference on Computational Geometry*, page 15, 1992.

[10] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 4.0*, June 2021.

[11] NVIDIA Corporation. CUDA Toolkit - project website. https://developer.nvidia.com/cuda-toolkit, 2022. [Online; accessed 13-September-2022].

[12] Stefan Ohrhallinger. Personal communication. TU Wien, 2022.

[13] Pix4D SA. Pix4D - website. https://www.pix4d.com/, 2022. [Online; accessed 13-September-2022].

[14] The Open MPI Project. OpenMPI - project website. https://www.open-mpi.org/, 2022. [Online; accessed 13-September-2022].

[15] T. Rauber and G. Rünger. *Parallel Programming: for Multicore and Cluster Systems*, chapter 4.2.1 Speedup and Efficiency. Springer Berlin Heidelberg, 2010.

[16] SchedMD LLC. Slurm Workload Manager - project website. https://slurm.schedmd.com/overview.html, 2021. [Online; accessed 13-September-2022].

[17] B. Schmidt, J. Gonzalez-Martinez, C. Hundt, and M. Schlarb. *Parallel Programming: Concepts and Practice*, chapter 9.1 Message Passing Interface. Elsevier Science, 2017.

[18] The CGAL Project. CGAL - project website. https://www.cgal.org/, 2022. [Online; accessed 13-September-2022].

[19] Stadt Wien. Wien Gibt Raum - project website. https://digitales.wien.gv.at/projekt/wiengibtraum/, 2022. [Online; accessed 13-September-2022].

[20] Xin Xiao. A direct proof of the 4/3 bound of LPT scheduling rule. In *Proceedings of the 2017 5th International Conference on Frontiers of Manufacturing Science and Measuring Technology (FMSMT 2017)*, pages 486–489. Atlantis Press, 2017/04.

[21] Cheng Chun You, Seng Poh Lim, Seng Chee Lim, Joi San Tan, Chen Kang Lee, and Yen Min Jasmina Khaw. A survey on surface reconstruction techniques for structured and unstructured data. In *2020 IEEE Conference on Open Systems (ICOS)*, pages 37–42. IEEE, 2020.

Real-Time Rendering

Real-time Rendering of Atmosphere and Clouds in Vulkan

Matěj Sakmary*
Supervised by: Jaroslav Sloup†

Department of Computer Graphics and Interaction
Czech Technical University in Prague
Prague / Czech Republic

Abstract

This work presents a Vulkan-based implementation rendering volumetric clouds and atmosphere. We combine previously published solutions to produce a single unified look. We use Raymarching as the main method to render both the atmosphere and clouds. Furthermore, we use multiple precomputed look-up tables (LUTs) proposed by Hillaire to speed up the rendering of the atmosphere. We enhance these methods with the option to render volumetric clouds using a precomputed three-dimensional texture setup storing procedurally generated noise. With our final solution, we can render images in a high dynamic range. We apply post-processing effects and use adaptive luminance to transform the image into a low dynamic range for presentation.

Keywords: Volumetric clouds, Real-time rendering, Atmosphere, Vulkan

1 Introduction

Having a realistic and believable atmospheric model when rendering dynamic environments in interactive applications is an important part of creating virtual worlds. The atmosphere and cloud configuration can instantly change the mood of the scene. This is especially important for applications that require dynamic time of day and weather. In addition, these effects are also interconnected and affect each other, making them even harder to simulate. Despite the gradual increase of computing power available in personal computers, simulating complex light interactions that produce the appearance of sky and clouds along with the constraint of displaying such effects in real time is still difficult. To avoid these problems, we are forced to adapt a number of approximations, gaining a significant reduction in the problem complexity. The goal of this work is to provide a complete solution to render dynamic clouds and the sky in real time. To achieve this, we combine multiple well-described techniques into a single solution.

In Section 2, we describe various approaches to rendering volumetric media along with their strengths and weaknesses. A short summary of the relevant topics in physics follows in Section 3. For brevity's sake we only provide a short description meant as a reference. More detailed descriptions, along with explanations, can be found in a textbook on these topics [11]. Lastly, in Sections 4 and 5 we propose the solution and describe our implementation. The images rendered by our implementation can be observed in Figures 7 and 8.

2 Related work

As we focus on real-time rendering, we will only describe methods that are relevant in this context. The most physically accurate method to render volumetric effects is to use path tracing [8, 10]. This method sends rays from the camera and follows them as they bounce when hitting objects in the scene until they reach a light source. Although using this method produces the most physically accurate effects, the computational complexity is very high. This is caused by the number of rays we need to trace to reduce the noise in the final image.

Another approach proposed by Hosek or Wilkie [7, 15] is to use fitted mathematical models. Methods leveraging this principle usually build a set of parameters from measured data used to evaluate the look of the sky. These models are very fast; however, due to the dependence on the data measured in the real world, these methods do not provide the option to change the parameters of the atmosphere. In addition, when the parameters of the atmosphere are changed, a new model has to be fitted. This is unsuitable for the goal of this work because we want our method also to visualize planets different from Earth.

Finally, many methods use raymarching to achieve their results [2]. Offering a good compromise between physical accuracy and speed, it is very popular in problems requiring rendering volumetric effects, such as clouds, mist, or atmospheres. Unlike path-tracing, ray marching does not spawn additional rays. Instead, multiple steps are taken along a ray, sampling the medium at each step. These medium samples are then used to calculate the final look of the ray-marched medium. Consequently, we chose to use ray marching in our implementation.

*sakmamat@fel.cvut.cz
†sloup@fel.cvut.cz

2.1 Atmosphere

The first methods for rendering physically based atmospheres evaluated only single scattering by ray marching the atmosphere from viewpoint for each pixel on the screen [13]. Although omitting multiple scattering has performance benefits, it does not produce realistic looking results, especially for more dense atmospheres, which results in overly dark scenes. Due to this, methods that take into account multiple scattering were introduced [3, 16].

Such methods usually rely on precomputing parts of the computation and storing them in 2D, 3D, and 4D tables called Lookup Tables (LUTs) to speed up the evaluation. This significantly improves the rendering time. Where previously the same evaluation was repeated hundreds of times, now it is only computed once at the beginning. The results are then accessed whenever needed. The main drawback of these methods is the inability to change the atmosphere parameters in real time. Whenever the atmosphere parameters change, all of the LUTs used have to be recalculated, which is a very expensive operation. This results in a long delay before seeing the changes. Another disadvantage is that, because the results are obtained by raymarching each pixel, the performance is tied to the resolution of the screen.

Hillaire et al. [6] introduced solutions to overcome the above-mentioned problems. The first proposal was a new method to evaluate multiple scattering inspired by a dual-scattering approximation used when simulating multiple scattering effects in hair. This reduced the time to precompute LUTs greatly, enabling the update of the atmosphere parameters with almost no delay. The second proposal was to precompute the final sky-view and the aerial perspective into fixed-size latitude/longitude textures, which are later sampled and upscaled. This effectively decouples the computation complexity from window resolution and introduces additional speed improvements. Bruneton [2] provides a good general summary and comparison of various sky models described in this section.

2.2 Clouds

We summarize previous approaches to rendering clouds that are most interesting or relevant to our work. One possible approach was to represent clouds as volumes of particles. For example, Yusov [17] presented a particle-based rendering method. The clouds were modeled using randomly rotated and scaled copies of a single reference particle. The complex optical properties of the reference particle were precomputed, making this process viable for use in real-time applications.

Another technique was presented by Bouthors et al. [1]. By combining meshes to represent low resolution cloud boundaries together with procedural volumetric hypertextures, which add the detail under the mesh boundary, an efficient cloud representation was achievable. When rendering, the cloud surface is covered with circular collectors that are used to evaluate the incoming light. Using this information along with a set of precomputed transfer tables, light is integrated. The cloud representation, however, is not trivial to tweak. This, along with the relatively high overall complexity of the described method, is why simpler methods were developed.

The more recent work by Schneider and Vos [12] uses a fully procedural set of volumetric noise textures to produce similar results. These noise textures are used to represent changes in density in a medium caused by clouds. The clouds are then rendered by ray marching the cloud volume and sampling the medium. This method allows to completely change the overall look of the cloud layer by only tweaking a few parameters while simulating dynamic lighting conditions caused, for example, by changing the time of day. Due to the above reasons, we use this method in our implementation.

3 Physical model

Light transport in participating media is a well-studied problem in computer graphics, described in detail in many articles. This section summarizes the fundamentals of light propagation in the atmosphere relevant to this work and is strongly motivated by works that previously describe these topics [5, 6, 11].

When electromagnetic radiation travels through the atmosphere, it collides with the molecules that make up the atmosphere. During this collision, part of the energy carried by the radiation is absorbed, part is reflected (scattered), and part is emitted. The amount of extinct, scattered and absorbed energy is given by the respective *extinction, scattering and absorption coefficients* denoted as β_e,β_a, and β_s. The *absorption coefficient* is defined as

$$\beta_a = \frac{4\pi n_i}{\lambda} \tag{1}$$

where λ is the wavelength of radiation in vacuum and n_i is the complex part of the *index of refraction*. Thus, this coefficient denotes the rate of energy attenuation per unit of distance at a point x. Similarly, we define a scattering coefficient. The extinction coefficient is then defined by the sum of the absorption and scattering coefficients

$$\beta_e = \beta_a + \beta_s. \tag{2}$$

To correctly compute attenuation over a path where the extinction coefficient varies, integrating the coefficient along the entire path of the ray is required. So, the amount of light that arrives at the point x_2 from the point x_1 given the intensity of light at this starting point and *extinction coefficient* β_e is given by equation 3

$$L(\lambda, x_2) = L(\lambda, x_1) \exp\left[-\int_{x_1}^{x_2} \beta_e(x)\, dx \right] \tag{3}$$

where λ is the wavelength of the radiation considered. The exponential term is often referred to as transmittance and

is denoted by T

$$T(x_1,x_2) = e^{-\int_{x_1}^{x_2} \beta_e(x)\,dx}. \tag{4}$$

We also have to consider the scattering effects of the atmosphere. Unlike absorption effects, when radiation is scattered away, it is added to the atmosphere at a different point. The radiation scattered away is not uniform in all possible directions. To represent the directional distribution of the scattered light, we use a *scattering phase function* denoted by $P(cos\theta)$ where θ is

$$\cos\theta = \vec{\omega}' \cdot \vec{\omega} \tag{5}$$

with $\vec{\omega}'$ being the incoming direction of the light and $\vec{\omega}$ being the direction of the ray we consider. We can formulate the scattering phase function so that it depends only on the parameter Θ because the particles in the atmosphere are spherical or randomly oriented.

Taking this into account, the scattering formula is denoted as follows:

$$dL_{scat}(\lambda,x,\vec{\omega}) = \int_{4\pi} \beta_s(y) P(cos\Theta) L(\lambda,x,\vec{\omega}')\,d\vec{\omega}'\,ds. \tag{6}$$

By combining the two previously described effects (Eq. 3 and Eq. 6), we get the following form:

$$L(\lambda,x,\vec{\omega}) = \underbrace{T(x,x_0)L(\lambda,x_0,-\vec{\omega})}_{direct\ light\ \ from\ the\ sun} + \underbrace{\int_x^{x_0} \beta_s(y)T(x,y)\int_{4\pi} P(cos\Theta)L(\lambda,y,\vec{\omega}')\,d\vec{\omega}'\,dy}_{in-scattered\ light\ \ along\ the\ ray} \tag{7}$$

where λ is the wavelength of the radiation considered, x is the origin of the ray, and $\vec{\omega}$ is the direction of the ray. The second term (i.e. *direct light from the sun*) almost directly corresponds to Equation 3. We rewrote the second part, corresponding to the in-scattered light, as follows. We can take the coefficient β_s from the inner integral, as it remains constant in the integrated area. Since we consider in-scattered light along a ray, as opposed to Equation 6 where we consider in-scattered light at a single point, we integrate over the entire ray and weigh the results by transmittance.

Next, we will describe two models used to substitute the real scattering phase function $P(cos\Theta)$. First, for particles that are much smaller than the wavelength of incident radiation, such as clear air molecules or ozone, the Rayleigh scattering phase function is used. We use the model proposed by Costa et al. [5]

$$P_R(\theta) = 0.7629(1+0.932\cdot cos^2(\theta))\cdot\frac{1}{4\pi} \tag{8}$$

Second, for particles comparable to or larger than the wavelength of the incident radiation, dust or water droplets, for example, Mie's theory was used. Larger particles, such as aerosols, tend to scatter light strongly forward. We use the double Henyey-Greenstein phase function approximation proposed again by Costa et al. [5]

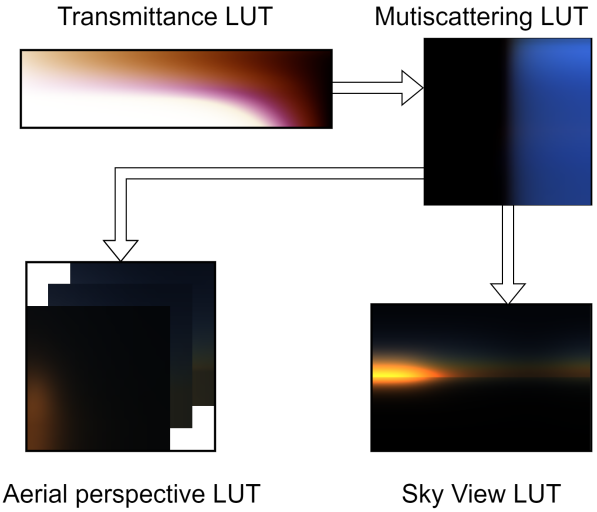


Figure 1: The order in which individual Sky LUTs are drawn. Please note that the color values of LUTs have been scaled in order to be properly visible.

$$P_M(\theta,g_1(\lambda),g_2(\lambda),\alpha(\lambda)) = \alpha * P_{fb}(...) + (1-\alpha) * P_{fb}(...) \tag{9}$$

$$P_{fb}(\theta) = \frac{(1+g_1^2(\lambda))}{(1+g_1^2(\lambda)-2g_1(\lambda)cos(\theta))^{\frac{3}{2}}}. \tag{10}$$

For more details on scattering or extinction coefficients, see [5].

4 Proposed solution

As mentioned above, to speed up the time taken to render the atmosphere, it is beneficial to precompute certain parts of the rendering equation and store them in multidimensional tables. We use the LUT setup proposed by [6], four LUTs storing precomputed parts of Equation 7. Individual LUTs and their dependencies can be seen in Figure 1.

4.1 Atmosphere precomputations

Transmittance LUT introduced by Bruneton et al. [3] is used to store the transmittance T described by Equation 4. When the atmosphere is ray-marched, the value of T is used frequently to model the atmosphere light attenuation. To compute this value, a second ray must be traced towards the light source. Given the overall smooth distribution of the atmospheric transmittance, we precompute the transmittance value for the entire atmosphere.

For **Multiscattering LUT** a new approach proposed by Hillaire [6] was used. We precompute the scattering contribution denoted by Equation 6 at several discrete points in the atmosphere. The incoming radiance from the Sun ($L(\lambda,x,\vec{\omega}$‘) in Equation 6) should be weighed by the

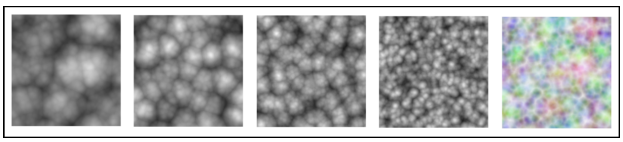


Figure 2: From left to right separate RGBA channels storing Worley noise and all of the channels combined together in the rightmost image.

transmittance. Here, we use the Transmittance LUT to retrieve the values instead of computing them directly.

Sky-View LUT represents the far sky mapped into a latitude/longitude texture that is much lower in resolution than the final image. This LUT stores the values of Equation 7. Similarly to the above, we use the Transmittance LUT to retrieve transmittance values. Additionally, we also interpolate values stored in Multiscattering LUT when marching each ray instead of computing in-scattered light along the ray (Equation 7). The highest visual frequency is introduced by the Sun. Thus, we orient the Sky-View LUT so that the sun is always present at the same position in the texture. We map the values non-linearly, by adding more samples near the horizon.

Lastly, we use **Aerial (AE) perspective LUT**. The Aerial perspective refers to how we see objects as they recede into the distance from the viewpoint. A 3D lookup table is precomputed. To parameterize along the z-axis, we use the distance from the viewing position. At each depth level, a 2D LUT is fitted to the camera view frustum. Each layer of the Aerial Perspective LUT contains the luminance of the atmosphere (Equation 7) and the average transmittance at the corresponding depth (Equation 4). Similarly to Sky-View LUT at each depth level we use the results stored in Transmittance and Multiscattering LUTs to speed up the computation.

4.2 Rendering process

The process of rendering a single frame is divided into four parts. The first part computes the four LUTs used to render the atmosphere. The second part draws all of the scene objects and terrain. Along with this, the atmosphere, its effects, and clouds are also rendered. The next step is to map the values from the HDR range into LDR that is used by the image presented to the screen. The final step renders the user interface that controls various parameters of the atmosphere and clouds. Moreover, our rendering process includes a fifth standalone part, which is to compute the Worley noise texture later used to draw the clouds. We reuse this texture instead of recomputing it each frame.

We based our work on a popular approach to cloud rendering, first introduced by Schneider and Vos [12], which relies on the use of inverted Worley noise. The computation of non-inverted Worley noise can be split into two parts. First, a number of points are randomly distributed in a desired volume for 3D texture. After this, for each

Look up table	Resolution	size
Transmittance LUT	256×64	128 KiB
Multiscattering LUT	32×32	8 KiB
Sky-View LUT	192×128	198 KiB
Aerial Perspective LUT	$32 \times 32 \times 32$	256 KiB
Total		590 KiB

Table 1: Parameterization and LUT sizes used to render the atmosphere.

voxel in the desired area, the distance to the nearest point was calculated and stored. Inverting Worley noise simply consists of storing $d_{max} - d$, where d_{max} is the maximum possible distance between a point and a voxel and d is the distance from the currently processed voxel towards the nearest point. We precompute multiple 3D textures that contain Worley noise with various frequencies. These textures are then sampled by raymarching the cloud.

We follow the method proposed by Lague [9]. It uses two 4-channel 16-bit float textures. Both textures store separate Worley noises with increasing frequencies in each of the RGBA channels. The red channel then stores Worley noise with the lowest frequency, and the alpha channel stores noise with the highest frequency. These textures can be seen in Figure 2.

The texture will have to be tiled multiple times to cover the entire skydome. This gives another requirement for the texture to be tileable (seamless) along all three dimensions.

5 Implementation

As in most performance-dependent applications, this work was implemented using C++. The Vulkan API was used as an interface to the GPU.

5.1 Application resources

In this section, we describe all the application resources and their format. We will mostly omit small uniform and storage buffers used only for parameterization, as they are multiple orders smaller than the LUT textures and have no real effect on the memory requirements of the application.

When rendering the atmosphere, four previously described LUTs have to be computed. We use a 16-bit RGBA texture for each LUT. The parameterization that we decided to use can be seen in Table 1.

In addition to the above, we use two volumetric textures, **Base Noise LUT** and **Detail Noise LUT**, which store Worley noise. Similarly to LUTs used to render the atmosphere, these textures store 16-bit floating point values in each of the channels. The parameterization, along with the size, can be seen in Table 2.

Look up table	Resolution	Size
Base Noise LUT	$256 \times 256 \times 256$	128 MiB
Detail Noise LUT	$128 \times 128 \times 128$	16 MiB
Total		144 MiB

Table 2: Parameterization and LUT sizes used to store Worley noise.

5.2 Main draw loop

Because most of the command buffers in our implementation are prerecorded and do not need to be reconstructed, the main purpose of the draw loop is to keep the GPU fed with as much work as possible. In order to do this, we have multiple *frames in flight*. By default, in our implementation, two frames are in flight at the same time.

At the start of our loop, we check if we do not already have more images in flight than we want. For this purpose, we create a fence for each frame in flight that we want to have. When we are sure that the number of frames in flight is less than the maximum specified values, we continue by acquiring the next image index from the swap chain. This prevents the above-mentioned issue of slowly overflowing our command queues.

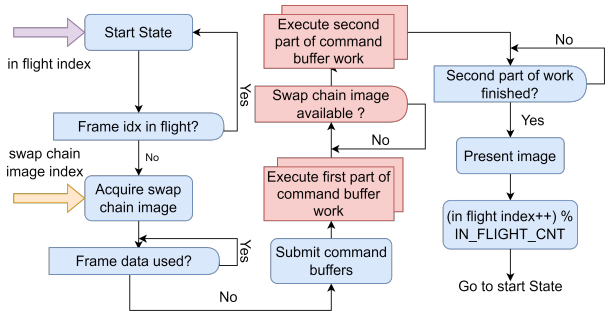


Figure 3: Flow diagram of the draw loop execution order. CPU parts as well as CPU-GPU synchronization points are colored blue. Similarly, GPU parts and GPU-GPU synchronization are colored red.

Because swapchain images might be returned out of order, we have an array of structures containing all of the data that change during the process of rendering one frame and a fence specifying whether the data are currently being used by some in flight frame. Whenever a new image is acquired from the swapchain, we check the corresponding frame data structure fence. Only after the fence has been signaled is an appropriate command buffer submitted to GPU. Whenever we finish rendering any frame, a structure fence is signaled, allowing another frame to be submitted. Figure 3 shows a flow diagram visualizing the entire draw loop.

For each frame, four command buffers are submitted to the GPU. The first pair of command buffers can start executing immediately. Since the second pair of command buffers writes directly into a swap chain image, we need to make sure that the corresponding image is available for us to write. We use an additional array of semaphores. Each semaphore is signaled when the presentation engine is finished using the corresponding image.

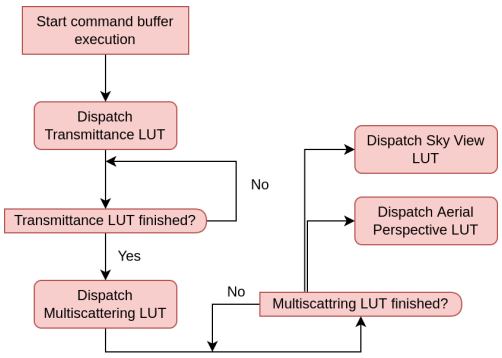


Figure 4: Flow diagram showing dependencies between individual operations in LUTs command buffer. All parts are either executed on GPU or are GPU-GPU synchronization, so they are all marked red.

5.3 Command buffer descriptions

The four parts of the proposed solution described in the previous section are directly linked to four command buffers that are submitted to the GPU for each frame. In this section, we provide a fairly detailed description of the commands that are submitted in each command buffer. We will also describe the GPU-GPU synchronization that takes place inside each of the command buffers. Think of this section as a description of Vulkan-specific parts in our implementation. This, of course, is not everything that is Vulkan-specific in our application; however, as we did not believe those other parts unique to our implementation, we decided to omit them.

In our implementation, we use compute shaders to fill out all LUTs. Each LUT is computed by one shader in one dispatch. The compute dispatch commands are recorded in the order shown in Figure 1 in the command buffer. Because there are data dependencies between individual LUTs, we need to introduce synchronization between the individual dispatch commands. We use pipeline barriers after each dispatch, waiting after each drawcall. This is to ensure that all of the compute work previously submitted has been finished before issuing another dispatch. Figure 4 shows the visualization of the execution order in this command buffer, as well as the synchronization performed between executions.

5.3.1 Worley noise command buffer

As mentioned above, sometimes an additional LUT command buffer may be submitted that computes the 3D Worley noise textures. We again opt for compute shaders when generating this texture.

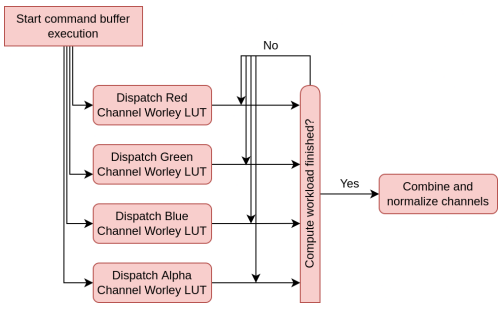


Figure 5: Flow diagram showing dependencies between and execution order of Worley noise command buffer.

Following the approach used by [9] in the first pass, we render Worley noise, followed by the second pass, which normalizes the values in the range between 0 and 1. We used a separate one-channel texture for each of the final four textures in the first pass. These four textures are combined into a single 4-channel texture in the second pass.

The only synchronization that we have is between the first and second passes. In our case, a single pipeline barrier is inserted. This makes sure that all writes and reads in the compute shader stage by previously called dispatches have finished before we normalize and combine all channels together. The entire execution process can be seen in Figure 5.

5.3.2 Sky command buffer

Next we render all objects in our scene and draw the sky with clouds. The ordering here is important; we first render all objects before drawing the sky and clouds. This is because drawing the sky, clouds, and atmosphere requires depth information about the rest of the scene. After all scene objects were rendered far sky is drawn where no object was drawn in the previous pass. Then, the clouds are rendered. Lastly, the aerial perspective is applied. We used the depth when raymarching the clouds as well as the index into the aerial perspective LUT.

In order for aerial perspective to correctly apply on clouds, we also need information about how far the clouds are from the viewing point stored in the depth texture. Because Vulkan does not allow a read and write the same texture from the same shader, we introduce a second depth texture. When rendering clouds, we use the first depth texture as an input and combine it with the depth of the rendered clouds. We write this result into the second output texture. Figure 6 shows how every pass reads or writes resources from the framebuffer.

We divide this command buffer into multiple subpasses and use subpass dependencies for image transitions as well as synchronization. Each subpass is responsible for one of the phases described above. After each subpass finishes, a set of barriers corresponding to Figure 6 is executed.

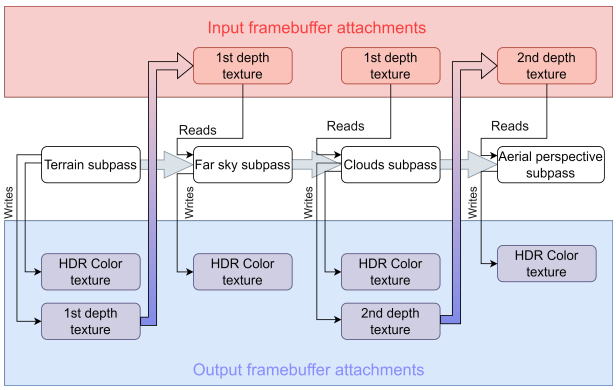


Figure 6: Visualization of writes and reads performed by each pass. We also show when transitions from output attachment into input attachment inside a framebuffer occur.

5.3.3 Post process and GUI command buffers

At the end of each frame, the post-processing of the final image and the drawing of the UI follow. To draw the UI, we use the ImGUI library [4]. The UI gives the user control over the parameters used while rendering the sky. Position of the Sun in the sky, scattering and absorption coefficients, atmosphere height, and falloff of Rayleigh and Mie particle density. As for the clouds, the height where the could layer starts, the thickness of the cloud layer, phase function parameters, and weights and scales of the noise textures used. It is also possible to control the relevant tonemapping parameters.

For the purposes of tonemapping, we first need to calculate the average luminance of the current image. We use a two-pass compute approach described by [14]. In the first pass, a histogram of the luminance values in the image is constructed. The second pass reads this histogram and calculates the weighted sum. This sum is then used to calculate the adaptive average luminance of the scene. Our post-process fragment shader then reads this average value and uses it for tonemapping. A pipeline barrier is inserted between the construction of the histogram and the calculation of the average luminance. A second pipeline barrier is inserted before tonemapping to ensure that the previous average luminance was written. The last command buffer draws the UI on the screen. We do not need any synchronization, everything is handled internally by the ImGUI implementation.

6 Results

In this chapter, we present the results obtained by using our implementation. We also provide the performance for Earth-like setup. The scene was tested on two computers - PC1 with AMD RYZEN 7 1700 & NVIDIA GTX 1080 and PC2 with Intel Core i7 7700HQ & NVIDIA GTX 1050 (mobile). Most of our frame budget was spent on raymarching clouds (see Table 3). Performance is

(a) Sparse cloud cover with the sun nearing sunset. Aerial perspective. Effects are clearly visible on the terrain towards the sun.

(b) Clouds during sunset. The clouds become much darker as not much sunlight reaches them through the atmosphere.

Figure 7: Images of clouds and atmosphere obtained by using Earth-like conditions.

highly dependent on the resulting cloud quality we want to achieve. In all of our benchmarks, we only had a single frame in flight. This slightly reduces the stability of the frame rate, but in return gives more consistent measurements.

6.1 Earth with medium cloud cover

Our first testing scene was an Earth-like atmosphere setup. We can see that we are only barely hitting 60 frames-per-second on PC1. The execution times of individual shaders can be seen in Table 3. The main bottleneck is the cloud rendering, which is expected as we raymarch each pixel. The resulting images can be seen in Figure 7.

6.2 Fictional planet

The second scene uses cloud and atmosphere parameters that are not based on reality. These settings are used to demonstrate the flexibility of our implementation. Given the procedural nature of our clouds combined with the parameterizable atmosphere, we are able to completely change the overall look and mood of the entire scene by tweaking a few values. Additionally, these fictional settings demonstrate the interconnected effects of both atmosphere and clouds, producing consistent results even when we change the values outside of the ranges we are able to observe in the real world. The rendered images of the fictional setup can be seen in Figure 8.

	PC1		PC2
Shader	1080p	720p	1080p
Transmittance LUT	63.9 μs	63.9 μs	233.7 μs
Multiscattering LUT	51.0 μs	51.1 μs	208.4 μs
Sky-View LUT	33.3 μs	32.4 μs	129.7 μs
AE Perspective LUT	56.2 μs	56.4 μs	184.9 μs
Draw Terrain	2.9 ms	2.9 ms	10.86 ms
Draw Far Sky	216.7 μs	104.7 μs	979.9 μs
Draw Clouds	11.6 ms	7.15 ms	43.7 ms
Draw AE Perspective	278.3 μs	125.9 μs	1.26 ms
Construct Histogram	228.1 μs	103.6 μs	415.3 μs
Sum Histogram	3.3 μs	3.3 μs	3.9 μs
Tonemapping	341.0 μs	147.3 μs	1.12 ms
Total	15.79 ms	10.73 ms	59.1 ms

Table 3: Average execution times of each shader for the Earth-like planet with clouds.

We have raised the Mie scattering and extinction coefficients, as well as the Rayleigh scattering coefficient, by almost two orders. Together with the increase in the distribution of particles throughout the medium, we are able to simulate a very dense atmosphere. We achieved the purplish-blue look of the atmosphere by leaving the blue-wavelength component of the Rayleigh scattering coefficient lower. As a result, most of the light in the red-green wavelength gets scattered away by the atmosphere before reaching the eye of the observer. To match the aerial perspective effects with the sky look, we also lowered the blue-wavelength component of the Mie absorption coefficient, allowing more blue light to penetrate the atmosphere.

7 Conclusion and future work

We have described the implementation of an interconnected system to render atmospheric effects. We leveraged the GPU for most of our computations, and thus reached real-time frame rates. The model described previously by Hillaire [6] was used to render the sky. In addition to this, a technique presented by Schneider and Vos [12] was implemented, which allows us to combine the atmosphere model with procedurally generated clouds.

The implemented solution allows visualization of miscellaneous settings ranging from ones based on reality to entirely fictional. Although our implementation relies on using multiple LUTs, it is still possible to change all the parameters during the application's run-time.

The most pressing issue of the implementation presented is the performance of rendering clouds. The current cloud raymarching implementation is naive. We do not take into account the distribution of the media to alter the step size or change the sample distribution. Furthermore, we do not temporally accumulate the raymarch

(a) Dense cloud cover with a thick cloud layer. The atmosphere is denser and the scattering and absorption coefficients of the particles were altered.

(b) Sun is near the horizon, and the atmosphere absorbs most of the light before it reaches the cloud layer.

Figure 8: Images of clouds and atmosphere obtained by using fictional conditions.

results across multiple frames, which would also bring a performance improvement, as the number of steps needed during the raymarch could be significantly reduced. Thus, we believe that optimizing cloud raymarching is a promising direction. Alternatively, we expect that adding hard and soft volumetric shadows along with godrays could improve the appearance and realism of the resulting images.

References

[1] Antoine Bouthors, Fabrice Neyret, Nelson Max, Eric Bruneton, and Cyril Crassin. Interactive multiple anisotropic scattering in clouds. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 173–182, 2008.

[2] Eric Bruneton. A qualitative and quantitative evaluation of 8 clear sky models. *IEEE transactions on visualization and computer graphics*, 23(12):2641–2655, 2016.

[3] Eric Bruneton and Fabrice Neyret. Precomputed atmospheric scattering. *Computer Graphics Forum*, 27(4):1079–1086, 2008.

[4] Omar Cornut. Dear imgui graphical user interface library. https://github.com/ocornut/imgui, 2023.

[5] Jonathas Costa, Alexander Bock, Carter Emmart, Charles Hansen, Anders Ynnerman, and Claudio Silva. Interactive visualization of atmospheric effects for celestial bodies. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):785—-795, 2021.

[6] Sébastien Hillaire. A scalable and production ready sky and atmosphere rendering technique. *Computer Graphics Forum*, 39(4):13—22, 2020.

[7] Lukas Hosek and Alexander Wilkie. An analytic model for full spectral sky-dome radiance. *ACM Transactions on Graphics (TOG)*, 31(4):1–9, 2012.

[8] Eric P Lafortune and Yves D Willems. Rendering participating media with bidirectional path tracing. In *Rendering Techniques' 96: Proceedings of the Eurographics Workshop in Porto, Portugal, June 17–19, 1996 7*, pages 91–100. Springer, 1996.

[9] Sebastian Lague. Coding adventure: Clouds. https://www.youtube.com/watch?v=4QOcCGI6xOU, 2019.

[10] Jan Novák, Andrew Selle, and Wojciech Jarosz. Residual ratio tracking for estimating attenuation in participating media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 33(6), November 2014.

[11] Grant W. Petty. *A first course in atmospheric radiation*. Sundog Pub., 2006.

[12] Andrew Schneider and Nathan Vos. The real-time volumetric cloudscapes of horizon: Zero dawn. *SIGGRAPH Course note in Advance in Real-Time Rendering in Games*, 2015.

[13] Jaroslav Sloup. A survey of the modelling and rendering of the earth's atmosphere. In *Proceedings of the 18th spring conference on Computer graphics*, pages 141–150, 2002.

[14] Alex Tardif. Adaptive exposure from luminance histograms. https://www.alextardif.com, 2019.

[15] Alexander Wilkie, Petr Vevoda, Thomas Bashford-Rogers, Lukáš Hošek, Tomáš Iser, Monika Kolářová, Tobias Rittig, and Jaroslav Křivánek. A fitted radiance and attenuation model for realistic atmospheres. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021.

[16] Egor Yusov. High performance outdoor light scattering using epipolar sampling. *GPU Pro*, 5:101–126, 2014.

[17] Egor Yusov. High-performance rendering of realistic cumulus clouds using pre-computed lighting. In *High Performance Graphics*, pages 127–136, 2014.

Foveated RTX Ray Tracing in Virtual Reality

Uroš Šmajdek*
Supervised by: Ciril Bohak†

Faculty of Computer and Information Science
University of Ljubljana
Ljubljana / Slovenia

Abstract

In this paper, we present a foveated ray tracing method for enhancing the viewing experience of virtual reality environments featuring dense molecular structures. Achieving immersion in virtual reality requires both high-resolution and high frame-rate rendering combined with the ability to generate realistic images. Our approach combines the power of the Nvidia RTX framework with a dynamic rendering rate that is based on the direction of the user's gaze. Foveated rendering is applied in the ray-generation stage of the ray-tracing pipeline, in order to reduce the ray-tracing rate depending on the distance from the center of the user's gaze on the image. This takes advantage of the way the human eye perceives visual detail, allowing us to spare processing power on areas that are largely imperceptible. The nature of the ray-tracing pipeline also allows us to easily apply any post-processing effects, such as screen-space ambient occlusion and temporal anti-aliasing, in a matter similar to the standard deferred rendering pipeline. We demonstrate our technique by rendering a model of SARS-CoV-2 on Meta Quest Pro in $3776 \times 3904 \times 2$ stereo resolution, where we observed a higher performance in comparison to the non-foveated counterpart of our method.

Keywords: Dense environments, ray tracing, foveated rendering, virtual reality

1 Introduction

Virtual reality (VR) is one of the key factors driving innovation in modern computer graphics. Development of immersive VR technologies has begun to encompass various fields from medicine [5, 10] to entertainment [30] and education [29], and as such, there is an increasing need for interactivity and real-time rendering. Additionally, over the last two decades, the pixel densities of high-quality head-mounted displays, also known as VR headsets, have dramatically increased from $263 \times 480 \times 2$ in 1997 (Forte VFX 3D) to $2448 \times 2448 \times 2$ in 2021 (HTC Vive Pro 2). To achieve the visual quality of a human eye, however, a display would require a resolution of about $32k \times 24k$ [12], making such achievement far beyond the reach of the current hardware and software, but nonetheless ensuring it will continue for decades to come. Not only that but to achieve immersion and limit the potential motion sickness, such rendering would have to exceed 60 updates per second.

One of the recent applications for such rendering is virtual visualization of the molecular structures of biological specimens [21, 1], which aims to immerse the user in the molecular environment, using highly detailed scenes coupled with interactivity. Such software can then be used for science dissemination to the broader public, an increasingly relevant topic since the Coronavirus pandemic. It can also aid in scientific presentations, or be used as a tool to gather additional information from such structures. On the other hand, dense molecular environments provide additional challenges, as scenes contain hundreds of thousands and up to billions of objects, that form closely packed structures, representing proteins, RNA, membranes, etc.

Foveated rendering presents one of the approaches to solving this problem. It describes the dynamic adaptation of rendering based on the user's gaze, specifically by limiting the details of the scene in the peripheral visual area that are largely imperceptible. The concept of foveated displays is not new, and gaze-reactive displays were already used in building flight simulators back in 2001 [25]. Until the recent generation of VR headsets, real-time eye tracking, which is required for its interactive use, was not commercially available, thus the approach was of lesser interest to a wider computer graphics community.

Hardware-accelerated RTX Ray Tracing is a recent breakthrough that promises to revolutionize the rendering scene, traditionally dominated by rasterization techniques. With the help of other Nvidia technologies, such as DLSS, it enables higher visual fidelity without compromising performance. Past research also indicates that the computational complexity of interactive ray tracing increases logarithmically with scene complexity [33], making it a compelling choice for addressing the above-mentioned interactive molecular visualization.

The main contributions of our work are:

- combining hardware-accelerate RTX Ray Tracing with Multi-spatial resolution-based foveated render-

*umajdek@gmail.com
†ciril.bohak@kaust.edu.sa

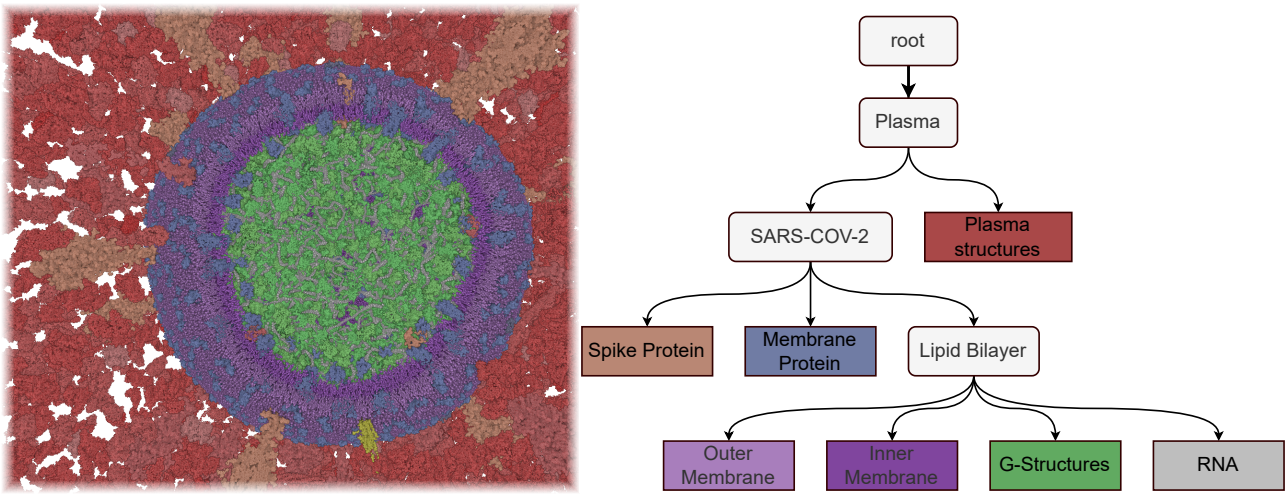


Figure 1: A visualization of dense molecular environment depicting SARS-COV-2 using our rendering technique (left), with individual structure labels and hierarchy (right).

ing,

- developing a pipeline that uses the technique for real-time visualization of dense molecular environments in virtual reality, and
- comparison and evaluation with the non-foveated ray tracing.

In section 2, we present the previous work on molecular visualization and resolution-based foveated ray tracing and differentiate our contributions from it. In section 3, we give a brief overview of the molecular environment we use as a basis. In section 4, we present our approach. The results and evaluation of our method are presented in sections 5 and 6. In section 7, we present the conclusions and give possible extensions as part of further work.

2 Related Work

Molecular Visualization: There are various tools for smaller-scale molecular visualization, such as VMD [11], Mol* [32], or PyMOL [31] which are unsuited for larger data sizes exceeding tens of millions of individual atoms. Over the years, numerous solutions to interactive molecular visualization have been proposed; glyph-based rendering [9], mapping structures onto a mesh geometry [36], using meshlets with probabilistic occlusion culling [13], instancing to repeat the recurring proteins in the scene [22, 6]. In 2015 Le Muzic et al. introduced cellVIEW [26], which depends on the LoD scheme and dynamic sphere primitives injection into primitives to be able to render 250 copies of the HIV virus model in blood plasma (16 billion atoms) at 60 FPS on NVIDIA GTX Titan. All of the above-mentioned techniques use procedural impostors to simplify the geometry and accelerate the rendering. Recently Alharbi et. al. [2] devised a solution that relies on hardware ray tracing instead and is able to render trillions of atoms using on-the-fly data construction and parallel rendering. In this paper, we expand on this paradigm but move the focus from scalability to higher performance, suitable for interactive VR visualization, which requires stereo rendering with a high and steady framerate. To this end, we supplement the hardware ray tracing with DLSS-enhanced foveated rendering but limit ourselves to data scales that directly fit into the GPU memory. To the best of our knowledge, there are no alternative foveated rendering techniques of dense molecular environments.

Multi-spatial resolution-based foveated ray tracing: Koskela et al. [20] proposed a theoretical estimate in 2016 that by integrating foveated rendering with ray tracing, 94% of the rays could potentially be omitted. This is based on the fact that ray tracing inherently supports spatial multi-resolution rendering, as it has the capacity to effortlessly adjust the number of rays emitted from a single pixel. Fujita and Harada [8] first implemented the foveated rendering system based on ray tracing, which relied on sparse sampling and kNN for image reconstruction. The system did not consider the eye sensitivity to contrast and lacked pertinent input from relevant user studies. To address these challenges, Weier et al. [38] combined ray tracing-based foveated rendering with reprojection rendering, using information from the previous frame to reduce the sampling rays for new frames. From there on out, different sampling models were proposed; Molenaar [24] traced rays based on the visual acuity fall-off model, Kim et al. [17] proposed a perceptually efficient pixel sampling method suitable for HMD ray tracing, which combined the Jin et al. [16] selective oversampling technique with the foveated rendering scheme and Koskela et al. [19] traced rays and denoised in Visual-Polar space and subsequently mapped the results to the screen space. Hybrid approaches also emerged with Blackmon et al. [4] using ray tracing to render the foveal region and rasterization to render the peripheral region. To address the lack of consideration for the displayed content, as opposed to the eccentricity, when

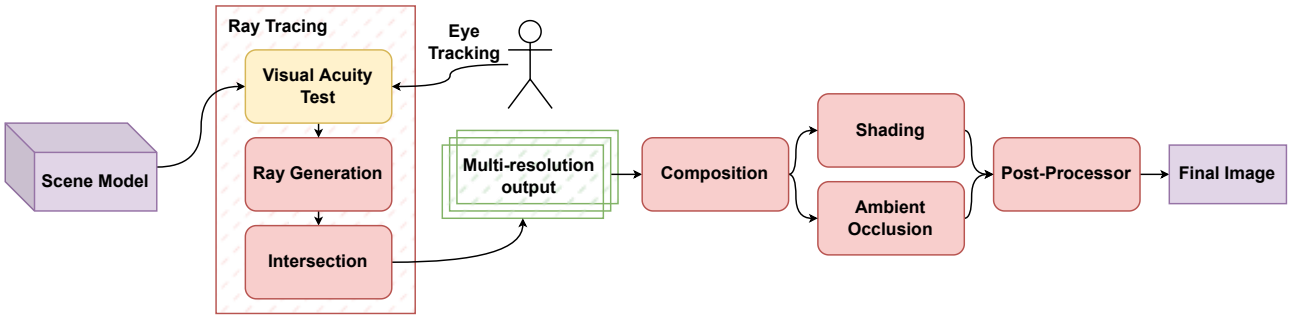


Figure 2: An overview of our technique for the foveated rendering of dense molecular environments in VR.

determining the number of rays emitted, Tursun et al. [35] proposed a new luminance-contrast-aware foveated ray tracing technique. The technique significantly reduced the number of traced rays but required a low-quality image to be generated for each frame, indicating areas with different luminances. Similarly, Yang et al. [39] introduced a trio of simple policies to achieve a variable ray tracing rate. The method integrates the foveate perspective, material, and depth information in order to dynamically reduce the number of tracing rays in certain steps in the ray tracing pipeline. Recently Wang et al. [37] compiled an in-depth overview of state-of-the-art foveated methods for rasterization, ray tracing, and volumetric rendering techniques. In our approach, we implement a low-overhead spatial policy to determine the local number of rays, similar to the one introduced in [39], but we further supplement it using a hardware-accelerated Nvidia Ray Tracing framework [18]. Additionally, we use traditional spatial and temporal anti-aliasing techniques in favor of Nvidia DLSS [23]. While there is no substantial contribution from individual methods, their combination, however, is a novelty. Moreover, some components of the pipeline utilize methods that we were unable to find elsewhere, including a single multi-resolution ray tracing pass and blending between foveal regions.

3 Molecular Environment

As input, we used the atom-based hierarchical 3D models of biological mesoscale organisms, such as viruses. The model may consist of an arbitrary number of instances, each composed of densely packed atoms, with the rest of the space implicitly filled with water molecules. We choose not to render those molecules as they are of less interest to the viewer and would render the scene completely packed. Those spaces are instead treated as empty, allowing the user to focus and travel between important structures found inside a virus. An example of a model's hierarchy can be seen in Figure 1. Additionally, employing such a hierarchical structure allows us to dynamically highlight different levels of structures based on distance and type, further enhancing the user experience.

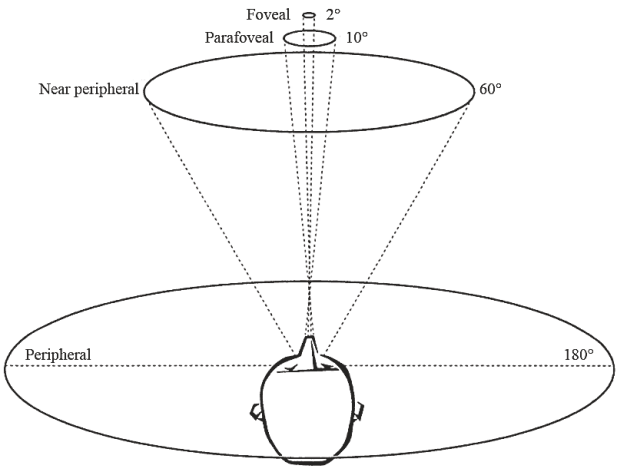


Figure 3: Schematic depiction of foveal/peripheral vision eccentric angles and regions by Ivanvcic et al. [14].

4 Foveated Ray Tracing

In this section, we describe our proposed foveated rendering system. An overview of the rendering pipeline can be seen in Figure 2, and we describe the individual stages in the following subsections. The pipeline is modular, the only static part being the visual acuity test and the hardware-accelerated ray tracing. Other parts can be easily swapped out or new ones added, which is made even easier by its strong resemblance to a classic deferred rendering pipeline.

4.1 Visual Acuity Test and Ray Generation

Visual acuity generally refers to our ability to distinguish the shapes and details of the object we are observing. One of its key properties is the foveal/peripheral vision, which recognizes that human visual acuity is not uniform over the entire visual field. Objects in the periphery are significantly harder to recognize than those in the center of the fovea [34], as illustrated in Figure 3.

We use this foveal/peripheral property of the visual acuity to determine the resolution at which we render different parts of the final image. To achieve this, we perform a vi-

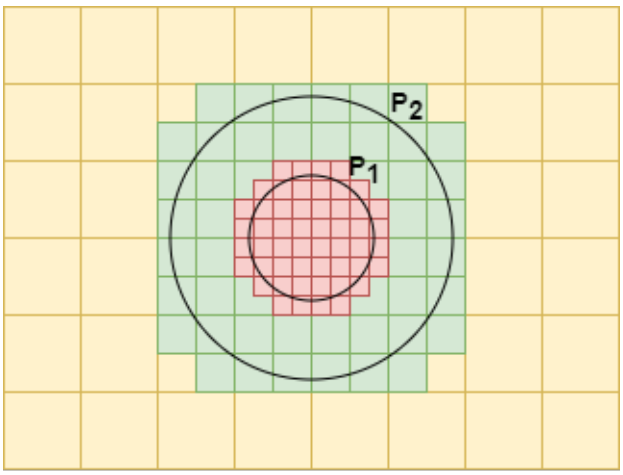


Figure 4: Visual representation of the multi-resolution rendering based on the eccentric angle from the user's gaze. The red squares in the center each cover one pixel, and larger squares denote rendering in lower resolution.

sual acuity test before the ray generation step, where we compare the eccentric angle of the final pixel relative to the center of the user's eye gaze. If we label the rendering resolution at pixel $\mathbf{x}$ as $R_\mathbf{x}$, we can parameterize the visual acuity test as:

$$R_\mathbf{x} = \begin{cases} 1 & : d_\mathbf{x} \leq P_1 \\ 1/4 & : P_1 < d_\mathbf{x} \leq P_2 \\ 1/16 & : d_\mathbf{x} > P_2 \end{cases}$$

where $d_\mathbf{x}$ denotes the eccentric angle at pixel $\mathbf{x}$, P_1 is set between foveal and parafoveal eccentric angle (2-10°), and P_2 is set between parafoveal and near peripheral eccentric angle (10-60°). The chosen resolutions conform to the standard mipmap sizes, normally used for texture rendering, which allows us to store and access the data more efficiently. A visual representation of the final image composition from different resolutions can be seen in Figure 4. Rendering resolution directly determines the number of rays generated in the ray-generation step of the ray-tracing pipeline.

4.2 Ray Tracing Pipeline

Our ray tracing pipeline heavily relies upon the Nvidia RTX ray tracing extension for Vulkan [18], which is the continuation of Nvidia OptiX [28], a general-purpose SDK for accelerating ray casting applications. Using the framework, we pack the atoms that build our model into *Accelerated Structures*, a specialized high-performance spatial structure built for ray tracing. The framework then takes care of optimizing the ray collision detection using bounding volume hierarchy traversal while we still retain full control of the ray generation and after-collision events. To be able to use deferred rendering techniques later, we store not only the colors of hit objects but also the hit location, the normal of the hit surface, and the hit object's numerical identifier.

To further reduce the overhead in the rendering pipeline, we only perform the ray tracing procedure once instead of separately for each resolution we render. We achieve this by inherently rendering at full resolution and then discarding the unneeded rays in the periphery.

4.3 Composition

During composition, we combine the 2D multi-resolution output of the ray tracing procedure. Since rendering at different resolutions results in spatial discontinuities between the rendering regions, we blend different resolutions at their border, as opposed to simply taking the highest resolution present at that location, which results in smoother transitions. To that end, we employ an alpha-blending technique to reduce the computational overhead of the procedure.

4.4 Shading

As our model does not depict an environment in which realistic lighting plays a role, we can simplify the illumination in ray tracing algorithm to focus solely on local illumination with Phong shading. This optimization not only improves performance by reducing the number of traced rays but also eliminates the probabilistic computations typically found in ray tracing algorithms, resulting in immediate convergence of the final result.

Another challenge that the molecular environment presents is the discrepancy between the macro-level and micro-level perspectives. At the macro level, we are interested in the various structures that make up our model, such as membranes or RNA, while at the micro level, we focus on individual atoms that comprise those structures. To seamlessly bridge these two levels, we modulate the color of each atom by interpolating between its inherent color, c_a, and the color of its corresponding structure, c_s, based on the atom's distance from the camera, d:

$$c = \begin{cases} (1 - \frac{d}{D_{max}}) \cdot c_a + \frac{d}{D_{max}} \cdot c_s & : d < D_{max} \\ c_s & : d \geq D_{max} \end{cases}.$$

Any atom located beyond a pre-determined distance, D_{max}, is assigned the color of its structure, making it easier to differentiate between the structures from a distance. The colors are user-defined.

4.5 Screen-Space Ambient Occlusion

As we are dealing with primarily mono-colored, locally illuminated base shapes of the same size, an ambient occlusion technique is needed to help the user distinguish between them and also improve depth perception. Screen Space Ambient Occlusion (SSAO) is well suited for this

task, as its complexity does not scale with the number of objects in the scene and is generally better in dense environments due to the improved locality of GPU memory accesses. Since our model only contains spheres, we use the hemisphere sampling variant of the technique, first described by Fillion and McNaughton [7].

4.6 Post processing

In the post-processing stage, we draw atom contours in order to ease the distinction between neighboring objects. This is especially important when working in dense molecular environments, as the high number of small objects is inherently difficult to distinguish. To determine whether to draw a contour at a pixel or not, we use previously computed object identifiers to check if all neighboring pixels belong to the same object. If they do not we draw a semi-transparent contour and blend it with the pixel's color.

5 Results

We evaluated the proposed technique on the SARS-CoV-2 model [27], which consists of 74,724,996 atoms. Experiments were rendered to a Desktop application window with a rendering resolution of 3840×2160, and the Oculus Pro VR headset with a stereo rendering resolution of $3776 \times 3904 \times 2$. All rendering was done using a Nvidia GeForce RTX 4090 graphics card with 24 GB of VRAM, an AMD Ryzen Threadripper PRO 3995WX 64-core processor, and 256 GB of RAM.

To discern the impact of the inclusion of foveated rendering, we measured the number of rendered frames per second (FPS). We performed 4 different tests for different combinations of the inclusion of foveated rendering and SSAO. During each experiment, we measured the average framerate over a duration of 30 seconds. To better represent a realistic usage scenario we slowly moved the camera around for the entire duration. As the framerate is restricted to 72 FPS on Meta Quest Pro, we chose the scene that closely matches this framerate to minimize the impact on final results. The results can be seen in Table 1. The addition of foveated rendering almost doubled the performance both on Desktop and in VR as long as SSAO is not used. Enabling the SSAO resulted in a massive performance drop, especially when foveated rendering was also enabled. For qualitative evaluation, we present renderings of all four states in Figure 5. The two left segments show the rendering without foveation and the two right segments with foveation. The top two segments show the result with SSAO disabled, and the bottom two segments show the result with SSAO enabled.

Display	SSAO	Foveated	FPS	rFPS
Desktop	No	No	288.3	-
		Yes	553.0	92%
	Yes	No	126.6	-
		Yes	159.0	26%
Oculus Pro	No	No	39.9	-
		Yes	71.8	80%
	Yes	No	24.6	-
		Yes	36.1	47%

Table 1: Performance evaluation of the proposed foveated rendering pipeline for dense molecular data. For reference, we rendered the same model without foveated rendering and also repeated both experiments without the inclusion of SSAO. The maximum refresh rate on Oculus Pro is 72 Hz, which represents the maximum measured framerate on the system. rFPS is defined as relative improvement using foveation on the specific display and SSAO setting.

6 Discussion

Our tests have revealed that foveated rendering by itself massively improves the performance (see Table 1). The results are less significant when the technique is used simultaneously with SSAO. We believe this is due to the lower resolution producing higher spatial discontinuities between neighboring pixels in low-resolution regions, resulting in a higher number of GPU cache misses. By itself, SSAO also massively reduces performance, as observed by the 38%-56% framerate drop when applied without foveation. This likely signifies that SSAO is not a suitable candidate for rendering dense molecular data in high resolutions.

From Figure 5, we conclude that the visual differences between the foveated and non-foveated results are rather minimal, especially close to the gaze direction. Internal user testing has shown that during movement, the aliasing in the distant object in the peripheral regions can be distracting. To address this issue, a performance-friendly solution would be to apply a blur filter to both peripheral regions in the post-processing stage. On the other hand, the figure also showcases the importance of ambient occlusion, without which it becomes difficult to distinguish between the objects, but we do not believe it justifies the performance loss in terms of user experience.

7 Conclusions

In this paper, we present a multi-spatial resolution-based foveated ray tracing method for the visualization of dense molecular environments in virtual reality. We evaluate the technique, both with and without the addition of ambient occlusion, showing a moderate to large increase in performance, depending on whether SSAO was used or not. We

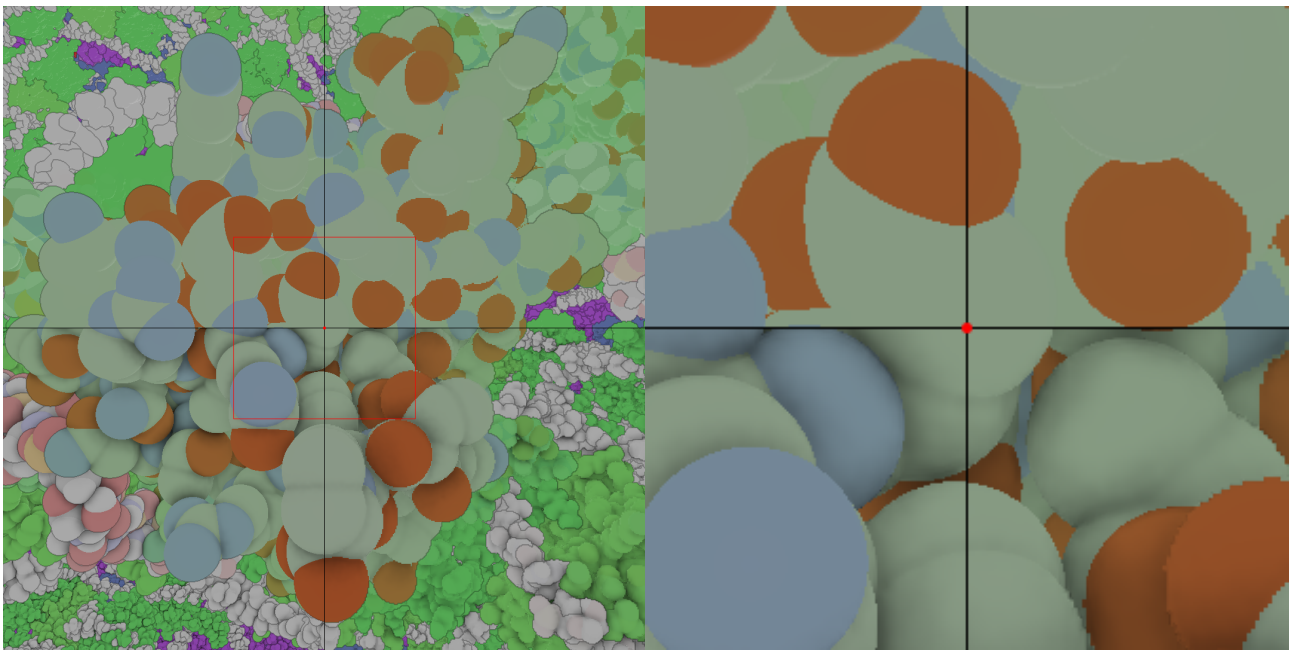


Figure 5: Individual image groups show a comparison between the non-foveated (left) and foveated (right) rendering and between no-SSAO (top) and with SSAO (bottom). The gaze direction is marked with a red dot. The right image group shows the magnification of the left one, denoted by the red square.

also showcase the need for ambient occlusion in terms of being able to tell the objects apart in a dense molecular environment.

As a future extension of this work more performant ambient occlusion techniques could be employed [3, 15], as the usage of SSAO resulted in a massive performance loss. Additionally, the method could be expanded by also considering the context of the scene during the visual acuity test, and thus being able to modulate the rendering resolution of different structures depending on other criteria, such as their depth and importance to the user.

References

[1] Ruwayda Alharbi, Ondrej Strnad, Laura Luidolt, Manuela Waldner, Ciril Bohak, David Kouril, Tobias Klein, Eduard Gröller, and Ivan Viola. Nanotilus: Generator of immersive guided-tours in crowded 3d environments. *IEEE Transactions on Visualization and Computer Graphics*, PP, 12 2021.

[2] Ruwayda Alharbi, Ondřej Strnad, Tobias Klein, and Ivan Viola. Nanomatrix: Scalable construction of crowded biological environments, 2022.

[3] Louis Bavoil. Deinterleaved Texturing for Cache-Efficient Interleaved Sampling. Technical report, NVIDIA Corporation, 2010.

[4] Steven Blackmon, Luke T. Peterson, Cuneyt Ozdas, and Steven J. Clohset. Foveated rendering, US Patent App. 15/372,589, 2017.

[5] Cléber Gimenez Corrêa, Fátima L. S. Nunes, Adriano Bezerra, and Paulo M. Carvalho. Evaluation of vr medical training applications under the focus of professionals of the health area. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC '09, page 821–825, New York, NY, USA, 2009. Association for Computing Machinery.

[6] Martin Falk, Michael Krone, and Thomas Ertl. Atomistic Visualization of Mesoscopic Whole-Cell Simulations Using Ray-Casted Instancing. *Computer Graphics Forum*, 32, 2013.

[7] Dominic Filion and Rob McNaughton. Effects & techniques. In *ACM SIGGRAPH 2008 Games*, SIGGRAPH '08, page 133–164. Association for Computing Machinery, New York, NY, USA, 2008.

[8] Masahiro Fujita and Takahiro Harada. Foveated real-time ray tracing for virtual reality headset. *Light Transport Entertainment Research*, 2014.

[9] Sebastian Grottel, Michael Krone, Christoph Müller, Guido Reina, and Thomas Ertl. MegaMol—A Prototyping Framework for Particle-Based Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 21(2):201–214, 2015.

[10] Min-Chai Hsieh and Jia-Jin Lee. Preliminary study of vr and ar applications in medical and healthcare education. *Journal of Nursing and Health*, 3, 2018.

[11] William Humphrey, Andrew Dalke, and Klaus Schulten. VMD: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33–38, 1996.

[12] Warren Hunt. Virtual reality: The next great graphics revolution. *Keynote Talk HPG*, pages 1–2, 2015.

[13] Mohamed Ibrahim, Peter Rautek, Guido Reina, Marco Agus, and Markus Hadwiger. Probabilistic Occlusion Culling using Confidence Maps for High-Quality Rendering of Large Particle Data. *IEEE Transactions on Visualization and Computer Graphics (Proceedings IEEE VIS 2021)*, 28(1):573–582, 2022.

[14] Snježana Ivančić Valenko, Vladimir Cviljušac, Sanja Zlatić, and Damir Modrić. The impact of physical parameters on the perception of the moving elements in peripheral part of the screen. *Tehnički vjesnik*, 26(5):1444–1450, 2019.

[15] Jorge Jiménez, Xianchun Wu, Angelo Pesce, and Adrian Jarabo. Practical real-time strategies for accurate indirect occlusion. *SIGGRAPH 2016 Courses: Physically Based Shading in Theory and Practice*, 2016.

[16] Bongjun Jin, Insung Ihm, Byungjoon Chang, Chanmin Park, Wonjong Lee, and Seokyoon Jung. Selective and adaptive supersampling for real-time ray tracing. In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, page 117–125, New York, NY, USA, 2009. Association for Computing Machinery.

[17] Youngwook Kim, Yunmin Ko, and Insung Ihm. Selective foveated ray tracing for head-mounted displays. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 413–421, 2021.

[18] Daniel Koch. Vulkan ray tracing final specification release. https://www.khronos.org/blog/vulkan-ray-tracing-final-specification-release, 11 2020. Accessed: 2023-02-11.

[19] Matias Koskela, Atro Lotvonen, Markku Mäkitalo, Petrus Kivi, Timo Viitanen, and Pekka Jääskeläinen. Foveated Real-Time Path Tracing in Visual-Polar Space. In Tamy Boubekeur and Pradeep Sen, editors, *Eurographics Symposium on Rendering - DL-only and Industry Track*. The Eurographics Association, 2019.

[20] Matias Koskela, Timo Viitanen, Pekka Jääskeläinen, and Jarmo Takala. Foveated path tracing. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Fatih Porikli, Sandra Skaff, Alireza Entezari, Jianyuan Min, Daisuke Iwai, Amela Sadagic, Carlos Scheidegger, and Tobias Isenberg, editors, *Advances in Visual Computing*, pages 723–732, Cham, 2016. Springer International Publishing.

[21] David Kouril, Ondrej Strnad, Peter Mindek, Sarkis Halladjian, Tobias Isenberg, Eduard Gröller, and Ivan Viola. Molecumentary: Adaptable narrated documentaries using molecular visualization. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 11 2021.

[22] N. Lindow, D. Baum, and H.-C. Hege. Interactive Rendering of Materials and Biological Structures on Atomic and Nanoscopic Scale. *Comput. Graph. Forum*, 31(3pt4):1325–1334, 06 2012.

[23] Edward Liu. Dlss 2.0 - image reconstruction for real-time rendering with deep learning. https://developer.nvidia.com/gtc/2020/video/s22698-vid, 2020. Accessed: 2023-02-11.

[24] Erik N Molenaar. Towards real-time ray tracing through foveated rendering. Master's thesis, University of Utrecht, 2018.

[25] Hunter A. Murphy and Andrew T. Duchowski. Gaze-contingent level of detail rendering. In *Eurographics*, 2001.

[26] Mathieu Le Muzic, Ludovic Autin, Július Parulek, and Ivan Viola. cellVIEW: a Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets. *Eurographics Workshop on Visual Computing for Biomedicine*, 2015:61–70, 2015.

[27] Ngan Nguyen, Ondřej Strnad, Tobias Klein, Deng Luo, Ruwayda Alharbi, Peter Wonka, Martina Maritan, Peter Mindek, Ludovic Autin, David S. Goodsell, and Ivan Viola. Modeling in the time of covid-19: Statistical and rule-based mesoscale models. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):722–732, 2021.

[28] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. Optix: A general purpose ray tracing engine. *ACM Trans. Graph.*, 29(4), 06 2010.

[29] Maria Paola Puggioni, Emanuele Frontoni, Marina Paolanti, Roberto Pierdicca, Eva Savina Malinverni, and Michele Sasso. A content creation tool for ar/vr applications in education: The scoolar framework. In *Augmented Reality, Virtual Reality, and Computer Graphics: 7th International Conference, AVR 2020, Lecce, Italy, September 7–10, 2020, Proceedings, Part II*, page 205–219, Berlin, Heidelberg, 2020. Springer-Verlag.

[30] Cedriss Saint-Louis and Abdelwahab Hamam. Survey of haptic technology and entertainment applications. In *SoutheastCon 2021*, pages 01–07. IEEE, 03 2021.

[31] Schrödinger, LLC. The PyMOL Molecular Graphics System, Version 1.8. unpublished, 11 2015.

[32] David Sehnal, Sebastian Bittrich, Mandar Deshpande, Radka Svobodová, Karel Berka, Václav Bazgier, Sameer Velankar, Stephen K Burley, Jaroslav Koča, and Alexander S Rose. Mol* Viewer: modern web app for 3D visualization and analysis of large biomolecular structures. *Nucleic Acids Research*, 49(W1):W431–W437, 05 2021.

[33] Philipp Slusallek and I Wald. State of the art in interactive ray tracing. *STAR, EUROGRAPHICS 2001*, pages 21–42, 2001.

[34] Hans Strasburger, Ingo Rentschler, and Martin Jüttner. Peripheral vision and pattern recognition: A review. *Journal of vision*, 11:13, 05 2011.

[35] Okan Tarhan Tursun, Elena Arabadzhiyska-Koleva, Marek Wernikowski, Radosław Mantiuk, Hans-Peter Seidel, Karol Myszkowski, and Piotr Didyk. Luminance-contrast-aware foveated rendering. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[36] Thomas Waltemate, Björn Sommer, and Mario Botsch. Membrane Mapping: Combining Mesoscopic and Molecular Cell Visualization. In *Eurographics Workshop on Visual Computing for Biology and Medicine*, 2014.

[37] Lili Wang, Xuehuai Shi, and Yi Liu. Foveated rendering: A state-of-the-art survey. *Computational Visual Media*, 9(2):195–228, 2023.

[38] Martin Weier, Thorsten Roth, Ernst Kruijff, André Hinkenjann, Arsène Pérard-Gayot, Philipp Slusallek, and Yongmin Li. Foveated real-time ray tracing for head-mounted displays. *Computer Graphics Forum*, 35:289–298, 10 2016.

[39] Jinyuan Yang, Xiaoli Li, and Abraham G. Campbell. Variable rate ray tracing for virtual reality. In *SIGGRAPH Asia 2020 Posters*, SA '20, New York, NY, USA, 2020. Association for Computing Machinery.

Optimization

Controlling 2D Laplacian Eigenfluids

Barnabás Börcsök*
Supervised by: László Szécsi †

Department of Control Engineering and Information Technology
Budapest University of Technology and Economics
Műegyetem rkp. 3., H-1111 Budapest, Hungary

Abstract

Understanding and modeling our environment is a great and important challenge, spanning many disciplines from weather and climate forecast through vehicle design to computer graphics. Physical systems are usually described by Partial Differential Equations (PDEs), which we can approximate using established numerical techniques. Next to predicting outcomes, planning interactions to control physical systems is also a long-standing problem.

In our work, we investigate the use of Laplacian eigenfunctions to model and control fluid flow. We make use of an explicit description of our simulation domain to derive gradients of the physical simulation, enabling neural network agents to learn to control the physical process to achieve desired outcomes.

Keywords: Computer Graphics, Modeling and Simulation, Fluid Simulation, Neural Networks

1 Introduction

Data helps us model and understand our world more truthfully. Enabling the processing of the ever-increasing volume of data, and running more precise simulations necessitate continuous engineering efforts.

Positioned at the crossroads of physical simulation and deep learning techniques, our work is inspired by current advances in physics-based deep learning. We investigate the general problem of controlling simulation parameters to achieve target outcomes. More concretely, many real-world applications require us to optimize for some parameters of a physics-based problem. Although such inverse problems have been around for quite some time in engineering applications, recent work showed remarkable results utilizing physical gradients to solve such problems. Examples include finding the best shape to minimize airfoil drag [2] and finding cloth simulation parameters for yielding desired simulation outcomes [9].

We present a novel method for controlling fluid simulations. We show that implementing a differentiable reduced-order physics simulation yields gradients that allow us to achieve speed-ups in the optimization process characteristic of reduced-order models, resulting in fast convergence times. We investigate different possibilities for control. After directly optimizing for parameters already present in the simulation technique (such as initial velocity and external force), we build up to adding a neural network (NN) for predicting control forces, and optimizing for its internal parameters. When optimizing for advection dynamics, we achieve significant speed-ups by utilizing point-wise samples: we keep the advection dynamics in the reduced-dimensional space, instead of reconstructing the velocity field on an $N \times N$ grid in each time step. Thus, we essentially decouple the optimization from the grid resolution, which can be reconstructed in any desired resolution without increasing the complexity of the optimization.

The source code of this project is available

*bborcsok@iit.bme.hu
†szecsi@iit.bme.hu
This work was supported by OTKA K-124124, and the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory.

at https://github.com/bobarna/controlling-2d-laplacian-eigenfluids.

2 Previous Work

2.1 Fluid Simulation

Most simulation methods are based on either an Eulerian (i.e. grid-based), or Lagrangian (i.e. particle-based) representation of the fluid. For advecting marker density in our fluid, as well as a comparative "baseline" simulation, we use Eulerian simulation techniques, mostly as described by Stam [13]. For an overview of fluid simulation techniques in computer graphics, see Bridson [1].

Reduced Order Modeling of Fluids. Dimension reduction-based techniques have been applied to fluid simulation in multiple previous works. Wiewel et al. [17] demonstrated that functions of an evolving physics system can be predicted within the latent space of neural networks (NNs). Their efficient encoder-decoder architecture predicted pressure fields, yielding two orders of magnitudes faster simulation times than a traditional pressure solver. Recently, Wiewel et al. [16] predicted the evolution of fluid flow via training a convolutional neural network (CNN) for spatial compression, with another network predicting the temporal evolution in this compressed subspace. The main novelty of Wiewel et al. [16] was the subdivision of the learned latent space, allowing interpretability, as well as external control over quantities such as velocity and density.

Eigenfluids. Instead of *learning* a reduced-order representation, another option is to analytically derive the dimension reduction and its time evolution. De Witt et al. [4] introduced a computationally efficient fluid simulation technique to the computer graphics community. Rather than using an Eulerian grid or Lagrangian particles, they represent fluid fields using a basis of global functions defined over the entire simulation domain. The fluid velocity is reconstructed as a linear combination of these bases.

They propose the use of Laplacian eigenfunctions as these global functions. Following their method, the fluid simulation becomes a matter of evolving basis coefficients in the space spanned by these eigenfunctions, resulting in a speed-up characteristic of reduced-order methods.

Following up on the work of De Witt et al. [4], multiple papers proposed improvements to the use of Laplacian eigenfunctions for the simulation of incompressible fluid flow. Liu et al. [10] extended the technique to handle arbitrarily-shaped domains. Jones et al. [7] used Discrete Cosine Transform (DCT) on the eigenfunctions for compression. Cui et al. [3] improved scalability of the technique, and modified the method to handle different types of boundary conditions. Cui et al. [3] refer to the simulation technique as *eigenfluids*, which we also adhere to in the following.

2.2 Differentiable Solvers

Differentiable solvers have shown tremendous success lately for optimization problems, including training neural network models [5, 6, 11]. Holl et al. [5] address grid-based solvers. They put forth Φ_{Flow}, an open-source simulation toolkit built for optimization and machine learning applications, written mostly in Python. After trying out multiple recent frameworks aimed at differentiable simulations [11, 6], we implement all of our experiments using Φ_{Flow} [5].

Physics-based Deep Learning. Despite being a topic of research for a long time [12], the interest in neural network algorithms is a relatively new phenomenon. This is especially true for the use of learning-based methods in physical and numerical simulations, which is a rapidly developing area of current research [2, 9]. Integrating physical solvers in such methods have been shown to outperform previously used learning approaches [15]. Drawing on a wide breadth of current research, Thuerey et al. [14] give an overview of deep learning methods in the context of physical simulations.

3 Background

In this section, we introduce the techniques, theory and notation underlying our methods for controlling eigenfluids in Section 4.

3.1 Fluid Simulation

The dynamics of fluids are governed by the Navier-Stokes Equations:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \quad (1)$$

where $\mathbf{u}$ is the velocity of the fluid, ρ is the density, p is the scalar pressure field, ν is the viscosity constant and $\mathbf{f}$ denotes external forces. For incompressible fluids, the divergence-freeness also has to hold, i.e. $\nabla \cdot \mathbf{u} = 0$. There are multiple established ways to simulate fluids, the most widespread being Eulerian (i.e. grid-based) and Lagrangian (i.e. particle-based) methods. We use established Eulerian methods [1] for advecting and comparison purposes.

The Laplacian Eigenfunction Method. A velocity field $\mathbf{u}(\mathbf{x})$ can be expressed via the linear combination of N global functions:

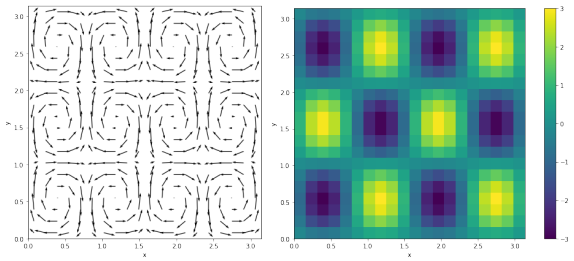
$$\mathbf{u}(\mathbf{x}) = \sum_{k=0}^{N} w_k \Phi_k(\mathbf{x}), \quad (2)$$

where the elements of $\mathbf{w} = [w_0, \ldots, w_N]$ are called *basis coefficients* and Φ_k are *basis functions*. In the following, we use $\mathbf{u} = \mathscr{R}\mathbf{w}$ to notate a velocity field $\mathbf{u}$ reconstructed from $\mathbf{w}$. De Witt et al. [4] propose the use of eigenfunctions of the vector Laplacian operator $\Delta = \nabla^2$. If we further require our basis fields Φ_k to be divergence-free and to satisfy a free-slip boundary condition, then our basis functions are fully characterized by

$$\nabla^2 \Phi_k = \lambda_k \Phi_k \quad (3)$$
$$\nabla \cdot \Phi_k = 0 \quad (4)$$
$$\Phi_k \cdot \mathbf{n} = 0 \text{ at } \partial D, \quad (5)$$

where $\mathbf{n}$ is the normal vector at the boundary ∂D of our domain D. On some domains, closed-form expressions exist. Denoting the two scalar components in the x and y directions $\Phi_{\mathbf{k}} = (\Phi_{\mathbf{k},x}, \Phi_{\mathbf{k},y})$, on the two dimensional $D \in [0,\pi] \times [0,\pi]$ square domain, $\Phi_{\mathbf{k}}$ can be written as

$$\Phi_{\mathbf{k},x}(x,y) = \eta_{\mathbf{k}}\big(k_2 \sin(k_1 x)\cos(k_2 y)\big) \quad (6)$$
$$\Phi_{\mathbf{k},y}(x,y) = -\eta_{\mathbf{k}}\big(-k_1 \cos(k_1 x)\sin(k_2 y)\big),$$



(a) Velocity field $\Phi_{(4,3)}$. (b) Curl field $\nabla \times \Phi_{(4,3)} = \phi_{(4,3)}$.

Figure 1: Visualizing $\Phi_{(4,3)}$ sampled on a 20×20 grid in our simulation domain $D = [0,\pi] \times [0,\pi]$.

where $\mathbf{k} = (k_1, k_2) \in \mathbb{Z}^2$ is the *vector wave number*, $\lambda_{\mathbf{k}} = -(k_1^2 + k_2^2)$ is the *eigenvalue*, and $\eta_{\mathbf{k}} = (-\lambda_{\mathbf{k}})^{-1}$ is a normalization parameter. As an example, $\Phi_{(4,3)}(x,y)$ is visualized in Figure 1.

Higher wave lengths corresponding to smaller scales of vorticity has a very literal meaning in our simulation. As we choose to truncate the spectrum of Φ_k at some number N, the error we incur is well defined: we lose the ability to simulate vortices smaller than a given scale. Also, as we will see later on, this correspondence to spatial scales of vorticity lets us control the viscosity (i.e. energy decay) in relation to the scales of vortices by modifying the base coefficients. By setting the magnitude of each basis coefficient to decay with a time constant equal to the eigenvalue, we get the physically correct behavior that small vortices dissipate faster than large vortices.

For the simulation technique, we further require the vorticity field $\omega = \nabla \times \mathbf{u}$ and a set of vorticity basis functions $\phi = \nabla \times \Phi$. Taking the curl gives us the vorticity basis fields:

$$\phi_{\mathbf{k}} = \nabla \times \Phi_{\mathbf{k}} = \begin{pmatrix} 0 \\ 0 \\ \sin(k_1 x)\sin(k_2 y) \end{pmatrix}. \quad (7)$$

As the velocity field $\mathbf{u}$ and vorticity field ω are orthogonal, the vorticity basis functions $\phi_{\mathbf{k}}$ have

Note: We use the wave length vector $\mathbf{k} = (k_1, k_2)$, as well as a single (non-vector) k for indexing over all of the basis fields – a slight, but very useful abuse of notation. This stems from the fact that a suitable mapping from vector wave length (k_1, k_2) to positive integers is necessary in an implementation.

only a normal component at the boundary and satisfy

$$\nabla^2 \phi_{\mathbf{k}} = \lambda_{\mathbf{k}} \phi_{\mathbf{k}} \tag{8}$$

$$\phi_{\mathbf{k}} \times \mathbf{n} = 0 \text{ at } \partial D. \tag{9}$$

From these properties, De Witt et al. [4] show a velocity-vorticity duality, letting us use the same $\mathbf{w}$ vector to reconstruct $\mathbf{u}$ and ω from Φ_k and ϕ_k, respectively. Furthermore, as both Φ_k and ϕ_k are divergence-free by construction (i.e. $\nabla \cdot \phi_{\mathbf{k}} = 0$), there is no need for a pressure projection in each time step, otherwise often present in fluid simulation techniques.

They also show that on Laplacian eigenfunctions, the inverse operator curl^{-1} takes the simple form $\Phi_k = -\lambda_k^{-1} \cdot \text{curl}(\phi_k)$, making the reconstruction of $\mathbf{u}$ from ω efficient.

Dynamics. The vorticity formulation of the Navier-Stokes Equations (Equation 1) is

$$\dot{\omega} = \text{Adv}(\mathbf{u}, \omega) + \nu \nabla^2 \omega + \nabla \times \mathbf{f}, \tag{10}$$

where $\omega = \nabla \times \mathbf{u}$ and $\mathbf{f}$ denotes external forces. $\text{Adv}(\mathbf{u}, \omega)$ represents the advection term, defined as $\text{Adv}(\mathbf{u}, \omega) \coloneqq \text{curl}(\omega \times \mathbf{u})$.

De Witt et al. [4] perform projection to a Laplacian eigenfunction basis by substituting the expansions $\omega = \sum_i w_i \phi_i$, $\mathbf{u} = \sum_j w_j \Phi_j$ and $\dot{\omega} = \sum_k \dot{w}_k \phi_k$ into Equation 10. With rearranging the terms through linearity of operators, they get

$$\sum_k^N \dot{w} \phi_k = \sum_i^N \sum_j^N w_i w_j \text{Adv}(\Phi_i, \phi_j) + \nu \sum_i^N \nabla^2 w_i \phi_i + \nabla \times \mathbf{f}. \tag{11}$$

As the $\text{Adv}(\Phi_i, \phi_j)$ terms are constant, we precompute them, and the results are stored in the ϕ_k basis, making up the elements of the $\mathbf{C}_k$ matrices for each basis field, each with $N \times N$ values:

$$\mathbf{C}_k[h,i] = \left(\nabla \times (\phi_h \times \Phi_i)\right) \cdot \phi_k. \tag{12}$$

Thus, the evolution of a fluid's velocity as the time derivative of the kth element of the coefficient vector $\mathrm{d}\mathbf{w}/\mathrm{d}t = \dot{\mathbf{w}}$ can be written as

$$\dot{w}_k = \mathbf{w}^T \mathbf{C}_k \mathbf{w} + \nu \lambda_k w_k + f_{w_k}, \tag{13}$$

where the advection term $\nu \lambda_k w_k$ is a point-wise exponential decay (derived via Equation 8) and f_{w_k} represents the external force $\mathbf{f}$ projected to the given basis. Any standard numerical technique can be used to integrate Equation 13 forward in time. However, De Witt et al. [4] describe a preferred technique that, in order to preserve kinetic energy, renormalizes the energy of the fluid simulation after each integration step. They show that due to the orthogonality of the basis functions, the total kinetic energy can be calculated as a sum of squared coefficients.

Algorithm 1 Eigenfluids: stepping $\mathbf{w}$ by Δt

```
e_1 = Σ_i^N w[i]^2                    ▷ store kinetic energy
for k = 1...N do
    ẇ[k] = w^T C_k w                  ▷ calculating advection
end for
w += ẇ Δt                             ▷ explicit Euler integration step
e_2 = Σ_i^N w[i]^2                    ▷ energy after time step
w *= sqrt(e_1 / e_2)                  ▷ renormalize energy
for k = 1...N do
    w[k] *= e^{λ_k Δt}                ▷ dissipate energy (viscosity)
    w[k] += f[k]                      ▷ add external forces
end for.
```

3.2 Neural Networks

The goal of neural networks (NNs) is to approximate an unknown function

$$\mathbf{f}^*(\mathbf{x}) = \mathbf{y}^*,$$

where $\mathbf{y}^*$ denotes *ground truth* solutions. $\mathbf{f}^*(\mathbf{x})$ is approximated by a neural network (NN) representation

$$\mathbf{f}(\mathbf{x}, \theta) = \mathbf{y},$$

where θ is a vector of *weights*, influencing the output of the NN. In the case of a fully-connected NN, we can write its i^{th} layer as

$$\mathbf{o}^i = \sigma\left(\mathbf{W}_i \mathbf{o}^{i-1} + b_i\right), \tag{14}$$

where $\mathbf{o}^i$ is the output of the i^{th} layer, and σ is a non-linear activation function, such as the rectified

linear unit (ReLU) function, and $\mathbf{W}_i$ and b_i are the weight matrix and the bias of layer i, respectively. We call $\mathbf{W}_i$ and b_i the parameters of the NN, and collect their values from all layers in θ.

Deep learning (DL) is about stacking multiple layers after each other, and finding θ parameters such that the outputs $\mathbf{y}$ of the NN match the $\mathbf{y}^*$ outputs of the original function $\mathbf{f}^*$ as closely as possible, as measured by some scalar-valued loss function $L\big(\mathbf{f}(\mathbf{x},\theta),\mathbf{y}^*\big)$. Using a mean square error for our loss function, we can write the optimization problem as:

$$\arg\min_{\theta} \|\mathbf{f}(\mathbf{x},\theta)-\mathbf{y}^*\|_2^2. \tag{15}$$

The chain rule gives us the derivates of composite functions, letting us calculate the gradients of the loss function L with respect to the weights θ (i.e. $\partial L/\partial\theta$). In Section 4.2.4, we optimize, i.e. *train* our NNs with Adam [8], a stochastic gradient descent (SGD) optimizer.

For the purposes of Section 4, before introducing NNs into the optimization loop, it is helpful to think of the derivative as *function sensitivity*, denoting how a small change in an input variable changes the output of the function. As introduced in Equation 15, for finding the optimal θ parameters of a NN, this is exactly what we need: how to tweak θ to reduce the output of a loss function. More generally, we can optimize w.r.t. any parameter of a function in the same manner, such as the initial velocity field of a fluid simulation. In Section 4, we will do exactly this.

4 Controlling Eigenfluids

In the following, we showcase different optimization scenarios of increasing complexity, investigating different aspects of controlling eigenfluids via differentiable physics (DP) gradients. Making use of the explicit closed-form description of a velocity field (Equation 6) to derive gradients used for optimization, we achieve a speed increase characteristic of reduced-order techniques.

4.1 Matching Velocities

To verify the feasibility of our technique before moving on to more involved setups, our most straightforward optimization scenario is finding an initial basis coefficient vector $\mathbf{w}_0 \in \mathbb{R}^N$ for an eigenfluid simulation using $N = 16$ basis fields, such that when simulated for t time steps, the reconstructed $\mathscr{R}\mathbf{w}_t = \mathbf{u}_t$ velocity field will match some precalculated $\mathbf{u}^* : [0,\pi]\times[0,\pi] \to \mathbb{R}^2$ target velocity field:

$$L(\mathbf{w}) = \big\|\mathscr{R}\mathscr{P}^t(\mathbf{w}) - \mathbf{u}^*\big\|_2^2, \tag{16}$$

where $\mathscr{P}^t(\mathbf{w}) = \mathscr{P}\circ\mathscr{P}\cdots\circ\mathscr{P}(\mathbf{w})$ is a composite function: the physical simulation of base coefficients $\mathbf{w}$, t times.

For the optimization, we initialize a $\mathbf{w}_{\text{init}} \in \mathbb{R}^N$ vector with random numbers (from a normal distribution), and run the eigenfluid simulation for t time steps, after which we measure the error as given by loss function 16. Relying on backpropagation [1] to derive the necessary gradients, we use the gradient descent (GD) optimization method to iteratively find a vector $\mathbf{w}_{optim}$, yielding a low scalar loss:

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda\nabla L^T(\mathbf{w}). \tag{17}$$

To be able to make some further evaluation of the end results possible, we step an eigenfluid solver for time t to precalculate the target $\mathbf{u}^*$ velocity field, sampled on a 32×32 grid. We denote the initial base coefficient vector of this reference simulation $\mathbf{w}^*$, but keep in mind that the optimization has absolutely zero knowledge of this value, as it sees only the $32\times32\times2$ velocity values of $\mathbf{u}^* = \mathscr{R}\mathbf{w}^*$ at time t. Also, these values could have been precalculated from any other kind of fluid simulation as well, or even just initialized randomly. Deriving $\mathbf{u}^*$ as the result of an eigenfluid simulation has the added benefit of exposing to us a solution $\mathbf{w}^*$ that we can use to compare with the solution of the optimizer.

Results. We test this setup on two scenarios, with differing the number of time steps t simulated: first with $t = 16$, and then with $t = 100$.

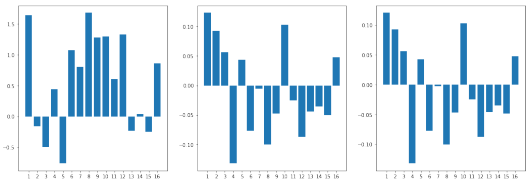
[1] By backpropagation, we refer to the reverse mode automatic differentiation technique of deriving the gradients w.r.t. any given parameter(s) of a composite function by applying the chain rule.

For $t = 16$ simulation steps, starting from a loss of around 400, the first 100 GD optimization steps with $\lambda = 10^{-3}$ reduced the loss to under 1.0, while 200 steps further decreased it to under $4 \cdot 10^{-4}$.

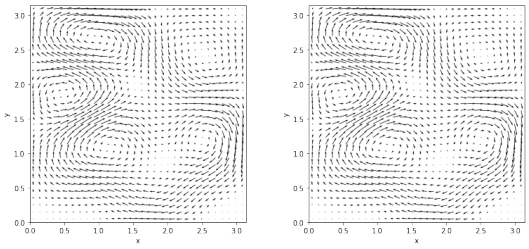
Naturally, this very basic method has its limits. Optimizing for initial coefficients for a target velocity field after 100 steps proved to be a substantially harder problem, as even a relatively small error can accumulate into major deviations over these longer time steps, resulting in much less stable gradients. With using the same learning rate, the optimization diverged almost instantly. With some tuning of the learning rate λ in the range of $[10^{-4}, 10^{-8}]$, we were able to get the loss below 0.14. (Starting from an initial loss of 320 from the random initialization.)

We visualize the results of these two scenarios in Figure 2. It is interesting to observe that even though the optimization had absolutely no knowledge of $\mathbf{w}^*$, only a comparison with a precomputed $\mathbf{u}^*$ velocity field at the target time step, the optimized $\mathbf{w}_{optim}$ vector already starts to look similar to $\mathbf{w}^*$. Keep in mind that this is not guaranteed at all. In some other cases of running this optimization setup, we also observed $\mathbf{w}_{optim}$s that are completely different from $\mathbf{w}^*$. Due to the physical constraints of the eigenfluids simulation, in these cases the optimization could not change any of the 16 values of $\mathbf{w}_{optim}$ locally in a way that would further reduce the loss below some small number, and was stuck in a local minimum of the parameter space.

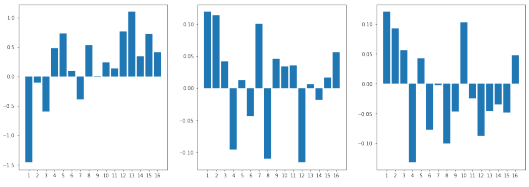
Although there are a number of ways to tweak this setup, we can already verify from these results that the flow of the gradients is working, and is ready to be tested in more advanced scenarios.



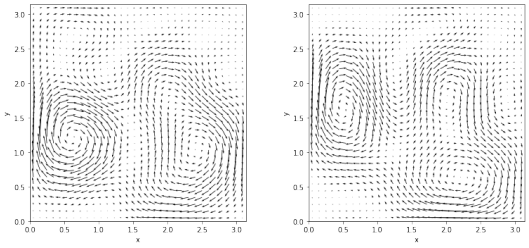
(a) $\mathbf{w}_{init}$, $\mathbf{w}_{optim}$, and $\mathbf{w}^*$, optimizing for velocity field after 16 time steps



(b) Target $\mathbf{u}^*$, and $\mathbf{u}^{16}$, reconstructed from $\mathscr{P}^{16}(\mathbf{w}_{optim})$



(c) Initial basis coefficients $\mathbf{w}_{init}$, $\mathbf{w}_{optim}$, and $\mathbf{w}^*$, optimizing for velocity field after 100 time steps



(d) Target $\mathbf{u}^*$, and $\mathbf{u}^{100}$, reconstructed from $\mathscr{P}^{100}(\mathbf{w}_{optim})$

Figure 2: Results of optimizing for an initial $\mathbf{w}_0$ basis coefficient vector that matches a target velocity field $\mathbf{u}^*$ when reconstructed after simulating for t time steps.

4.2 Controlling Shape Transitions

Advection of some scalar quantity in a fluid is an abstract problem that describes many real-world phenomena, like ink in water or smoke in the air. We define a density function $\psi(\mathbf{x})$ over a simulation domain D. In a fluid with velocity $\mathbf{u}$, and $\nabla \cdot \mathbf{u} = 0$ holding (i.e. the fluid is incompressible), the advection is governed by the equation

$$\frac{\partial \psi}{\partial t} + \mathbf{u} \cdot \nabla \psi = 0. \tag{18}$$

We define each shape with

$$\psi(\mathbf{x}) = \begin{cases} 1, & \text{inside the shape} \\ 0, & \text{outside the shape.} \end{cases} \tag{19}$$

In Eulerian fluid simulation methods [1], both $\mathbf{u}$ and ψ are sampled on grids, numerically approximating the evolution of the field quantities. The work of Holl et al. [5] formulated the shape transition problem in an Eulerian representation, with explicitly simulating the shapes as scalar marker densities being advected by the velocity field of the simulated fluid. Instead, playing to the strengths of an eigenfluids simulation, our method proposes sampling the density function at discrete particle positions, thus rephrasing the process in a Lagrangian way. In the context of Laplacian eigenfluids, a Lagrangian viewpoint is especially inviting, as the explicit description of the fluid velocity $\mathbf{u}$ (Equations 2 and 6) allows us to reconstruct $\mathbf{u}$ only partially, while keeping the simulation of the fluid dynamics in a reduced dimensional space. In a forward physics simulation, this can already lead to substantial speed-ups, but this formulation seems especially promising when the backpropagation of variables is desired, such as the optimization scenarios introduced herein.

We formulate three different control problems, each with a different mean to exert control over the fluid simulation.

- In Section 4.2.2, similarly to the problem statement in Section 4.1, we are looking for an initial coefficient vector $\mathbf{w}_0$, such that when simulated for t time steps, the reconstructed velocity field $\mathscr{R}\mathbf{w}_t = \mathbf{u}_t$ advects some initial shape into some target shape.

- In Section 4.2.3, we optimize for some force vector $\mathbf{f} \in \mathbb{R}^{t \times N}$, such that $\mathbf{f}_t \in \mathbb{R}^N$ applied as external force to each time step of an eigenfluid simulation, it yields the desired outcome.

- Finally, in Section 4.2.4, we generalize the problem to looking for a function that exerts the necessary control force at time t, such that particles currently at positions $\mathbf{p}_t$ end up at target positions $\mathbf{p}_{t+1}$ at the next time step. We formulate this third task as a neural network (NN) model in the form $\mathbf{f}(\mathbf{p}_t, \mathbf{p}_{t+1}, \mathbf{w}_t, \theta)$, also passing in the current basis coefficient vector $\mathbf{w}_t$, and optimizing for its parameters θ to yield the desired outcome (as introduced in Section 3.2).

In each of these tasks, a velocity field $\mathbf{u} = \mathscr{R}\mathbf{w}$ advects a set of initial points $\mathbf{p}_0 = \left[\mathbf{p}_0^0, \ldots, \mathbf{p}_0^i\right]$ to line up with target positions $\mathbf{p}_t = \left[\mathbf{p}_t^0, \ldots, \mathbf{p}_t^i\right]$ after time t. Using a mean-square error, our loss function becomes

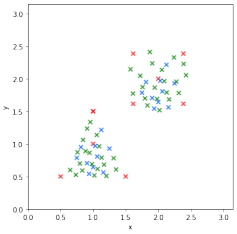
$$L(\mathbf{w}, \mathbf{p}_0, \mathbf{p}_t) = \left\| \mathscr{P}^t(\mathbf{p}_0, \mathbf{w}) - \mathbf{p}_t \right\|_2^2. \tag{20}$$

4.2.1 Sampling

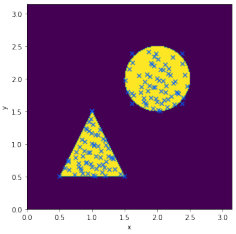
As we neither want to lose too much information about our original function nor do we want to keep track of an unnecessary number of points, the feasibility of our method necessitates an efficient sampling of $\psi(\mathbf{x})$. We use a simple rejection-based sampling technique. We generate random points $\mathbf{p}_{\text{sample}} \in [0,1] \times [0,1]$, rejecting them if they lie outside the shape.

As we consider shape transitions given start and target shapes S_0 and S_t, it is important to take into consideration the connection between these shapes. To balance finding spatial correspondences between the shapes, while still approximating their unique shapes, we sample O overlapping, and U unique points. For the *overlapping* points, we accept only $\mathbf{p}_{\text{sample}} \in S_0 \cup S_\text{t}$, i.e. we reject points that are not inside both shapes. For the *unique* points, we sample a different set of points for each shape. To generate low-discrepancy, quasi-random 2D coordinates, we use Halton sequences.We further generate $T = 5$ *trivial* points that are hand-picked to best resemble the given shape, as well as line up between different shapes. We choose these to be the center, upper right, upper left, lower left, and lower right corners of the shapes.

In conclusion, our final set of $\mathbf{p}_0$ initial, and $\mathbf{p}_t$ target sample positions are given by concatenating the O overlapping, U unique, and T trivial points for each shape, resulting in two sets of sample points $\mathbf{p}_0, \mathbf{p}_t \in \mathbb{R}^{2(O+U+T)}$. Figure 3 shows the result of our sampling strategy for a triangle and a circle shape.



(a) $O = 30$ (blue), $U = 30$ (green), and $T = 5$ (red) points.



(b) Sample points over $\psi_{\text{triangle}} + \psi_{\text{circle}}$.

Figure 3: Sampling strategy for transitioning from a triangle to a circle. Halton series with base $(2,7)$ and $(3,11)$ were used to generate the overlapping and unique positions, respectively.

4.2.2 Optimizing for Initial Velocity

As Equation 20 introduced the problem, our goal is to find an initial velocity field $\mathscr{R}\mathbf{w} = \mathbf{u}$ that advects points $\mathbf{p}_0$ to line up with target positions $\mathbf{p}_t$ after t steps. We can write optimizing for base coefficients $\mathbf{w}$ as:

$$\arg\min_{\mathbf{w}} \left\| \mathscr{P}^t(\mathbf{p}_0, \mathbf{w}) - \mathbf{p}_t \right\|_2^2. \tag{21}$$

Making use of the differentiability of our physical simulator $\mathscr{P}$, and the multivariable chain rule for deriving the gradient of the above $\mathscr{P}^t$ function composition, we can derive its gradient with respect to the initial coefficients:

$$\frac{\partial \mathscr{P}^t(\mathbf{w}, \mathbf{p})}{\partial \mathbf{w}}.$$

Finally, we simply iterate a GD optimizer to find a (good enough) solution for the minimization problem of Equation 21:

$$\mathbf{w}_{\text{better}} = \mathbf{w} - \lambda \frac{\partial L(\mathbf{w}, \mathbf{p}_0, \mathbf{p}_t)}{\partial \mathbf{w}},$$

where L is the same as in Equation 20:

$$L(\mathbf{w}, \mathbf{p}_0, \mathbf{p}_t) = \left\| \mathscr{P}^t(\mathbf{p}_0, \mathbf{w}) - \mathbf{p}_t \right\|_2^2.$$

The main difficulty of this non-linear optimization problem lies in that we have no control over the natural flow of the fluid besides supplying an initial $\mathbf{w}_0$ vector. We showcase two different setups in Figure 4, with the details of both experiments described in Table 1.

Table 1: Details of the two optimization scenarios shown in Figure 4.

	Figure 4a	**Figure 4b**
N	16	36
Sampling size for smoke simulation	32	32
Eigenfluid initialization time	6.19 sec	68.47 sec
Time for 51 optimization steps	108.05 sec	230.48 sec
Initial loss	2.3	2.19
Final loss	0.08	0.09
Number of overlapping points O	0	30
Number of unique points U	0	30
Number of trivial points T	5	0

4.2.3 Control Force Estimation

In this scenario, we optimize for a force vector $\mathbf{f} \in \mathbb{R}^{t \times N}$, such that $\mathbf{f}_t \in \mathbb{R}^N$ applied as external force at each time step t of an eigenfluid simulation, initial positions $\mathbf{p}_0$ will be advected to target positions $\mathbf{p}_t$ after t time steps:

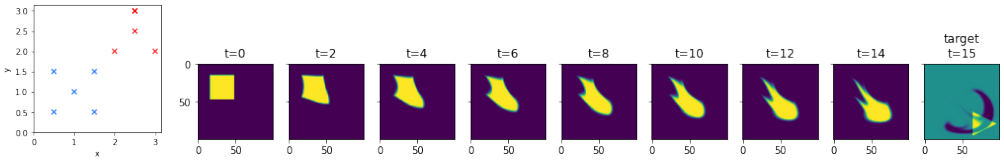
$$\arg\min_{\mathbf{f}} \left\| \mathscr{P}^t(\mathbf{p}_0, \mathbf{w}, \mathbf{f}) - \mathbf{p}_t \right\|_2^2,$$

where $\mathscr{P}^t(\mathbf{p}_0, \mathbf{w}, \mathbf{f}) = \mathscr{P} \circ \cdots \circ \mathscr{P}(\mathbf{p}_0, \mathbf{w}, \mathbf{f})$ denotes simulating the physical system for t time steps, applying $\mathbf{f}_t$ force at each time step. Results of the optimization are shown in Figure 5.

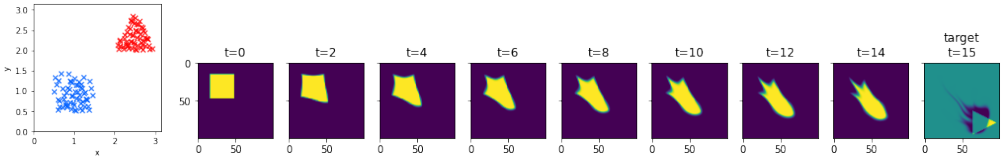
4.2.4 Neural Network Training

We generalize the control force estimation (CFE) problem by defining a function $\mathbf{f}(\mathbf{p}_t, \mathbf{p}_{t+1}, \mathbf{w}_t)$: $\mathbb{R}^{2 \cdot 2(O+U+T)+N} \to \mathbb{R}^N$, that given particles at positions $\mathbf{p}_t$ and velocity field $\mathbf{w}_t$ at time t, gives a force that advects the particles to positions $\mathbf{p}_{t+1}$ in the next time step. Its inputs are the (x,y) coordinates of the points, and the N basis coefficients, giving $2 \cdot 2(O+U+T)+N$ values, where O, U, and T denote the number of overlapping, unique, and trivial sample points, respectively, as introduced in Section 4.2.1.

We approximate function $\mathbf{f}$ with a CFE NN $\mathbf{f}(\mathbf{p}_t, \mathbf{p}_{t+1}, \mathbf{w}_t, \theta)$. Each layer is constructed as described in Equation 14 with ReLU non-linearities in-between. Figure 6 gives an overview of our NN architecture. As the input size of the NN is dependent on the specific problem, the number of trainable parameters also varies, and a new NN has to be trained when using a different number of basis fields, or different number of total sample points. As an example, for $N = 16$ basis fields, and 75



(a) Using $O = 0$, $U = 0$, $T = 5$ sampling points and $N = 16$ basis fields.



(b) Using $O = 30$, $U = 30$, $T = 0$ sampling points and $N = 36$ basis fields.

Figure 4: In the least complex scenario, we solve the shape transition problem by optimizing for an initial coefficient vector $\mathbf{w}$ without any further control over the simulation. Time evolution over 16 time steps of two different simulation set-ups are shown, with different number of sample points and basis fields used. Initial (blue) and target (red) sample points are shown.

sample points, the NN has 337 392 trainable parameters. Testing the setup, we overfit the NN to a single training sample. Plotting the results of the time evolution in Figure 7, we observe that a reduced degrees of freedom can yield comparable, or even better results with the same setup, and training time. Using an Adam optimizer [8] with learning rate 10^{-3}, the results shown in Figure 7 were achieved in 260 epochs. The training took 53.94 seconds.

Training. We generate 2000 samples, using 1800 for training, and 200 for validation. Using $N = 16$ basis fields, we train the NN for the CFE problem detailed above. At the end of the training, we generate further data the NN has not seen during training to further test generalization. Using an Adam [8] optimizer with learning rate 10^{-3}, the results shown in Figure 8 were achieved in 260 epochs. The training took 1201.74 seconds (20 minutes). As we did not experience any overfitting issues during training, no additional regularization schemes were applied.

5 Results & Discussion

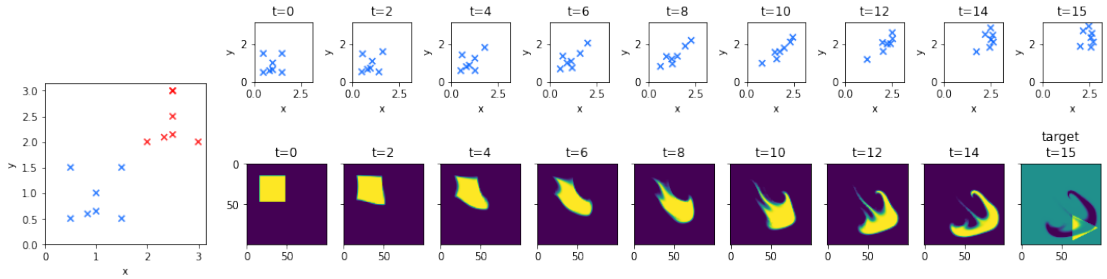
After introducing gradient-based optimization in the context of eigenfluids (Section 4.1), we proposed a novel approach to control shape transitions (Section 4.2) in the reduced-dimensional fluid simulation. Starting with individual optimization problems (Figures 4 and 5), we showed that NNs can not only give comparable results to a set of problems, but they also generalize beyond the examples seen during training (see Figure 8).

Owing to the reduced-order nature of the approach, we achieved speed-ups that usually result in convergence times of minutes even in the case of more advanced setups (and sub-minute, or seconds in the more straight-forward ones).

Although not a silver bullet, we believe that this approach complements and connects existing techniques in a new and exciting way, offering a fresh perspective on thinking about NNs as universal function approximators.

Generalizing to 3D. All of the introduced methods generalize to 3D in a straightforward way. As shown by Cui et al. [3], the Laplacian eigenfluids technique is a viable option for simulating three-dimensional incompressible fluid flow.

Target Trajectory. We estimate the trajectory as a linear interpolation between start and end positions. Recalculating the trajectory based on the actual path taken after applying the control forces at each time step might lead to more natural transition paths. Alternatively, the effects of implementing a predictor-corrector scheme as introduced by Holl et al. [5] could be investigated.



(a) Sample points.

(b) Optimized trajectory of the sample points underlying the optimization (top). Smoke advection is shown for qualitative comparison, reconstructed on a 100×100 grid (bottom).

Figure 5: Direct force optimization results with 16 time steps, and using $N = 16$ basis fields. We observe that although the sample points (blue) were advected close to their target positions (red), using only 5 sample points was not enough to approximate the underlying higher dimensional advection dynamics.

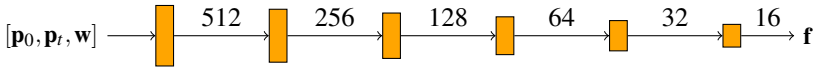


Figure 6: The CFE NN transforms the input vector of size $2 \cdot 2(O+U+T)+N$ into a force vector $\mathbf{f}$ that can be added to the $\mathbf{w}$ coefficients as external force. (The architecture for $N = 16$ fields is shown.) Each layer is linear, with outuput sizes matching each following input size. A ReLU non-linearity is applied after each layer.

References

[1] R. Bridson. *Fluid simulation for computer graphics, Second Edition*. K Peters/CRC Press, 2015. doi: 10.1201/9781315266008. URL https://doi.org/10.1201/9781315266008.

[2] Li-Wei Chen, Berkay A. Cakal, Xiangyu Hu, and Nils Thuerey. Numerical investigation of minimum drag profiles in laminar flow using deep learning surrogates. *Journal of Fluid Mechanics*, 919:A34, 2021. doi: 10.1017/jfm.2021.398.

[3] Qiaodong Cui, Pradeep Sen, and Theodore Kim. Scalable Laplacian Eigenfluids. *ACM Trans. Graph.*, 37(4), jul 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201352. URL https://doi.org/10.1145/3197517.3201352.

[4] Tyler De Witt, Christian Lessig, and Eugene Fiume. Fluid Simulation Using Laplacian Eigenfunctions. *ACM Trans. Graph.*, 31(1), feb 2012. ISSN 0730-0301. doi: 10.1145/2077341.2077351. URL https://doi.org/10.1145/2077341.2077351.

[5] Philipp Holl, Vladlen Koltun, and Nils Thuerey. Learning to Control PDEs with Differentiable Physics. In *ICLR*, 2019. URL https://ge.in.tum.de/publications/2020-iclr-holl/.

[6] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable Programming for Physical Simulation. *ICLR*, 2020.

[7] Aaron Demby Jones, Pradeep Sen, and Theodore Kim. Compressing Fluid Subspaces. In Ladislav Kavan and Chris Wojtan, editors, *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*. The Eurographics Association, 2016. ISBN 978-3-03868-009-3. doi: 10.2312/sca.20161225.

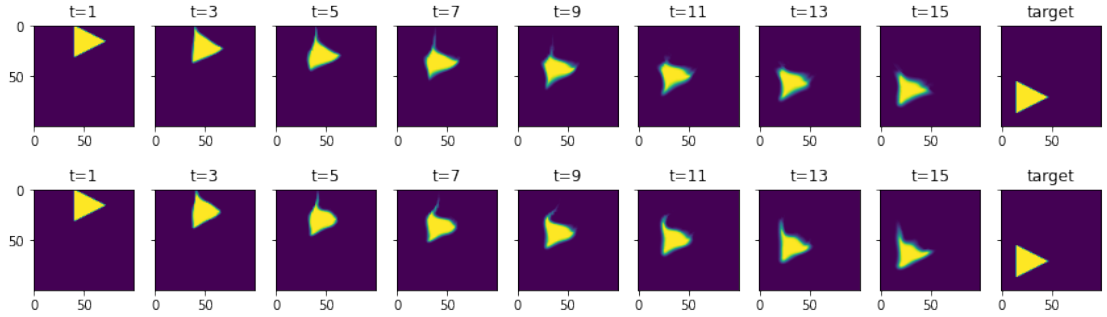


Figure 7: Time evolution of simulating two overfitted CFE NNs to a single shape transition for 16 time steps, using the same 75 sample points, but different number of basis fields (top: $N = 16$, bottom: $N = 36$).

[8] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 12 2014.

[9] Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. DiffCloth: Differentiable Cloth Simulation with Dry Frictional Contact. *ACM Trans. Graph.*, 42(1), oct 2022. ISSN 0730-0301. doi: 10.1145/3527660. URL https://doi.org/10.1145/3527660.

[10] Beibei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. Model-Reduced Variational Fluid Simulation. *ACM Trans. Graph.*, 34(6), nov 2015. ISSN 0730-0301. doi: 10.1145/2816795.2818130. URL https://doi.org/10.1145/2816795.2818130.

[11] Miles Macklin. Warp: A High-performance Python Framework for GPU Simulation and Graphics. https://github.com/nvidia/warp, March 2022. NVIDIA GPU Technology Conference (GTC).

[12] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

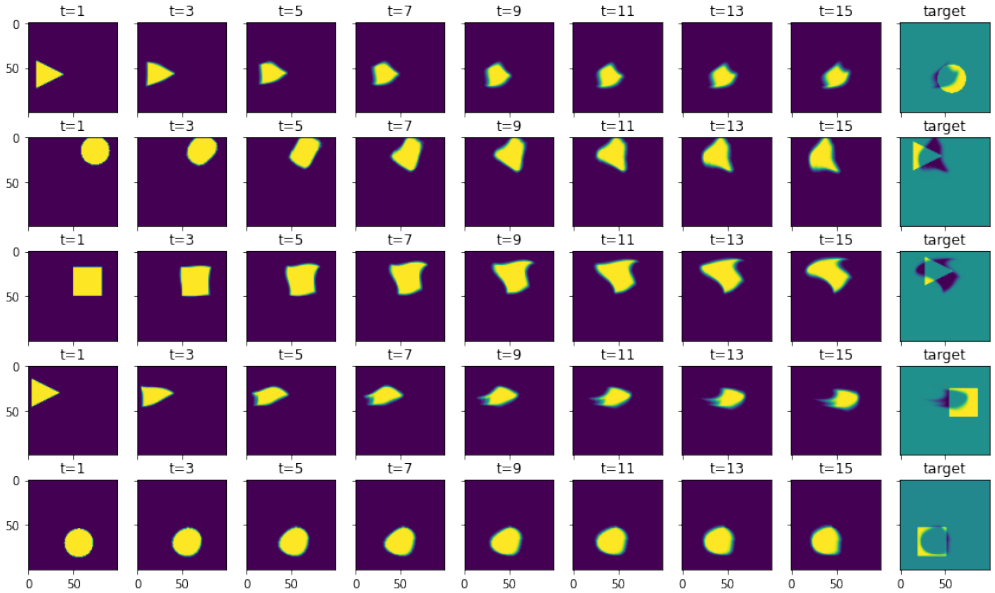
[13] Jos Stam. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, page 121–128, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0201485605. doi: 10.1145/311535.311548. URL https://doi.org/10.1145/311535.311548.

[14] Nils Thuerey, Philipp Holl, Maximilian Mueller, Patrick Schnell, Felix Trost, and Kiwon Um. *Physics-based Deep Learning*. WWW, 2021. URL https://physicsbaseddeeplearning.org.

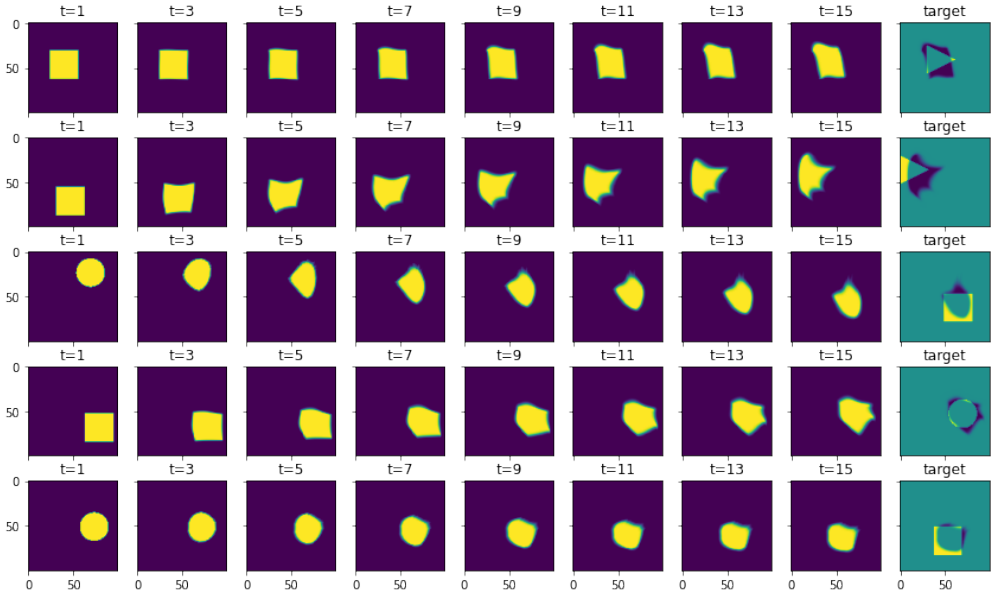
[15] Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. In *Advances in Neural Information Processing Systems*, 2020.

[16] S. Wiewel, B. Kim, V. C. Azevedo, B. Solenthaler, and N. Thuerey. Latent Space Subdivision: Stable and Controllable Time Predictions for Fluid Flow. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '20, Goslar, DEU, 2020. Eurographics Association. doi: 10.1111/cgf.14097. URL https://doi.org/10.1111/cgf.14097.

[17] Steffen Wiewel, Moritz Becher, and Nils Thürey. Latent Space Physics: Towards Learning the Temporal Evolution of Fluid Flow. *Computer Graphics Forum*, 38, 2019.



(a) Performance on **training data**. (Randomly sampled.)



(b) Testing on previously unseen **test data**. (Randomly sampled.)

Figure 8: Randomly sampled time evolution of controlled shape transition tasks. Using $N = 16$ basis fields, sampling the smokes on a 32×32 grid, approximating them with $O = 30$ overlapping, $U = 40$ unique and $T = 5$ trivial sample points, through 16 time steps $t = [0 \dots 15]$.

Translucent Material Parameter Estimation

Saip-Can Hasbay[(1)]
Supervised by: David Hahn[(2)]

(1) University of Vienna, (2) TU Wien

Abstract

In this paper, we present a workflow for optical material parameter estimation based on real-world images, numerical optimization, and differentiable rendering (using *Mitsuba 3*). We primarily focus on *translucent* materials and demonstrate our approach both for synthetic and real-world data. Specifically, we estimate parameters for two well-known material models: either a *Principled BSDF* (a surface model based on Burley's Disney BSDF), or a *Rough dielectric BSDF*, describing volumetric *homogeneous* participating media.

In order to solve the inverse rendering problem of acquiring suitable material properties from a (set of) given reference image(s), we present a software tool that handles the entire material reconstruction workflow. Furthermore, we also propose an experimental workflow to acquire the necessary reference images and potentially the 3D scene geometry. Our results show that our approach works on well-known materials such as acrylic glass, as well as newly designed materials, such as alginate, from experimental materials research.

Keywords: Numerical Optimization, Material Parameter Estimation, Differentiable Rendering, Appearance Modelling

1 Introduction

Early work in computer graphics relied on phenomenological, often physically implausible, shading models and focused primarily on rendering images under direct, or purely diffuse illumination [26, 2, 27]. The driving force behind many rendering methods was artistic expression rather than physical accuracy. However, with more powerful hardware supporting modern ray tracing algorithms [23, 14], photo-realistic and physics-based rendering has become not only feasible but de-facto standard in many fields, both within and outside of computer graphics.

Given a virtual scene description, the main goal of physics-based rendering is to create images matching the way our eyes perceive the world. The ability to create images that are hard to separate from reality, however, hinges on accurate knowledge of the optical material parameters of objects in the scene.

Scattering occurring at the surface of objects is modelled by the *bidirectional scattering distribution function* (BSDF). Statistical micro-surface models provide physics-based, parametric BSDF formulations that describe the material properties of an object in the scene. For translucent materials, ray tracing methods, as well as material descriptions, have been extended to cover *volumetric scattering* in addition to surface effects. While both surface and volumetric BSDF parameters can be obtained from careful measurements of real-world materials, or instead formulated in a data-driven (rather than a parametric) framework, these approaches heavily rely on specialized laboratory equipment. However, this precise approach might not always be feasible or cost effective. For example, research in materials engineering might produce new compounds for specific applications, whereas aging and weathering effects might alter a material's appearance over time.

Figure 1: Reconstruction results for a surface BSDF model from synthetic data (left) and real-world alginate material (right). Dragon model by Delatronic [6], License: CC-BY.

Recent work on differentiable and inverse rendering, on the other hand, has enabled the identification of optical material parameters by applying (gradient-based) optimization methods in order to match the rendered result to an object's real-world appearance. This optimization approach is particularly suited to estimating parameters from only a few reference images. In this work, we rely on the *Mitsuba 3* renderer [11], a publicly available, state-of-the-art differentiable rendering system, to solve material parameter estimation problems. However, as a research-oriented system, Mitsuba 3 requires a lot of domain-specific knowledge to operate. To facilitate the parameter estimation workflow, we implement a software tool that combines a gradient-based optimization algorithm with the necessary data management and rendering interface.

In the remainder of this paper, we first provide an overview of previous work on physics-based (inverse) rendering and accumulate relevant background information. The subsequent sections then detail the experimental and computational aspects of our proposed approach. Lastly, we present and analyze our results from both synthetic and real-world data. In summary, our contributions are:

- A software tool[1] that incorporates the full power of the Mitsuba 3 renderer and various features to accomplish the task of inverse rendering.
- A gradient-based optimization procedure to acquire material properties from an image, including a careful evaluation of how optimization hyper-parameters and rendering quality affect the results.
- A workflow to increase the reproducibility of our results, encompassing both synthetic as well as real-world examples.

2 Related Work

Physics-based rendering has its origins in the seminal paper by Kajiya [12], formulating the now well-known rendering equation. Veach [32] introduced the path-integral form, which provides the theoretical foundation of modern Monte Carlo (MC) ray tracing methods [25]. In order to arrive at a photorealistic rendering, an accurate description of how various materials interact with light must be formulated. Different analytical material models have been introduced to account for the light-scattering effects from surfaces [34, 9, 11]. Similarly, models for participating media describe volumetric scattering effects as light passes through the material [25, 11]. Physics-based rendering itself (which we also refer to as *forward* rendering in this context) is concerned with computing accurate images using the rendering equation, given a scene description, including all materials.

Conversely, the question for *inverse* rendering is how to find an appropriate scene description in order to obtain a desired result. In particular, finding the parameters of a scattering model corresponding to a real-world material, has been a long-standing research problem. While some previous work has addressed this problem in a data-driven way [19, 15, 28], inverse rendering methods in contrast, seek to determine the parameters of established material models, resulting in a more constrained search space that ensures the use of physics-based models. Early work on inverse rendering for material parameter estimation [8] used material dictionaries, aiming to approximate the appearance of real-world materials as a combination of dictionary entries. More recently, advances in differentiable rendering [22, 33, 21, 16, 37] have enabled inverse rendering applications by applying gradient-based optimization directly on the material parameters.

Initially, differentiable renderers have applied code-level automatic differentiation to find gradients of either the rendered image, or an objective function defined on that image, with respect to input parameters. In order to reduce the computational cost and memory footprint of automatic differentiation, analytic back-propagation formulations have been developed [22, 33]. One issue that has received a lot of attention in recent research, is that derivatives of pixel values (which are integrated according to the rendering equation) can contain discontinuous integrands if a silhouette edge moves across the pixel due to a change of scene parameters [16, 36, 17, 38].

Modern ray tracing renderers use importance sampling to reduce noise due to MC sampling, based on either the material's scattering behaviour, the light sources, or combining both through multi-importance sampling. Consequently, similar sampling strategies have also been investigated for differentiable ray tracing. Zeltner et al. [36] analyzed numerous potential approaches and mathematically classified various estimators for differential light transport. Prior to the work by Vicini et al. [33], many techniques used statistically biased gradient estimation methods. Their work proposed a new back-propagation algorithm that runs in constant memory and linear computation time. Additionally, this method provides a way to handle highly specular materials such as smooth dielectrics and conductors. Our work builds upon their method, which is implemented in Mitsuba 3 [11].

Inverse Rendering for translucent material reconstruction. The work by Gkioulekas et al. [8] is one of the initial inverse rendering research regarding translucent material reconstruction. They introduced an optimization framework to measure the bulk scattering properties of homogeneous materials. Their primary focus was describing homogeneous materials by two scalar values and one angular function: scattering coefficient, absorption coefficient and phase function. Similar to our project, Gkioulekas et al. [8] also incorporated gradient-based optimization with MC rendering. However, they specifically used stochastic gradient descent for the optimization. Moreover, in their optimization procedure, they used a material dictionary to invert the radiative transfer equation. The material dictionary contained a variety of common materials, including liquid, solid, and publicly-available collections of tabulated phase functions.

Yang et al. [35] proposed another inverse rendering approach for heterogeneous translucent materials from a single input photograph. Their method can obtain the material distribution and estimate material parameters from the provided reference image. Yang et al. introduced an iteration process for optimizing the scattering and absorption coefficient. Deng et al. [7] introduced another approach to reconstruct translucent objects using differentiable rendering. They extended previous methods using a bidirectional scattering-surface reflectance distribution function (BSSRDF). To handle the noise introduced by the BSSRDF integral, they proposed a dual-buffer method for evaluating the loss during optimization. In this work, we make use of the dual-buffer method and observe improved optimization convergence over standard error metrics (see our results in §5).

[1] Access at https://github.com/sapo17/BachelorThesis.

3 Parameter estimation

In this section, we briefly recap the underlying theory of the rendering process and the core principle of differentiable ray tracing. Next, we present the material parameters of interest, and finally state the problem we address.

3.1 Background and theory

Let us start with the well-known rendering equation as formulated by Kajiya [12], also referred to as the light transport equation (LTE). As we are primarily interested in translucent materials, we consider the angular form of the LTE, see also [25, 12]:

$$L(p,\omega_o) = \underbrace{L_e(p,\omega_o)}_{Emission} + \underbrace{\int_{S^2} L_i(p,\omega_i) f(p,\omega_i,\omega_o) d\omega_i}_{Scattering}. \quad (1)$$

Here $L(p,\omega_o)$ describes the radiance (power per area, solid angle, and color channel) leaving a point p in direction ω_o due to light emitted at or scattered through p. The incident radiance results from recursively applying the LTE such that $L_i(p,\omega_i) = L(p',-\omega_i)$, where p' is a point directly visible from p in direction ω_i. Monte Carlo (MC) ray tracing approximates this recursive integral as a sum of randomly sampled scattering rays. The same concept has been extended to volumetric rendering, including scattering and attenuation in participating media [25].

Most importantly for us, the function $f(p,\omega_i,\omega_o)$ encodes the material's optical properties, and is commonly referred to as the *bidirectional scattering distribution function* (BSDF). For convenience of notation, we assume the cosine term accounting for the projected solid angle is included in the BSDF. The BSDF is composed of a reflective part, the bidirectional reflectance distribution function (BRDF) and a transmissive part, the bidirectional transmittance distribution function (BTDF).

Inverse and differentiable Rendering. So far we have summarized how a virtual scene, including material descriptions, can be rendered via ray tracing. We now move on to the problem of inverse rendering, where we aim to find scene parameters, such that the resulting image fulfills certain requirements. In particular, our problem is finding the parameters of the material models, such as to match given reference images. Inverse rendering is commonly defined as an optimization problem of the form

$$\min g(y(x)), \text{ subject to } h(x) \leq 0, \quad (2)$$

where g defines the optimization objective (or loss) function on the rendered image y given scene parameters x, and h defines additional constraints (e.g., min/max parameter values).

In order to solve this optimization problem more efficiently, differentiable rendering provides derivatives $(\delta y/\delta x)$, which allows gradient-based optimization techniques to successively improve the parameters with respect to the specified objective [38]. In the simplest form, the objective takes the L2 norm of the difference between the rendered and the reference image, thus $g(y) = ||y - y_{ref}||^2$. The *dual buffer method* by Deng et al. [7] proposes an alternative objective function that is better suited to handle the noise introduced by the MC rendering. Rather than taking one differentiable rendering step, the dual buffer method takes two independent rendering steps y_1, y_2, per iteration. Instead of the standard L2 loss, we now compute the loss in each iteration as $g(y_1,y_2) = (y_1 - y_{ref}) \cdot (y_2 - y_{ref})$.

Finally, in order to acquire the gradients required for optimization, Mitsuba 3 implements a path-replay back-propagation method [22, 33]. Formally, we take the derivative of Eq. (1) with respect to the scene parameters x:

$$\begin{aligned}\delta_x L(p,\omega_o) = &\underbrace{\delta_x L_e(p,\omega_o)}_{Emission} \\ &+ \int_{S^2} [\underbrace{\delta_x L_i(p,\omega_i) f(p,\omega_o,\omega_i)}_{Transport} \\ &+ \underbrace{L_i(p,\omega_i)\delta_x f_x(p,\omega_o,\omega_i)}_{Material}] d\omega_i.\end{aligned} \quad (3)$$

This equation describes the scattering of derivatives analogous to the LTE. The individual terms are:

- Emission: differential radiance is emitted when the emitted radiance L_e depends on x.
- Transport: differential radiance *scatters* in the same way as normal radiance, according to the BSDF f.
- Material: surfaces with a parameter-dependent BSDF emit differential radiance proportional to incident radiance.

Note that along each ray, we find $\delta L_i(p,\omega_i) = \delta L(p',-\omega_i)$, analogous to the forward rendering. While in an abstract sense, the gradient of the objective function could be computed as $\delta g/\delta x = (\delta g/\delta y)(\delta y/\delta x)$, doing so would be computationally expensive due to the large size of $\delta y/\delta x$ (which can be thought of as a matrix of derivatives for each image pixel with respect to each scene parameter). Instead, the back-propagation method uses the (partial) derivative of the objective function $\delta g/\delta y$ as a source of "adjoint radiance" which is then traced from the virtual camera into the scene, scattered according to Eq. (3) and "collected" at the scene parameters.

In this work, we rely on the ADAM optimizer [13] to solve the parameter estimation problem. This method is a general-purpose method that applies gradient descent with momentum and an adaptive strategy to estimate the learning rate per parameter. As such it is well suited to non-linear optimization problems that may contain spurious local minima. Using a momentum strategy can prevent getting stuck in these local minima and increase the chances of finding a good solution.

3.2 Microfacet models and Disney BSDF

Modeling surface reflection and transmission often originates from the idea that rough surfaces can be represented

as a collection of small microfacets. The scattering of light from a group of microfacets is statistically modelled by microfacet-based BRDFs [25]. Walter et al. [34] introduced a microfacet distribution function, called GGX, which has been shown to be equivalent to the microfacet distribution introduced by Trowbridge and Reitz [30]; see also [24]. Microfacet models not only require the distribution of facets, but also need to account for some facets being obscured by others (also known as *masking*) and some will be *shadowed*. Smith's masking-shadowing function [25] addresses these effects. More recently, Walter et al. [34] reviewed microfacet theory and extended it to simulate transmission through rough surfaces. *Mitsuba 3* [10] implements this variant in its Rough-dielectric BSDF model, which we make use of in this project. Furthermore, *Mitsuba 3* allows two types of participating media in the scene: homogeneous and heterogeneous, which can be used to simulate translucent materials such as milk or skin, but also clouds and fog [10]. Therefore we combine a Rough-dielectric BSDF and a homogeneous interior medium to simulate and optimize volumetric scattering effects in translucent materials.

Burley et al. [3] proposed a general-purpose BRDF (also known as Disney Principled BRDF). They compared their new model with measured materials in terms of the microfacet models [29, 5, 34]. Their BRDF blends metallic and dielectric BRDF models and includes a microfacet reflection with anisotropic roughness and an optional secondary clearcoat reflection. The Disney BSDF [9] extends the dielectric BRDF with integrated subsurface scattering and blends in a specular BSDF based on a new specular transmission parameter, arriving at a unified BSDF model including refraction and subsurface scattering effects. For the specular BSDF, they follow the GGX model and extend the microfacet reflection to refraction. *Mitsuba 3* [10] implements the model proposed by Burley et al. [9] in its Principled BSDF material, which we also use in this project.

3.3 Material parameters

Following the summary of material models, we now briefly list the parameters of these models we use in our optimization pipeline. For surface-only materials, i.e. the *Disney Principled BSDF*, we have:

- base colour (albedo) c (can be textured),
- surface roughness α,
- specular transmission σ_s (blending between the BRDF and BTDF lobes), and the
- index of refraction η.

Note that we do not use secondary reflection effects such as clear-coat materials, or metallic reflections here. For volumetric rendering, i.e. *Rough Dielectric BSDF* with homogenous medium [34, 11], we additionally consider

- a *volumetric* albedo colour (texture) and
- an extinction coefficient σ_t describing the absorption as light traverses the material.

3.4 Problem statement

In summary, our goal is to estimate optical material parameters $x = \{c, \alpha, \sigma_s, \eta, \sigma_t\}$ in a given 3D scene, such that the rendered result best approximates one (or more) reference image(s). Therefore, we require

- a (set of) reference image(s) and
- a Mitsuba scene file, including the initial guess for the material parameters, and virtual cameras corresponding to the reference images as input to our system.

To solve this problem, we combine differentiable rendering with a gradient-based optimization procedure. To use our approach, however, we first need to meet the above-mentioned requirements, which we discuss in the first part of the next section.

4 Method

In this section, we first introduce the experimental and then the computational part of our approach.

4.1 Scene and image acquisition

Here, we focus on 3D scene construction and reference image acquisition. For validation tests where the Bunny [31] model is in use, we utilized the readily available Mitsuba scene from their documentation [11] and modified the parameters we introduced in §3.1. However, for synthetic data, users may also construct a scene in a 3D modelling tool such as Blender [4]. For example, for the Dragon [6] model test case in §5.1, we constructed a scene in Blender and exported it using Mitsuba's Blender Add-on [20]. For validation tests, we then rendered (using Mitsuba) the acquired scenes to get the reference images.

The real-world case is more complicated. During imaging, a controlled environment is crucial to accurately determine all scene parameters that could affect the resulting image, particularly scene geometry and light position(s).

We propose multiple approaches, *first*, users may reconstruct the imaged environment manually. This task can get seriously complicated and requires relatively adequate skills in modelling. For the alginate [18] material test cases, we only partially had this difficulty. We were provided with a "material scanner"—a small light-proof container with fixtures for camera and light placement—where we photographed the alginate samples. Fortunately, we were also provided with the corresponding 3D model that we used in Blender. However, we still had to approximate the model of the sample alginate materials, camera positions, and the radiance values of the lights. Please note that we processed images from the real-world to remove the background.

Our *second* approach utilizes a photogrammetry tool Metashape [1]. Using Metashape, we not only acquired the geometry of the imaged bird statue (third row in Fig. 5) but also the camera positions from photographs. Using Metashape, (1) we first loaded images and used the *Align*

Images functionality, (2) then used the *Build Mesh* functionality, (3) next, exported mesh and camera locations with the X3D format, (4) and lastly, imported the X3D file into Blender. Once we had a scene in Blender, we utilized the Mitsuba Blender Add-on [20] to export and use it in our tool. We found our second approach useful specifically for *opaque* materials.

Another technique we employed involved *naive* vertex position optimization that is supported by Mitsuba 3 and our tool. As shown in the fourth row of Fig. 5, from a scaled-sphere object the optimization recovers a *rough* bird statue. Geometry reconstruction on its own is a fascinating and challenging research topic, and we will omit the details in this work.

4.2 Optimization

We are now ready to introduce the computational part of our approach, combining the differentiable renderer Mitsuba 3 with an ADAM optimizer to estimate the material parameters.

Using our software tool, we first load a virtual 3D scene file, which includes the initial material parameters x_0. We also load the (set of) reference image(s), y_{ref}, which will be used in the objective function (selecting either the L2 norm or the dual buffer method as introduced earlier). Next, we select the material parameters of interest (x_0), which get assigned to a newly initialized ADAM optimizer by our tool. Optionally, we also select the following optimization hyper-parameters (default values in braces). The *maximal number of iterations* for the optimization (100); the number of *samples per pixel (spp = 4)* for each rendered image; the *loss tolerance* ε, where optimization stops if the loss $g < \varepsilon$ (0.001); the *learning rate*, which adjusts the step size at each iteration (0.03); and *minimum and/or maximum clamp values*, which define box constraints for specific parameters. For example, an RGB color value must be in the interval $[0,1]$ per channel; by default, most parameters[2] are constrained to $[0,1]$. Note that virtual camera and reference image resolutions are overridden by our tool according to the aspect ratio of the loaded image and restricted to $(256 \times AspectRatio, 256)$. This restriction originates mainly from memory limitations.

Initializing $x_i = x_0$, we then run the following optimization loop for each camera pose (i.e. reference image): (1) Perform a differentiable rendering step with respect to x_i resulting in an image y_i. (2) Evaluate the objective function $g(y_i)$. (3) Back-propagate $\delta g/\delta y$ using Mitsuba 3, to obtain $\delta g/\delta x_i$. (4) Take an ADAM optimization step to find updated parameters $\tilde{x}_{i+1}$. (5) Ensure legal values for x_{i+1} by clamping $\tilde{x}_{i+1}$ using box constraints. (6) Update the scene with x_{i+1}. Repeat until either the loss tolerance, or the maximal iterations are reached.

Additionally, our software tool also provides convenience functions. To streamline the above-mentioned process, we propose a mini-tool with a GUI [3]. To execute the optimization, users can simply load a Mitsuba 3 scene and a (set of) reference image(s), and select the appropriate material parameters. Our tool automatically selects integrators and scene resolution, sets default hyperparameters, and presents differentiable scene parameters for optimization upon scene loading. At the end of the optimization, our tool provides the necessary visualizations and ability to export the obtained results.

5 Results

In this section, we examine results produced using our software tool and optimization approach. In the first part, we evaluate results from synthetic data, where ground-truth solutions are available for comparison. In the second part, we show results for real-world data. Our tool uses the NVIDIA CUDA variant of Mitsuba v3.2.1, and all timings reported in the following have been measured on NVIDIA GeForce RTX 2060 graphics card.

5.1 Validation tests

In this section, we examine material reconstruction from synthetic data. These tests start from a virtual scene, with given ground-truth parameters. The optimization must then recover the correct material parameters from a dark and opaque initial guess. First, in Figs. 2, 3, we show successful results using the method described in §4.2. Table 1 also shows the corresponding initial and optimized parameter values and optimization hyperparameters. Please note that we use a single reference image for the Bunny and four images for the Dragon test cases.

In all cases, we observe that the dual buffer method [7] is extremely useful in reconstructing the object's material properties, allowing for *less* restrictive constraints.

Furthermore, we observe that noise inherent to MC sampling affects the optimization procedure, and a sufficient number of samples per pixel (*spp*) is required to obtain good convergence. We also use box constraints on certain parameters, i.e. clamping to minimum or maximum values to prevent some parameters from moving to physically implausible values. Especially the index of refraction (η) often suffers from these issues. We believe this is due to total internal reflection introducing discontinuous jumps in light propagation paths. Another useful strategy to resolve these issues is to split the optimization process into two parts: first we optimize a group of parameters that work well together (for instance excluding η). We then continue from the optimized values from the first part and include all parameters. Result using this procedure is shown in the last row of Fig. 2.

Having established that our method is capable of recovering ground-truth material parameters, we now show

[2] The interested reader may find the default values in the *constants.py* file of our repository.

[3] We expect the use of this tool for academic purposes. The design and HCI related topics are beyond the scope of this paper.

Case	Params.	Init.	Opt. (succ.)	Ref.	Opt. (unsuc.)	Hyper. (succ.)	Hyper. (unsuc.)	Loss / Comp. time (succ.)	Loss / Comp. time (unsuc.)
Bunny (P)	c	0.01, 0.01, 0.01	0.415, 0.853, 0.999	0.412, 0.824, 0.999	0.397, 0.833, 0.999	*spp*=8, DBM, ε=0.0001	*spp*=16, MSE	0.0002 / 48.9s	0.0098 / 35.6s
	α	0.5	0.001	0.01	0.001				
	σ_s	0.02	0.904	0.9	0.912				
	η	1.54	1.495	1.49	1.747				
Dragon (P)	c	bitmap	bitmap	bitmap	bitmap	*spp*=16, DBM, $h(\sigma_s)_{min}$=0.11	*spp*=8, DBM	0.0015 / 231.3s	0.0528 / 180.9s
	α	0.7	0.001	0.001	0.135				
	σ_s	0.1	0.973	1	0.001				
	η	1.64	1.507	1.49	1.815				
Bunny (R)	c	0.01, 0.01, 0.01	0.469, 0.844, 0.999	0.412, 0.824, 0.999	0.599, 0.924, 0.999	*spp*=16, ε=0.0001, DBM, $h(\eta)_{max}$=1.55	*spp*=8, DBM	0.0005 / 237.3s	0.0025 / 154.6s
	α	0.5	0.014	0.01	0.001				
	σ_t	0.98	0.502	0.4	0.585				
	η	1.544	1.503	1.49	1.919				
Dragon (R)	c	volume	volume	volume	volume	*spp*=4, DBM, 2-stage	*spp*=16, DBM	0.0016 / 241.2s	0.0456 / 498.3
	α	0.7	0.011	0.01	0.307				
	σ_t	0.98	0.357	0.4	0.26				
	η	1.544	1.484	1.49	1.549				

Table 1: Results from Figs. 2, 4; DBM=Dual Buffer Method, MSE=Mean Squared Err., P=Principled BSDF, R=Rough dielectric BSDF, succ.=successful, unsuc.=unsuccessful.

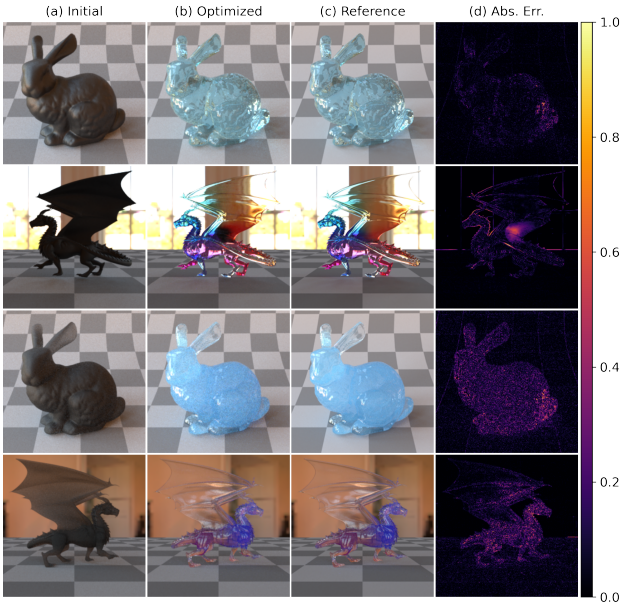


Figure 2: Our successful attempts for the Principled (first and second rows) and Rough dielectric (third and fourth rows) BSDF. For corresponding parameter values please refer to Table 1.

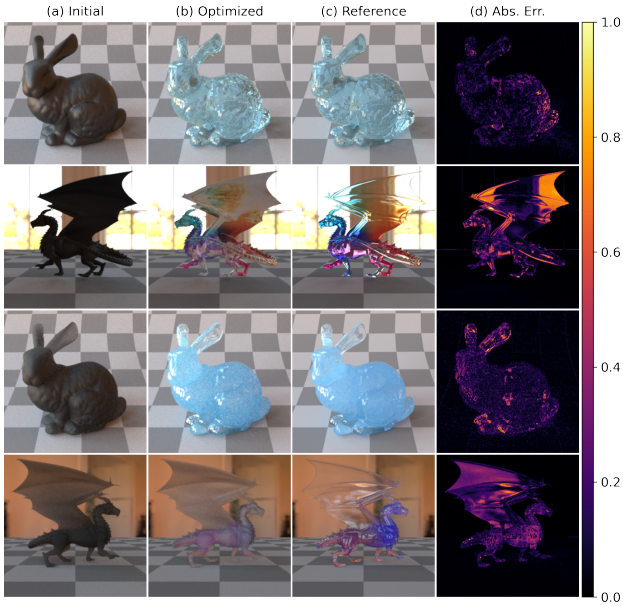


Figure 4: Our unsuccessful attempts for the Principled (first and second rows) and Rough dielectric (third and fourth rows) BSDF. For corresponding parameter values please refer to Table 1.

additional comparisons and discuss potential pitfalls during translucent material reconstruction in a short ablation study, Figs. 4, 3 and Table 1. We compare the dual buffer method (2×8 spp) to the single-image L2 error metric (1×16 spp) on the Stanford bunny using the Principled BSDF (first row in Fig. 2 and 4 respectively). Although the visual result might look acceptable, the numerical results (Table 1)—specifically for the η parameter—are not satisfactory. Note also the instability of the convergence compared to our successful attempt (Fig. 3).

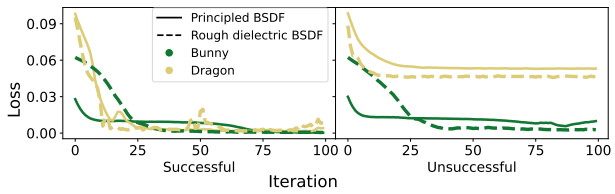


Figure 3: Convergence plots (synthetic-data) from (un)successful attempts. See also Figs. 2, 4.

On our second test case, the Dragon using a surface BSDF, we compare the optimization behaviour for different samples per pixel (16 vs. 8) value, and relaxing the minimum clamp value for the specular transmission (σ_s) parameter. As shown in Fig. 4 (second row), the lower sample count results in a visually worse appearance. As shown in Table 1 (second row), the numerical results are also not satisfactory.

Using the volumetric material model, we again test the influence of noise on the optimization procedure. In the

Case	Params.	Init.	Opt.	Hyper.	Loss / Comp. time
Alginate (G)	c	0.1 0.1 0.1	bitmap	spp=8,	0.0135/ 52.7s
	α	0.7	0.699	DBM, 2-stage	
	σ_s	0.1	0.429	$h(\alpha)_{max}$=0.7,	
	η	1.54	1.156	$h(\sigma_s)_{min}$=0.4	
Alginate (B)	c	0.1 0.1 0.1	bitmap	spp=8,	0.0033/ 51.1s
	α	0.7	0.501	DBM, 2-stage	
	σ_s	0.1	0.573	$h(\alpha)_{max}$=0.6,	
	η	1.54	1.397	$h(\sigma_s)_{min}$=0.5	
Birdy (R)	c	bitmap	bitmap	spp=4,	0.2711/ 347.8s
	α	0.5	0.246	MSE	
	η	1.5	1.435		
Birdy (SS)	c	bitmap	bitmap	spp=4,	0.1511/ 516.6s
	α	0.5	0.813	MSE,	
	η	1.5	1.172	lr:p=0.0003	
	p	ss	shape		

Table 2: Results for real-world data. G=Green, B=Blue, R=Reconstructed (using Metashape), SS=Scaled Sphere, lr=learning rate.

third row of Fig. 4, we observe visually acceptable results, whereas numerical results degrade when reducing the samples from 16 spp to 8. In this experiment, the η parameter initially increases (moving away from the expected reference value), and subsequently, the extinction coefficient (σ_t) remains not recovered accurately. Also note, although the resulting convergence plot and image (Figs. 3, 4) indicate a successful attempt, the numerical results, specifically the η parameter is far from the reference value (third row in Table 1). Consequently, one should note that—specifically in the case of real-world data, where the target value is unknown—one cannot fully rely on the resulting visual representations. Therefore for accurate results, one must also consider the plausibility of the resulting numerical values.

Finally, applying the volumetric material to the more complex Dragon scene in the fourth row of Fig. 4, we attempt to optimize all parameters of interest simultaneously (as opposed to the successful case presented earlier, where we optimize in two stages). Interestingly the material's roughness (α) fails to reduce sufficiently from an initially high value (fourth row in Table 1), resulting in a visually noticeable mismatch between the optimization result and the reference. Consequently, in some cases, we suggest separating the optimization procedure into two parts, to acquire acceptable results, as shown in the last row of Fig. 2.

5.2 Real-World applications

In this section, we perform material reconstruction from real-world data. In particular, we aim to acquire optical material properties for two novel alginate specimens [18]. As these alginate specimens are relatively thin, we use the *Principled* BSDF [3, 9], which has proved easier to optimize in the test cases discussed previously.

Please note that we use a single for the alginate and multiple reference image(s) for the bird statue test cases. Furthermore, note that interreflections play a role in the alginate test cases, albeit we kindly remind that they were photographed in the material scanner without the influence of external objects or light sources. On the other hand, interreflections do not play a role in the bird statue experiments. Note also in the bird statue experiments we specifically use Mitsuba's constant emitter as the primary light source.

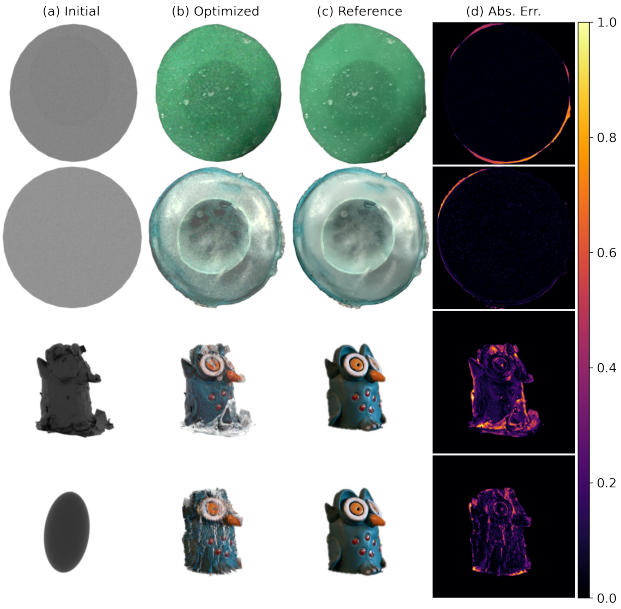


Figure 5: Results from alginate [18] materials (1. and 2. row) and a bird statue (3. and 4. row). For corresponding parameter values please refer to Table 2.

Figures 5, 6 shows results for both (green and blue) translucent alginate specimens. We again employ the strategy to split the optimization into two parts for both results.

We first only allow one constant RGB colour, and in the second stage extend the optimization to a bitmap texture image. We obtain visually acceptable results, even though the virtual geometry is not perfectly matched to the real-world images, resulting in errors along the outer edge of the specimens, seen in the absolute error images in Fig. 5. Table 2 summarizes the numerical results corresponding to Fig. 5.

For both cases, we apply specific minimum and maximum clamp values for the α and σ_s parameters. Without these additional constraints, we obtained visually identical results, albeit with physically unreasonable parameter values. Note also how the two-stage strategy is noticeable in the convergence plots (Fig. 6). For both alginate materials, we initialize an RGB texture with the previously optimized base color (c) at the start of the second stage (iteration 50). In the second stage of the optimization, we also observe a

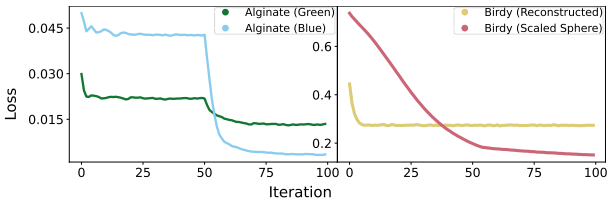


Figure 6: Convergence plots from real-world data.

noticeable decrease in loss. Consequently, the texture plays a significant role in representing the object as it contains a substantial amount of information.

Finally, in Fig. 5 and Table 2, we also show the results of a more complex opaque object. In Fig. 5, the third row showcases results utilizing the geometry obtained through Metashape [1], where only the c, η, and α parameters of the object were optimized. The fourth row features results using a (scaled) sphere object, with optimization of c, α, η, and vertex positions (p) parameters. Note that while we optimize the p parameter of the scaled sphere object, we use a box constraint, namely the bounding box value of the reconstructed object (third row in Fig. 5).

In conclusion, we acknowledge the complexity of real-world material and geometry reconstruction and emphasize the importance of a comprehensive data acquisition process that accounts for various factors, including lighting, image acquisition, object and light positions, and geometry reconstruction. However, as shown in Fig. 1, with accurate measurements, and thorough experimentation, material and geometry reconstruction can be successfully achieved.

6 Conclusion

In this paper, we describe a material reconstruction pipeline using the Mitsuba 3 differentiable renderer. Our software tool is capable of reconstructing material properties from a scene description file and multiple images. We focused on translucent material reconstruction, which we validated using synthetic data and demonstrated on real-world specimens. We tested our approach on both surface-only *Principled* BSDF, as well as a volumetric *Rough dielectric* BSDF with homogeneous participating media.

A difficulty we noted was the scene and image acquisition complexity. Regarding translucent materials, we observed that the noise introduced by MC sampling affected the parameter estimation procedure. Employing the dual buffer method noticeably improved our results. Our analysis showed that certain parameters, like the index of refraction, created discontinuities in scattering behaviour and made it difficult to achieve accurate convergence. In some cases, we found that two-stage optimization was necessary. When optimizing a texture for a material's albedo, highly localized parameters can have a stronger impact than global material parameters, which can lead to inaccurate reconstruction. Our findings showed that additional constraints, such as imposing a max. clamp value on some parameters, were necessary for certain scenarios.

In the future, we aim to improve our geometric modelling and image acquisition procedures, which will allow for a more accurate representation of physics-based reality. Based on the evidence obtained from our results, we conclude that our approach will be beneficial for different translucent material reconstruction tasks in various applications.

Acknowledgements

We thank Prof. Peter Ferschin (TU Wien) and his former master's student Cheng Shi for providing the material scanner. We thank Prof. Stavric and colleagues at TU Graz for the alginate samples. We also thank Prof. Torsten Möller (University of Vienna), for his help during the topic discovery phase. Lastly, thanks to the Mitsuba team at EPFL in Switzerland, which provided an incredible tool for this author and the whole computer graphics community.

References

[1] AgiSoft. Metashape, 2022. `https://www.agisoft.com/`.

[2] James F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192–198, jul 1977.

[3] Brent Burley. Physically-based shading at disney. *ACM Trans. Graph. (SIGGRAPH)*, 2012.

[4] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation.

[5] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. In *Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '81, page 307–316, New York, NY, USA, 1981. Association for Computing Machinery.

[6] Delatronic. Dragon. `https://blendswap.com/blend/15891`, 21.08.2015.

[7] Xi Deng, Fujun Luan, Bruce Walter, Kavita Bala, and Steve Marschner. Reconstructing translucent objects using differentiable rendering. In *ACM SIGGRAPH 2022 Conference Proc.*

[8] Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. Inverse volume rendering with material dictionaries. *ACM Trans. Graph.*, 32(6), nov 2013.

[9] Stephen Hill, Stephen McAuley, Brent Burley, Danny Chan, Luca Fascione, Michał Iwanicki, Naty Hoffman, Wenzel Jakob, David Neubelt, Angelo Pesce, and Matt Pettineo. Physically based shading in theory and practice. In *ACM SIGGRAPH 2015 Courses*.

[10] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[11] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. Mitsuba 3 Renderer, 2022. `https://mitsuba-renderer.org`.

[12] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, aug 1986.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[14] Chris Lattner and Vikram Adve. LLVM: A compilation framework for lifelong program analysis and transformation. pages 75–88, Mar 2004.

[15] Jason Lawrence, Aner Ben-Artzi, Christopher DeCoro, Wojciech Matusik, Hanspeter Pfister, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. *ACM Trans. Graph.*, 25(3):735–745, jul 2006.

[16] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph.*, 37(6), dec 2018.

[17] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.*, 38(6), 2019.

[18] Ivan Marjanovic, Elizabeta Samec, Hana Vasatko, and Milena Stavric. Alginate in architecture: An experimental approach to the new sustainable building material. In *Art and Science Applied: Experience and Vision*, volume 2 of *SmartArt*, chapter 21, pages 394–406. 2022.

[19] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 22(3):759–769, jul 2003.

[20] Baptiste Nicolet. Mitsuba blender add-on. `https://github.com/mitsuba-renderer/mitsuba-blender`, 2022.

[21] Merlin Nimier-David, Thomas Müller, Alexander Keller, and Wenzel Jakob. Unbiased inverse volume rendering with differential trackers. *ACM Trans. Graph.*, 41(4), jul 2022.

[22] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *ACM Trans. Graph.*, 39(4), 2020.

[23] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.

[24] Matt Phar. Let's stop calling it ggx. `https://pharr.org/matt/blog/2022/05/06/trowbridge-reitz`, 2023. [Online; accessed 22-February-2023].

[25] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation (3rd ed.)*. 3rd edition, oct 2016.

[26] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975.

[27] Peter Shirley and Steve Marschner. *Fundamentals of Computer Graphics*. 3rd edition, 2009.

[28] Tanaboon Tongbuasirilai, Jonas Unger, Joel Kronander, and Murat Kurt. Compact and intuitive data-driven brdf models. *The Visual Computer*, 36(4):855–872, 2019.

[29] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces∗. *J. Opt. Soc. Am.*, 57(9):1105–1114, Sep 1967.

[30] T. S. Trowbridge and K. P. Reitz. Average irregularity representation of a rough surface for ray reflection. *J. Opt. Soc. Am.*, 65(5):531–536, May 1975.

[31] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proc. SIGGRAPH '94*, page 311–318. ACM, 1994.

[32] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162.

[33] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Path replay backpropagation: Differentiating light paths using constant memory and linear time. *ACM Trans. Graph.*, 40(4):108:1–108:14, 2021.

[34] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proc. of the 18th EGSR'07*, page 195–206.

[35] Jingjie Yang and Shuangjiu Xiao. An inverse rendering approach for heterogeneous translucent materials. In *Proc. of the 15th ACM SIGGRAPH*, page 79–88.

[36] Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. Monte carlo estimators for differential light transport. *ACM Trans. Graph.*, 40(4), 2021.

[37] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM Trans. Graph.*, 39(4):143:1–143:19, 2020.

[38] Shuang Zhao, Wenzel Jakob, and Tzu-Mao Li. Physics-based differentiable rendering: From theory to implementation. In *ACM SIGGRAPH 2020 Courses*.

Visualization

Scatterplot Visualization of Hierarchically Clustered Data Points

Daniel Gruncl*
Supervised by: Ing. Ladislav Čmolík, Ph.D.†

Faculty of Electrical Engineering
Czech Technical University in Prague

Abstract

The focus of this paper is real-time visualization of and interaction with scatterplots with hundreds of thousands of data points, where the points are organized into a hierarchy of clusters. We present a technique that automatically selects a color palette for the clusters of the selected level. The level of the cluster hierarchy and the color palette are dynamically adjusted by zooming the scatterplot. Furthermore, the technique improves visibility of the displayed clusters by reducing occlusion of the overlapping clusters. We demonstrate the visualization technique using two real medical datasets containing 2D coordinates of hundreds of thousands points.

Keywords: hierarchical data, cluster visualization, scatterplot, dynamic color palette

1 Introduction

Applications of different clustering algorithms in biomedicine play a key role in advancement of data analysis. Namely, protein-protein interactions (PPI)[9] and protein expression, genomic sequence analysis, MRI image analysis etc., require different cluster analysis and assumptions. Hierarchical clustering as a method of cluster analysis aims to build a hierarchy of similar clusters to identify informative natural clusters of observations, adds additional complexity to this problem. The visualization aims to present the properties of such datasets, that is the hierarchical structure, the density of the clusters and their mutual overlaps as well. This is important for physicians to distinguish substantial overlap in diseases spectrum [1].

The clustering algorithms can be divided into hierarchical and partitioning. Hierarchical clustering algorithms can be subdivided into agglomerative (bottom-up clustering) and divisive (top-down), which perform recursive partitioning. Since the boundaries of the clusters cannot be objectively defined, hundreds of clustering algorithms have been proposed, each with different priorities. Thus, it cannot be said which algorithms are better or worse, as the algorithm's performance is often dependent on the characteristics of the information demanded as well as on the dataset itself.

To present the data, reduction from n-dimensional to 2D or 3D space is needed. Based on the deformation of space caused by the dimension reduction, the dimension reduction algorithms can be divided into linear, nonlinear, and those that have been implemented in both linear and nonlinear variants. PCA is a commonly used linear method that flattens the data along axes of minimal variance. t-SNE, a nonlinear method, aims to separate clusters in the data and avoid overlap of clusters of different categories, which may distort the data as the clusters might have overlapped in the original data. MDS has been implemented both as linear and nonlinear. This method reduces dimension while minimizing distortion of mutual distances between data points. An overview of the clustering and dimension reduction algorithms is provided by Wenskovitch et al. [10].

In this paper, we present our progress on the scatterplot visualization of large datasets of medical observations represented as n-dimensional data points organized into a hierarchy of clusters. Hierarchy is given as a tree, whose leaves contain points in 2D space. The hierarchy subdivides data points into clusters based on their mutual proximity. At the same time, some of the nodes in the hierarchy maintain the assignment of all their child nodes to a certain population. These nodes are disjunctive and provide complete coverage, meaning that every point in the data set is associated with exactly one population. Therefore, while the nodes of the hierarchy signify spatial proximity and clustering in the data points, some of the nodes also categorize the clusters into populations. Overlaps of clusters of different populations signify interactions between the populations.

The challenges of visualizing such observations are the scale of the data, namely the high count of points, causing heavy overdraw, and the high number of categories, which is only amplified by the hierarchical structure of clusters. The next challenge is the overlap of points of different categories, which makes the separation of points difficult. The characteristics of the data also rule out the utilization of position as a visual channel, as the data are represented as a set of hundreds of thousands points, whose original n-dimensional position is projected into 2D using dimension reduction techniques (e.g., PCA, NMF, t-SNE). This leaves us to separate clusters only by color since high point

*gruncdan@fel.cvut.cz
†cmolikl@fel.cvut.cz

count limits utilization of visual channels such as size and shape of points as well. We propose a visualization technique that allows for the analysis of such data. The proposed approach has three main contributions:

1. Real-time visualization rendering, allowing interactive close-up examination of visualized data.
2. Color separability of clusters and their identification in the hierarchy.
3. Identification of cluster density and their overlaps.

2 State-of-the-art

The overdrawing issue can be reduced by data abstraction techniques, such as binning, contouring the density function (that is, converting dense parts of the clusters into areas while outliers are displayed as points), or subsampling of the data.

Heimerl et al. [5] designed a technique of binning data points into a honeycomb grid, with each cell containing a bar graph or pie chart representing the distribution of individual categories in the cells. The background of the cell can also be saturated to communicate the density of the data points. However, the often complex structure of clusters in our data and rapid changes in cluster density, would require a high number of cells to keep the visualization representative. Breaking continuous areas of single color into a number of small icons, coupled with the much higher number of categories than Heimerl et al. [5] planned for, would lead to visual clutter and decrease the discernability of categories significantly.

Chen et al. [4] in their article demonstrate subsampling by sampling a density function, reducing overdrawing while preserving density information and relative representation of data categories compared to original data. Visualization by contouring the density is also provided, but overlapping colored areas causes color mixing, thereby significantly reducing the discernability of colors while losing density information of as well. However, sampling a density function seems feasible and may be employed in further developments of our work.

The color separability of many categories can be improved by using a dynamic color palette, which takes advantage of situations where only some categories have significant representation on the screen. Such a technique was developed by Waldin et al. [8]. The technique, named Chameleon, also provides hierarchical subdivision of the color space, which too is desired for our purposes.

Chameleon is designed for coloring the internal structures of viruses and cells. Here, at the highest level of the hierarchy are the structures of the virus, such as the lipid envelope and the capsid, in which the individual proteins can be distinguished after zooming in, as well as the domains and the atomic structure of the proteins. When gradually zooming in, only one or a few structures from higher levels of the hierarchy can fit on the screen, so there is no need to divide the color space among all of them but only among the visible ones. Thus, for each structure on a higher level, a larger part of the color space can be allocated, which can be further divided to be allocated to lower structures, as can be seen in Figure 1.

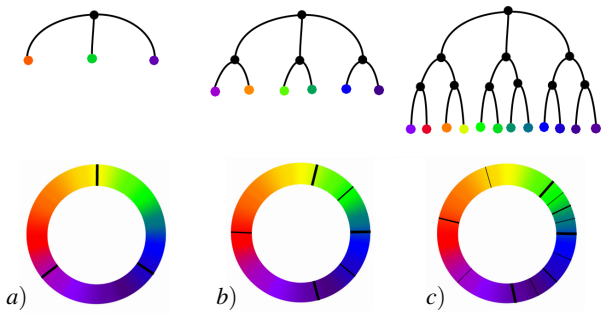


Figure 1: Hierarchical subdivision of color when zooming in on certain cluster. a) Color wedges are allocated only for the top level of hierarchy. b), c) With increased zoom, lower levels of hierarchy are visualized with distinct colors. The size of the color wedges corresponds to the hypothetical relative representation of individual clusters and subclusters on the screen.

3 Our Approach

We have decided to visualize the data as-is, without abstraction or resampling. A dynamic color palette with hierarchical coloring is employed to improve cluster discernabilty, while rendering points with transparency will allow for visualization of density and overlaps of the clusters. The visualization is GPU accelerated to achieve interactive real-time rendering.

3.1 Dynamic color palette

The method of coloring data points is based on the dynamic color palette developed by Waldin et al. [8], but with regards to the different characteristics of the data, several modifications have been made. The first modification is given by the fact that the data are individual points, not forming distinguishable continuous objects such as proteins. Therefore, it is impossible for the color spaces of individual hierarchy nodes to overlap, thereby reducing the space available to nodes at lower levels of the hierarchy. Furthermore, the number of hierarchy levels can be greater than it is in the case of visualizing a virus or cells. Therefore, the dynamic palette method [8] is applied at all levels of the hierarchy, except for the last one, where we do not have to consider the needs of any subclusters, and thus. maintaining color discrenibility remains the only priority. The color space is thus – in correspondence to the hierarchy – recursively divided into smaller and smaller sections (Figure 1).

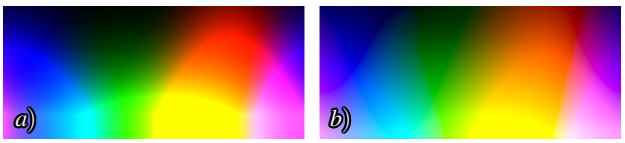


Figure 2: a) HCL implementation by Zeileis et al. [11]. b) HCL implementation from Catalano framework [3] satisfies the property of isoluminance better, notably in the dark green part of the spectra.

To prevent one dominant cluster from taking up all the color space, Waldin et al. [8] have imposed limit on the maximal size of the color wedge allocated to a single cluster. This way, the maximal size of one single color wedge at the k-th level of the hierarchy is a^k, where a, ranging from 0 to 1, is the size limit. To maintain the distinguishability of colors at lower levels of the hierarchy, the size limit a was set to a value higher than recommended by Waldin et al., namely 0.75 instead of 0.5. Also, because the number of subclusters may vary significantly across clusters, the size limit may be too high for some clusters with only a few subclusters. Thus, we have imposed an additional size limit at the top level of the hierarchy. The limit is the number of descendants n_i of a i-th cluster at the top level of the hierarchy multiplied by an appropriately chosen constant 0.08. In conclusion, the maximal size for cluster at k-th level of the hierarchy being descendant of i-th top-level cluster is

$$min(0.08n_i, 0.75) \cdot 0.75^{k-1}. \quad (1)$$

The original method sets size of color wedge allocated to individual clusters proportionally to the number of subclusters visible on the screen. In our case, even at a high level of zoom, most subclusters are present on the screen due to the wide scattering of the clusters. This results in almost no color space being released when zooming. Therefore, we have decided to size the color wedge proportionally to the number of points in the given cluster located on the screen. Allocating by the number of points is very aggressive, which achieves a good distinction of points according to their belonging in the hierarchy. On the other hand, in certain situations, the assigned colors may change significantly when the view changes, possibly confusing the user. The limits imposed on the maximal size of color wedges reduced this artifact. Other applicable methods are normalizing the number of points in individual clusters, weighting by the square root of the number of plotted points, or blending uniform and weighted color allocation.

3.2 Color model

HCL is a family of color models having channels hue, chroma and luminance. HCL models are isoluminant and perceptually uniform and thus often used in computer visualizations [8][4][7]. We have used the implementation of the HCL color model developed by Zeilies et al. [11],

Figure 3: Top line: coloring using hue only. Bottom line: coloring using hue and brightness.

which contains HCL-to-RGB conversion chain. Part of the chain is XYZ-to-RGB conversion, where we have used Catalano's implementation [3] instead, since in our experience it better satisfies the property of isoluminance (see Figure 2). The HCL color model is implemented as CIELAB with axes $A*$ and $B*$ transformed to polar coordinates. The hue channel is used to differentiate the clusters, as it is the only one that does not visually imply sorting.

But, since the first level of hierarchy of visualized data contains over twenty nodes, comparing to the data, for which the dynamic color palette developed by Waldin et al. [8] was designed, where first level of hierarchy only contains up to ten nodes, the hue channel alone proved to be insufficient to reliably distinguish clusters of data points (see Figure 3). This is also due the used model CIELAB not being perfectly hue uniform [6] and in the area of the blue hues, distance between distinguishable hues is greater than it is for the other color hues.

Therefore, two channels, hue and luminance, are used in coloring the first level of the hierarchy. The chroma channel is not utilized because it interferes with the hue. Hue is used as the primary channel, where each group uses a different hue. Luminance is used as a complementary channel to avoid blending in with the background. Because the technique of color optimization used by Waldin et al. is only designed for optimizing color palette in one dimension, a low number (five) of fixed brightness levels is used, which are assigned to successive nodes in the cluster hierarchy. The order of the used luminance levels is permuted so that adjacent sections in the hue channel are not assigned adjacent luminance levels, thereby increasing the tonal distance of the colors assigned to successive clusters. All nodes in the hierarchy are colored using the same luminance level as their parent. The use of two channels improved the separability of clusters, as can be seen in Figure 3.

3.3 Data density

Visual encoding using the luminance channel excludes the rendering of points using transparency, because in the case of a white[1] (achromatic in general) background, luminance interferes with transparency, that is, the difference in the brightness of the colors of individual clusters is not discernible; darker clusters only look sparser. Also, when transparency is used, colors will be mixed, which will worsen the distinguishability of individual clusters, and new colors may even appear. Therefore, in order to

[1] Visualization uses black background by default, as it makes clusters stand out better, white background mode was added for print.

preserve the distinctness of the individual clusters, the rendering is performed without transparency.

However, this introduces two new problems: the dependence of the visualization results on the rendering order (Figure 4a, b), which distorts the information about clusters overlap and loss of the data density information.

The rendering order issue was resolved by performing a depth test and assigning a random depth to each point. This way, the overlap of the groups depends on their density; a denser group has a statistically greater chance that out of the number of points falling within one pixel on the screen, its point will have the smallest depth of all and thus will be displayed over the others. This technique also achieves proper visualization of cluster overlaps without mixing colors. Figure 4 compares the differences on two overlapping clusters, blue one linearly falling off in down direction and brown one linearly falling off in left direction.

We will now demonstrate that the probability of a cluster being on top in a given pixel corresponds to its relative representation at the pixel. The depth of cluster X at pixel p contributing to the pixel with l samples is the minimum of l samples of uniform random variable with range 0 to 1. The probability of one such sample being lower than $x \in \langle 0;1 \rangle$ is x. Therefore, probability of l samples being lower than x is x^l, which is the CDF function of cluster depth at the pixel. Thus, when cluster X is contributing to a pixel with l samples, and all other clusters (Y) combined contribute with k samples, probability of X being on the top is calculated as difference between two random variables with CDF functions

$$F_X(x) = x^l, \tag{2}$$

$$F_Y(y) = y^k. \tag{3}$$

The sum of two random variable is calculated as a convolution. The formula for sum can be modified to difference as such:

$$P(X+Y \leq z) = \int_{-\infty}^{\infty} f_X(x) F_Y(z-x) dx, \tag{4}$$

$$P(X-Y \leq z) = \int_{-\infty}^{\infty} F'_X(x) F_Y(z+x) dx, \tag{5}$$

although in case of difference, the formula is not commutative anymore. The formula can be further modified to

$$P(X \leq Y) = \int_0^1 l x^{l-1} x^k dx. \tag{6}$$

The result of the integral is

$$P(X \leq Y) = \frac{l}{l+k}. \tag{7}$$

As we can see now, when $k+l$ is normalized to 100, the probability of X being on the top is $l\%$.

As for visualizing the data density, the transparency settings of the plotted points have been made available to the user so that s/he can view the density if necessary, even though this distorts the information about the points belonging to groups. In this mode, it is necessary to disable the depth test so that no points are neglected and to disable the use of the luminance channel, as it would distort the perceptual transparency.

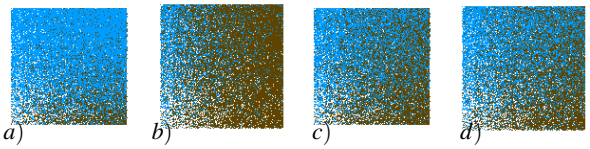


Figure 4: a), c) Blue over brown. b), d) Brown over blue. a), b) No depth test. c), d) Depth test with randomized point depth.

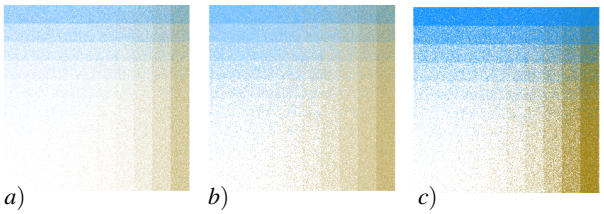


Figure 5: a) Single-pass rendering. b) Two-pass rendering. c) Single-pass with lower transparency causes order-dependent result. Unlike in Figure 4, an exponential falloff is used here.

When visualizing data density using transparency, we find that for high transparency settings, very sparse areas where there are only a few points, surrounded by a black background, are hard to see, and at low transparency settings, the difference in density between the rims and the centers of the clusters cannot be discerned. We cannot increase transparency, as due to alpha blending, clusters drawn last would cover earlier drawn clusters, as can be seen in Figure 5c, top right, where the blue cluster incorrectly covers the brown cluster. Therefore, a non-linear dependence between transparency and density is needed. We implemented this as rendering with transparency in two passes (comparison in supplementary video 2) .[2] At first, all points are rendered with a transparency lower than the user set, and on each screen pixel, only the first point that fell into the pixel is rendered. Then all the data is re-rendered over the previously drawn data using the transparency level set by the user. In both passes, the points are drawn in the same order so that the distortion given by color mixing is the same in both passes.

3.4 Color assignment order

Looking at Figure 6, we notice that some very close or overlapping clusters of points are colored in similar colors, thus merging and creating the impression of a single

[2] https://drive.google.com/drive/folders/1yLt3SFyOHZMTO00O3J20oP4wAha0KPrk?usp=sharing

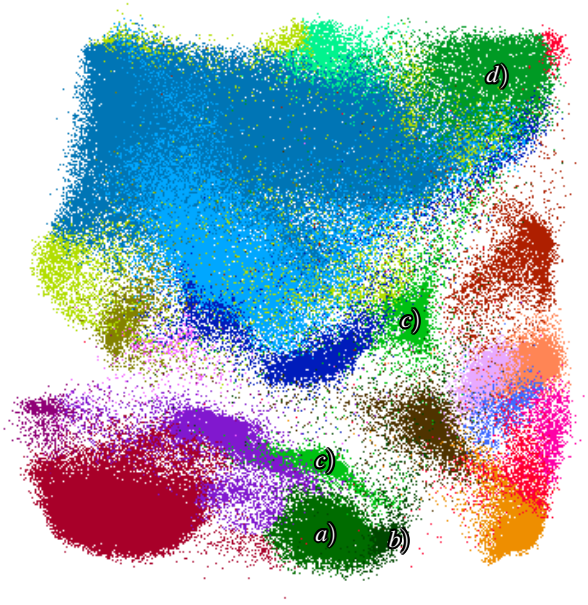


Figure 6: a),b) Co-located groups colored with a similar shade. c) Disjointed cluster. c) and d) might create impression of one disjointed cluster.

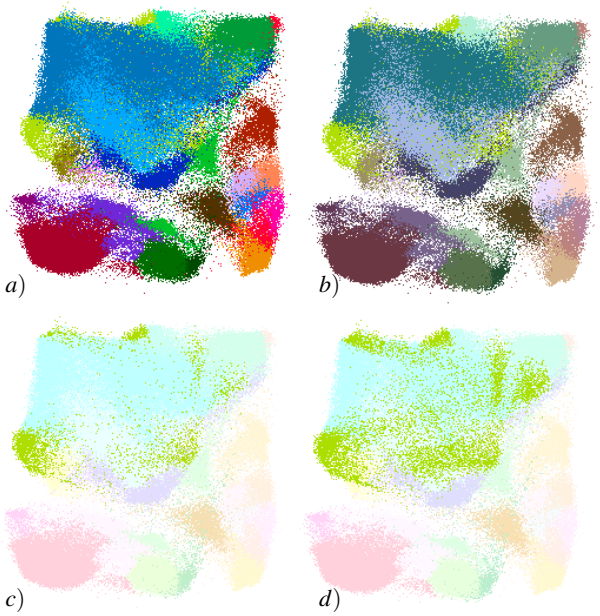


Figure 7: A comparison of context highlighting methods: a) No context highlighting. b) Highlighting with chroma. c),d) Highlighting with chroma and luminance. d) Highlighting cluster is brought to front, thus not overlapped by any cluster.

group. For example, the two clusters, a) and b), are colored in a similar shade of green. An easy distinction between those two clusters could be achieved by permuting the order of colors so that the co-located clusters are colored by colors distant in the color space, as proposed by Lu et al. [7]. But this may create the problem of two distant clusters dyed with a similar color creating the impression of one disjointed cluster, as shown in Figure 6 by clusters c) and d). Similar shades of color are easier to recognize if placed close to each other. Thus, we decided not to permute the color spaces.

3.5 Context selection

Waldin et al. [8] mention improving the discernability of user-selected area of interest by desaturation of the surroundings. In our case, the context selection aims to give a full overview of clusters of the hierarchy (see supplementary video 2, time 0:04) . However, since the mere desaturation of the surroundings turned out to be an insufficiently discriminating channel (see Figure 7b), the luminance channel was additionally used to brighten of the surroundings. Desaturation needs to be used together with brightening, as brightening alone makes colors appear more saturated. Saturated background is undesirable, as it draws users' attention away from the selected context. Desaturation and brightening are already sufficiently distinctive, but there remains the problem of covering the highlighted cluster with a denser cluster (7c), which can be solved by moving the highlighted cluster to the foreground, but this results in the suppression of the context, and we lose information about the background of the marked cluster (7d). The user is given the option to switch between these two methods since it cannot be conclusively said which one is better. Comparing to 7a, we can now see that the context selection gives us an overview of the cluster, which would otherwise remain unknown.

4 Results

An application visualizing clusters of real measured hierarchically grouped data in real-time was created. Running on AMD Ryzen™ 5 3500U APU with integrated Radeon™ Vega 8 GPU, we have measured 30 FPS on a dataset with 250,000 points and 68 leaf nodes in the hierarchy and 45 to 60 FPS on a dataset with 800,000 points and 50 leaf nodes, which shows that the number of categories has a higher performance impact than the number of points. This is due to synchronization between CPU and GPU, caused by querying the number of drawn points after each node drawn. Therefore, smooth running of the visualization on low-end hardware might require reworking the querying to be made for all nodes at once and thus making only one query per frame.

Due to the unavailability of hierarchy data, cluster data were additionally divided into subclusters using the Kmeans++ method [2], resulting in three levels of hierarchy. The application is written in the Java language and utilizes GPU accelerated rendering trough the JOGL library, which is a Java binding for OpenGL.

The visualization allows to distinguish individual clus-

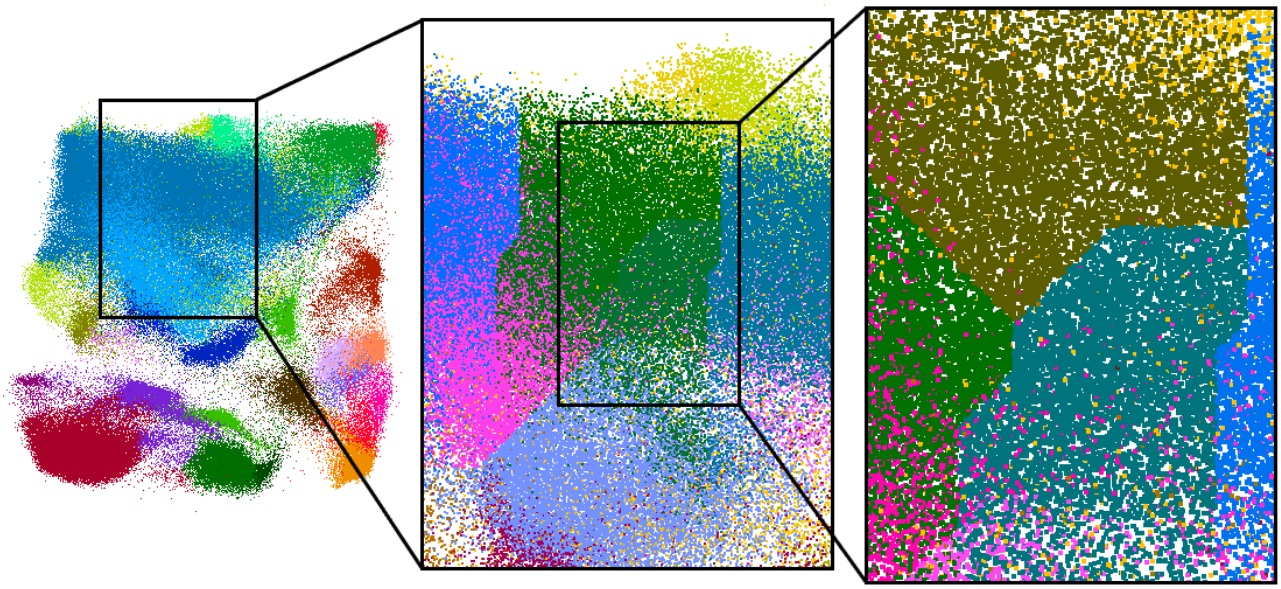


Figure 8: Hierarchical coloring.

ters in a multilevel hierarchy, as shown in Figure 8, where zoom at three consecutive levels is depicted. Individual clusters at any level of the hierarchy can be highlighted (Figures 9, 7) by double-clicking on the cluster of interest. To make selection easier and to allow the selection of an entire subtree or of a cluster overlapped by other clusters, a visualization of the hierarchy as a tree was added to the visualization (Figure 9). The selection is done by double-clicking on a node in the tree.

Figures 10b and 11b demonstrate improvements achieved by two-channel coloring and nonlinear transparency. A comparison of these two figures also shows differences between discernability-based visualization (Figure 10) and density-based visualization (Figure 11). The density based visualization allows for precise perception of the shapes of the two large blue clusters at the top left of the visualizations, but at a cost of lower discernability of the clusters' category.

4.1 Limitations

An unsolved issue is the color inconsistency of the dynamic color palette at high zoom. In most cases, the color difference between adjacent subclusters is satisfying, but typically in very scattered clusters, there are cases where the color shades of the subclusters are very far from each other, and significant color changes occur when the view is moved or zoomed in or out (supplementary video 1) . In a related issue, color differences between clusters having a relatively low number of points are small, making distinguishing them difficult, as can be seen in Figure 6, clusters c) and d). However, context highlighting can be used to resolve this issue.

Since only the division of groups at the population level of the hierarchy was available at the time of design, it was necessary to extend the hierarchy artificially for testing purposes. To subdivide the available data into a multi-level hierarchy, the K-means++ [2] method was used, which is a heuristic hierarchical clustering method often used in statistical data analysis.

5 Conclusion

A visualisation of hierarchically clustered multi-categorical data of medical measurements was created. Such a visualisation is important for medical diagnoses and development. The visualised properties are the density of the clusters, their mutual overlaps and the hierarchical structure.

The nodes of the hierarchy are separated by color with color resolution of twenty-six clusters. Due to the large number of categories, it was necessary to use two color channels, hue and brightness, to achieve resolution. After zooming in, up to seventy clusters can be distinguished with the use of the dynamic color palette. It takes advantage of the fact that the vast majority of these seventy clusters are not visible on the screen when zoomed in. When zooming in, it was necessary to increase the size of the rendered points, which is proportional to the degree of zoom, in order to maintain color resolution. In the application, it is possible to specify at which zoom levels coloring with different colors should be activated at individual depths of the hierarchy.

Due to the mutual exclusion of cluster discriminability and density visualization, two visualization modes were created. Density-focused mode renders points with transparency, while discriminability-focused mode renders fully opaque points. Both modes respect the mu-

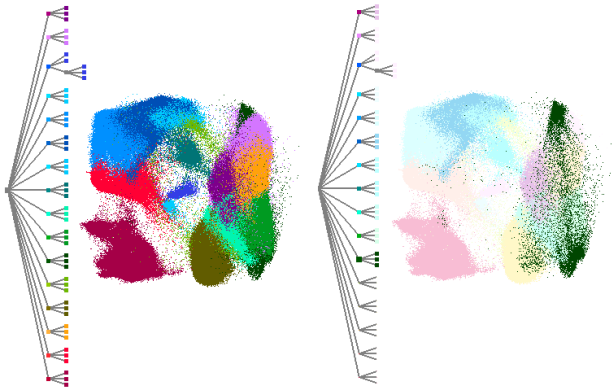


Figure 9: Context selection and hierarchy visualization. The selected hierarchy can also be seen it the tree.

tual overlap of the clusters, i.e., that where several clusters overlap, the color distribution corresponds to the density of the individual clusters. In the case of opaque rendering of points, the overlap visualization was achieved by activating the depth test and randomizing of the point depth.

5.1 Further developments

Visualisation of the hierarchy can be augmented to display the relative frequency of points in individual clusters by scaling individual nodes in the hierarchy visualisation proportionately.

The dynamic color palette can be extended to two dimensions, that is both hue and luminance being assigned dynamically based on the situation. This would improve the utilization of the available color space and the discernability of the clusters as well.

A mentioned but unexplored method is to place labels in the visualization, which would make it easier to distinguish and identify individual clusters. The labeling of the clusters needs to consider the density and overlapping of the clusters, as naive label placement would produce ambiguous or confusing labeling. Čmolík and Bittner [12] propose a method that evaluates places of possible label anchoring based on local opacity salience, overlap salience, and the distance from the edge of the labeled object.

The technique of subsampling the data might also be helpful in achieving the visualization of the density of clusters while maintaining the distinctness of the clusters. A suitable subsampling technique is described by Chen et al. [4], aiming specifically at improving the visualization of overlapping clusters in a multi-class scatterplot.

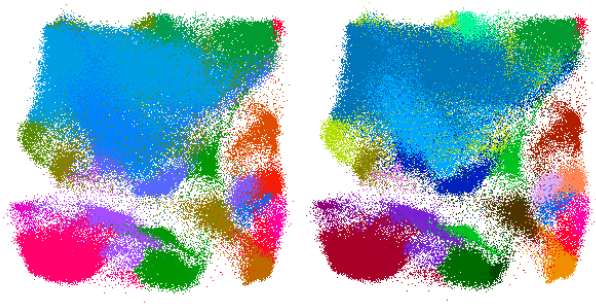


Figure 10: a) Coloring using hue only. b) Coloring using hue and brightness.

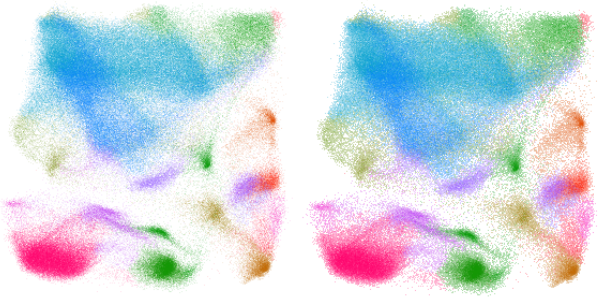


Figure 11: a) Single-pass rendering. b) Two-pass rendering.

References

[1] Hany Alashwal, Mohamed El Halaby, Jacob J Crouse, Areeg Abdalla, and Ahmed A Moustafa. The application of unsupervised clustering methods to alzheimer's disease. *Frontiers in Computational Neuroscience*, 13:31, 2019.

[2] Jason Altschuler. Github - jasonaltschuler: Improvements on classical kmeans clustering, 2013. https://github.com/JasonAltschuler/KMeansPlusPlus [online; accessed 2023-01-05].

[3] Diego Catalano, Robert Theis, and Yuri Pourre. Github - diego catalano/catalano-framework: Framework, 2013. https://github.com/DiegoCatalano [online; accessed 2023-01-05].

[4] Haidong Chen, Wei Chen, Honghui Mei, Zhiqi Liu, Kun Zhou, Weifeng Chen, Wentao Gu, and Kwan-Liu Ma. Visual abstraction and exploration of multi-class scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1683–1692, 2014.

[5] Florian Heimerl, Chih-Ching Chang, Alper Sarikaya, and Michael Gleicher. Visual designs for binned aggregation of multi-class scatterplots, 2018.

[6] Rolf G. Kuehni. Hue uniformity and the cielab space and color difference formula. *Color Research & Application*, 23(5):314–322, 1998.

[7] Kecheng Lu, Mi Feng, Xin Chen, Michael Sedlmair, Oliver Deussen, Dani Lischinski, Zhanglin Cheng,

and Yunhai Wang. Palettailor: Discriminable colorization for categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):475–484, 2021.

[8] Nicholas Waldin, Mathieu Muzic, Manuela Waldner, Eduard Gröller, David Goodsell, Autin Ludovic, and Ivan Viola. Chameleon. *Eurographics Workshop on Visual Computing for Biomedicine*, 2016, 09 2016.

[9] Jianxin Wang, Jun Ren, Min Li, and Fang-Xiang Wu. Identification of hierarchical and overlapping functional modules in ppi networks. *IEEE Transactions on NanoBioscience*, 11(4):386–393, 2012.

[10] John Wenskovitch, Ian Crandell, Naren Ramakrishnan, Leanna House, Scotland Leman, and Chris North. Towards a systematic combination of dimension reduction and clustering in visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):131–141, 2018.

[11] Achim Zeileis, Jason C. Fisher, Kurt Hornik, Ross Ihaka, Claire D. McWhite, Paul Murrell, Reto Stauffer, and Claus O. Wilke. colorspace: A toolbox for manipulating and assessing colors and palettes. *Journal of Statistical Software*, 96(1):1–49, 2020.

[12] Ladislav Čmolík and Jiří Bittner. Real-time external labeling of ghosted views. *IEEE Transactions on Visualization and Computer Graphics*, 25(7):2458–2470, 2019.

Interactive Visual Analysis of Anomalies in Simulations of Energy Flow

Fabrizia Bando-Bechtold*
Kresimir Matkovic (supervisor)†

VRVis Research Center, Vienna, Austria

Abstract

At a time when the demand for alternatives to gas and oil is steadily increasing, the automotive industry faces the challenge of designing vehicle systems, which are as efficient as possible. Vehicle system simulation is used for concept analysis, subsystem design, and virtual component integration. The high-level simulation model accurately depicts the energy flow between different vehicle parts over an extended period of time. The the resulting data is presented as a time-dependent edge-weighted graph. The analysis and exploration of such data is a tedious task.

We propose a novel approach for exploring and analyzing energy flow data at different levels of detail to assist engineers and guide them to the events of interest. We employ automatic anomaly detection techniques and propose intuitive navigation to time steps of interest. We also propose expandable cards or labels that depict current and overall data. Users can interactively and dynamically choose how much information is displayed on the labels by changing their level of detail. We provide four levels of detail, each giving more information than the previous level. The first level depicts the current amount of energy; the second level shows sparklines for the energy flow over the entire time interval; the third level shows a timeline of anomaly occurrences; level four shows the sparkline zoomed around the current time frame; finally, level five depicts expands the label to a large display of the previous four levels of detail of the flow data for the selected element.

The new approach is implemented as an interactive web application. We are currently evaluating it with domain experts. Since the initial feedback is very positive, we expect rapid adoption of the newly proposed approach by automotive industry professionals.

Keywords: Visual Analytics, Anomaly Detection, Flow Graph Visualization

*bechtold@vrvis.at
†matkovic@vrvis.at

1 Introduction

Creating and designing a vehicle is a difficult task in and of itself. Consumer concerns about the environment and the high operating costs of traditional fuel-powered vehicles exacerbate the problem even further. Engineers working in the automotive industry must therefore design systems for cars that are both cost-effective and environmentally responsible.

Once a system design is finished and before an actual vehicle is built, engineers want to know how the potential system performs during a drive. They want to collect and analyse data extracted from drives which are longer or shorter and mimic different geographic and environmental conditions. Driving on a hilly road or through snow, for instance, is different from driving on a highway in good weather. The car behaves differently in a traffic jam than it does on a smooth ride. The energy flowing between the system's components is calculated over time for each of these various settings. The analysis problem for hybrid cars becomes even more difficult, because some system components, like the battery, can be both sources and sinks of energy flow during a single driving simulation. The potentially large cardinality of the set of parameters alone makes the analysis task intimidating. Engineers want to collect this data for various driving cycles, which is not feasible to do from real life test drives. Therefore, in the design phase of a vehicle, when engineers carefully examine potential system component layouts, simulation plays a significant role.

A simulation is a technique that mimics a real-world system or process and tracks its evolution [2][17]. It is typically run on a computer and is based on a model - often mathematical - displaying key characteristics of a process. Simulations are frequently used when a real-world process, such as monitoring car system components during a test drive, is too complex and expensive to build or provide analytical solutions for. Computer simulation is also used in vehicle engineering to improve the aerodynamic properties of a component [12], the turbulent combustion system [20], the influence of tail structure on the rear field [13].

While simulations are frequently used in research and design, analyzing such data can be time-consuming and challenging, particularly when dealing with multivari-

ate time-dependent data. As a result, linear analysis of such time-dependent data takes a long time and is labor-intensive.

We propose an analysis system, that can reduce the engineers workload by conducting automatic anomaly detection on the data and visually guiding them to incidents, which require more thorough investigation. The goal of anomaly detection is to identify events that differ from the norm [15][16]. Anomaly detection is often accomplished by employing AI and training a model on a particular test data set. Our approach employs anomaly detection techniques, which are only dependent on the input data. This has the advantage of being applicable to any given time-series without the need for humans to identify anomalous occurrences beforehand. We use anomaly detection to direct the engineers' attention to specific spots in the simulated test drive. The engineer might focus on the time frame around the occurrence to assess whether or not the system is behaving strangely.

Research on the effectiveness of visualizations shows, that visualizations exploit a humans' cognitive capabilities and therefore increase the users performance for spotting anomalous behavior [14]. Visual Analytics, especially when guided, [6][18] supports humans in identifying patterns and points of interest, discover previously unknown connections, gaining fresh insight and making data-driven decisions. Interactivity can further enhance the users performance.

Interactive visualization has become a well-established tool to support engineers in the exploration and analysis of complex data. We, therefore, propose an interactive visualization approach to further support engineers in the analysis of energy flow data. In a collaboration with domain experts, we designed a novel visual analysis system to support engineers in their work flow. The new approach unifies automatic anomaly detection and interactive visualization to guide engineers to events of interest.

Our proposed system can be generalized as a flow graph. It consists of nodes, which represent the various car's components, and edges, which represent the energy flowing between the components. In order to cope with the complexity of the data, we abstract and give an overview of the data in carefully designed visualization elements. Additionally we provide the user with the option to increase the displayed information, by preparing various visualizations showing different details and only displaying them on demand. The detected anomalies are displayed in a timeline, giving a summary of the anomalies.

The summarized main contributions of this paper are:

1. interactive visualization approach for the analysis of energy flow data,
2. augmented and extended flow graph visualization
3. anomaly detection to guide users to incidents of interest.

2 Related Work

Our proposed application is closely related to the existing software **AVL CRUISE™ M**[1] [7] developed by **AVL List GmbH ("AVL")**. The software is used to design a car's system and then simualte diverse test runs. After the simulation, engineers can analyse the behaviour of the system by watching a replay, which shows a simple graph consisting of all the car's components as nodes and the flow of energy between them as edges. The current energy value of a component is showcased as two scalar values next to the component as incoming flow and outgoing flow, which is also visually encoded by the thickness of the edge. The dynamic flow is further depicted by huge arrows moving along the edge, with the arrow head facing the direction of the flow, which can change during the drive. Due to the redundant information incoming and outgoing labels, the screen gets easily cluttered. We plan to solve this by encoding all information in one label and decreasing the size and quantity of the arrows. The visualization itself is static and the user is unable to query further information. We plan to implement diverse interactions such as various playback modes, and interaction with and selection of the components. The biggest limitation of the original software, though, is the linear playback mode without any indication of where and when during the simulation incidents of interest happened. For this we plan extend the visualization with automatic anomaly detection and displaying them on a time line. This allows the user to only watch the important parts of the visualization.

Matković et al. [11] introduce diverse visualization techniques to process monitoring, namely history encoding, multi-instruments and level of detail. Additionally to displaying the current value, the authors encode values of the near past into their visualization (history encoding) by adapting a bar chart. Instead of encoding the current value as a bar, which takes up a whole column or row, they encode the value as a heavily saturated line. Like this, they can add more data points by adding lines and gradually decreasing the saturation for points further in the past. This feature could be useful for our application to show the recent change in energy flow to give the user an idea of how the values changed, especially around an anomaly. Matković et al. display several data values within one virtual instrument, to easily compare related values and quickly spot divergences (multi-instruments). Another advantage of this approach is that it saves screen space, allowing for showcasing a lot of information in a condensed and easy to grasp manner. We make use of this feature in our anomaly timeline (see Section 5) by displaying different types of anomalies as a stacked one-dimensional scatter chart. Focus and context approaches

[1] https://www.avl.com/cruise-m

coarsely represent the entire information in the available screen space and give different levels of detail for different degrees of interest. Users can see information with the highest degree of interest in the highest level of detail, and areas of low interest can be depicted as only a scalar value. We employ this focus and context approach for displaying condensed information of the flow of energy for each component. Users can manually set the level of detail for each component and see different visualizations, depending on the selected level.

Cambridge Intelligence developed a `JavaScript`-based toolkit for scalable timeline visualizations that reveal patterns in time data, mostly aimed at cyber security and fraud detection analysis. The main feature is a scalable timeline, which highlights events occurring between two entities. Its goal is to guide analysts in detecting patterns and abnormal behaviour in events. The graph consists of rows, each row dedicated to an entity. An event is encoded as a (directed) line between the two rows of the respective entities on a low zoom level and an aggreated heat map on a higher. The software can be extended to show a graph or network of the underlying data. Since the view of the graph and the timeline are linked, users can easily query subsets of the data. While the graph visualization seems powerful enough, it does not allow dynamic playback, or encoding aggregated information or statistics into the graph itself. The timeline is a powerful tool to investigate events, but it uses a lot of screen space. As our main focus is on the visualization of the graph, and only enhancing it with a timeline, this does not seem feasible. Though the aggregated timeline could be useful for future extensions of the **Energy Flow Explorer**. [5]

3 Energy Flow Data and Analysis Requirements

During a drive, a car is subjected to many different external influences. Temperature, weather, traffic and such affect the performance of the car's components and the whole system's behaviour. Testing and analysing the behaviour in the real world is not desirable due to unpredictable weather and difficult reproducibility. Furthermore, for the automotive industry simulations are vital to save costs and reduce time to market their products.

A driving simulation imitates the drive with a hybrid car from point A to point B under specific environmental circumstances, or with a given set of system parameters. For the simulation of our test data the software **AVL CRUISE™ M**[2][7] developed by **AVL List GmbH ("AVL")**[3] was used. It is a "multi-disciplinary vehicle system simulation tool", which allows for "Powertrain Concept Analysis","Control Function Development and Calibration", "Vehicle Simulation on Testbeds" as well as "Sub-system Analysis" [9]. The latter can be used to perform detailed design layouts and optimizations of subsystems. It serves as the basis for our proposed analysis tool, the **Energy Flow Explorer**.

The **AVL CRUISE™ M** software allows to simulate the flow of energy between the components of the hybrid car and is able to replay and show the amount of flow between components. Figure 1 depicts the simulation player of the software, where the "Mass flow [kg/s]" is distributed within a small test-set containing eight components: an energy source (Boundary 1) and a target (Boundary 2). The figure shows the direction and the amount of flow entering and leaving a component on a small label attached to each component. The view only allows for linear playback of the simulation and analysing the simulation at a specific point in time. The user has to watch the whole animation to analyse the simulation and is provided with only very limited information; no data aggregation or statistics are performed. The visualization also gives no indication of abnormal behaviour or the change of energy flow throughout the whole simulation.

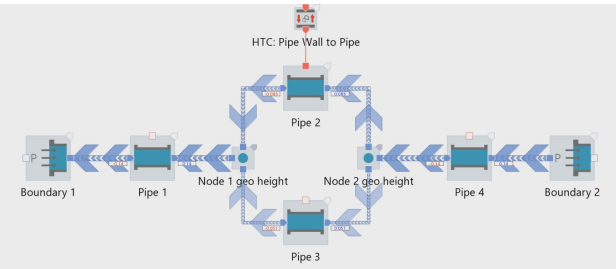


Figure 1: A frame captured from the replay of a small simulation using the software **AVL CRUISE™ M**. It shows the direction and amount of the energy flowing between components at the time of the capture.

The **Energy Flow Explorer** is an application with the goal to solve the previously stated limitations and provide better support for engineers from the automotive industry in their analysis work.

Before designing the architecture of the system, we thoroughly examined the analysis process together with domain experts and abstracted the following requirements:

- **R1 - Automatic Data Processing**: load multiple simulation results from different sources
- **R2 - Data Aggregation**: support different types of anomaly detection and data aggregation
- **R3 - Model Structure**: store system information in simple and easy to query format
- **R4 - Simulation Player**: show energy flow between components throughout simulation
- **R5 - Improved "Data-Ink Ratio"** [18]: improve data density and encoding in the visualization

[2]https://www.avl.com/cruise-m
[3]www.avl.com

- **R6 - Anomaly Visualization**: show anomalous behaviour in multiple levels of detail
- **R7 - Multi-Resolution Data Graphs** [18]: show different levels of detail on the data with overview first and more detail on demand

To conform to these tasks, the designed system has to fulfill the following criteria: read input data of different formats; have the possibility to perform data analysis techniques and data aggregation; perform queries on the data and the model itself; display various non-standard visualizations and provide user interaction. Since the system should support powerful data analysis as well as user interaction, a `client-server` architecture following the request-response model is employed:

1. *Data Handling & Analysis*: a python server is responsible for reading the source data, generating an easy-to-query model data structure, carrying out data analysis and responding to user queries (Requirements T1-T3).
2. *Data Visualization & User Interaction*: a JavaScript-based web-application displays the processed data in an interactive dashboard, where the user is able to freely manipulate and explore the data (Requirements T4-T7).

Section 4 describes the first part of the system *Data Handling & Analysis*, whereas the second part - *Data Visualization* - is described in section 5.

4 Data Handling & Analysis

Our application requires several types of input data. First, a graph model consisting of information about each component, such as physical properties, meta data and ports (connection between two components). Second, the used parameters for the simulation, which is required for selecting subsets in the data when comparing multiple simulations (see Section 6 - Future Work). Third, the actual flow data for each component for the whole simulation.6 - Future Work).)
We chose `Python`[4] due to its flexibility and vast data manipulation libraries. We employ a `Flask`[5] server as it enables easy creation of websites *Data Handling & Analysis*: Once the user selects a simulation case-set, the server is responsible for parsing the model data and generating the component-flow model before performing anomaly detection on each component and sending the processed data to the user.
Python provides multiple libraries for anomaly detection. We used PyCaret [1], which provides a flexible and extensive framework for anomaly detection of time-series. Here it is possible to choose from a multitude of machine learning based anomaly detection algorithms. We chose `Isolation Forest` or `IForest` [10], an unsupervised model. Using randomly chosen characteristics, an Isolation Forest processes randomly sub-sampled data in a tree structure. As they required more cuttings to separate, samples that travel further into the tree are less likely to include anomalies and samples that end up on shorter branches tend to be anomalies. The second approach for anomaly detection is a simple `Min-Max Threshold` model, where the user can set a percentile, at which data points below or above are marked as anomalous. Lastly, we chose to use the `Modified Z Score` [8]. A z-score in statistics indicates how many standard deviations a result deviates from the mean. However, unusually big or tiny data values can have an impact on z-scores, so using a modified z-score is a more reliable method of identifying outliers, as it is based on the median rather than the mean.

Engineers desire a system where they can analyze multiple simulation runs for the same component system but with different parameter settings at the same time and get a feeling for the special features of the system as well as its peculiarities and hidden correlations between components. These aspects were already considered for the design of the system architecture, but not yet fully realized at the publishing date of this paper.

The python server is split into the following components to perform these tasks:

- *DataFileParser*:
  The results generated by the **AVL CRUISE™ M** come in different formats - ranging from `.csv` and `.xml` to their own file format `.gid` - and vary greatly in their structure. The *DataFileParser* provides all necessary functionality of requirement **R1 - Automatic Data Processing** to retrieve data from all provided raw data files. Additionally it provides methods to persist and load the generated models and data to skip time-consuming processes such as performing anomaly detection - a resource-heavy task - in future analysis sessions of the same case-set.
- *CaseSetManager*:
  This manager is responsible for invoking all operations happening on the server. It functions as an organiser for the selected case-sets by loading the model data and passing it to the *ComponentManager*, where it is rearranged into a easy-to-query structure. It then gives the order to aggregate the data with the selected parameters and finally arranges all relevant data and information into `.json` format to send to the client.
- *ComponentManager*:
  The systems model consists of different units: `Components`, `Ports`, and `Connections` with additional information on the flow direction (OUTgoing, IN-coming or NEUTRAL).

[4] www.python.org/
[5] flask.palletsprojects.com

The *ComponentManger* stores all this information and provides methods to perform diverse queries on the components, such as selecting only `Components` associated with a specific flow, and re-structure the query result in different formats needed by the client and hence fulfilling requirement **R3 - Model Structure** (see Section 3 - 3).

- *DataAggregation*:
  To visualize more information than just the simulation results, the data needs to be processed and aggregated. *DataAggregation* is responsible for reading every component's data and storing the wrangled time-series data. It performs the above mentioned anomaly detection algorithms, which can be extended to any other outlier detection method. Additionally, data aggregation methods are available to calculate parameters such as the minimum, cumulative sum or integral, though these are mostly used for the exploration of multiple simulation ensembles (see section 6 - Future Work). This part of the system fulfills requirement **R2 - Data Aggregation** (see Section 3).

Once all the required data has been compiled, the server sends the data in `.JSON` format as a response to the client. The server then waits for requests from the client, and responds adequately with data updates or setting changes.

5 Data Visualization & User Interaction

The main motivation for our proposed application is to improve the existing simulation player used by domain experts to guide them to interesting incidents in the mimicked drive. Therefore, the minimum criteria for our visualization is to show the system component layout with its connections and an animation of the calculated results displayed with each component (requirement **R4 - Simulation Player** of Section 3). The original animation is linear, where users can change the playback speed. It also tends to clutter due to redundant information and spacious visualizations. To ease the analysis process domain experts desire more succinct information at one glance (requirement **R5 - Improved "Data-Ink Ratio"** of Section 3) and a visual feedback on where events of interest occurred during the simulation (requirement **R6 - Anomaly Visualization** of Section 3). They desire an overview of the simulation, but at the same time more detailed information in certain areas (requirement **R7 - Multi-Resolution Data Graphs** of Section 3). Lastly, the new application should be interactive, enabling the user to focus on specific areas, if desired.

For the client we opted for a web application, as it has the advantage of not requiring the user to download any specific software, is platform independent and many versatile data visualization libraries exist. We used `HTML` in combination with vanilla `JavaScript` and `CSS`. Additionally, we use `d3.js`, "a JavaScript library for manipulating documents based on data", as it is written in `JavaScript` and allows web developers "to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document" [4].

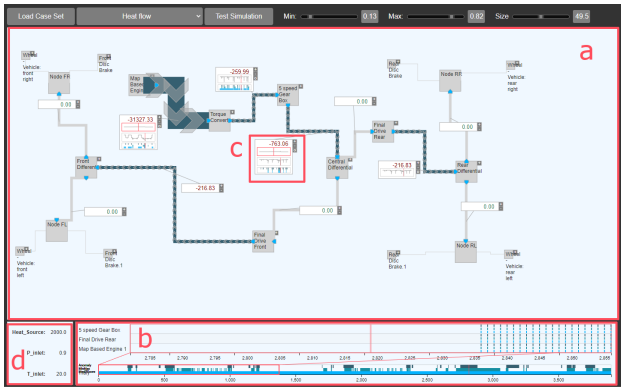


Figure 2: The proposed web application - **Energy Flow Explorer** with its four main windows. a) the *Simulation Player* shows the system layout, flows and *Data Label's*. b) the *Anomaly Timeline* depicts detected anomalies of the simulation. c) *Data Labels* give detailed information on demand. d) the *Parameter View* shows the applied parameters for the simulation.

To comply with all requirements we split the application into four visualization parts as indicated by the red rectangles in Figure 2: **a)** `Simulation Player`, **b**) `Anomaly Timeline`, **c**) `Data Label` and **d** `Parameter Table`.

First, the `Simulation Player` - the biggest visualization in Figure 2 **a** - shows the layout of used components in the selected simulation. Only the relevant components are shown as gray boxes, as not all components may be affected by the selected energy flow. Figure 2 **a**, e.g., shows the **Energy Flow Explorer**, where a model with 30 components was loaded but only 20 are visible, as only those have *'Heat Flow'*, the select flow parameter. Users can display the inactive components on demand. All components are connected by orthogonal flow lines, forming a planar graph drawing. A flow is defined as a connection between one component with outgoing flow and one with incoming flow, and displays the data from the outgoing component. The simulation software calculated a scalar value for each flow at every point in time during the simulated drive. These scalar values are reflected in the thickness of the flow lines; the thickness is interpolated between the minimum and maximum flow thickness, which can be set by the user. Although thickness is a visual encoding with low discernability of small changes ([3][19]), we use it here to give a visual feedback on the current en-

ergy distribution throughout the system, as size in general is a good visualizer for the quantitative data. In Figure 2 **a**, it is immediately visible, that there is a lot of energy flowing between two components in the upper left part, while only low amount of energy flows between the rest of the components. Here it is less important how big the change is between two frames, than how big the current value is. Additionally, the color of the flow is *gray*, if the current value is zero and *navy blue* otherwise. As the direction of the flow can change during the simulation, the current flow direction is indicated through animated half-transparent arrows along the flow line. These features realize requirement **R5 - Improved "Data-Ink Ratio"** (see Section 3).

The user can choose to animate the simulation, which complies with requirement **R4 - Simulation Player** (see Section 3). This updates the current flow value and consequently the size, color, and arrow position and direction

The `Anomaly Timeline` in Figure 2 **b** is dedicated to showing the points of interest, or anomalies, and is called `Anomaly Timeline`. As the name indicates, it shows at which point in time an anomaly occurred. The `Anomaly Timeline` has three rows, as we performed three different anomaly detection techniques (IForest anomaly detection, Min-Max Threshold, Modified Z Score), each dedicated to one technique. An anomalous point is indicated by a line and positioned relative to the point in time when it occurred in the simulation, with the left-most point translating to the start and the right-most to the end of the simulation. The `Anomaly Timeline` gives an overview of the anomalies of all components, as can be seen in Figure 6. If the user wants to see the anomalies of a specific component, they can add the component to the timeline. The associated anomalies will then appear on top of the summary timeline. In Figure 2 **b** three components have been added. This component-based timeline shows not every anomaly of a component, but only those within a specified time frame, i.e. x frames before and after the current time stamp, creating a zoomed anomaly timeline. A zoom-window linked to the time frame appears on the summary timeline as soon as one component has been added to the zoomed timeline (see Figure 3). The `Anomaly Timeline` complies to requirements **R6 - Anomaly Visualization** and **R7 - Multi-Resolution Data Graphs** (see Section 3) by showing more details of anomalies when requested. The user can move the zoom-window along the timeline, as well as change the size. A mouse-over effect reveals the anomaly type, component and time-stamp of an anomaly. The mouse-over visually links the referenced component by highlighting the corresponding `Data Label` in red for a short time.

A `Data Label` is an interactive and responsive visualization inherent to the **Energy Flow Explorer**. Each flow has a `Data Label`, which is an expandable graph, showing diverse information of the flow at multiple levels of detail (LOD) (see in Figure 2 **c**). The label always shows the scalar value associated with the current time stamp (see the first `Data Label` in Figure 4). The scalar value is either green or red, depending on if the difference in change since the last time stamp was positive or negative. The user can change the level of detail of the `Data Label` by clicking on the **[+]** button or **[-]** button on the right side. Showing the scalar value is level of detail one (LOD 1). LOD 2 shows a *Sparkline* of the whole simulation results for the associated component. This gives the user an overview of the components behaviour. Increasing the level of detail to 3 reveals an `Anomaly Timeline`, showing the anomalies exhibited by the component. A final increase in LOD shows a zoomed sparkline, centered around the current time stamp, with the same time frame as the zoomed timeline in visualization **b** - `Anomaly Timeline`. When playing the simulation, this time frame synchronizes with the zoom window and therefore stays always up-to-date. The right-most graph in Figure 4 depicts a `Data Label` at LOD 4, showing the scalar value, the zoomed sparkline, overview sparkline and anomalies. To not clutter the whole visualization, these `Data Labels` are kept small to give an overview, which can impede the analysis. To counter that, the user can enlarge a `Data Label` so that it gives more room and details to the graphs. Figure 5 shows the enlarged view of component *'Map Based Engine'*. In the future we plan to provide a pin feature, where the user can select maxed `Data Labels` and pin a smaller version of them to the side of the explorer, similar to "sticky notes". Future work might also include an extension of displayed information, such as multiple spark lines, which show additional statistical features e.g. the first derivative.

The user can choose to change the LOD for selected components, or globally set the LOD in the settings menu. In Figure 6 the global LOD level has been set to 4, the maximum. This change in level of detail complies with requirements **R5, R6** and **R7** (see Section 3).

The last part of the web application, the `Parameter Table` (see Figure 2 **d**), is dedicated to the parameters used in the simulation, as domain experts want to examine the same model with different parameters set. It shows the used parameters and corresponding values. When analysing ensembles of simulations the user will be able to see a summary here, as well as have the ability to brush the data here, i.e. select subsets of the simulations.

The user is able to animate the changes of values throughout the simulation in a linear manner with variable playback speed and pause whenever an anomaly is hit. Still, watching the whole simulation, even at increased speed, can take a long time, depending on the length of the simulated drive. To avoid this, we provide a play-

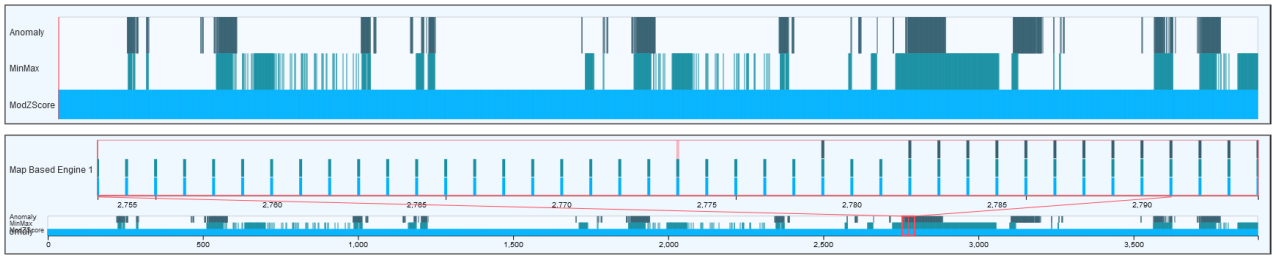


Figure 3: The *Anomaly Timeline* displays the anomalies, subdivided by anomaly type; one row each. A line indicates the occurrence of an anomaly somewhere in the system at the given point in time. The top figure shows the summarized *Anomaly Timeline* showing all anomalies of the system, subdivided by anomaly types Iforest Anomaly, MinMax Threshold and Modified Z Score. The second figure shows on the bottom the summarized timeline, and on top the timeline of a specific, user selected, component. Here the timeline is zoomed according to the zoom window (indicated by the orange rectangle in the summarized timeline) and shows all three anomaly types that arose in this component.

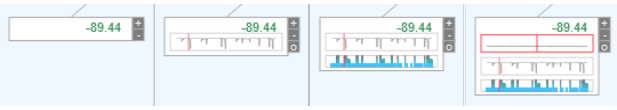


Figure 4: A *Data Label* showing its four levels of detail (LOD). First, on the left, only the current energy flow value is shown (LOD 1). Next to it, the label is extended to LOD 2, showing the whole, unzoomed sparkline. For LOD 3, the third *Data Label*, the anomaly timeline corresponding to this component is added. The right-most *Data Label* is extended by a zoomed version of the sparkline, centered around the current time stamp (LOD 4).

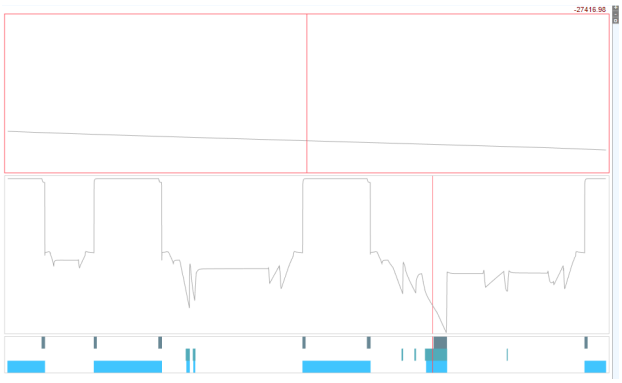


Figure 5: The **Energy Flow Explorer** with a selected Data Label enlarged, giving more room and revealing more details of how the component was affected throughout the simulation.

back option to accelerate the animation until the current time frame is near an anomaly. Before hitting the anomalous point, the animation slows down, so that the user can carefully watch the following time frames and examine the changes. The user can choose that, when hitting an anomaly, the `Data Label` associated with the component where the anomaly arose automatically expands, creating a visual link to the anomaly occurrence in the system.

6 Conclusions & Future Work

A vehicle undergoes many cycles of evaluation and adaptation before it can be cleared for production. The evaluation process is based on several simulations of a drive through various terrains and under divers conditions. The analysis process is a tedious task and current systems leave engineers wanting. We propose an interactive web application, which alleviates this task by aggregating the simulation results as well as performing anomaly detection techniques on it in a python-based server. The processed data is displayed in an interactive web visualization with four main views: the *Simulation Player* showing the system layout, giving playback control and encoding additional information in the flows (lines connecting two system components). The *Anomaly Timeline* turns the users attention to points of interest in the simulation. The *Data Labels* give the user additional information on a flow and component at multiple levels of detail, if desired. The parameters used for the simulation are depicted in the *Parameter View*. The whole application is highly interactive and gives the user an overview as well as detailed information on demand and non-linear animation of the simulation.

The system is currently being extended to cater to the analysis of multiple simulations concurrently, where the user will also be able to further investigate subsets of selected simulations brushing and linking. We also plan to include a visualization showing the balance of energy flowing into a component versus leaving it, revealing loss or gain in energy. We plan on doing a thorough evaluation together with domain experts, but first feedback has been very positive.

7 Acknowledgements

VRVis is funded by BMK, BMDW, Styria, SFG, Tyrol and Vienna Business Agency in the scope of COMET -

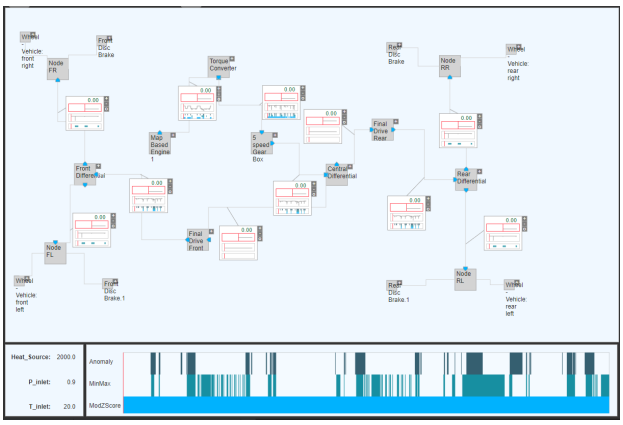


Figure 6: The **Energy Flow Explorer** showing the system components and associated *Data Labels* at the highest level of detail. Below the *Parameter View* and summarized *Anomaly Timeline*

Competence Centers for Excellent Technologies (879730) which is managed by FFG.

References

[1] Moez Ali. *PyCaret: An open source, low-code machine learning library in Python*, April 2020. PyCaret version 1.0.0.

[2] Jerry Banks, John Carson, Barry L. Nelson, and David Nicol. *Discrete-Event System Simulation (4th Edition)*. Prentice Hall, 4 edition, December 2004.

[3] J. Bertin and W. Berg. *Semiology of Graphics*. University of Wisconsin Press, 1983.

[4] Mike Bostock. Data-driven documents. `https://web.archive.org/web/20201031193629/https://d3js.org/`. [Online; accessed 14.02.2023].

[5] Cambridge Intelligence. KronoGraph.

[6] Brian Fisher. *Illuminating the Path: An R&D Agenda for Visual Analytics*, pages 69–104. National Visualization and Analytics Ctr (January 1, 2005), 01 2005.

[7] Wolfram Hasewend. AVL CRUISE. *ATZ Automobiltech. Z.*, 103(5):382–392, May 2001.

[8] B. Iglewicz and D.C. Hoaglin. *How to Detect and Handle Outliers*. ASQC basic references in quality control. ASQC Quality Press, 1993.

[9] Ingo Lütkebohle. AVL CRUISE™ M. `https://www.avl.com/cruise-m`, 2009. [Online; accessed 27.01.2023].

[10] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[11] Krešimir Matković, Helwig Hauser, Reinhard Sainitzer, and Eduard Gröller. Process visualization with levels of detail. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pages 67–70, 01 2002.

[12] Gabriel Moldovan, Alessandro Mariotti, Laurent Cordier, Guillaume Lehnasch, M. Salvetti, and Marcello Meldi. Multigrid sequential data assimilation for the large-eddy simulation of a massively separated bluff-body flow. *pre-print*, 12 2022.

[13] Xian Qu and Jun Xie. Simulation analysis for effect of rear structure of hatchback car on rear field characteristics. *Journal of Physics: Conference Series*, 1815(1):012001, feb 2021.

[14] Maria Riveiro. Evaluation of normal model visualization for anomaly detection in maritime traffic. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 4, 04 2014.

[15] Lukas Ruff, Jacob Kauffmann, Robert Vandermeulen, Gregoire Montavon, Wojciech Samek, Marius Kloft, Thomas Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, PP:1–40, 02 2021.

[16] K. Santhosh, Debi Dogra, and P. Roy. Anomaly detection in road traffic using visual surveillance: A survey. *ACM Computing Surveys*, 53:1–26, 12 2020.

[17] Young-Kyoon Suh and Lee Kiyong. A survey of simulation provenance systems: modeling, capturing, querying, visualization, and advanced utilization. *Human-centric Computing and Information Sciences*, 8, 12 2018.

[18] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, 2 edition, 2001.

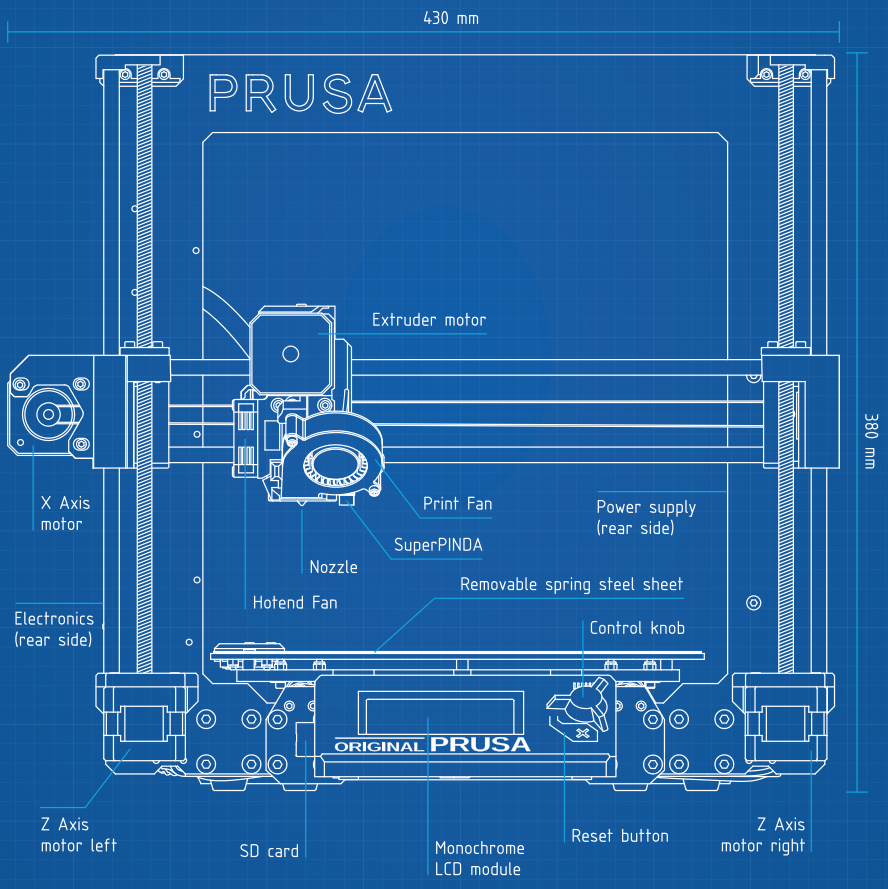
[19] Johan Wagemans, Jacob Feldman, Sergei Gepshtein, Ruth Kimchi, James R. Pomerantz, Peter A van der Helm, and Cees van Leeuwen. A century of gestalt psychology in visual perception: Ii. conceptual and theoretical foundations. *Psychological bulletin*, 138 6:1218–52, 2012.

[20] Kamila Zdybał, Giuseppe D'Alessio, Gianmarco Aversano, Rafi Malik, Axel Coussement, James Sutherland, and Alessandro Parente. *Advancing Reacting Flow Simulations with Data-Driven Models*, pages 304–329. Cambridge University Press, 01 2023.

Partners of CESCG 2023

CHANGE THE WORLD WITH 3D PRINTING

BE PART OF THE DEVELOPMENT OF THE WORLD'S BEST 3D PRINTERS!



PRUSA3D.COM

PRUSA RESEARCH
by JOSEF PRUSA

WHO ARE WE LOOKING FOR?

Working with us is a truly remarkable adventure. We develop, invent, and test... We have no limits on what is possible and what is not. We have only one particular goal: **to make the best 3D printers in the world!**

That's why we're looking for **skilled software and hardware developers** to join our development department. Interested?
Take a look at our development teams that could use a hand:

FIRMWARE

WE HAVE SEVERAL DEDICATED TEAMS. THESE TEAMS DEVELOP SPECIALIZED FIRMWARE FOR OUR FDM AND SLA PRINTERS.

C++ AND PYTHON

PRUSASLICER

WE DEVELOP OUR OWN SLICING SOFTWARE FOR PRINTING DATA PREPARATION AND WORKING WITH 3D OBJECTS.

C++

PRUSA CONNECT

WE ARE DEVELOPING A TOOL TO CONTROL THE ENTIRE 3D PRINTING ECOSYSTEM REMOTELY.

C++ AND PYTHON

WEB DEVELOPMENT

WE HAVE SEVERAL WEBSITES, ALL DEVELOPED IN-HOUSE.

PHP, PYTHON, DEVOPS (K8S), JAVASCRIPT (REACT, ANGULAR)

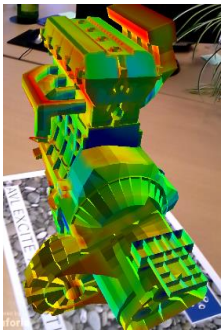
WHAT TOOLS DO WE USE?

We use **GitHub** for software development and internal projects, **JIRA** for progress tracking, and Confluence for documenting work. We don't send dozens of emails to each other, we communicate through **Slack**. Our company also offers a well-equipped lab and workshop with a large variety of tools: oscilloscopes, spectrometers, lasers, CNC, etc. You can also use revolutionary and exciting technologies such as robotic arms or a cybernetic dog from Boston Dynamics. It's up to you what you can do with them.

VISIT OUR WEBSITE FOR CURRENT OPEN JOB OPPORTUNITIES

WWW.PRUSA3D.COM

zentrum für
virtual reality und visualisierung
forschungs-gmbh



VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH

The VRVis Research Center is a joint venture in research and development for virtual reality and visualization. VRVis was founded in 2000 as part of the Austrian Kplus program to bridge the gap between academic research and commercial development as well as to supply the necessary transfer of knowledge between the academic community and industry. The competence center VRVis is funded by BMVIT, BMDW, the Vienna Business Agency, Styria and the Styrian Business Promotion Agency (SFG) within the scope of COMET – Competence Centers for Excellent Technologies. The program COMET is managed by FFG.
The company is located in Vienna, Austria. Today, around 70 researchers together with about 20 students do high-level applied and basic research in four different areas.

The Team

VRVis consists of internationally experienced researchers in the areas Visual Analytics, Complex Systems, Smart Worlds and Multiple Senses. Their outstanding experience and knowledge in these topics qualify them for the innovative research they are performing. The research areas are headed by key researchers who manage these areas, define goals and projects for this area, as well as conduct the defined research together with their staff. Most members of the research teams are young researchers, whose creativity and ingenuity is the key to the success. Beyond that VRVis has a friendly and inclusive company culture, which translates into great teamwork and –spirit, also outside of the office (e.g. our running teams).

Research Program

The scientific research program consists of the previously mentioned research areas in which thematically matching projects are conducted. Each research area realizes application projects on the one hand and basic research for these application projects on the other hand.

Working at VRVis

VRVis is always looking for students, junior and senior researchers who want to join the team. VRVis is offering regular positions as well as internships, diploma and PhD theses in cooperation with universities. For more detailed information or currently open positions visit our website at www.vrvis.at.

Selection of Partners

Scientific Partners:

- Vienna University of Technology
- Graz University of Technology
- University of Vienna

Industrial Partners:

- AVL List GmbH
- AGFA Healthcare GesmbH
- Austria Power Grid AG
- Geodata Ziviltechniker GmbH
- HILTI Corporation
- ÖBB-Infrastruktur AG
- RHI Feuerfest GmbH
- Zumtobel Lighting GmbH
- and many more

Currently, VRVis is again extending its industrial base with new partners from several new fields.

Additional Information and Contact

Please visit our website for detailed information about the research program or current projects at www.vrvis.at or contact us at office@vrvis.at or via phone +43 (1) 908 98 92.