Graz University of Technology

Institute for Computer Graphics and Vision

Dissertation

Optical phenomena in participating media as metaphors for scientific visualization

---

Rostislav Khlebnikov

Graz, Austria, June 2014

Thesis supervisor

Prof. Dr. Dieter Schmalstieg

External referee

Prof. Dr. Daniel Weiskopf

To Elena and my Parents

I don't think it had ever occurred to me that man's supremacy is not primarily due to his brain, as most of the books would have one think. It is due to the brain's capacity to make use of the information conveyed to it by a narrow band of visible light rays. His civilization, all that he had achieved or might achieve, hung upon his ability to perceive that range of vibrations from red to violet. Without that, he was lost.

John Wyndham, The Day of the Triffids

Abstract

Visualization of spatially fixed data is of great importance in many disciplines, as it allows to use the huge power of human visual system to spot the crucial dependencies and trends hidden within the data that would be difficult or almost impossible to detect automatically. However, the algorithms that transform the raw data to the image sequence which is presented to the analyst may both promote or impede the effective data exploration.

This thesis aims to develop the visualization algorithms which synergize with the human visual system by employing natural phenomena as the metaphors. The intuition behind this approach is that the millions of years of evolution of the complex visual system have made it a very powerful instrument for solving a *limited* amount of tasks necessary for survival. Therefore, if the data is presented in a familiar form, the human visual system will extract the available information very efficiently. With this basic idea in mind, we propose three novel approaches to data visualization at three different levels of abstraction from the original natural phenomenon metaphor.

First, we explore a very literal approach at using the natural phenomena in visualization. We make the next step in bringing realistic rendering to interactive DVR by allowing parallel creation of irradiance cache on GPU, which allows efficient rendering of scattering in participating media. Second, we use the crepuscular rays phenomenon in a context that would not be feasible in nature to create an efficient approach for visualization of safe access paths towards tumors. Even though the path safety computation is conceptually and visually similar to formation of crepuscular rays in real world, we impose additional useful information into the ray intensity by using computational schemes different from realistic exponential decay of light in participating media. And third, we use the visual appearance of smoke as an abstract metaphor to separate the available space between multiple variables. While the overall appearance of our visualization is similar to that of smoke, it is not driven by physical processes, but rather by the data that needs to be displayed. Using the controlled user studies, we show that regardless of the fact that the second and third approaches do not replicate all the physical processes inherent to the

natural phenomena used as metaphors, the human visual system is still flexible enough to interpret the data more efficiently than the existing ad-hoc approaches.

We believe that using natural phenomena as metaphors for data visualization algorithms is a promising research direction. Because the scientific community does not yet fully understand how do the human brain in general and the human visual system in particular work, and therefore perceptually optimal visualizations cannot yet be designed, we can imitate the visual appearance of the natural phenomena to create efficient and accessible methods for visualizing data.

Keywords. Data visualization, direct volume rendering, Monte-Carlo volume rendering, multivariate visualization.

Kurzfassung

Die Visualisierung von räumlich invarianten Daten spielt in vielen Disziplinen eine immer größere Rolle da es uns ermöglicht, das immense Leistungsvermögen des menschlichen visuellen Systems dahingehend auszunutzen, für Algorithmen nicht offensichtliche Abhängigkeiten und Trends zu identifizieren. Jedoch muss man bei der Entwicklung solcher Visualisierungsalgorithmen Vorsicht walten lassen, da die gewählte Transformation von Daten zu Bildern die visuelle Analyse und Erforschung zwar deutlich erleichtern, aber auch erschweren kann.

Diese Arbeit zielt darauf ab, Visualisierungsalgorithmen zu entwickeln, welche Naturerscheinungen als Metaphern zur Synergie mit dem menschlichen visuellen System verwenden. Die Intuition hinter diesem Ansatz ist, dass in Millionen von Jahren der Evolution des äußerst komplexen visuellen Systems ein sehr mächtiges Instrument für die Lösung einer begrenzten Menge von Aufgaben - vorrangig zur Sicherung des Überlebens - entstand. Aufgrund dieser Entwicklung ist das menschliche visuelle System sehr effizient darin, Information aus bereits bekannten visuellen Repräsentationen abzuleiten. Auf Basis dieser Grundidee schlagen wir drei neue Ansätze für die Datenvisualisierung vor, welche auf drei verschiedenen Ebenen der Abstraktion eines ursprünglichen metaphorischen Phänomens beruhen.

Zuerst untersuchen wir einen sehr direkten Ansatz der Verwendung von Naturphänomenen in der Visualisierung indem wir den nächsten Schritt in Richtung realistischer Darstellung gehen. Durch paralleles Erstellen eines “Irradiance Cache” während direkter Volumsvisualisierung können wir die Lichtausbreitung in optischen Medien, wie z.B. Wolken, effizient berechnen. In einem weiteren Schritt verwenden wir Dämmerungsstrahlen in einem naturfernen Kontext um sichere Zugangswege zu Tumoren in menschlichen Körpern anzuzeigen. Obwohl diese Methode in ihrem Konzept und auch visuell der Idee der Dämmerungsstrahlen sehr ähnlich ist, erlegen wir den Strahlen noch zusätzliche Information auf welche vom üblichen exponentiellen Abfall der Lichtstärke

in optischen Medien abweicht. Im letzten Teil nutzen wir die visuellen Eigenschaften von Rauch als abstrakte Metapher zur Separierung des Raums zwischen mehreren Variablen. Obwohl das generelle Erscheinungsbild unserer Visualisierungsmethode dem von Rauch ähnelt, spielen die physikalischen Prozesse nur eine untergeordnete Rolle. Vielmehr beeinflussen die Daten, die wir darstellen wollen, den Prozess. In kontrollierten Studien mit Testnutzern zeigen wir, dass die zweite und dritte Methode trotz des Ausschließens mancher physikalisch korrekteren Methoden zur Berechnung der verwendeten Naturphänomene die Dynamik und Flexibilität des menschlichen visuellen Systems effizient ausnutzen können, um die Interpretation der angezeigten Daten im Vergleich zu herkömmlichen Methoden erleichtern.

Wir glauben, dass die Verwendung von Naturphänomenen als Metaphern für Algorithmen zur Visualisierung großer Datenmengen eine vielversprechende Forschungsrichtung bietet. Da die Wissenschaft noch nicht alle funktionalen Zusammenhänge des menschlichen Gehirns, und insbesondere des visuellen Systems, versteht, ist die Erstellung von Visualisierungsmethoden, welche die menschliche Wahrnehmung optimal unterstützen äußerst schwierig. Nichtsdestotrotz können wir Naturphänomene visuell imitieren und damit effektive und leicht zugängliche Methoden zur visuellen Exploration von Daten kreieren.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....

(Unterschrift)

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....

date (signature)

Acknowledgments

I would like to thank all the people who were of great support for development of this thesis. First and foremost I would like to express my gratitude to my supervisor, Prof. Dieter Schmalstieg, whose insight and creativity have sparked the ideas present in this thesis and whose organizational talent allowed great collaboration with my peers and let me focus completely on the thesis.

I would also like to express my gratitude to my colleagues, as it was absolute pleasure to work with them. The discussions with them allowed the rough ideas to flourish and take a refined shape, in which they are presented in this thesis. First of all, I am grateful to Bernhard Kainz, who I was working with for several years from the very beginning of my studies at TU Graz. His creativity, composure and diverse knowledge in many fields taught me a lot and helped me gain the confidence in my work. Many other colleagues helped me overcome the hurdles and to achieve the goals that stood before me when writing this thesis: Manuela Waldner for helping me go through the tough times of doubt, Philip Voglreiter and Markus Steinberger for the heated discussions and great teamwork, Judith Muehl for introducing me to the world of science at TU Graz, Marc Streit for the insight in the field of information visualization. And many other fellow PhD students whose work created the unforgettable feeling of being involved in the scientific community: Markus Tatzgern, Eduardo Veas, Eric Mendez, Stefan Hauswiesner, Alexander Bornik, Tobias Langlotz, Stefanie Zollmann and many many others.

This work was funded by the European Union in FP7 VPH initiative under contract number 223877 (Project "Imppact") and the Austrian Science Fund (FWF) (Project P23329 "Managed Volume Processing on the GPU").

I am very grateful to my family and especially my parents, who have shown me the beauty of science and made an immense effort to allow me to get the best education possible. Finally, I dedicate this work to my amazing wife, Elena. Her tremendous support and encouragement helped me in finishing this thesis, as she always knows when it is

necessary to take a short time off to gain a huge amount of energy for the next effort in bringing a new idea to life.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Overview

Contents

1.1 Introduction

The growing amount of data that needs to be analyzed requires significant efforts from the research community to develop algorithms and tools that allow gaining insight into these data. In this regard, a variety of methods for automatic computational analysis of data have been developed. However, on the stage of data exploration and forming of hypotheses, computational algorithms are still inferior to the analysis that can be provided by the human operator, given the right tool. This is due to the immense power of the human visual system that allows a person to notice obscure patterns and trends in the data when presented visually. Therefore, concise and clear methods for visualizing data are still an active focus of research.

Even though the human visual system (HVS) is very powerful, it has its limits in flexibility, because in the evolutionary process, its sole purpose was to provide information about the natural environment to the human brain. Taking this specialization of the human system into account helps designing visualization methods that exploit the power of the HVS effectively.

In fact, many data visualization techniques already use this principle. For example, conventional direct volume rendering [105] interprets the volumetric data as a partici-

pating medium that emits, absorbs and scatters the light [112]. And even though this approach is synthetic, because, for example, a human body does not represent a participating medium, and the transparency of some tissues during visualization is artificial, the fact that direct volume rendering is extensively used in a variety of applications shows that such an approach is viable. In this regard, tissue transparency that reveals the inner structures of the human body to the observer may be seen as a metaphor, which, although not realistic, allows using the power of HVS for better comprehension of the data at hand.

These considerations lead to the idea that other optical phenomena arising from light rays travelling through transparent media could provide useful visualization metaphors. In this work, we explore several examples of this idea, ranging from realistic optical phenomena to rather metaphorical ones.

This thesis is a first step towards the exploration of a design space of visualization metaphors based on optical phenomena. Initial results indicate that users can intuitively understand such metaphors and find them useful. We therefore hope that optical phenomena will provide a fruitful avenue of future visualization research.

Figure 1.1: Photo of "Body Worlds" exhibition (left, cropped from "Body Worlds: Vital" by Matty Stevenson, licensed under CC BY), and an example of the image generated using our approach in Chapter 3 (right).

The first approach simulates light scattering in transparent materials for achieving hyper-realistic images of the interior of the human body, visually similar to the physical process of plastination for revealing inner anatomical structures (commonly known as "Body Worlds", see Fig. 1.1, left). This approach is based on known Monte-Carlo techniques, and we show how it can be efficiently computed in real-time on modern GPU architectures.

Figure 1.2: The photo of crepuscular rays (left, photo by Brocken Inaglory, licensed under CC BY), and an example of the image generated using our approach in Chapter 4 (right).

The second approach mimics the optical phenomenon knows as crepuscular rays, or god rays (see Fig. 1.2, left), to facilitate planning of minimally invasive interventions. To achieve this, we regard the target structure (e.g., tumor) as a light source. The vulnerable structures detected from various imaging modalities are then considered as blocking the light emitted by the target structure. Finally, the intensity of light on the outside of the body guides a surgeon in the choice of safe locations where surgical probes could be placed.

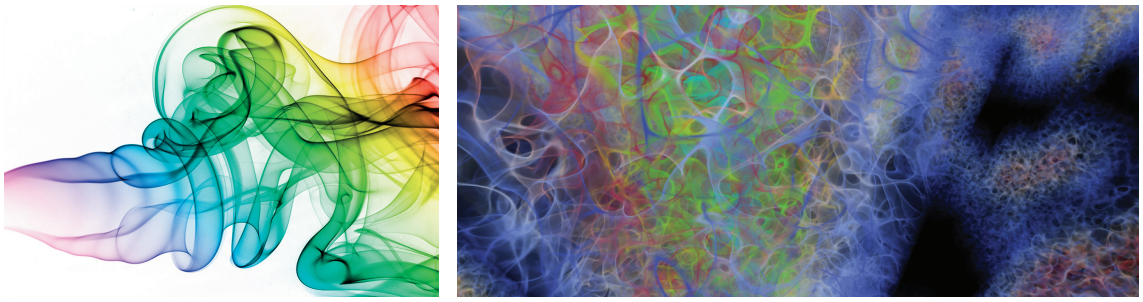


Figure 1.3: The photo of colored smoke (left, "Horizontal smoke" by Andreas Feldl, licensed under CC BY), and an example of the image generated using our approach in Chapter 5 (right).

The third approach builds on the observation that while smoke is mostly transparent, the denser parts can carry color, which can be easily distinguished by the observer (see Fig. 1.3, left). Furthermore, smoke plumes exhibit noticeable internal structures. Therefore, it is possible to simultaneously perceive the smoke color and structure along with the objects within it. This effect can be leveraged for multi-variate visualization in 2D and 3D, and has useful properties such as scale invariance. Therefore, for visualization of three-

dimensional data, we use several smoke-like textures to separate the space between the variables and then to visualize each variable with color while avoiding color intermixing between the variables. For two-dimensional data, we additionally display two variables using the orientation and frequency of a smoke-like texture.

1.2 Individual publications and collaboration statement

Aside from the supervisor of this thesis, Prof. Dieter Schmalstieg, many colleagues have contributed to the work presented in this thesis. The publications used in this thesis are mentioned in this section.

- Khlebnikov, R., Voglreiter, P., Steinberger, M., Kainz, B., and Schmalstieg, D. (2014). Parallel irradiance caching for interactive Monte-Carlo direct volume rendering. *Computer Graphics Forum. In press.*

  The author of this thesis contributed the main idea, the expected gain priority formula, a large portion of implementation as well as the novel irradiance extrapolation method and object-space locking approach. Philip Voglreiter implemented the multi-reference octree to store the cache entries, proposed the screen-space locking approach and collected the necessary experimental data. Markus Steinberger implemented the queue sorting and time allocation for different procedures within the Softshell framework. Jaime Garcia Guevara provided additional related work analysis. Bernhard Kainz assisted in writing the paper. Dieter Schmalstieg refined the final version of the paper and coordinated the team. This work has been used in this thesis in Chapter 3.

- Khlebnikov, R., Kainz, B., Steinberger, M., and Schmalstieg, D. (2013). Noise-based volume rendering for the visualization of multivariate volumetric data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2926–2935

  The author of this thesis contributed the majority of this work including the main idea, parameter computation for zoom-independent visualization, opacity mapping approach, most of the implementation and the design of the controlled experiments for evaluation. Berhnard Kainz extended the implementation for the time-dependent data, assisted with conducting the user studies and with writing the paper. Markus Steinberger provided the statistical analysis of the user study data. Dieter Schmalstieg coordinated the work and revised the final version of the paper. This work has been used in this thesis in Chapter 5.

- Khlebnikov, R., Kainz, B., Steinberger, M., Streit, M., and Schmalstieg, D. (2012). Procedural texture synthesis for zoom-independent visualization of multivariate data. *Computer Graphics Forum*, 31(3):1355–1364

  The author of this thesis contributed the vast majority of this work including the main idea, opacity redistribution and filtering approaches, the implementation, and designed and conducted the controlled experiments for evaluation. Bernhard Kainz assisted with writing the paper. Markus Steinberger provided the statistical analysis of the user study data. Marc Streit provided the climatology data and processed it to be used in our rendering system. Dieter Schmalstieg coordinated the work and revised the final version of the paper. This work has been used in this thesis in Chapter 5.

- Khlebnikov, R., Kainz, B., Muehl, J., and Schmalstieg, D. (2011a). Crepuscular rays for tumor accessibility planning. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2163–2172. *Karl-Heinz Höhne Award for Medical Visualization 2012, 3^{rd} place.*

  Bernhard Kainz contributed the basic idea of using crepuscular rays for computation of safety of access paths, evaluated the method and implemented the 2D and 3D visualization in software. The author of this thesis derived the mathematical formulation for various accumulation approaches suitable for usage in medical practice as well as implemented the 3D path safety computation using the GPU. Judith Mühl extended the related work section of the paper and was the author of the EU IMPPACT proposal, which has funded this research. Dieter Schmalstieg was the team coordinator and edited the paper. This work has been used in this thesis in Chapter 4.

1.3 Additional publications

Apart from the publications mentioned in Section 1.2, the author of this thesis has also contributed to publications listed below.

- Hauswiesner, S., Khlebnikov, R., Steinberger, M., Straka, M., and Reitmayr, G. (2012). Multi-GPU image-based visual hull rendering. In Childs, H., Kuhlen, T., and Marton, F., editors, *EGPGV*, pages 119–128. Eurographics Association

The author of this thesis implemented and optimized various approaches to computing visual hulls on multiple GPU's.

- Kainz, B., Steinberger, M., Hauswiesner, S., Khlebnikov, R., and Schmalstieg, D. (2011b). Stylization-based ray prioritization for guaranteed frame rates. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, NPAR '11, pages 43–54, New York, NY, USA. ACM

- Kainz, B., Steinberger, M., Hauswiesner, S., Khlebnikov, R., Kalkofen, D., and Schmalstieg, D. (2011a). Using perceptual features to prioritize ray-based image generation. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 215–215, New York, NY, USA. ACM

  The author of this thesis revised the papers and contributed to creation of video content.

- Khlebnikov, R., Kainz, B., Roth, B., Muehl, J., and Schmalstieg, D. (2011b). GPU based on-the-fly light emission-absorption approximation for direct multi-volume rendering. In Laramee, R. and Lim, I. S., editors, *Eurographics 2011 - Posters*, pages 11–12

  The author of this thesis revised the final version of the paper.

- Khlebnikov, R. and Muehl, J. (2010). Effects of needle placement inaccuracies in hepatic radiofrequency tumor ablation. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 716–721

  The author of this thesis derived the necessary formulations and implemented the finite-element solver for coupled radio-frequency ablation simulation.

Chapter 2

Related work

Contents

Natural phenomena that can be observed in our daily life are caused by the light interacting with physical objects in our surroundings. Therefore, in order to efficiently display such phenomena and to use them for purposes of scientific data visualization, it is necessary to understand the underlying physics of light-matter interaction. In this chapter, we first explore the equations describing this interaction. Then we show the simplifications employed to efficiently solve these equations as well as the range of natural phenomena that can be displayed using these simplifications. Finally, we describe the most relevant approaches for direct volume rendering, ray casting and Monte-Carlo rendering, in more detail.

2.1 Physics of light

In context of scientific visualization, a common approach is to use geometrical optics as a main model for describing light behaviour. Even though geometrical optics itself is a simplistic model compared to more advanced models such as electromagnetic or quantum optics, it is suitable for a majority of visualization applications. Furthermore, regardless of its relative simplicity, simulation of optical effects that can be achieved with geometrical

optics still requires tremendous computational effort, and therefore further simplifications for particular tasks are necessary. We discuss some of these simplifications in section 2.4.

Geometrical optics assumes that light travels in a straight line within a homogeneous medium and is reflected and refracted at media interfaces. The law of reflection states that the incident and the reflected rays are coplanar and that the angle of incidence is equal to the angle of reflection. The law of refraction states that the incident and the refracted ray are coplanar and the ratio of sinuses of angle of incidence and angle of refraction is equal to the constant refractive index for the materials forming the interface (see Fig. 2.1, left). In addition to reflection and refraction, light may also be absorbed by the media, which effectively transforms the light energy into the internal energy of the absorber, reducing the light intensity.

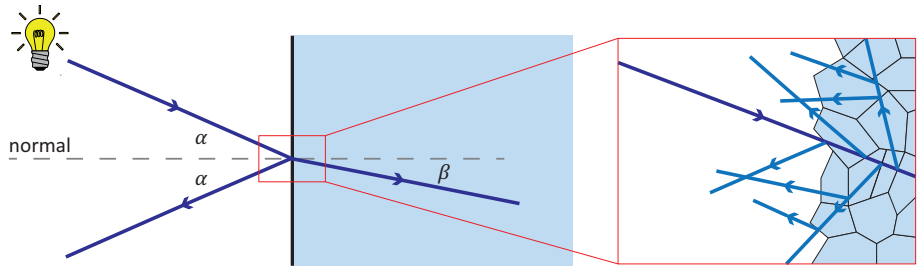


Figure 2.1: Perfect reflection and refraction at the interface between two media with incoming and reflection angles α and refraction angle β (left). Diffuse reflection at a surface of an opaque object caused by the micro-structure of the surface layer (right).

2.1.1 Integral formulations

Even though geometrical optics describes the light-object interactions on a basis of single rays, some effects are better formulated using stochastic approach. Such effects include diffuse reflections and scattering within a participating medium.

Diffuse reflections are mostly caused by the fact that incoming rays are reflected and refracted many times in the boundary layer of material before exiting it (see Fig. 2.1, right). These reflections and refractions are caused by the microscopic structure of the material, and thus the outgoing direction for a particular incident ray direction requires precise knowledge of the material structure. Furthermore, the primary incoming light ray may be simultaneously reflected and refracted by the material, leading to even more complex ray behaviour, which influences the outgoing direction. Obviously, specification of the material properties as well as the computation of the outgoing direction for each incoming ray is not feasible for straightforward computation of illumination in large scenes.

This behaviour can be, however, summarized using a bidirectional reflectance distribution function (BRDF) [121]. A BRDF specifies the amount of light that is reflected at an opaque surface given the incident and outgoing directions (see Fig. 2.2).

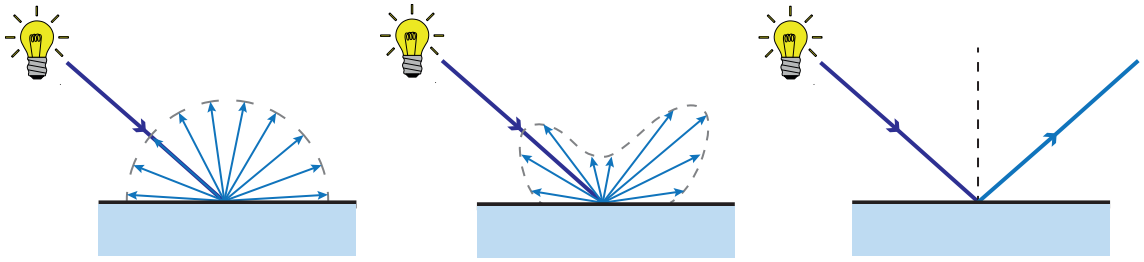


Figure 2.2: Examples of BRDF functions: ideal diffuse, where the light is reflected equally in all directions (left), perfect mirror (right), and a general BRDF (center).

Similarly, when the light travels through a participating medium, the rays may be scattered by the particles composing this medium. Therefore, similar effects are present and have to be taken into account when rendering. However, modelling single rays and particles is not feasible, and therefore the notion of a phase function is used to summarize the interaction of light with the whole collection of particles comprising the medium. Phase functions are described in more detail in section 2.3.

2.2 Natural phenomena explained by geometrical optics

Using the formulations outlined in the previous section, a variety of natural phenomena may be described using geometrical optics. In this section, we discuss several natural phenomena explained by geometrical optics, describe the approaches to their rendering and discuss their application to scientific data visualization. As light interaction with participating media is crucial to direct volume rendering and to our methods, we analyse the scattering effects in more detail in the next section.

2.2.1 Transparency

Transparency is a fundamental property of various materials that allows one to see *inside* the solid objects. Furthermore, in data visualization, the materials that are naturally opaque may be artificially made transparent in order to see inside the objects that would not allow this in the real world, for example, allowing the physician to observe the internal organs of a human body without an intervention.

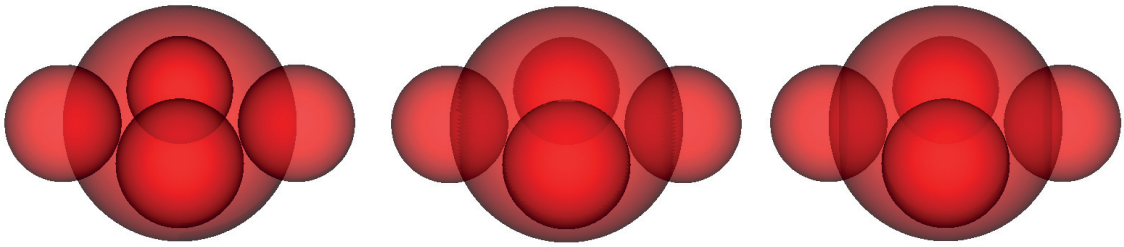


Figure 2.3: Comparison of rendering of transparent objects with neither depth sorting or depth peeling (left), depth sorting only (middle), and depth peeling (right). Figure by Lars Friedrich, licensed under CC BY.

However, displaying transparent objects often requires additional efforts compared to rendering of opaque objects only. For instance, one approach to correct rendering of a scene containing transparent polygons requires sorting the primitives based on the distance to the virtual camera. Alternatively, order-independent approaches to rendering transparent geometry may be applied. Such approaches include depth peeling [11] and stochastic transparency [47]. Including transparency into the rendering system allows creating more realistic and visually appealing images (see Fig. 2.3).

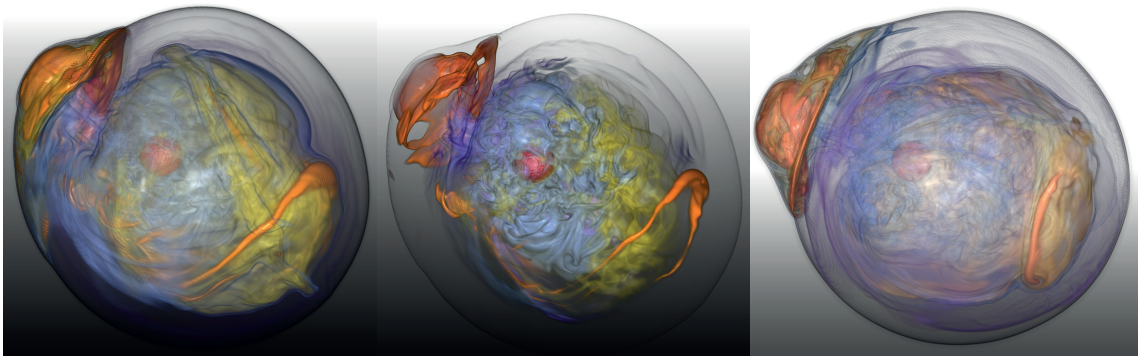


Figure 2.4: An example of different transfer functions applied to the same dataset: manual 1D transfer function with gradient magnitude modulation (left, [33]), manual transfer function based on occlusion spectrum (center, [33]), and semi-automatic visibility-driven transfer function (right, [34]).

The notion of transparency is crucial for visualization of volumetric data. Usually, the transparency of different materials is specified using a *transfer function* [44, 105]. The transfer function is manipulated in order to reveal the features of interest within the volumetric dataset, while completely hiding the materials obstructing the view or de-emphasizing them to a large extent to provide the contextual information (see Fig. 2.4). Therefore, the correct design of the transfer function is very important when visualizing

volumetric data, which may be a very difficult and tedious task depending on the nature of the data. Consequently, automatic or semi-automatic transfer function design is an important research topic in the visualization community. However, this topic goes beyond the scope of this thesis, and we refer the reader to a recent survey of various transfer function design approaches presented by Arens and Domik [7].

2.2.2 Shadows

According to laws of geometrical optics, light travels in a straight line until it encounters an interface between two materials with different optical properties. Therefore, the shadowing effects can be directly explained with geometrical optics (see Fig. 2.5).

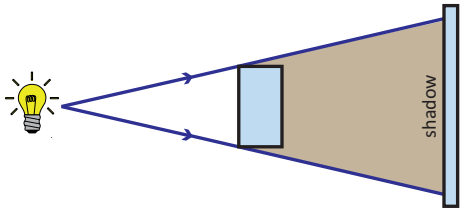


Figure 2.5: An illustration of formation of hard shadow from a point light source.

Shadows are the areas where the light from a source cannot reach a surface due to an object between the surface and the light source. As shadows can greatly improve the perception of shapes and relative positions of the objects within a scene, including them to a rendering system is important.

Figure 2.6: An example of hard shadows (left) and soft shadows (right) [64].

Simple cast shadows from opaque sources were introduced in the early years of development of computer graphics. For example, in 1978, Williams has proposed an algorithm for displaying curved shadows on curved surfaces using a Z-buffer [171]. In 1988, Blinn proposed to use a projective transformation to compute a shadow from a distant light

source, which requires fewer memory and computations compared to the Z-buffer approach [20]. However, the development of efficient shadowing algorithms is still an active area of research in the effort to widen the range of supported light sources, accommodate large scenes and improve the shadow quality (see Fig. 2.6). We refer an interested reader to an excellent state of the art reports by Hasenfratz and colleagues [64] and Scherzer and colleagues [140], as well as the course notes by Eisemann and colleagues [46] for more details on the current research in real-time shadow mapping algorithms.

Figure 2.7: A comparison of various volume illumination approaches in DVR [76].

In the context of interactive volume data visualization, shadowing effects could not be included until the recent years, due to the complexity of computing shadows in a translucent volume. However, the growing performance of the consumer-level GPU's and advances in general-purpose GPU programming have made inclusion of shadow computations feasible for DVR (see Fig. 2.7).

Hadwiger and colleagues proposed a GPU implementation of deep shadow maps for DVR, which allows to achieve near real-time framerates with high quality shadowing [59]. Ropinski and colleagues compared three approaches to computing shadows for GPU-based interactive volume ray casting: shadow rays, shadow maps and deep shadow maps [136]. Shadow rays allow to produce very high quality images, the computational overhead is very high compared to the other techniques. Shadow mapping achieves highest framerates among the three methods, but does not support correct shadows from semi-transparent objects. Deep shadow maps achieve a good balance between quality and performance, but require careful parameter selection for each dataset. Further work in shadowing for direct volume rendering is directed towards increased realism with the goal of improving the perception of 3D volume-rendered images and increasing performance to allow rendering high-resolution volumetric datasets with high quality shadowing effects. For an excellent

overview of the advances in advanced illumination in volume rendering, the reader is encouraged to consult recent surveys by Jönsson and colleagues [75, 76].

2.2.3 Reflection and refraction

Surfaces reflecting light are widespread in the natural environment. Furthermore, often the reflection and refraction effects are exposed on the same surfaces, one example being the water surface. Therefore, adding reflections and refractions in the rendered image is often necessary to create a realistic rendering.

Figure 2.8: Reflections on concave objects (left, [52]), caustics (center, [174]), and a virtual mirror used for laparascopic intervention (right, [17])

The reflection and refraction effects share similar traits in the geometrical relationships that describe them. Therefore, the algorithms that are able to render the reflected geometry only need a slight modification to be used for rendering the refracted geometry.

One of the approaches to rendering the reflections consists of two steps: creation of virtual reflected geometry and rendering of both actual and reflected geometry from the perspective of virtual camera. The complexity of algorithms for creating reflected geometry depends on the type of a reflecting surface. For example, a simple planar reflector requires very little computational effort – it is only necessary to reflect the positions of the objects within the scene across a planar surface, which can be done using a single reflective matrix transformation. Similarly, for refractions formed by a planar surface, a single translation-rotation matrix is required to transform the scene objects. In contrast, curved reflections and refractions may require decomposition of surfaces into concave and convex parts and more involved computation of the transformed vertex positions [52, 53, 125] (see Fig. 2.8, left).

One particular effect caused by reflective or refractive behaviour of surfaces are caustics. Caustics are caused by focusing and defocusing of light rays by curved surfaces. As caustics are very prominent and eye-catching, a lot of effort was made to facilitate their interactive

rendering. For instance, Ernst and colleagues [50] use interpolated caustic volumes to determine the pixels for which the caustic intensity has to be estimated, which is then done by considering the caustic wavefront. Yu and colleagues estimate the caustic surfaces by parametrizing rays using their intersections with two parallel planes [176]. The energy flux on the receiver surface is then estimated using the ray characteristic equation to produce sharp and clear caustics in real-time. Wyman and Nichols use adaptive caustic maps and deferred shading to render high quality caustics at interactive frame rates [174] (see Fig. 2.8, center). A good overview of the techniques for rendering caustics is given by Prast and Frühstück in their recent survey [130].

In the context of rendering medical data, reflections may be used to improve understanding of complex scenes. This is achieved by placing virtual mirrors into the scene to provide the user with multiple viewpoints integrated within a single view. For example, virtual mirrors improve the accuracy of medical tool insertion for laparascopy [17] (see Fig. 2.8, right).

2.3 Scattering in participating media

Light scattering dominates the visual appearance of participating media observed in the real world, such as clouds or smoke. The main feature of participating media that needs to be considered to render them realistically is that light constantly interacts with the medium, while it travels through it, and not only at material interfaces. As scattering in general and natural phenomena based on it, such as smoke and crepuscular rays, serve as metaphors for the visualization methods in this thesis, we discuss scattering in more detail. In this section, we analyse the different ways in which light interacts while travelling through a participating medium and discuss the approaches to their rendering.

2.3.1 Light interaction with participating media

In order to derive relatively simple mathematical formulations for light-medium interactions, it is necessary to make some simplifying assumptions about the properties of the scattering media. The most widespread model assumes that the participating media consist of microscopic equally-sized particles. The individual photons are absorbed, reflected or emitted by these particles to form the final image on the camera film.

2.3.1.1 Intensity reduction

Assume a light source that emits photons towards the camera (see Fig. 2.9). In vacuum, the light intensity at the camera film can be easily computed directly by integrating over the surface of the light source. However, if a participating medium is present on the way from the light source towards the camera film, it can alter the light intensity reaching the film.

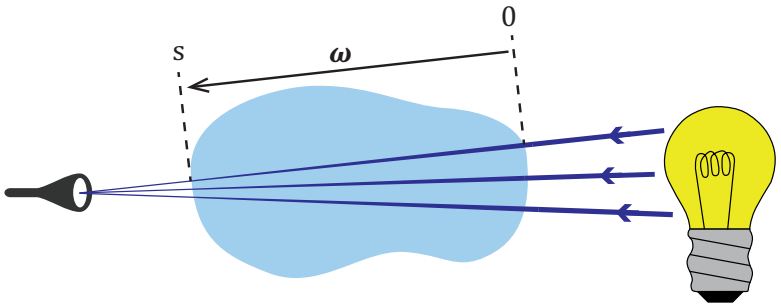


Figure 2.9: Rays from light travelling through participating medium. The ray within the participating medium is parametrized by distance from the entrance point 0 until the exit point s along its travel direction $\boldsymbol{\omega}$.

The amount of light *absorbed* by the medium depends on the density and size of particles that compose the participating medium. These properties can be summarized in the absorption coefficient σ_a of the medium, which denotes the fraction of absorbed photons when the light travels a unit length through the medium. Consider an infinitesimally thin slab of the medium that lies on the way of light travelling in direction $\boldsymbol{\omega}$ towards the camera. The light intensity $L(\mathbf{x} \rightarrow \boldsymbol{\omega})$ will be then decreased by the absorption within the participating medium by the factor depending on the absorption coefficient σ_a, which can be expressed in differential form as:

$$(\nabla \cdot \boldsymbol{\omega}) L(\mathbf{x} \rightarrow \boldsymbol{\omega}) = -L(\mathbf{x} \rightarrow \boldsymbol{\omega}) \sigma_a(\mathbf{x}), \tag{2.1}$$

where $(\nabla \cdot \boldsymbol{\omega})$ represents the directional derivative of L along $\boldsymbol{\omega}$.

In addition to being absorbed, the light can also be scattered in a different direction by the particles, leading to a similar light intensity reduction on the camera film. The fraction of the photons scattered by the medium per unit length is called the scattering coefficient σ_s, and the process itself is called *out-scattering*. Consequently, the overall fraction of light per unit length blocked by the participating medium can be computed as the sum of absorption and scattering coefficients, which is called the extinction coefficient:

$\sigma_t(\mathbf{x}) = \sigma_a(\mathbf{x}) + \sigma_s(\mathbf{x})$.

Furthermore, in a heterogeneous medium, the particle density and, consequently, the absorption and scattering coefficients are functions of position. Therefore, to compute the light intensity actually reaching the camera film without being obstructed, it is necessary to solve the differential equation 2.1 using the full extinction coefficient $\sigma_t(\mathbf{x})$:

$$L(s) = L(0) \cdot e^{-\int_0^s \sigma_t(\mathbf{x}(t)+\omega t)dt} \tag{2.2}$$

Here, the segment from the entrance to the exit point of the volume along the direction $\boldsymbol{\omega}$ is parametrized using the distance from the entrance point (see Fig. 2.9), and the second term

$$T(s) = e^{-\int_0^d \sigma_t(\mathbf{x}+\omega t)dt} \tag{2.3}$$

is the *transmittance*, which shows the actual fraction of light intensity that reaches the camera film. In the context of direct volume rendering, this quantity is usually referred to as *transparency*.

2.3.1.2 Intensity increase

In addition to the processes reducing the final light intensity reaching the camera film, the opposite processes also take place within the medium. The opposite of the absorption is the light *emission* process, and the opposite of the out-scattering is *in-scattering* process, wherein the light from various directions is deflected by the particles towards the observer.

Emission takes place when the internal energy of the medium is converted to light by emitting photons. This process can be described using an equation similar to Eq. (2.1):

$$(\nabla \cdot \boldsymbol{\omega})L(\mathbf{x} \rightarrow \boldsymbol{\omega}) = L_e(\mathbf{x} \rightarrow \boldsymbol{\omega})\sigma_a(\mathbf{x}), \tag{2.4}$$

where $L_e(\mathbf{x} \rightarrow \boldsymbol{\omega})$ is the intensity of light emitted at position $\mathbf{x}$ in direction $\boldsymbol{\omega}$.

Unlike out-scattering, for many participating media, in-scattering cannot be described by the means of a simple scalar coefficient, because the amount of light scattered in the direction towards the camera depends on the angle between the incident and scattered rays. This dependency is usually described by the means of a *phase function*, which is described in more detail below. If we consider, however, that the total light intensity scattered at position $\mathbf{x}$ in the direction $\boldsymbol{\omega}$ is $L_i(\mathbf{x} \rightarrow \boldsymbol{\omega})$, then the change in light intensity

due to in-scattering can be expressed as

$$(\nabla \cdot \boldsymbol{\omega}) L(\mathbf{x} \rightarrow \boldsymbol{\omega}) = L_i(\mathbf{x} \rightarrow \boldsymbol{\omega}) \sigma_s(\mathbf{x}), \tag{2.5}$$

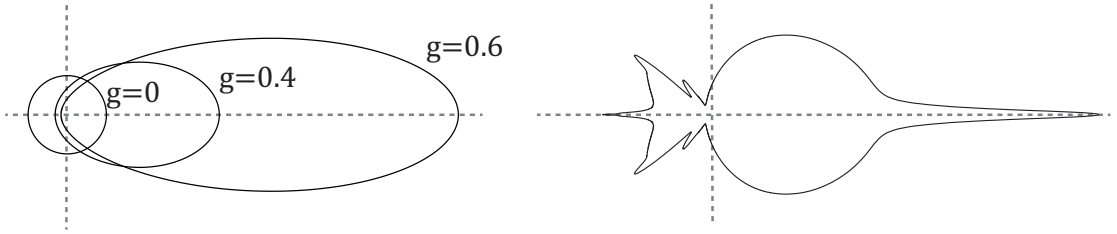


Figure 2.10: Henyey-Greenstein phase function for different values of parameter g (left, [22]). The isotropic phase function is a special case of Henyey-Greenstein function for $g = 0$. A realistic phase function for clouds (right, [15]).

A phase function for the spherical or randomly oriented particles depends only on the angle θ between the incoming and outgoing directions. The concrete form of a phase function is dictated by the size of the particles in relation to the wavelength of incoming light. The simplest case is an isotropic phase function, where the light is scattered equally in all directions (see Fig. 2.10, left), which is similar to fully diffuse reflection from a surface:

$$p_I(\mathbf{x}, \theta) = \frac{1}{4\pi}, \tag{2.6}$$

where the factor $\frac{1}{4\pi}$ comes from normalization over a whole sphere of directions. Note that we consider the phase function to be applied only to the portion of light that is actually scattered by the medium, which is defined by the scattering coefficient σ_s.

Another commonly used phase function model is the Henyey-Greenstein phase function [67]:

$$p_{HG}(\mathbf{x}, \theta) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos(\theta))^{\frac{3}{2}}}, \tag{2.7}$$

where the parameter $g \in [-1, 1]$ is used to smoothly interpolate between strong backward and strong forward scattering. It is easy to notice that if $g = 0$, the Henyey-Greenstein phase function corresponds to isotropic scattering (see Fig. 2.10, left).

Various other phase function models can be used to display different scattering media realistically, such as the Rayleigh scattering model for particles much smaller than the wavelength [110] or the Lorenz-Mie model [116] for particles of size comparable to the wavelength. However, for the purposes of direct volume rendering, where the displayed materials do not have a counterpart in the real world, isotropic phase function and Henyey-

Greenstien – or its less computationally intensive approximation, Schlick [19] – phase function provide sufficient realism.

2.3.1.3 The radiative transfer equation

Considering all the four possible interactions of light with participating media together, the summary change in light intensity at a position $\mathbf{x}$ can be expressed as:

$$(\nabla \cdot \boldsymbol{\omega})L(\mathbf{x} \rightarrow \boldsymbol{\omega}) = \quad -L(\mathbf{x} \rightarrow \boldsymbol{\omega})\sigma_a(\mathbf{x}) - L(\mathbf{x} \rightarrow \boldsymbol{\omega})\sigma_s(\mathbf{x}) \tag{2.8}$$

$$+L_e(\mathbf{x} \rightarrow \boldsymbol{\omega})\sigma_a(\mathbf{x}) + L_i(\mathbf{x} \rightarrow \boldsymbol{\omega})\sigma_s(\mathbf{x}), \tag{2.9}$$

which forms the radiative transfer equation (RTE) [28].

In order to render the final image, it is necessary to compute integral of equation (2.9) for each pixel constituting the image:

$$L(\mathbf{x} \leftarrow \boldsymbol{\omega}) = \quad T(s)L(\mathbf{x}_s \rightarrow -\boldsymbol{\omega}) + \tag{2.10}$$

$$\int_0^s T(t)\sigma_a(\mathbf{x}(t))L_e(\mathbf{x}(t) \rightarrow -\boldsymbol{\omega})dt + \tag{2.11}$$

$$\int_0^s T(t)\sigma_s(\mathbf{x}(t))L_i(\mathbf{x}(t) \rightarrow -\boldsymbol{\omega})dt, \tag{2.12}$$

where $L(\mathbf{x} \leftarrow \boldsymbol{\omega})$ is the light intensity reaching the image element, and $L(\mathbf{x}_s \rightarrow -\boldsymbol{\omega})$ is the light intensity at the entrance point to the medium, which for direct volume rendering applications is often equivalent to background color. This integral formulation is usually referred to as the *volume rendering integral*.

2.4 Methods for rendering participating media

Rendering the participating media with the full extent of the effects occurring within them, which corresponds to solving the full radiative transfer equation, requires tremendous computational efforts. Therefore, a variety of methods have been developed to incorporate these effects into off-line rendering systems. An excellent overview of these techniques can be found in a survey by Cerezo and colleagues [27].

As direct volume rendering considers the volumetric data to be artificial participating media, it is possible to employ these methods to display it to the user. However, as this thesis deals with visual encodings for volumetric data exploration, interactivity is an important requirement imposed on the rendering approaches. Therefore, it is necessary to make some simplifying assumptions about the properties of the involved participating

Figure 2.11: Deterministic rendering of crepuscular rays in the atmosphere (left, [42]), and stochastic Monte-Carlo rendering with radiance caching (right, [72]) with the inset showing difference between results without radiance caching (top right) and with it (bottom right).

media and the lighting configuration to limit the range of effects that can be achieved, thus reducing the amount of computations and subsequently improving the rendering performance. In this section, we review such approximations and the techniques that allow rendering participating media with these approximations at interactive frame rates.

2.4.1 Deterministic methods

In his seminal work, Marc Levoy proposed the classical volume rendering formulation, which in its essence computes light transport in participating media considering only emission, absorption and surface-like shading [105]. The simplification of the volume rendering integral is made in the in-scattering term: the light is considered to reach the scattering point without any obstructions and only directly from the light source. The amount of light scattered in the direction of observer is computed using a simple Phong surface illumination model [127] using the data gradient as the surface normal. This simplification allows computing the final pixel color using a Reimann sum simply by marching along the view ray with very few computations at each sampling point. Kajiya and Von Herzen propose to use a two-pass approach to include global illumination effects for rendering of volumetric datasets [80]. During the first pass, the radiance is estimated for each voxel, which is consecutively integrated along view rays during the second pass. However, at the time of the original publication, the computation time for rendering a single frame was still in the order of minutes and thus not applicable to interactive visualization of dynamic scenes.

Therefore, many approximations for improved computational performance were proposed. For example, ambient occlusion is used for efficient shadow computation within

a local neighborhood [39, 137, 147], and summed area tables provide an approximation including global shadowing [141]. Global illumination can also be approximated by a diffusion process [151]. Zhang and colleagues propose to use a convection-diffusion partial differential equation [178]. While the performance and range of effects achieved by this approach is high, it can only handle certain light types. Weber and colleagues apply an approach using virtual point lights (VPL) in interactive volume rendering [169]. Even though the resulting image is visually appealing, interactive visualization severely restricts the VPL number.

Introduction of various optimization techniques, such as early ray termination and empty space skipping [96, 106, 107], as well as rapid growth of GPU performance, has allowed various implementations which achieved interactive frame rates. Such approaches include texture-based volume rendering [25, 48, 131] and GPU-based ray casting [170]. With time, interactive rendering of large-scale datasets [16, 45, 57] and multiple datasets with polyhedral boundaries [14, 23, 77] has also become feasible.

Furthermore, it became possible to apply more sophisticated optical models for interactive volume rendering. The most widespread sequence of optical models of increasing complexity was summarized in the work by Max [112]. Starting with simplistic emission-only, absorption-only, and emission-absorption models, Max extends the mathematical formulations to include single scattering, shadows and multiple scattering, which were used by various approaches computing complex global illumination in direct volume rendering. A half-angle slicing approach allows approximating multiple forward scattering from a single light source for texture-based volume rendering [89]. Similarly, directional [142] and multi-directional [150] occlusion shading approximate multiple forward scattering from a headlight or a light positioned within a hemisphere defined by the view vector, while simultaneously avoiding artefacts introduced by half-angle slicing. Spherical harmonic lighting [108] allows interactive rendering of materials with arbitrary phase functions with colored area light sources. Numerous other approaches were proposed for computing advanced volumetric illumination in interactive volume rendering, and we refer the reader to the state of the art report by Jönsson and colleagues for their classification and in-depth analysis [76].

2.4.1.1 Phenomenon-specific methods

For rendering some of the effects arising from scattering in participating media, specific deterministic algorithms were developed. For example, computer graphics solutions for

rendering crepuscular rays were introduced in research by Max [113], who first computed these rays for photorealistic rendering, and by Nishita *et al.* [123], who invented the airlight integral to compute the effects of light scattering by air particles. Instead of computing volumetric light rays, volumetric shadows can also be computed, as demonstrated by Baran *et al.* [10]. The corresponding computer graphics research question originates in the computation of exact from-region visibility as researched by Nirenstein *et al.* [122]. Due to the high complexity of this problem, it is most often approached by sampling the region, as performed, for example, by Wonka *et al.* [173].

2.4.2 Stochastic methods

In addition to deterministic approaches discussed above, stochastic methods represent a distinct body of work on rendering, capable of displaying complex interaction of light with participating media. The main distinction of stochastic rendering approaches lies in the numerical integration of the whole volume rendering integral or its parts, which is done in a stochastic manner. Some of these approaches were also applied for interactive direct volume rendering tasks.

The idea behind Monte-Carlo integration is to compute the integrand on a set of samples randomly chosen from the domain and then to estimate the integral value by averaging the obtained results. This process is guaranteed to converge to a true solution due to the *Strong Law of Large Numbers*. The main advantage of such approach is that the amount of computations required to achieve the desired integration accuracy, when the dimensionality of the domain is high, is much lower than for deterministic quadrature-based methods, which is the case for rendering of the participating media. Using this advantage, Rezk-Salama used the Monte-Carlo approach to compute physically-based scattering for direct volume rendering and showed high-quality renditions of set of isosurfaces obtained from volumetric data [139]. More recently, Kroes and colleagues demonstrated that a full progressive Monte-Carlo approach is feasible for interactive direct volume rendering with a complex global illumination model supporting area lights and arbitrary phase functions [95]. We use their framework as the base for our method presented in Chapter 3 and therefore, we describe it in more detail therein.

Photon mapping approaches [73] share similar traits with Monte-Carlo approaches, because they also use random sampling to speed up the computation of the full volume rendering integral. The photon mapping approaches generally consist of two stages – photon tracing and photon gathering. In the photon tracing stage, the paths of individual

photons are traced from the light sources until they are scattered within the scene. The photons are then stored and reused to query the lighting information by estimating the photon density during the gathering stage, which can be done using either deterministic or Monte-Carlo rendering. As both stages may be computationally demanding, the trade-off between them is necessary to use the photon mapping approaches in interactive systems. In the context of interactive DVR, Jönsson and colleagues propose Historygrams to increase the speed of photon map re-computation for transfer function changes [74]. The photon gathering step is still quite time consuming, leading to low frame rates when the camera moves. Precomputed radiance transfer approaches [149] allow to overcome the high runtime cost. Zhang and colleagues propose to use precomputed photon maps for high quality interactive volume data visualization [177]. While the rendering performance is very high, full re-computation of the photon map is still required upon a transfer function change.

2.4.2.1 Irradiance and radiance caching

Among other approaches to accelerating the Monte-Carlo rendering, such as multiple importance sampling [36, 97, 162], irradiance caching techniques have proven to be very effective. The irradiance caching technique [167] takes advantage of the fact that the indirect irradiance field is mostly smooth. The irradiance and some additional quantities, such as the irradiance gradient [166], are computed for a sparse set of cache points, which are then used during rendering to interpolate the irradiance, avoiding costly integration over all directions. Křivánek and colleagues proposed radiance caching which stores and interpolates direction-dependent radiance using spherical harmonics [100]. Later, Jarosz and colleagues extended the radiance caching approach for use with participating media [72].

2.5 Summary and context

Natural phenomena received a lot of attention from the researchers in computer graphics and data visualization communities. In both fields, the goal is improving the realism of the computer generated pictures to either be more visually pleasing or improving the perception of the data. There exists a significant body of work that makes rendering of a wider variety of natural phenomena possible as well as increases the performance of existing approaches.

In this thesis, we present contributions in two different directions. On the one hand, in chapter 3, we show how the existing irradiance caching approaches can be modified to be brought to high-performance interactive DVR. These contributions go in line with the other research aimed at improving realism in data visualization by considering more complex natural phenomena. On the other hand, in chapters 4 and 5, we propose to look at the natural phenomena in visualization from a novel perspective. We use them as metaphors for developing the new visualization approaches, which convey the information in a very effective way by using visual forms familiar to the human eye.

Chapter 3

Scattering simulation

Contents

In this chapter, we explore a very literal approach at using the natural phenomena in visualization. We make the next step in bringing realistic rendering to interactive DVR by allowing parallel creation of irradiance cache on GPU, which allows efficient rendering of scattering in participating media.

3.1 Motivation

As discussed in section 2.4.2, stochastic methods allow achieving very high quality complex global illumination effects that can be used for the purpose of direct volume rendering. Monte-Carlo Volume Rendering (MCVR) is particularly suitable for this task, because the rendering can be done in a progressive manner. This allows showing intermediate results, thus enabling interactive data exploration. However, the first few iterations may be very noisy, which forces the user to suspend interaction until higher levels of convergence are achieved.

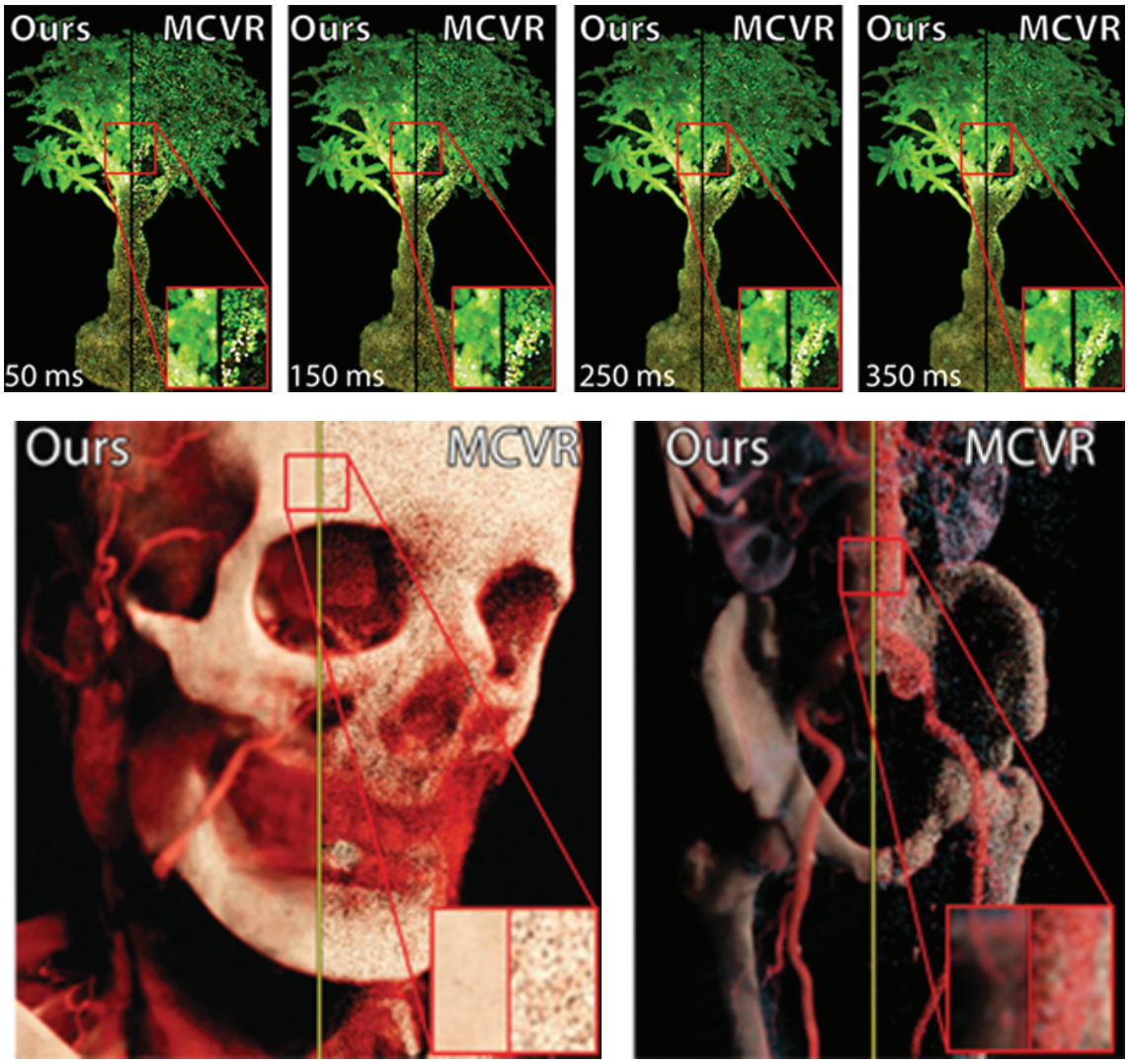


Figure 3.1: Three volumes with enlarged areas rendered with Irradiance Caching (left half) and normal Monte Carlo Volume Rendering (right half) after the same time. The enlarged areas show a significant reduction in noise with our approach.

Irradiance [167] and radiance [100] caching techniques significantly improve the convergence rate by storing and reusing the illumination computation results for nearby pixels. This does not reduce the quality, because the illumination field varies smoothly in large parts of natural scenes. An additional advantage of such caching techniques is that the cache can be built during rendering – once we encounter a position within the scene not covered by the cache, we compute a new cache entry and then proceed with rendering. However, for interactive DVR, multiple problems prevent using traditional techniques.

First, the majority of modern DVR systems employ the GPU to achieve interactive

frame rates. The massive parallelism required to effectively utilize the power of the GPU is achieved by dedicating one thread per pixel and executing the threads in the SIMD manner in thread blocks. Thus, if all the threads within a thread block compute a similar cache entry at the same time, a lot of computational effort is wasted since many redundant entries are created. Second, even if the cache density is controlled, displaying the results for the thread block will stall until all the necessary cache entries are created, which affects interactivity significantly.

In this chapter, we propose an integrated approach that allows to build the irradiance cache in parallel to rendering on a single GPU, without leading to excessively dense caches or rendering stalls. To achieve compatibility with most DVR applications, we formulated the following goals:

- Irradiance caching should not interfere with the interaction and the visualization, allowing for efficient data exploration. This includes both virtual camera motion as well as interactive transfer function design.
- Caching should improve the convergence rate of the MCVR without sacrificing its quality.
- Memory footprint of the cache should be minimal.

These goals are contradictory to some extent. For instance, a very low memory footprint may be achieved at the expense of quality or interactivity. Achieving a good balance in a GPU-friendly way is not trivial, as multiple interlocking tasks compete for the GPU time and must be scheduled. In this regard, we present the following contributions:

- We demonstrate object- and screen-space approaches which allow the new cache requests to be handled in a massively parallel MCVR system without rendering stalls or creating excessively dense cache on a single GPU.
- We show a prioritization scheme that allows to distribute the available computational power between building the irradiance cache and the actual rendering.
- We propose a modification to the exponential irradiance extrapolation approach, which is more accurate for scenes with high irradiance field gradients.

Overall, we show that our method improves the convergence rate and, consequently, the visualization quality of MCVR, without reducing its interactivity.

3.2 Background

In this section, we describe the works particularly relevant to this chapter. More specifically, we discuss the specific implementation details of creating the irradiance cache in parallel and its storage as well as the Exposure Render progressive Monte-Carlo volume rendering framework, which we build upon.

3.2.1 Irradiance cache implementation

There have been attempts at bringing irradiance caching to parallel systems, mostly for multiple nodes using MPI. Robertson and colleagues [135] propose distributing cache generation over several nodes and use frame to frame coherence for geometric scenes to reduce the computational demand. However, single nodes still compute entries for screen regions serially, which limits the overall parallelism of the approach. In addition, cache synchronization among nodes is problematic. More recently, Debattista and colleagues [38] propose separating rendering from irradiance calculations. They share parts of the locally computed cache of single nodes at particular time intervals, which leads to issues with latency and cache misses for entries computed on a different node. This has also been observed by Kohalka and colleagues [92]. Gautron and colleagues proposed the radiance cache splatting approach to enable efficient use of the GPU for rendering with radiance cache [56]. Debattista and colleagues propose a wait-free data structure for populating irradiance cache on parallel systems with shared memory [37]. However, both methods create the cache entries sequentially for the whole screen or a large tile and do not handle participating media.

The shape of the influence zone of cache entries affects the memory footprint necessary to achieve high quality results. We need to consider the total number of cache entries and the amount of data stored for each of them. The most widespread shape is spherical [72]. The goal of storing very little information for each cache entry is affected, if the cache density increases significantly in the areas of high frequency illumination changes. Furthermore, spherical influence zones cause excessive cache density in all directions, even if irradiance changes rapidly in only one direction. Therefore, an adaptive shape of the influence zone is more suitable for highly inhomogeneous media, which, for instance, can be observed in direct volume rendering of scientific data. Ribardière and colleagues proposed adaptive records for irradiance caching methods, applied to surface data [132] and volume data [133] rendering, where the influence zones are adapted according to geometrical features and irradiance changes.

3.2.2 Exposure Render

We use the *Exposure Render* method presented by Kroes and colleagues [95]. The Exposure Render system applies the Monte Carlo approach to solve the radiance transfer equation for the purpose of interactive progressive volume rendering.

The solution of the radiative transfer equation may be obtained using a Monte-Carlo approach. In the Exposure Render framework, it is implemented as follows:

- For each pixel on the screen, a single ray is cast into the scene through a random position within this pixel.
- Each ray propagates through the volume to find a single tentative collision point (scatter event).
- For this event, the illumination is estimated by casting a ray towards a randomly selected sample located at one of the lights within the scene. Similarly to the previous step, if a collision point with the medium is found, the point is considered unlit. Otherwise, this light sample in conjunction with sampling a bidirectional reflectance distribution function (BRDF) and/or a phase function is used to compute the illumination and considered in the current estimate for this pixel.
- The current estimate is incorporated into the final pixel color using a running average approach.

In this chapter, we accelerate the illumination estimation using an irradiance cache. For isotropic phase functions, the irradiance can be computed as the integral of the incident radiance over the sphere of directions:

$$E(\mathbf{x}) = \int_{\Omega} L(\mathbf{x}, \boldsymbol{\omega}) d\boldsymbol{\omega} \tag{3.1}$$

3.3 Parallel irradiance cache management

The main objective of our approach is to generate an irradiance cache in a massively parallel way, while concurrently performing DVR on a single GPU. As alternating between DVR and cache management would introduce a considerable latency, we want to *schedule* all involved tasks concurrently. In this way, we can provide immediate feedback to camera movements using progressive updates, while building and adjusting the irradiance cache at the same time.

To enable this goal, we distinguish between three procedures, which are executed on the GPU in parallel (Fig. 3.2):

The rendering procedure implements the basic Exposure Render approach with a modified illumination computation step. For each scattering event, we check whether cache information is available. If so, we extrapolate the irradiance and use it for shading. If not, we generate a cache entry request and use default MCVR shading instead.

The cache creation procedure handles incoming cache entry requests. It computes new cache entries by sampling the irradiance field in the local neighborhood of the cache request center.

The cache update procedure eliminates discontinuities in the estimated irradiance field. This discontinuities can arise in the areas of high frequency changes in the irradiance field, which are not captured by the values computed during cache entry creation.

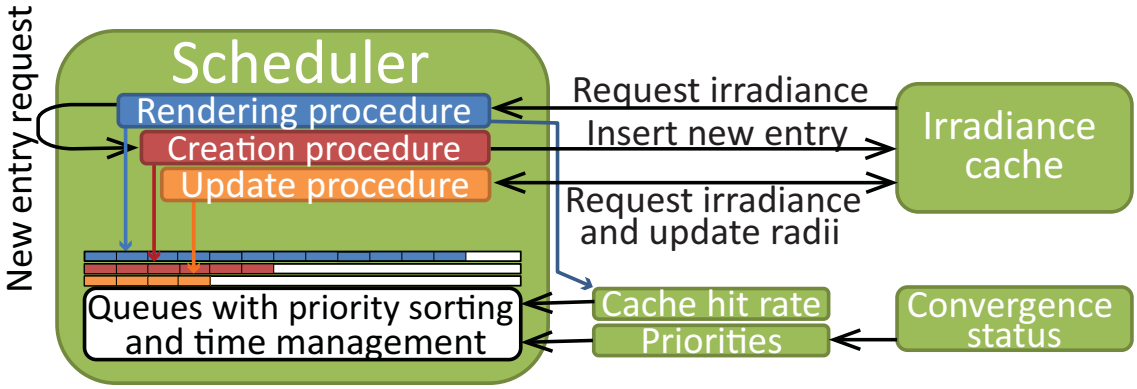


Figure 3.2: Overview of our system. Three procedures run in parallel on a single GPU. The scheduler distributes time slots for each of them, based on the cache hit rate. The rendering procedure is also prioritized, based on screen-space convergence information.

3.3.1 GPU Scheduling

Using the traditional execution model based on GPU shaders or SIMD languages like CUDA, a dynamic concurrent execution cannot be set up easily. Even with the most recent *dynamic parallelism*, which allows kernel launches from the GPU, we were unable to implement concurrent caching and rendering efficiently. The overhead of dynamic

parallelism itself and of finding a mapping between launched threads and actual work was too high in our experiments. Thus, we use *Softshell* [152], an open-source framework, which occupies the GPU to provide custom scheduling in software.

Softshell uses a *persistent megakernel* built on top of CUDA. In this approach, a single kernel continuously occupies the GPU, with each thread spinning in a loop. An idle thread draws a new task from a queue implemented in software. The queues support parallel insertion and removal using a fixed-size ring buffer with atomically operated front and back pointers and overflow/underflow protection. Individual slots are assigned to threads with atomic additions on these pointers. This design provides simultaneous access to an arbitrary number of threads without waiting.

To allow for different scheduling policies, we associate an individual queue with each procedure as shown in Figure 3.3. Depending on which queue is chosen for dequeuing, different scheduling policy can be employed. For instance, different quotas can be assigned to different procedure types. Moreover, queues can be sorted to reflect priorities within one procedure type, enabling an out-of-order execution.

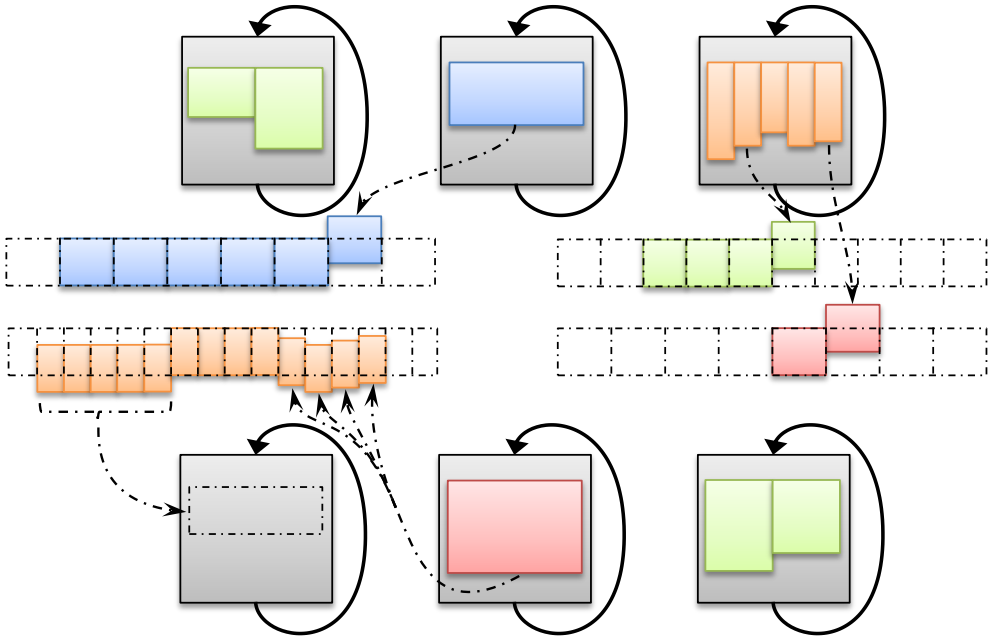


Figure 3.3: Our approach consists of worker-blocks, continuously drawing tasks from queues. We keep one queue per procedure, which is essential for a divergence free execution of tasks with different granularities.

The priority queues offer the possibility to influence the execution order to, e.g., control

the amount of resources invested into each individual procedure or focus processing power on more important image regions. As the persistent threads megakernel can be controlled from outside, it is possible to interrupt the GPU execution at a certain point in time. We use this feature to enable a frameless rendering approach [18] aiming at a fixed refresh rate.

3.3.2 Cache entry creation

We take the following steps to create a single cache entry:

- Compute the local coordinate frame
- Estimate the incoming irradiance at the center and at six points offset along the axes of the local coordinate frame
- Compute validity radii for each of the six axis directions

Below, we first show how we estimate the irradiance at the necessary positions. Next, we show how to extrapolate the irradiance to arbitrary positions. Then, we explain how we choose validity radii. Finally, we justify the selection of the local coordinate frame.

Estimation of irradiance values. The incoming irradiance is estimated with Monte Carlo integration using multiple importance sampling with power heuristics [161]. We control the number of samples N for the Monte Carlo integration using the standard deviation based error estimate with 95% confidence interval [69]:

$$1.96\sqrt{\frac{S^2(N)}{N}} < \frac{\gamma}{\gamma+1}E(N), \tag{3.2}$$

where γ is the acceptable relative error. $E(N)$ and $S^2(N)$ are the average and the variance of irradiance estimates, which are computed incrementally [54]. Note that we use the relative error estimate due to the fact that the human eye is most sensitive to relative and not absolute values of illumination change according to Weber's law of just noticeable differences [155, p. 37]. Therefore, larger absolute errors are admissible for brighter regions.

Irradiance extrapolation and validity radius. We assume that the irradiance changes exponentially in the local neighborhood of the cache entry center $\mathbf{c}$, similarly to Jarosz and colleagues [72]. Using the error metric shown in Eq. (3.2), we estimate the

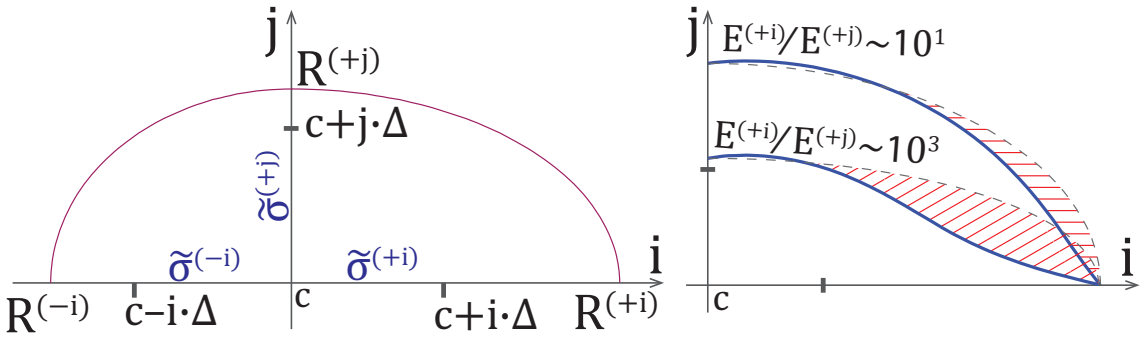


Figure 3.4: (Left) We compute the tentative extinction coefficients $\tilde{\sigma}$ as well as radii R for six directions $(\pm\mathbf{i}, \pm\mathbf{j}, \pm\mathbf{k})$. (Right) When interpolating local extinction coefficients, the ellipsoidal shape may be inconsistent with the actual validity radius if the ratio of irradiances at different axes is very high (inner graph). Therefore, to avoid visual artefacts, we discard the influence of a cache entry in areas beyond the interpolated validity radius (red hatching). However, for lower ratios (outer graph), the ellipsoidal shape is a good estimate for the actual validity radius.

irradiance values at $\mathbf{c}$ and at positions $(\mathbf{c} + \xi \cdot \Delta)$ for $\xi \in (\pm\mathbf{i}, \pm\mathbf{j}, \pm\mathbf{k})$, where $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ are the unit vectors of the local coordinate frame at $\mathbf{c}$, and Δ is the computation offset (see Fig. 3.4, left). Then, for each of six directions, we fit a 1D exponential function of form $E(x) = E(\mathbf{c}) \cdot \exp(-\tilde{\sigma}_t x)$, where $\tilde{\sigma}_t$ is the tentative local extinction coefficient, using two computed data points $\{0; E(\mathbf{c})\}$ and $\{\Delta; E(\mathbf{c} + \xi \cdot \Delta)\}$:

$$\tilde{\sigma}_t^{\xi} = -\ln\left(\frac{E(\mathbf{c} + \xi \cdot \Delta)}{E(\mathbf{c})}\right) \cdot \frac{1}{\Delta} \tag{3.3}$$

To perform the extrapolation to an arbitrary position $\mathbf{p}$, we first compute its coordinates $\mathbf{p}' = (p'_{\mathbf{i}}, p'_{\mathbf{j}}, p'_{\mathbf{k}})$ in the local coordinate frame. Then the tentative local extinction coefficient in the direction $\mathbf{p}'$ is computed using barycentric coordinates:

$$\tilde{\sigma}_t^{(\mathbf{p}')} = \frac{|p'_{\mathbf{i}}| \cdot \tilde{\sigma}_t^{(\pm\mathbf{i})} + |p'_{\mathbf{j}}| \cdot \tilde{\sigma}_t^{(\pm\mathbf{j})} + |p'_{\mathbf{k}}| \cdot \tilde{\sigma}_t^{(\pm\mathbf{k})}}{|p'_{\mathbf{i}}| + |p'_{\mathbf{j}}| + |p'_{\mathbf{k}}|}, \tag{3.4}$$

where the sign of $(\pm\mathbf{i}, \pm\mathbf{j}, \pm\mathbf{k})$ corresponds to the sign of $(p'_{\mathbf{i}}, p'_{\mathbf{j}}, p'_{\mathbf{k}})$. Finally, the extrapolated irradiance value is computed as $E(\mathbf{p}') = E(\mathbf{c}) \cdot \exp(-\tilde{\sigma}_t^{(\mathbf{p}')} |\mathbf{p}'|)$.

Given this extrapolation approach, we compute the validity radius for each half-axis such that the extrapolated irradiance at the edge of the influence zone differs by not more

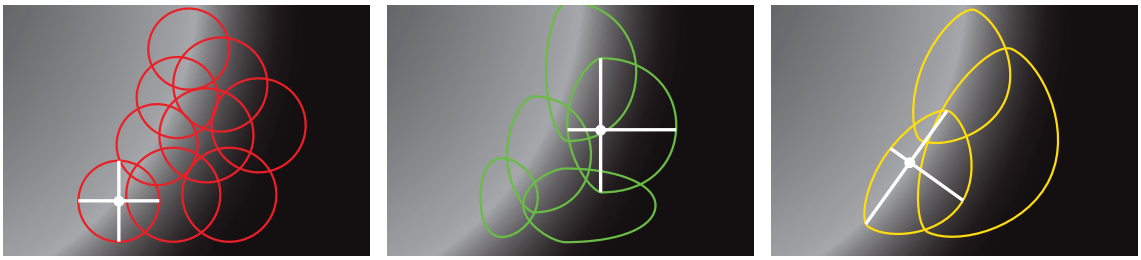


Figure 3.5: Influence of the entry shape on the cache density. The white lines denote the radii corresponding to the local coordinate frame of an entry. If the entry is represented as a union of elliptical sections, (middle) less cache entries are required compared to the spherical ones (left). Orienting these shapes along the opacity gradient (right) reduces the cache density even more because in many cases, the largest irradiance gradient coincides with the opacity gradient. Orienting the entries allows for larger validity zones, since the radius has to be small only along one axis.

than one ϵ relative to the estimated irradiance at position $(\mathbf{c} + \xi \cdot \Delta)$:

$$R^{\xi} = \Delta \cdot \frac{\ln(E_{\epsilon}/E(\mathbf{c}))}{\ln(E(\mathbf{c} + \xi \cdot \Delta)/E(\mathbf{c}))}, \tag{3.5}$$

where

$$E_{\epsilon} = \begin{cases} (1+\epsilon)E(\mathbf{c} + \xi \cdot \Delta) & \text{if } E(\mathbf{c}) \leq E(\mathbf{c} + \xi \cdot \Delta) \\ (1-\epsilon)E(\mathbf{c} + \xi \cdot \Delta) & \text{if } E(\mathbf{c}) > E(\mathbf{c} + \xi \cdot \Delta). \end{cases} \tag{3.6}$$

We define the shape of the cache entries as a union of elliptical sections with distinct radius along each of half-axes of the local coordinate frame (Fig. 3.4, left). While an ellipsoid is a good estimate, the actual validity radius for the interpolated local tentative extinction coefficient may be smaller than the corresponding ellipsoid radius. With our exponential extrapolation approach, this may lead to errors in case the ratio of the estimated irradiance along the half-axes is large (Fig. 3.4, right). Therefore, we discard the influence of a cache entry if the point lies outside the validity radius of the interpolated local tentative extinction coefficient.

If a point lies within the validity zone of multiple cache entries, we compute the log-space weighted average of the per-entry extrapolation results [72]:

$$E(\mathbf{p}) \approx \exp\left(\frac{\sum_{k \in C} \ln(E_k) w(d_k)}{\sum_{k \in C} w(d_k)}\right) \tag{3.7}$$

where the weight is computed as $w(d) = 3d^2 - 2d^3$ with $d_k = 1 - \|\mathbf{p'}_k\|/R_k(\mathbf{p'}_k)$, $\mathbf{p'}_k$ are

the coordinates of the point in the local coordinate frame of the cache entry k, and E_k is the irradiance extrapolated using the cache entry k. Since the shape of cache entry in each octant is an ellipsoid section, the radius is computed as:

$$R(\mathbf{p}') = \|\mathbf{p}'\| \cdot \left((p'_{\mathbf{i}}/R_{\mathbf{i}})^2 + (p'_{\mathbf{j}}/R_{\mathbf{j}})^2 + (p'_{\mathbf{k}}/R_{\mathbf{k}})^2 \right)^{-0.5}$$

Local coordinate frame. To minimize the effect of errors introduced by the 3D interpolation of the local tentative extinction coefficient, it would be preferable to orient one of the axes of a cache entry's local coordinate frame along the irradiance gradient. The exact orientation would require knowing the gradient in advance. This is a very expensive computation, requiring to store the local coordinate frame with each cache entry. However, we have observed that in many cases the largest change in irradiance is related to the variation of the extinction coefficient. Therefore, orienting the local coordinate frame along the gradient of the extinction coefficient is a good approximation of the optimal orientation. Furthermore, as the computation of extinction coefficients is deterministic and can be done using a simple central differences approach, we can easily recompute it on demand rather than expend additional memory for its storage.

We have compared the cache size for spherical, as well as non-oriented and oriented pseudo-ellipsoidal influence zones. The results have shown that spherical zones indeed lead to very high cache densities (more than four times larger). Oriented and non-oriented zones in many cases produce very similar cache sizes. However, we observed that the cache size for oriented zones was 5-10% smaller than that of the non-oriented ones for optically dense materials. Because using oriented zones does not require additional storage and little computational effort, we use oriented cache entries throughout the remainder of this paper.

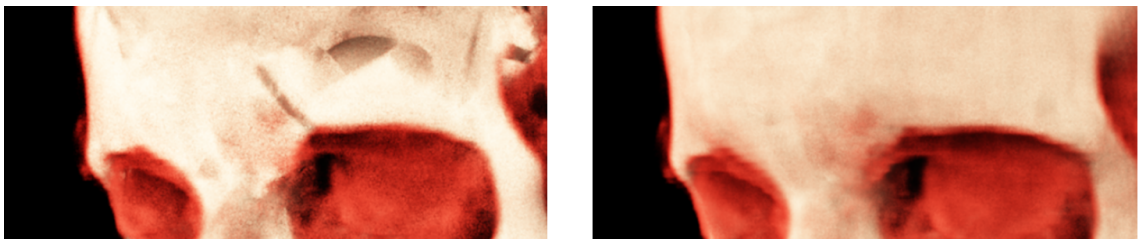


Figure 3.6: (left) Visible artefacts can be produced if extrapolated irradiance does not capture the actual irradiance field. (right) The update procedure removes such artefacts.

3.3.3 Cache update

While computing the validity radius gives a good approximation for the cases where the irradiance varies smoothly, it can still be erroneous to a certain degree (Fig. 3.6, left). Deviations result from considering global illumination information only partially during the estimation of incoming irradiance in the local neighborhood of the cache entry centre. To avoid the resulting artefacts, we incorporate a cache update procedure into our system. The cache update is done in a similar way as proposed by Křivánek and colleagues [98].

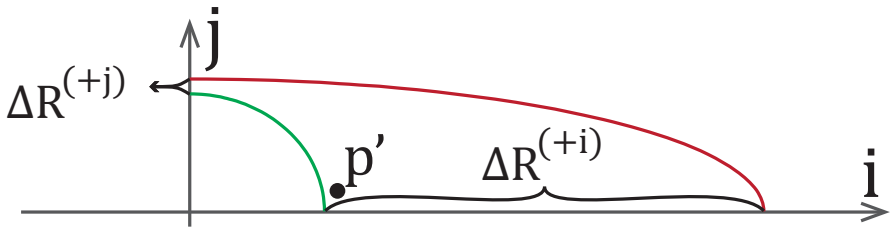


Figure 3.7: Reduction of radii in the local coordinate frame of the cache entry to exclude point $\mathbf{p}'$ from the validity area in the corresponding octant of the cache entry. The radii are reduced proportionally to the coordinates of point $\mathbf{p}'$ in the local coordinate frame. In this case, $|p'_{\mathbf{i}}|/\Delta R^{(+\mathbf{i})} = |p'_{\mathbf{j}}|/\Delta R^{(+\mathbf{j})}$.

The update procedure searches for scatter events in the same way as the main rendering procedure. Whenever the predictions of overlapping cache entries conflict at the scatter event position, the contribution of the cache entry with the lowest weight is removed from the computation by reducing its radii in the corresponding octant. The radii are reduced proportionally to the event's coordinates in the local coordinate frame (see Fig. 3.7), i.e. , $|p'_{\mathbf{i}}|/\Delta R_{\mathbf{i}} = |p'_{\mathbf{j}}|/\Delta R_{\mathbf{j}} = |p'_{\mathbf{k}}|/\Delta R_{\mathbf{k}}$. We repeat this process, until all conflicts are resolved, with the special case of only one remaining influence.

3.3.4 Cache entry storage

We store the cache entries in a multi-reference octree, similarly to Křivánek and Gautron [99]. Even though the multi-reference octree requires additional storage space, its performance is significantly higher than that of its single-reference counterpart [81]. Since we only need to store additional references rather than full entries, the impact on the memory footprint is small.

We use oriented bounding boxes enclosing the cache entries to find the nodes for which to store a reference. Additionally, as the update procedure may change the extent of a cache entry influence zone and reduce the number of influenced octree nodes, we rebuild

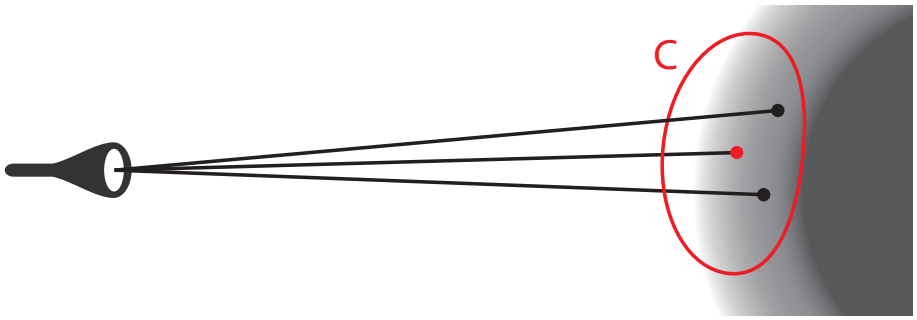


Figure 3.8: Neighboring rays may issue cache entry creation requests at very similar positions. We avoid redundant cache entries by allowing only one cache entry to be created per octree node. Once a cache entry C is created, creation requests within the influence of C will be ignored.

the octree from scratch at particular time intervals (Table 3.1). This overall reduces the total number of references (Section 3.6).

3.4 Parallelization of cache entry computations

Neighboring rays are likely to issue a scattering event at similar positions, especially if they encounter optically dense material (Fig. 3.8). This may lead to too dense and redundant cache regions despite low frequency irradiance variation.

To solve this issue, we evaluated two different approaches. Method 1 forbids simultaneous creation of more than one entry per octree node. Since the octree resides in the object-space of the volume, we call this approach *object-space locking*. Method 2 is based on the fact that the creation of new cache entries is driven by image-based ray generation. Hence, locking parts of the screen in form of superpixels is an alternative. We call this scheme *screen-space locking*.

For both versions, upon requesting a new entry, we first try to lock the corresponding primitive (node or superpixel) using atomic operations, similar to Debattista and colleagues [37]. Upon success, we issue the request for creating a new cache entry, and, as soon as its estimation has finished, we insert it into the octree and release the lock. Requests for already locked primitives are discarded and the sample of the scattering event is shaded with MCVR.

3.4.1 Object-space locking

In this technique, each octree node can be separately locked. If we cannot extrapolate the irradiance from the cache, we check whether we can lock the node furthest down in the branch enclosing the event position. Improving over the method of Debattista and colleagues [37], however, our approach also allows parallel, asynchronous determination of the positions where new cache entries need to be created.

The effects of this approach are twofold. First, it allows parallel creation of multiple cache entries in distant parts along a single ray. Second, the octree adapts its local resolution to cache density. (Fig. 3.9).

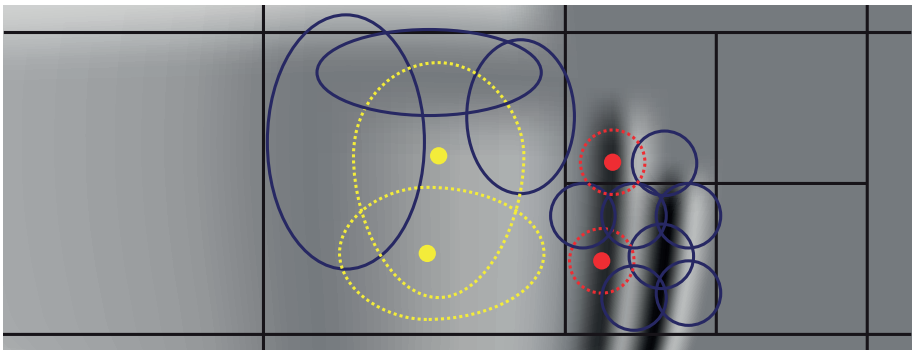


Figure 3.9: As the octree nodes are small in the areas of high cache density, more cache entries can be created in these areas in parallel. This does not lead to the creation of redundant cache entries, because the new entries are likely to have small radii as a high cache density suggests high frequency of irradiance change in this area. In this illustration, the simultaneous creation of two entries, shown in red, will be allowed, while it will be forbidden for the two yellow ones, even though the distance between them is the same.

3.4.2 Screen-space locking

In contrast to the aforementioned method, *screen-space locking* operates on rectangular regions of the screen (superpixels). For an optimal distribution of requests, we use two restrictions. First, each superpixel may only issue one request at a time. We limit the total number of cache entry requests throughout the whole screen to balance workload.

Only a fraction of all rays in a superpixel are actually in need of a new entry, since some may have encountered available cache information in a particular iteration. The ratio of requests to total rays bears information on the projected cache distribution in the current region. We only issue creation of a new cache entry, if a randomized rejection test based on the ratio passes. This method allows for accessing the same node of the octree

from different superpixels on the screen. Consecutively, for preventing conflicts, we use atomic operations on the reference list per node.

3.5 Priorization

To control the assignment of available processing power to procedures, we use a two-level priorization. First, we dynamically adjust the ratio of processing time spent on each of the three procedures based on the cache hit rate. Second, we use different priorities for the individual image regions during the rendering procedure to direct processing power towards parts further from convergence.

Cache status directed scheduling. If the number of cache entries ensures that the majority of lighting computation requests can be extrapolated from the cache, it is not necessary to allocate time for creating additional cache entries. To achieve this goal, we dynamically adjust the time frame assigned to the individual procedures based on the cache hit rate (h). We estimate h over a small time interval to consider multiple depths for each pixel. The time frames are then computed as: $t_c = t_{c,min} + (1 - h^{e_c}) \cdot (t_{c,max} - t_{c,min})$, where t_c corresponds to the relative time frame assigned to the cache creation procedure, while $t_{c,min}$ and $t_{c,max}$ are the minimum and maximum time that should be used for cache creation, respectively. e_c allows to control a non-linear translation from hit rate to assigned time. Based on the intuition that the number of required cache updates should be proportional to the number of newly created cache entries, we use the same formula with different minimum and maximum times for the cache update procedure. The relative time spent on rendering simply corresponds to the remaining time (parameters given in Table 3.1).

Rendering image priorities. Based on the geometric relations of the scene and lighting, the convergence rate of individual parts of the image may differ strongly. For instance, a region completely in shadow may converge to black within a single iteration, while complex light interactions from multiple light sources will result in very slow convergence rates. Thus, we want to provide a more uniform convergence rate across the entire image. We do that by estimating the expected gain from running another iteration per region. If we assume that the sample variance does not change after an additional iteration, then the

error estimate will be reduced by

$$\Delta E = 1.96 \left(\sqrt{\frac{S^2(N)}{N}} - \sqrt{\frac{S^2(N)}{N+1}} \right) \tag{3.8}$$

Using ΔE, we compute the average expected error reduction for each block and use it directly as the rendering priority.

3.6 Implementation

As mentioned in section 3.3, an essential part of our approach is the concurrent execution of cache creation, cache update and rendering. Using the Softshell [152] scheduling framework, we generate and manage work descriptors for all three procedures on the GPU. Each procedure is run in blocks of 128 threads.

The *rendering procedure* divides the screen into blocks of 16×8 pixels and performs ray casting with one pixel per thread. A sufficient number of blocks are created initially to cover the entire screen. For each scattering event, we query information from the irradiance cache. In case no entry exists, we use either of the locking methods described in Section 3.4. In case a new request should be generated, we generate a work descriptor for the cache creation procedure and hand it over to the scheduling framework. After all threads of the rendering procedure have added their contribution to each pixel estimate, we hand the work descriptor for this screen block back to the scheduling framework. In this way, rays for the same screen region will again be traced at a later point in time, implementing the progressive rendering method.

All 128 threads in the *cache creation procedure* work on a single cache entry. Each thread casts seven rays towards randomly selected lights – one ray for the central position and one for each of the six positions offset in both directions of the axes of local coordinate frame. Then, the results are combined using shared memory. If the stopping criterion is met (Eq. (3.2)), the newly created cache entry is inserted into the octree, which uses atomic operations to avoid potential race conditions. Otherwise, the process is repeated. The work descriptor is simply removed after execution.

Similar to the rendering procedure, the *cache update procedure* also starts with thread blocks covering the entire screen. Each thread randomly selects one of the covered pixels and casts a ray into the scene. At the scatter event, the procedure checks whether cache entries are in conflict with each other and reduces radii if necessary. The random selection of a pixel avoids updating the cache in neighboring object-space positions. When the

Table 3.1: The parameter values used in our experiments

Cache entry creation and update		
Estimation accuracy	γ	0.1
Validity radius threshold	ϵ	0.1
Position offset	Δ	2 voxel
Minimum influence zone radius	-	0.01 voxel
Maximum influence zone radius	-	16 voxel
Cache entry update threshold	-	0.1
Procedure time distribution		
Max. creation procedure fraction	$t_{c,max}$	15%
Min. creation procedure fraction	$t_{c,min}$	5%
Max. update procedure fraction	$t_{u,max}$	2%
Min. update procedure fraction	$t_{u,min}$	1%
Exponent for fractions update	e_c	3
Screen-space locking		
Max.# simult. created cache entries	-	1000
Max.# simult. created entries/superpixel	-	1
Miscellaneous		
Octree rebuild interval	-	1000 ms
Cache hit rate update interval	-	100 ms

entire block has finished, the block is re-emitted for scheduling. In this way, cache entries for the entire image are consistently checked and improved.

To incorporate execution time into the priority scheduling process, we measure the number of cycles of each individual procedure execution using the clock cycle counter provided by CUDA. We then update the current estimate of the overall time spent on each of the three procedures using atomic addition. Using these measurements we update the procedure priorities, and steer the scheduling in such a way that they receive predefined portions of the overall available processing time. For instance, we could use 20% for cache creation and 80% for rendering.

3.7 Results

We evaluate several conditions of our proposed method in comparison to plain MCVR. In the following, we discuss cache creation in static scenes and during interaction, and also evaluate convergence rates for a fully built cache. Furthermore, we provide measurements

for comparing the behavior of object- vs. screen-space locking. For our experiments, we used the following datasets: Bonsai ($512 \times 512 \times 182$), Manix ($512 \times 512 \times 460$), Macoessix ($512 \times 512 \times 460$), in full, half and quarter resolution each.

In Table 3.3, we list both the total number of cache entries for our three tested volumes as well as the total number of references stored in the multi-reference octree for object-space locking. During all our experiments, we measured an average of six to seven references for each cache entry. In fact, these additional five to six references per entry, stored as indices, are a good trade-off memory-wise. Single referencing induces the necessity of searching through neighbor nodes on each level along an octree branch, leading to significantly higher number of costly global memory accesses, decreasing the performance.

Extrapolation accuracy for single cache entries. We compared the radiance extrapolation accuracy computed using the method by Jarosz and colleagues [72] with our method (Section 3.3.2). The radiance for our method is obtained by multiplying the irradiance value by the isotropic phase function value and scattering coefficient. Fig. 3.10 shows that with increasing global density, i. e., growing gradient magnitudes, the Jarosz method, which is relying on only a single gradient value, becomes insufficient to capture the high frequency changes in the irradiance field. By comparison, our method produces drastically lower error, quickly amortizing the memory and computation overhead (a factor of 2-4×) required for the more complex cache entry creation. Higher gradients induced by the lack of background illumination cause similar effects.

Behavior during and after interaction. During interaction, new regions of the dataset may appear which are not covered by the cache yet. However, we implicitly exploit frame-to-frame coherence, since previous regions rarely vanish instantly, and we generate new entries on-the-fly. As expected, stopping interaction leads to quickly and drastic decrease in error, whereas plain MCVR starts from scratch. The right panel of Figure 3.11 shows the average error rates for the Bonsai during 360 ° rotation and then stopping interaction.

Convergence rates in static scenes. Static scenes starting with an empty cache and without user interaction are a hard case for a cache-based algorithm, since a lot of noise is accumulated initially. We consider three different lighting setups. First, we use a single small, distant area light. Second, we use a setup with three very intense, large area lights close to the volume. Third, we use four rather dim, large area lights surrounding the

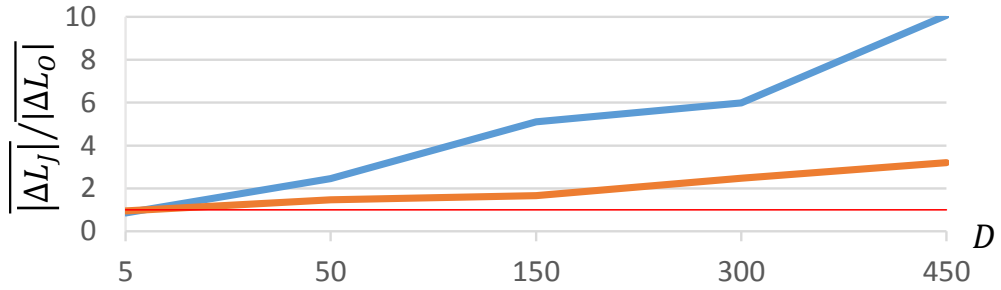


Figure 3.10: The ratio of average relative error of extrapolated radiance values computed using the approach presented in [72] ($\overline{|\Delta L_J|}$) to the ones computed with our method ($\overline{|\Delta L_O|}$) for scene with a single light source (blue) and with additional background illumination (orange). The red line shows the value of 1 which means no difference in average errors of two methods. The horizontal axis shows the global density factor D, which is used to convert transfer function values in range $[0, 1]$ to the density of participating medium. Our method is significantly more accurate for higher gradient magnitudes, which are caused by larger global volume density factor D or by the light setup. Lower estimation errors for our method lead to the lower number of cache entries required to represent the irradiance field.

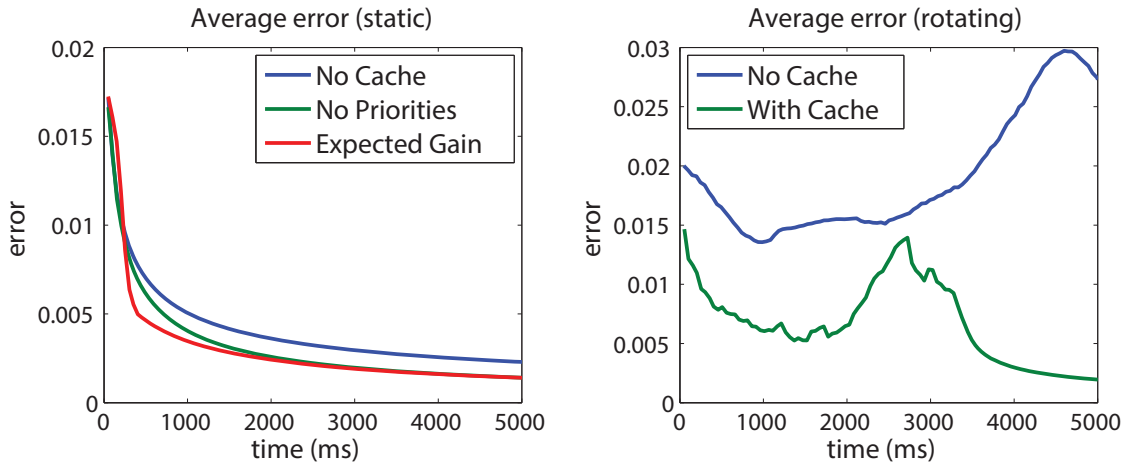


Figure 3.11: (Left) Average error progression observed for the static Bonsai with initially empty cache. Both rendering with (red) and without priorities (green) achieve faster convergence than plain MCVR (blue). (Right) Average error observed during rotating Bonsai by 360 degrees. Even during interaction, our method (green) has lower error compared to plain MCVR (blue). Stopping interaction (3300 ms) leads to rapid convergence due to present cache information.

Table 3.2: Comparison of Monte Carlo error estimates to rendering using the Cache. The numbers express the ratio of Monte Carlo error over Irradiance Caching error, averaged over 200 frames. We used the static scene as the most harsh environment for our algorithm, but nonetheless achieve drastically lower average error for all cases. 'Full' relates to the full volume resolution as described before, and 'Half' for the down sampled counterpart. Cases (a), (b), and (c) represent different light setups. Case (a) uses a single distant sun, case (b) uses three intense area lights, and case (c) uses four dim area lights. The *Mac.* abbreviation represents the macoessix dataset.

No priorities									
	Full ; 1280x720			Full ; 1920x1080			Half ; 1920x1080		
	Bonsai	Manix	Mac.	Bonsai	Manix	Mac.	Bonsai	Manix	Mac.
(a)	1.586	1.493	1.651	1.423	1.448	1.642	1.947	1.895	1.423
(b)	1.381	1.227	2.668	1.282	1.235	2.728	3.361	1.707	2.549
(c)	1.192	1.230	2.560	1.233	1.198	2.416	4.032	2.028	2.864
Expected Gain									
	Full ; 1280x720			Full ; 1920x1080			Half ; 1920x1080		
	Bonsai	Manix	Mac.	Bonsai	Manix	Mac.	Bonsai	Manix	Mac.
(a)	1.619	1.549	1.696	1.652	1.566	1.742	1.986	2.291	1.579
(b)	1.338	1.284	2.565	1.383	1.313	2.777	3.517	2.400	2.757
(c)	1.284	1.245	2.458	1.339	1.224	2.568	3.894	2.049	2.950

volume.

For each of these light setups, we have measured the convergence rate of our approach compared to plain MCVR using the error estimate shown on the left side of Eq. (3.2). This definition of the error per pixel in screen-space allows us to locally estimate the difference to an optimal solution. Since we observed very similar outcomes for all scene experiments, we provide the average ratio of plain MCVR versus our irradiance cache error over 200 frames in Table 3.2 and left panel of Figure 3.11 (plots in Appendix A). Note that despite the difficult setting, our method consistently outperforms plain MCVR event without priorities.

Object-space vs screen-space locking. As discussed in Section 3.4, we evaluate two different methods for preventing creation of excessive amount of redundant cache entries at once. In our first measurement, we rotate the volume by 360 degrees and record the number of cache entries and tree references. We show the results for the detailed structures of the Bonsai dataset. In a second, more extensive test, we focus on static scenes showing Manix and Bonsai with varying parameters. These include volume resolution (512^3, 256^3, 128^3),

screen resolution (1920×1080, 1280×720, 640×480) as well as background illumination (on, off), while the viewpoint is fixed. Our tests show very marginal differences in convergence rates. For low screen resolutions, screen-space locking seems to create entries in more beneficial locations and achieves the desired cache hit rates and convergence values slightly faster. For high screen resolutions, object-space locking creates less redundant entries and achieves high cache hit rates faster with comparable cache sizes. For detailed outcomes of all parameter combinations, please refer to Appendix A.

Table 3.3: Cache sizes observed for object-space locking. The reference counters refer to the total number in the multi-reference octree.

Volume	**Cache Entries**	**References**
Bonsai	32421	193939
Manix	30755	206807
Macoessix	24774	150660

3.8 Discussion

We have presented a method to create an irradiance cache for volumetric data in parallel, concurrently to MCVR. Furthermore, we have proposed two techniques for preventing the creation of redundant cache entries, thus keeping our memory footprint small. Additionally, we have discussed a new irradiance extrapolation method, which outperforms previous approaches. Our experiments illustrate a drastic increase in convergence rate when compared to standard MCVR, even in the most difficult situations for our approach. Especially during and after interaction, our method achieves a significant improvement over MCVR, outperforming it by up to four times.

We also look forward to extending this approach to incorporate multiple scattering, as well as adopt spherical harmonics to allow for anisotropic phase functions and camera-dependent effects. Additionally, a promising research direction includes investigating different data structures for caching, because despite the speed-up we have already achieved, the multi-reference octree still poses a bottleneck for cache lookup.

Chapter 4

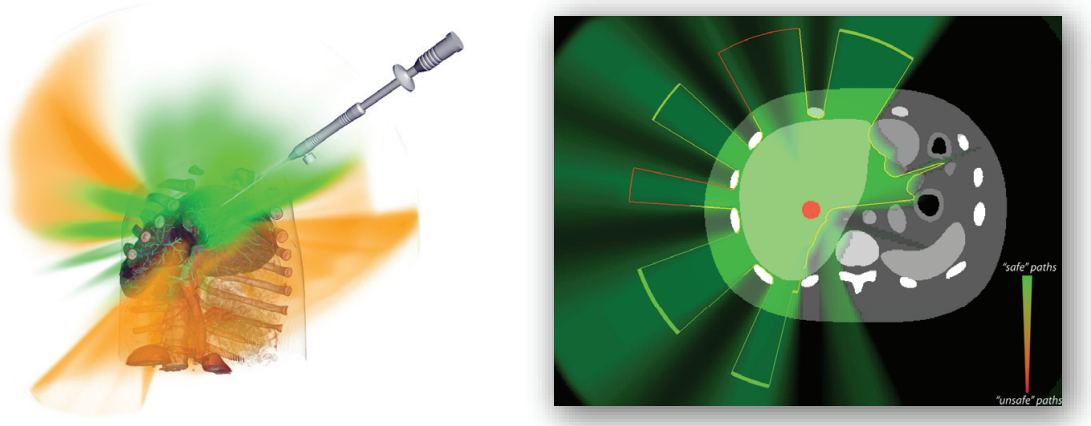
Crepuscular rays simulation

Contents

As opposed to the previous chapter, where the natural phenomenon was used in a very realistic manner, in this and the following chapters, we use natural phenomena as metaphors in a context that would not be feasible in nature to create useful visualization techniques. In this chapter, we use the crepuscular rays phenomenon to create an efficient approach for visualization of safe access paths towards tumors. Even though the path safety computation is conceptually and visually similar to formation of crepuscular rays in real world, we impose additional useful information into the ray intensity by using computational schemes different from realistic exponential decay of light in participating media.

4.1 Motivation

Planning an optimal access path to harmful structures, such as tumors, within the human body is essential in many medical procedures. In modern clinical practice, the treating



(a) (b)

Figure 4.1: Regarding a tumor as a volumetric light source we estimate the safety of *all* straight access paths in the region of interest based on the light intensity reaching every point. We display the safety information in 3D using direct volume rendering (a) and in 2D slice views that allow millimeter-accurate intervention planning, which are widespread in medicine (b). The 3D view (a) shows the safe (green) and the medium-safe paths (yellow). The 2D slice view (b), shows only the safe paths from the volumetric representation and provides additional information about the extent of safe areas using color-coded thick lines. Thin red lines signify that the available leeway is very small and that the physician must be very precise to use such access areas. In contrast, thick green lines mean that less precision is required, as the corresponding safe area is large.

physician makes a decision about the trajectories of medical tools or the areas of resection for a particular patient. This decision is usually based solely on the physician's experience with similar interventions and general knowledge of the vulnerable anatomical structures within the human body. This empirical approach leads to a strong dependence of the treatment result on the experience of the clinician. Hildebrand *et al.* [68] have shown that operator experience has a *significant* influence on the treatment outcome of minimally invasive radio frequency ablations of malignant liver tumors. Mueller *et al.* [119] and McDonald *et al.* [114] showed similar effects for coronary interventions and spine surgery. Hence, in certain cases, this unsupervised access-path planning approach may be harmful or even *deadly* for the patient. Even when a navigation system is used for guidance during an intervention (e.g., in neurosurgery), the access path is still chosen empirically by the performing physician, and thus the treatment outcome depends on the physician's experience.

Trajectory planning is a very complex procedure, because of the huge amount of available data that must be taken into account. These data are produced using modern imaging

modalities such as computed tomography (CT), magnetic resonance imaging (MRI) and its derivatives such as diffusion-tensor imaging (DTI). Furthermore new imaging modalities (e.g., dual-energy CT) are still being developed, and thus the information load on the physician planning the operation will continue to increase. Therefore, it is crucial to create systems that can integrate these data sources and assist the physician in choosing the access path. A fully automatic determination of the *best* access path is certainly desirable, but it is currently not possible in every case, because of the enormous number of degrees of freedom. For this reason, we propose using visualization to assist the physician's decision-making. This visualization must be able to display all of the available medical data simultaneously to let the physician concentrate on the planning itself. Furthermore, it is beneficial to visualize information about the accessibility of the target structure along with the medical data. However, this visualization should only provide auxiliary information to the physician, leaving the final decision to the human operator.

In this chapter, we propose a novel multi-stage tumor-accessibility visualization approach that makes it possible to evaluate the safety of *all* possible straight access paths and to display this information in an intuitive way, thus allowing the performing physician to make a more informed decision without limiting treatment options. The intuitiveness of our visualization originates from a basic metaphor: we think of a tumor as an omnidirectional volumetric light source that is placed in an isotropic scattering medium. The vulnerable structures act as completely opaque or semi-transparent obstacles that block the light 'emitted' by the target structure. This model produces ray bundles in the safe regions and 'shadows' in the unsafe regions. These ray bundles are similar to the widespread and thus easily comprehensible natural phenomenon usually referred to as '*god rays*' or '*crepuscular rays*' (e.g., see Figure 4.2). We have refined this basic metaphor to provide valuable information to a physician and have implemented our method as a complete accessibility visualization system. We present the following technical contributions in this work:

- We propose an algorithm for the evaluation of *all* access paths to a tumor with respect to their safety. It takes into account the potential risks extracted from all of the available volumetric datasets (Section 4.3.2). We show that our method is flexible and can be adapted for various uses without changing the core algorithm. However, in this chapter, we mostly refer to the medical use because of the available datasets and the given evaluation possibilities.

- We propose an algorithm for computing the amount of available leeway in the *safe*

areas and show how this information can be saved in a form suitable for visualization (Section 4.3.3).

- We show how the information regarding the safety of access paths and the amount of leeway can be presented to a physician to assist in decision-making. Our interactive visualization system combines 3D representation for a good overview of all access paths with widely used 2D slice views for millimeter-accurate planning (Section 4.3.4).

We evaluate the applicability of our method for clinicians using real and artificial datasets for various medical tasks. We also compare the paths extracted by our algorithms for 19 real interventions with actual paths chosen by a highly skilled physician in an unsupervised environment. The evaluation results show strong evidence that our method not only reflects the possible access path choices of an experienced doctor, but also has a good chance for a high acceptance rate in clinical practice.

(a) (b)

Figure 4.2: The basic idea comes from crepuscular rays formed by the shape of the trees or by the frame of a window. The sunlight is scattered in the dusty air, and thus an observer gets an impression of separate ray bundles. (Images taken at Watkin's Glenn, NY, by the authors (a) and inside Bayleaf Farmstead by 'There and back again', Antony Scott)

4.2 Background

Preoperative planning. Whereas navigation systems for intra-operative assistance are quite common in neurosurgery [40], preoperative planning, besides the proper integration

of multiple modalities into navigation systems, remains an open problem. For example, Brunenberg *et al.* [24] and Essert *et al.* [51] calculate safe paths for deep-brain stimulation, and Navkar *et al.* [120] and Shamir *et al.* [146] calculate them for general minimally invasive brain surgery. In contrast to our approach, all of these algorithms consider only a single point as a possible target or entrance position. For simpler surgery tasks, which sometimes require only one image modality, several active path safety evaluation systems exist: Villard *et al.* [165] optimize the needle position for ablating malignant liver tumors by minimizing the size of an ellipsoid, which models the necrosis zone, needed to cover a tumor plus a safety margin. The optimization process is constrained by a collision detection algorithm to avoid the vital organs and uses an 'insertion window' that is drawn by a radiologist on the scanned patient's skin. Altrogge *et al.* [3] propose a similar approach, but they predict the necrosis zone more accurately using finite element simulation. Vancamberg *et al.* [160] use FEM-based simulation of tissue and needle deformations to compute the best needle path for performing breast biopsies. The chosen path is constrained to be at a safe distance from blood vessels. Schumann *et al.* [145] generate a list of access paths using a set of 2D constraint maps. The paths are ranked for suitability using multiple criteria and empirically determined weighting factors. These paths are then displayed one-by-one in slices of the original CT volume. Whereas these approaches try to make a decision for a physician, using computation-based prediction, our work concentrates on using visualization to assist the planning of interventions, where current automatic approaches are not suitable.

Intraoperative guidance. Another approach to assisting physicians is the provision of additional information during the intervention. Viard *et al.* [164] show how tracking of an inserted needle can be achieved in an MRI-based setup. Colchester *et al.* [30] use a localizer superimposed on a video stream for the same task. Hansen *et al.* [63] propose an approach that augments real scenes in an operating room with a distance-controlled illustrative visualization of risk structures using a projector. Chentanez [29] proposes a method for simulating deformable tissue and needles that can be used to guide a rigid or steerable needle to reach the target and avoid vulnerable structures. These strategies help in performing the intervention through better visualization and control of the current situation, but they do not guide the surgeon towards the most feasible access paths and cannot show alternative corridors.

Access path visualization. Rieder *et al.* [134] visualize a possible access path by cutting out all tissue in a cylindrical volume between the target and entry points. Baegert *et al.* [8, 9] visualize safe access areas through the abdominal skin of a patient as holes in a fully opaque surface. The decision for making such holes is based on optimization of a set of candidate access paths with regard to multiple criteria. A similar technique is used by Villard *et al.* [165], where the 'insertion window' is visualized as cut-away skin. Brunenberg *et al.* [24] compute safe paths for a neurosurgery scenario and visualize them as opaque geometry on the brain surface in a 3D rendering that displays vulnerable structures only. A selected path can be further inspected using cutting planes orthogonal to the probe axis. Vaillant *et al.* [158] evaluate a cost function along straight paths between the outer brain boundary and a target point. The computed values are then mapped to a color scale of the rendered brain surface. Although all of these access-path visualization techniques are easy to interpret, it should be noted that they make it difficult to observe the full needle trajectory or the vulnerable structures within the body, or they lack context information.

4.3 Method

To illustrate our idea, we propose the following thought experiment. Imagine that a person needs to find and pull out a small object from a dark and dusty room with many obstacles in the way. A blind search for the best location from which to reach the object will be difficult and error-prone. However, if the object is a lamp that is switched on, then the beams of light that form in the dusty air will guide the searching person to the best path to retrieve the lamp. The best path will be easily identifiable: it is where the light beams both are strong (which signifies few obstacles in the way of the light) and cover a large area (which implies that there will be more space to operate in).

In the following subsections, we describe how we employ this metaphor for the tumor-accessibility visualization task. The description will follow the basic data flow in our application, which is outlined in Figure 4.3. We will begin by describing how to define the target and blocking structures (the 'preprocessing' stage, Section 4.3.1). Then, we describe how the basic metaphor can be adapted to convey useful values to a physician and provide an algorithm for computing this information (the 'path safety' stage, Section 4.3.2). During the development of our method, we continuously consulted with clinicians to ensure that we would fulfill their needs. They drew our attention to the insufficiency of a simple projection of 3D data for 2D slice visualization. In section 4.3.3, we propose an algorithm

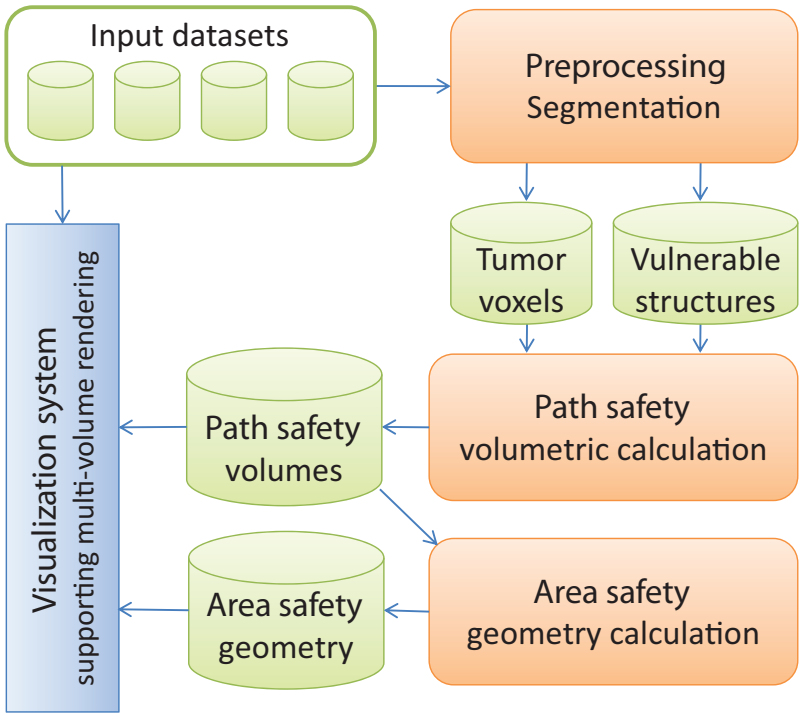


Figure 4.3: The overview of the required steps for our visualization method.

for evaluating the extent of *safe* areas that is designed to solve this problem (the 'area safety' stage). Last, in section 4.3.4, we describe the visualization system that allows a physician to explore the data interactively.

4.3.1 Preprocessing

Preprocessing of the input datasets is necessary to classify the intensity values of the scanned volumetric datasets by their vulnerability. This type of classification is usually referred to as *segmentation*. This step depends heavily on the application and is described in detail for two examples in section 4.4.1.

Often, a rough segmentation by thresholding the intensities to find impassable structures is sufficient. Thresholding can detect bone structures, which have very high intensity values in suitable imaging modalities (e.g., CT) or lungs, which usually have very low intensity values. However, this simple method is not applicable to distinguish more complex structures such as, for example, bronchial tubes, vessel trees, or neurological structures in the brain. Segmentation of these structures is an active area of research, and increasingly accurate algorithms are becoming available. Therefore, we have decided to leave the preprocessing as a completely independent and replaceable part of our method. In

the subsequent steps, we can deal with all types of binary and continuous volumetric segmentation results.

Another application-dependent variable is the definition of the vulnerability of various structures. This definition must be made once by medical experts for each intervention type. Although a continuous scale for *vulnerability* would be easy to implement, our medical partners have suggested using a few discrete levels of vulnerability. Therefore, we use four to six levels for the examples in this chapter, where low values denote non-vulnerable structures and high values denote impassable structures.

After the segmentation and classification of the relevant structures is completed, the safety margin may be added to the blocking information. The safety margin is necessary because of the inherent uncertainty that is caused by the change of relative organ position between the planning and actual intervention stages and the inherent uncertainty of segmentation methods. This change occurs due to respiration for abdominal interventions and due to brain shift, caused by the opening of the skull, for brain surgery. In our examples, we use a Gaussian smoothing filter with an empirically determined kernel size of 5 mm to add a smooth safety margin around all the blocking structures. The exact parameters of the safety margin can be determined by the end users for every type of application.

4.3.2 Path safety

We could use the light-scattering metaphor directly, but we want to convey to the user more valuable information than simply the amount of light reaching a certain point in space. Therefore, we generalize the light-scattering equation and search for a mathematical expression that maintains the basic 'beams of light' idea, but also conveys information about dangerous structures on the way to the tumor. For convenience, we will use the light metaphor to describe the computed values in the rest of this section.

The amount of scattered light reaching a viewpoint from a direction $\mathbf{r}$ is obtained using the low-albedo volume rendering integral [115]:

$$I(\mathbf{r}) = \int_0^L I_s(s) \cdot e^{-\int_0^s \tau(t)\mathrm{d}t} \mathrm{d}s, \tag{4.1}$$

where L is the length of ray $\mathbf{r}$, I_s is the intensity of light scattered by particles in the air in the direction of the viewpoint, and τ is the extinction coefficient.

The definition of I_s depends on the value that is to be visualized and, in general, can

be arbitrarily complex. However, for this chapter, we define I_s using two functions. The first function, which we call the *ray accumulation* function, governs the computation of the amount of light I_p from a point source that reaches a certain point in space. To compute the light intensity reaching that point from the entire target structure, we represent the target structure as a union of an infinite number of point sources. The way that the intensities from different point sources are combined into the final intensity I_s is described by the *ray combination* function. The definitions of ray accumulation and ray combination functions that we found useful in the case of medical interventions are described further (see Figure 4.4 for comparison) in the following.

The **surface visibility value** shows the percentage of the target structure surface facing a certain point that can be reached from that point without any significant blocking structures on the way:

$$I_p^{perc}(\mathbf{p}, \mathbf{x_0}) = \mathbf{I}_{[0,\epsilon)} \left(\int_C b(\mathbf{x}) \, \mathrm{d}s \right) \tag{4.2}$$

$$I_s^{perc}(\mathbf{x_1}) = \frac{\iint_{\partial \mathfrak{T}_+} I_p^{perc}(\mathbf{x}, \mathbf{x_1}) \, \mathrm{d}S}{A(\partial \mathfrak{T}_+)} \cdot 100\%, \tag{4.3}$$

where C is a line segment from the point source $\mathbf{p}$ to a point in space $\mathbf{x_0}$, $b(\mathbf{x})$ is the blocking value function, $\mathfrak{T}$ is the target structure, $\mathbf{I}(x)$ is the indicator function, ϵ is the desired threshold for insignificant blocking values, and $A(\partial \mathfrak{T}_+)$ signifies the area of $\partial \mathfrak{T}_+$. Also, $\partial \mathfrak{T}_+$ is the part of the target structure's surface visible from the point $\mathbf{p}$, considering only self-occlusion of the target structure (see Figure 4.5):

$$\partial \mathfrak{T}_+ = \left\{ \mathbf{x} \in \partial \mathfrak{T} : \textstyle\int_C \mathbf{I}_{\mathfrak{T}}(\mathbf{p}) \mathrm{d}s = 0 \right\}. \tag{4.4}$$

We refer to it as the *front-facing* part of a surface. We also define the *back-facing* part of a surface, which we use further, as:

$$\partial \mathfrak{T}_- = \left\{ \mathbf{x}' \in \partial \mathfrak{T} : \textstyle\int_{C'} \mathbf{I}_{\mathfrak{T}}(\mathbf{p}) \mathrm{d}s = 0 \right\}, \tag{4.5}$$

where C' is the ray starting at point $\mathbf{x}$ in the direction from $\mathbf{p}$ to $\mathbf{x}$.

Integral surface blocking value: The surface visibility value, which is very easy to understand, is not able to distinguish cases where several blocking structures lie on the same ray to the tumor (see Figure 4.4(b, e, II)), which may lead to failure to provide information about safe paths in complex cases, where the target structure is strongly occluded. To handle such cases, we propose functions that show the total weighted volume

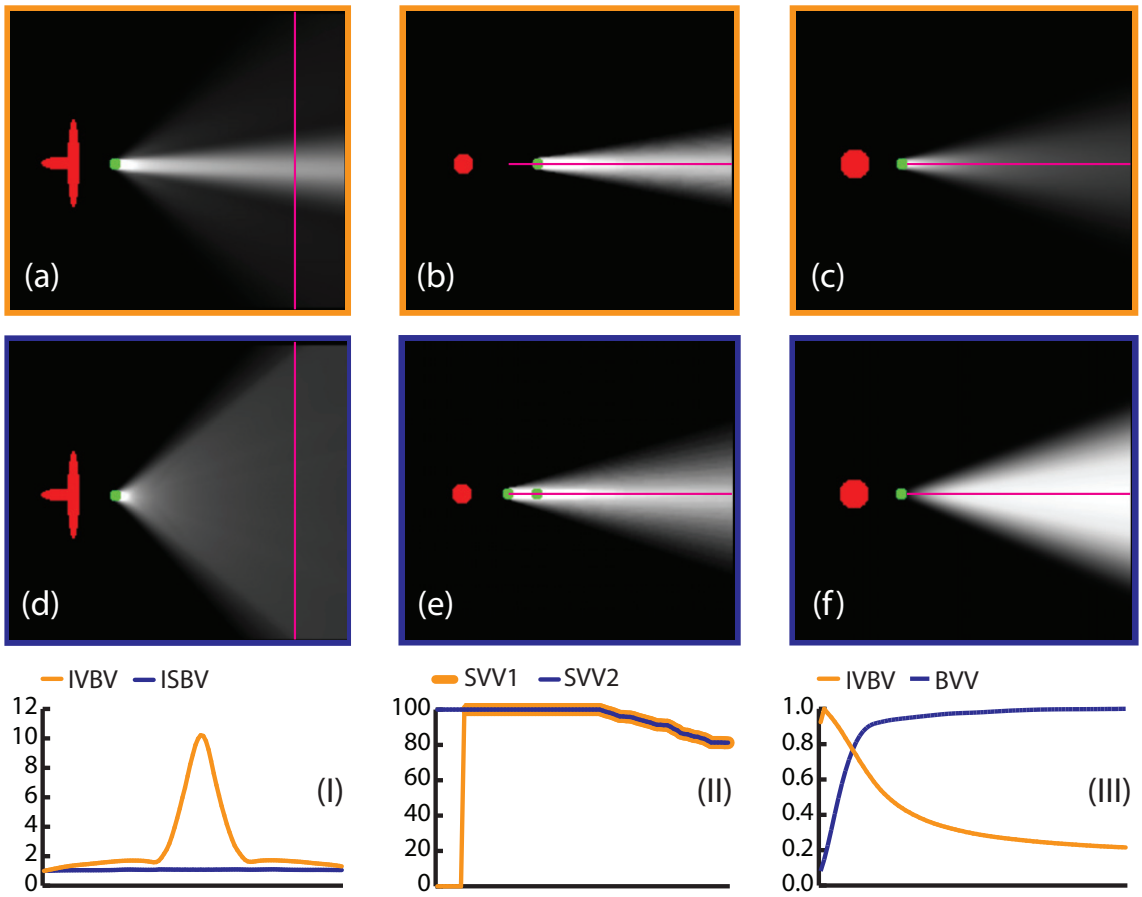


Figure 4.4: Comparison of the behavior of various value-accumulation strategies. The images show the slices of the path safety volumes computed for different configurations of the target structure (shown in red) and blocking structures (shown in green). Panels (a) and (c) show the integral volume blocking value (IVBV), (b) and (e) show the surface visibility value (inverted, for better grayscale perception) with one (SVV1) and two (SVV2) blocking structures, (d) shows the integral surface blocking value (ISBV), and (f) shows the blocker volume value (BVV). The plots (I), (II) and (III) show the comparison of profiles along the lines in corresponding columns. The ordinate axis in the plots shows the ratio of the value to the minimum value along the entire line (I), the actual computed value (II) and the ratio of the value to the maximum value along the entire line (III).

of blocking structures on the way from a point in space to the front-facing part of the target structure's surface:

$$I_p^{surf}(\mathbf{p}, \mathbf{x_0}) = \int_C b(\mathbf{x})\, \mathrm{d}s \tag{4.6}$$

$$I_s^{surf}(\mathbf{x_1}) = \iint_{\partial\mathfrak{T}_+} I_p^{surf}(\mathbf{x}, \mathbf{x_1})\, \mathrm{d}S, \tag{4.7}$$

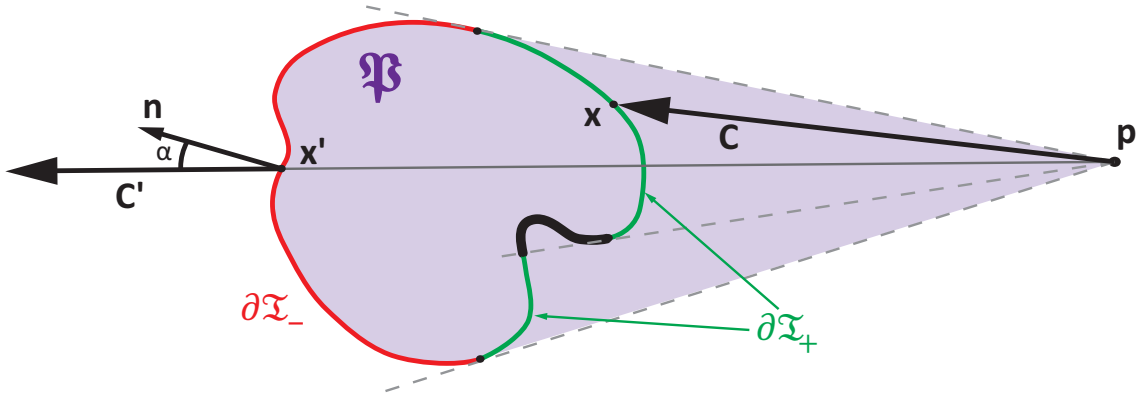


Figure 4.5: Illustration of various structures used in our method: $\partial\mathfrak{T}_+$ is the front-facing and $\partial\mathfrak{T}_-$ is the back-facing part of the surface; $\mathbf{n}$ is a surface normal; and $\mathfrak{P}$ is the pyramid, with apex at point $\mathbf{p}$ and base $\partial\mathfrak{T}_-$.

Integral volume blocking value. In some cases, such as tumor surgery, the blocked *volume* of the target structure is of interest. The functions to compute this information are defined similarly to the integral surface blocking value:

$$I_s^{vol}(\mathbf{x_1}) = \iiint_{\mathfrak{T}} I_p^{vol}(\mathbf{x}, \mathbf{x_1})\,\mathrm{d}V, \tag{4.8}$$

where function I_p^{vol} is the same as in Eq. (4.6). Because we consider straight paths only, the integral volume blocking value can be also defined as:

$$I_p^{vol'}(\mathbf{p}, \mathbf{x_0}) = \int_C b(\mathbf{x})\,\mathrm{d}s \cdot \int_C \mathbf{I}_{\mathfrak{T}}(\mathbf{x})\,\mathrm{d}s \tag{4.9}$$

$$I_s^{vol'}(\mathbf{x_1}) = \iint_{\partial\mathfrak{T}_-} I_p^{vol'}(\mathbf{x}, \mathbf{x_1})\,\mathrm{d}S. \tag{4.10}$$

Using this definition, the subsequent evaluation of these integrals is less computationally intensive.

The **blocker volume value** shows the total weighted volume of blocking structures on the way from a point in space to the entire target structure. It can be computed as a volume integral over the pyramid $\mathfrak{P}$, with the apex as the point and the base as the back-facing part of the target structure's surface (see Figure 4.5). After conversion from

an integral in a spherical coordinate system, the blocker volume value can be defined as:

$$I_p^{block}(\mathbf{p}, \mathbf{x_0}) = \int_C b(\mathbf{x}) \cdot d^2(\mathbf{x}) \cdot \cos(\alpha)\, \mathrm{d}s \tag{4.11}$$

$$I_s^{block}(\mathbf{x_1}) = \iint_{\partial\mathfrak{T}_-} I_p^{block}(\mathbf{x}, \mathbf{x_1})\, \mathrm{d}S, \tag{4.12}$$

where α is the angle between the normal to the surface at point $\mathbf{x_0}$ and the ray.

Discretization. We assume that the scattering is isotropic, and therefore the amount of light scattered at each point in space is viewpoint-independent. Consequently, we can precompute the values of I_s in the region of interest in an offline step and store them as a volumetric dataset (in the rest of the chapter, we refer to this volume as the *path safety volume*). This volume is then used by a multi-volume rendering system [77] to evaluate equation (4.1). We choose the region of interest to be a bounding sphere of all the vulnerable structures, centered at the centroid of the target structure. We use a regular grid to cover the target domain and set its spacing to the minimum of the spacings of the input datasets.

Then, for every voxel of the output volume, we cast rays through the blocking volumes to every voxel of the target area, which changes according to the function being computed. For each ray, we accumulate the value using a discretized ray accumulation function. Figure 4.6 illustrates this process. The sampling distance is chosen to be half of the smallest input volume spacing, which will automatically satisfy the Nyquist criterion for all the input volumes. The values accumulated for different rays are then combined with a discretized version of the ray accumulation function to produce the final value of I_s that is stored in the path safety volume.

The discretization of all of the ray accumulation and ray combination functions is done in a similar way. For example, for the case of the blocker volume value, the discretized version of the ray accumulation function (Equation (4.11)) is:

$$I_p^{block} = \sum_{i=1}^{N_s} b_i \cdot V_{sample_i} \tag{4.13}$$

$$V_{sample_i} = s^2 |\cos(\alpha)| \frac{i^2}{N_s^2} l_{step}, \tag{4.14}$$

where N_s is the number of samples along the ray, b_i is the i^{th} sampled blocking value, s

is the spacing of the target structure volume, l_{step} is the length of a single step along the ray, and α is the angle between the normal to the surface and the ray direction. V_{sample_i} is the volume of the i^{th} sample (see Figure 4.6). Note that no blocking structures will be missed when casting the rays, because the Nyquist criterion is satisfied, and the maximum distance between cast rays does not exceed the spacing of the blocking volumes.

The discretized version of the ray combination function (Equation (4.12)) is defined as:

$$I_s^{surf} = \sum_{j=1}^{N_v} I_{p_j}, \tag{4.15}$$

where N_v is the total number of rays cast from the voxel, and I_{p_j} is the corresponding ray value.

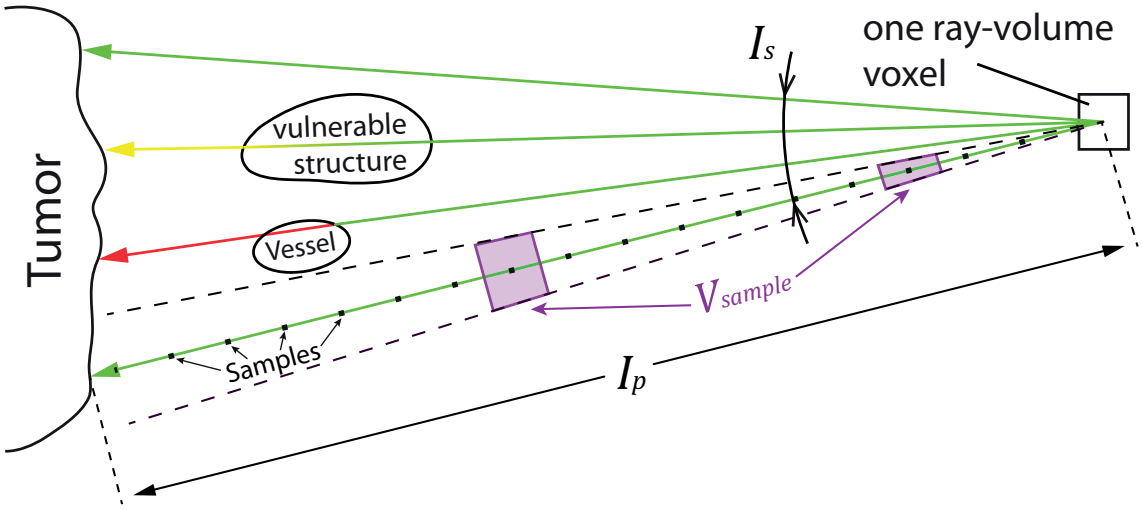


Figure 4.6: Illustration of the calculation of the integral voxel value for one voxel. Impassable or very dangerous structures raise the accumulated 'ray value' I_p quickly to high values. Other vulnerable structures gradually increase the accumulated value based on their size. Separate ray values are then combined to a voxel value I_s. V_{sample} is the approximated volume of the sample, which increases with distance from the voxel. It is used to avoid weakening of the effect of dangerous structures with distance for the blocker volume value.

4.3.3 Area safety

Many physicians often prefer using multi-planar reconstruction views instead of, or in combination with, a 3D view. To convey the safety information in 2D views, we display the relevant slices of the path safety volume. However, because the 2D slices lack depth

information, it is difficult to determine the overall extent of a safe area (unlike 3D, where safe areas can be identified at a glance). To address this problem, we propose the following algorithm:

1. Approximate the target structure surface using a geometry.
2. Extrude each vertex of the geometry in the direction of its normal until a certain distance or an 'unsafe' point is reached.
3. Classify vertices as 'safe' and 'unsafe', and determine the connected regions containing only 'safe' vertices.
4. Weight each vertex based on the size of the connected safe cluster to which it belongs.
5. Display a corresponding slice of the computed geometry to provide additional information about the overall extent of the safe regions.

The criterion for considering a certain point in space to be 'unsafe' can be varied based on the application. Usually, a point is considered unsafe, if the path safety at the point exceeds a certain threshold. However, we can also use the blocking value datasets to determine if the point is unsafe. In this way, the described algorithm can be used as a fast preview of the path safety. However, it cannot act as a replacement for the path safety computation method, as it takes only a few paths into account and cannot distinguish between medium-safe and dangerous paths.

We approximate the segmented tumor with an ellipsoid to avoid non-manifold faces and overlaps during the extrusion process. We align the ellipsoid's axes with three main axes of the tumor, which are extracted using principal component analysis of the tumor voxel coordinates. We tessellate the ellipsoid using a regular grid in a spherical coordinate system, i.e., the angular distances between the vertices of the ellipsoid are equal in both angular directions. The tessellation level is chosen in such a way that after extrusion the maximum edge length does not exceed the spacing of the output volume of the path safety calculation.

Every vertex of the ellipsoid is subsequently extruded in the direction of its normal vector, until a certain distance or an unsafe point is reached. We choose normal direction in order to avoid nonmanifold and intersecting faces in the resulting geometry. We then separate the vertices of the resulting geometry into two classes: the 'safe' class, which contains the vertices that have been extruded up to the fixed distance, and the 'unsafe' class, which contains the vertices whose extrusion was stopped, because the safety criterion

was violated. Because of this binary separation, we can directly define disjoint connected safe regions. For each of these regions, we compute the mass and save it as a scalar property with every vertex belonging to the region. The algorithm for the mass calculation uses the discrete form of the divergence theorem with the general assumption that the surface is watertight, as described by Alyassin *et al.* [5]. In the rest of the chapter, we refer to the resulting geometry with weighted vertices as the *area safety geometry*.

4.3.4 Visualization system

Our visualization system contains two significant parts that are designated for separate stages of the intervention planning procedure. The first part is intended to provide a comprehensive overview of the safety of available paths and uses a 3D representation. The second part is designed for millimeter-accurate planning of the intervention when the access area is usually already chosen and uses a 2D slice representation. We allow the physician to switch between these modes interactively or use them together in a single visualization, combining the advantages inherent in both perspectives.

Path safety visualization. We use a multi-volume rendering system [77] to visualize the output of the path safety computation algorithm along with the original 3D body scans. Our rendering system also allows the addition of the segmentation results as translucent geometry if desired.

A simple one-dimensional transfer function is sufficient to control the visualization of the path safety volume. Depending on the value encoded into the path safety volume, the safe paths have either low values (in the case of integral volume or surface blocking value and blocker volume value) or high values (in the case of surface visibility value). We assign green to the very safe paths, yellow to the medium-safe paths, and red to extremely dangerous paths. The blue-to-yellow color scheme can be used for color-blind users, and our system also allows the definition of arbitrary color mappings. The user can remove certain safety classes from the visualization by setting their opacity value to zero to reduce the visual clutter. As the values in the path safety volume represent the intensity of 'emitted light' reaching a point in space, we do not perform additional lighting or shadowing during the ray casting of this volume. An example of the visualization of only the very safe and medium-safe paths can be seen in Figure 4.1(a).

Area safety visualization. For 2D visualization, we display slices of the available datasets using cutting planes with the desired orientations. The default layout uses axial,

coronal, and sagittal orientations of the cutting planes, as these orientations are widespread in medicine. However, the user can choose an arbitrary orientation during the interaction. The 2D interaction scheme itself is similar to the one used in MITK [172].

The slices of the original datasets, segmented datasets, and path safety volume are displayed with a contour that results from the intersection of the cutting plane and the area safety geometry. We render this contour as an OpenGL triangle strip with varying thickness and color. For the color mapping of the available leeway, we use the same color scheme as for the three-dimensional path safety rendering. The thickness of the line segments is also defined by the area safety information. After discussion with our medical partners, we fixed the upper limit for the line thickness to be one centimeter in the patient coordinate system. Figure 4.1(b) shows a sample image of an axial slice of an artificial dataset rendered using our method.

4.4 Implementation

Precomputation. The path safety computation requires the evaluation of all possible and impossible access paths. This process is very computationally intensive, and a full evaluation might take up to several years using conventional CPU-based iterative approaches. Because the evaluation of the safety level of every voxel is completely decoupled from the evaluation for other voxels, our algorithm is very well suited for parallel computation. We implemented our path safety computation algorithm using NVidia's CUDA to utilize the available processing power of the GPUs. The lookup volumes, including vulnerable structures and impassable structures, are stored in the GPU memory in the available texture units, allowing us to use the advantages of hardware trilinear interpolation and texture caching. The CUDA kernel for the fast path safety volume computation is outlined in Algorithm 1. We use 3D thread blocks with a size of $8 \times 8 \times 8$, which results in 512 threads per block. Every thread block processes a $8 \times 8 \times 8$ block of voxels. This layout allows better thread coherency and less texture-cache misses as compared to 1D ($\sim$20% slower) or 2D ($\sim$3% slower) voxel blocks. Each thread computes the path safety for exactly one output voxel.

The area safety computation algorithm is not as computationally intensive. Therefore, for simplicity and clarity of the implementation, we use the Visualization Toolkit (VTK [144]) for geometry extrusion and clustering.

Algorithm 1 The path safety volume computation kernel in pseudo-code. The sampling of the volumes is performed in the global coordinate system. Ω refers to a volume containing all information about the vulnerability of structures and $\Omega(p)$ to the volume sample at point p; $\mathfrak{T}$ defines a volume of tumor voxels and $\mathfrak{T}(p_t)$ the tumor voxel at point p_t; I_{xyz} and A_{xyz} are the virtual attenuated light intensities emitted by the tumor. Θ defines the output volume and $\Theta(p_o)$ the sample at position p_o. $\mathbf{M}_\Theta$ is the transform matrix from the local coordinate system of Θ to the global coordinate system. δ is the normalized direction of a virtual light ray scaled by the step size γ. The implementation of accumulateValue, combineValues, and finalizeValue functions as well as the values of *initialIntensity* and *initialVoxelValue* depend on the value being computed (see Section 4.3.2).

function castRay($p, \delta, \Omega, N_{steps}$)
  $A_{xyz} \leftarrow initialIntensity$
  for N_{steps} **do**
    $p \leftarrow p + \delta$ ▷ advance along ray
    $A_{xyz} \leftarrow$ accumulateValue($A_{xyz}, \Omega(p)$)
  end for
  return A_{xyz}
end function
function computeAccessibility(Ω,$\mathfrak{T}$)
  setup computation grid
  for all GPU THREADS **do** ▷ is done in parallel on the GPU
    $p \leftarrow$ thread's index in the computation grid
    $p \leftarrow \mathbf{M}_\Theta \cdot (p \cdot spacing(\Theta))$ ▷ get global coordinates
    if $p \notin$ sphere inscribed in Θ **then**
      return
    end if
    $I_{xyz} \leftarrow initialVoxelValue$
    $\gamma \leftarrow 0.5 \cdot \min(spacing(\Omega))$ ▷ Nyquist criterion
    for $p_t \in \mathfrak{T}$ **do**
      $\delta \leftarrow normalize(p_t - p) \cdot \gamma$ ▷ calculate step direction
      $I \leftarrow$ castRay$\left(p, \delta, \Omega, \left\lceil \frac{||p_t - p||}{\gamma} \right\rceil\right)$
      $I_{xyz} \leftarrow$ combineValues(I_{xyz}, I)
    end for
    $\Theta(p) \leftarrow$ finalizeValue(I_{xyz})
  end for
end function

Rendering. We use the Medical Imaging Interaction Toolkit (MITK [172]) as the core medical visualization system. MITK provides all of the necessary basic functionality for the visualization of medical data. We use MITK's built-in OpenGL-based 2D slice view functionality (achieved through 3D texture mapping) to display the abdominal data from

an arbitrary imaging modality, and the path safety volume. Additionally, for area safety visualization, we project the geometry's normal vectors onto the desired cutting planes and use them together with information about the current cluster size to render thick lines as OpenGL quad-strips. Thus, the projected normal vectors define the direction of the quads, and the additionally stored information about the mass of an area defines their size and therefore the thickness of the lines. For 3D visualization, we have integrated the polyhedral CUDA-based rendering system proposed by Kainz *et al.* [77] as a separate view in the widget system provided by MITK.

Performance. Although the direct geometric area safety calculation is performed in a couple of seconds on the CPU ($O(n)$ complexity, where n is the number of vertices in the mesh approximating the tumor), the path safety volume calculation depends strongly on the number of voxels in the segmented tumor ($O(n^3 \cdot m)$ complexity, where n is number of voxels per dimension of the output volume, and m is the number of tumor voxels). As the voxel values are computed independently, we are able to split the output path safety volume into several parts and utilize multiple GPUs simultaneously (tumor and vulnerable structures information is duplicated on each GPU). With this approach, the evaluation of path safety takes $600ms$ per tumor voxel and approximately five minutes for the whole tumor on our test system (Intel QuadCore 3.16 GHz, 12 GB RAM, NVidia Quadro6000 and GTX580) for 512^3 input data sets and a tumor with an average diameter of 2 cm (508 target tumor voxels). Note that we do not need to update this volume during visualization and planning and that this computation must be performed only once per intervention.

The required time for preprocessing depends strongly on the desired application and the available segmentation algorithms. During our experiments, these steps took from a few seconds (clear structures, segmented by thresholding) up to several minutes, including user input (brain DTI, fMRI evaluation, and sophisticated vessel-segmentation algorithms).

Memory requirements. Our method requires only the resulting segmentation volumes as an input. Whereas a pure binary segmentation can be stored as a one-bit single-value scalar field that combines all segmentation results, segmentations that also include levels of vulnerability and smooth safety margins have to be stored with a bit-depth at least comparable to that of the input volumes. Nevertheless, intersecting vulnerabilities can also be summed up and subsequently saved in a single volume. Overall, the required additional memory is not more than that of the input volume with the highest resolution. For our examples, we stored the blocking volumes in a floating-point 3D texture. The

biggest dataset we used (pig abdomen) amounted to 512MB. The memory requirements of the geometric representation of access areas are negligible compared to those of the path safety volume. Whereas the path safety volume might require several hundred megabytes of storage space, the area safety geometry occupies approximately 10 MB, even at a very high polyhedral resolution ($\sim$600.000 faces).

4.4.1 Selected segmentation procedures

The preprocessing of the input data varies depending on the target application area. As the main focus of this paper lies in the medical domain, we will describe the preprocessing step for the medical data used for planning the RFA of liver tumors and brain tumor resection. Note that the operation of our path safety computation system is fully decoupled from the algorithms used for the determination of vulnerable and target structures, and, thus, these algorithms can easily be replaced.

Preprocessing for minimally invasive liver interventions. In modern practice, both contrast-enhanced and native CT scans are used to plan an RFA intervention. We extract the following information from these scans:

- **Target structure.** The tumor is segmented using the contrast-enhanced arterial phase CT scan using the algorithm from Hame [62].
- **Safe structures, level zero.** The structures that can be safely penetrated during the RFA are the liver itself (with exceptions for the vessels, see below), which is segmented using the algorithm described by Alhonnoro *et al.* [2], and the skin, which is segmented using thresholding.
- **Vulnerable structures, level one.** In this case, the less vulnerable structures are small venous vessels of $5 - 10mm$ that are segmented from contrast-enhanced CT scans using the algorithm by Alhonnoro *et al.* [2].
- **Vulnerable structures, level two.** Arteries and large venous vessels ($> 10mm$) constitute more vulnerable structures. Organs whose penetration is possible but undesirable (such as lungs, with very low intensity values) are classified as vulnerable and segmented using thresholding. Other organs can also be defined as level three structures. This definition depends on the choice of the performing physician.
- **Impassable structures, level three.** Bones and metal implants are extracted using the native CT scan by simple threshold-based segmentation (very high intensity

values). The organs that must not be penetrated during the intervention (such as heart and bowel) are also considered to be impassable and are segmented based on the liver segmentation results.

Preprocessing for brain tumor surgery. For the case of brain tumor surgery planning, various MRI and CT scans are usually taken into account. For our examples, we use perfectly registered and segmented datasets provided during the VIS-contest 2010. The datasets are therefore courtesy of Prof. B. Terwey, Klinikum Mitte, Bremen, Germany. With tools provided by MedINRIA*, we assign six discrete levels of vulnerability for the following segmentations:

- **Target structure.** The target structure is the tumor whose resection is being planned. It has been segmented by a professional by hand.
- **Safe structures, level zero,** include the skull, skin and other non-neurological structures, such as low-tumor-lesion thickness areas (extracted from FLAIR and SWI datasets), white matter with no relevant connecting fiber bundles (fMRI, DTI), and non-stimulated gray matter areas (fMRI).
- **Vulnerable structures, level one,** are weakly stimulated gray matter areas (fMRI), areas with no fiber bundles (DTI) and no vessels (CE T1).
- **Vulnerable structures, level two,** are moderately stimulated gray matter areas (fMRI), areas with no fiber bundles (DTI) and no vessels (CE T1).
- **Vulnerable structures, level three,** are areas containing only a few fiber bundles (DTI) and no vessels (CE T1).
- **Vulnerable structures, level four,** are highly stimulated gray matter areas (fMRI) or areas containing an average count of fiber bundles (DTI) or small vessels (CE T1).
- **Impassable structures, level five,** we consider the following structures to be impassable, as damaging them would be deadly for the patient or would severely influence the brain function: highly stimulated gray matter areas (fMRI), dense fiber bundles (DTI), and thick vessels (CE T1).

Preprocessing for the engine dataset example. The preprocessing for a dataset such as this one is simple. Structures made of metal are usually not penetrable by a tool.

*http://www-sop.inria.fr/asclepios/software.php

Consequently, all areas in the dataset with a higher intensity value than that used for air can be considered impassable.

4.5 Results

We evaluated our accessibility visualization approach in several discussions with radiology experts during the implementation process. Our algorithms were successively refined using this expert knowledge. The results of this refinement process are different value-accumulation strategies for the calculation of safe and unsafe paths as they are presented in Section 4.3.2.

Qualitative user study. To verify the suitability of the chosen path and area safety visualizations for the prospective users, we conducted an online user study among 31 medical doctors of whom 15 have fully completed our survey. Of those 15 participants, 13% of the participants have been female [2/15] and 86% – male [13/15], with 80% expert knowlege in radiology, 26% neurology, 26% surgery, and 20% computer science and visualization (multiple fields of expertise have been possible). Their professional experience was between five and ten years for 46% [7/15], and more than ten years for 20% [3/15].

We divided the survey into two parts. During the first part, we evaluated the overall acceptance of our method in 3D and 2D, based on an eight-value Likert scale. We chose eight values to avoid neutral answers, which can occur for odd numbered scales (scales without midpoints are not less reliable than those scales with them [4, 6]). Our method was presented as turns in the pitch and yaw directions for 3D visualizations and as a complete scroll through all available slices for 2D slice-based visualization. The results in Figure 4.9 show that the 2D methods were preferred by the participants in our survey. This result can be explained by the high number of participating radiologists who are specially trained to work with 2D slice visualizations. The most successful 3D visualization was the one showing only the safe paths in 3D. The addition of medium-safe paths and impassable paths decreased the acceptance by most participants. However, the wide range of results indicates that the information desired to be seen in 3D depends on personal preferences and should remain adjustable by the user, as is the case in our system. For 2D slice-based visualization, an augmentation with our proposed area safety visualization method and a combination with the projection of the path safety volume received the highest grades. The projection of the path safety volume onto 2D slices alone showed lower acceptance on average, but the acceptance also showed large variation, which provides evidence for

significant differences in personal preferences. Therefore, we assume that an adjustable visualization is also the best choice for 2D views.

In the second part, we investigated the influence of our method on the choice of access paths for two cases: one liver and one brain tumor. We presented different options for access paths and observed how the decision changed with different accessibility visualizations. In the presented images, the green region shows the safest 20% of the paths, using blocker volume value as a measure. The questionnaire provided several choices for access paths, both feasible and dangerous, which were chosen in advance by a radiology expert and a neurology expert. We measured the number of feasible paths that were chosen by the participants. In Figure 4.7, paths 'c,d,e' are the safest paths. For this example, the area safety calculation had the greatest impact. For example, path 'd' is safe but has very little leeway, which was shown with thin red lines by our algorithm. This factor motivated our survey participants to choose different paths, when area safety information was shown in addition to the anatomical data. The same evidence can be seen in Figure 4.8, which shows an example of brain tumor surgery. Here, paths 'a,l,k' are the safest ones, but, again, 'k' has very little leeway, which is visible in the area safety visualization. This factor influenced the participants' decision to avoid the path 'k'. Although no user chose a dangerous path using our methods, the tendency to choose a safe path increases when path safety and area safety are also displayed, as shown in Figures 4.7 and 4.8.

Overall, our method proved to be well-received by the participants in our survey. The results are summarized in Figure 4.9, which outlines the general expected acceptance in clinical practice, and in Figures 4.7 and 4.8, which show the impact of our method on the expert decisions. Finally, we asked the participants if they would prefer a combination of the shown visualization methods. Most participants (> 80 %) would prefer a combination of a 3D volumetric representation, as shown in Figure 4.1(a), and a projection of the safe areas on 2D slices (both area safety and path safety information), as shown in Figure 4.1(b).

Gold standard comparison. In addition to the qualitative user study, which we performed to gain a subjective acceptance evaluation of the proposed accessibility-visualization method, we studied the accuracy of the displayed safety information, proposed by our algorithms, on 19 real abdominal CT-guided RFA interventions. To perform this study, we applied our visualization method to all available patient datasets and the vulnerable structures segmented from them. Subsequently, we compared the calculated path and area safety to the needle trajectory actually chosen for the intervention. This straight trajectory is available in a separate registered CT scan.

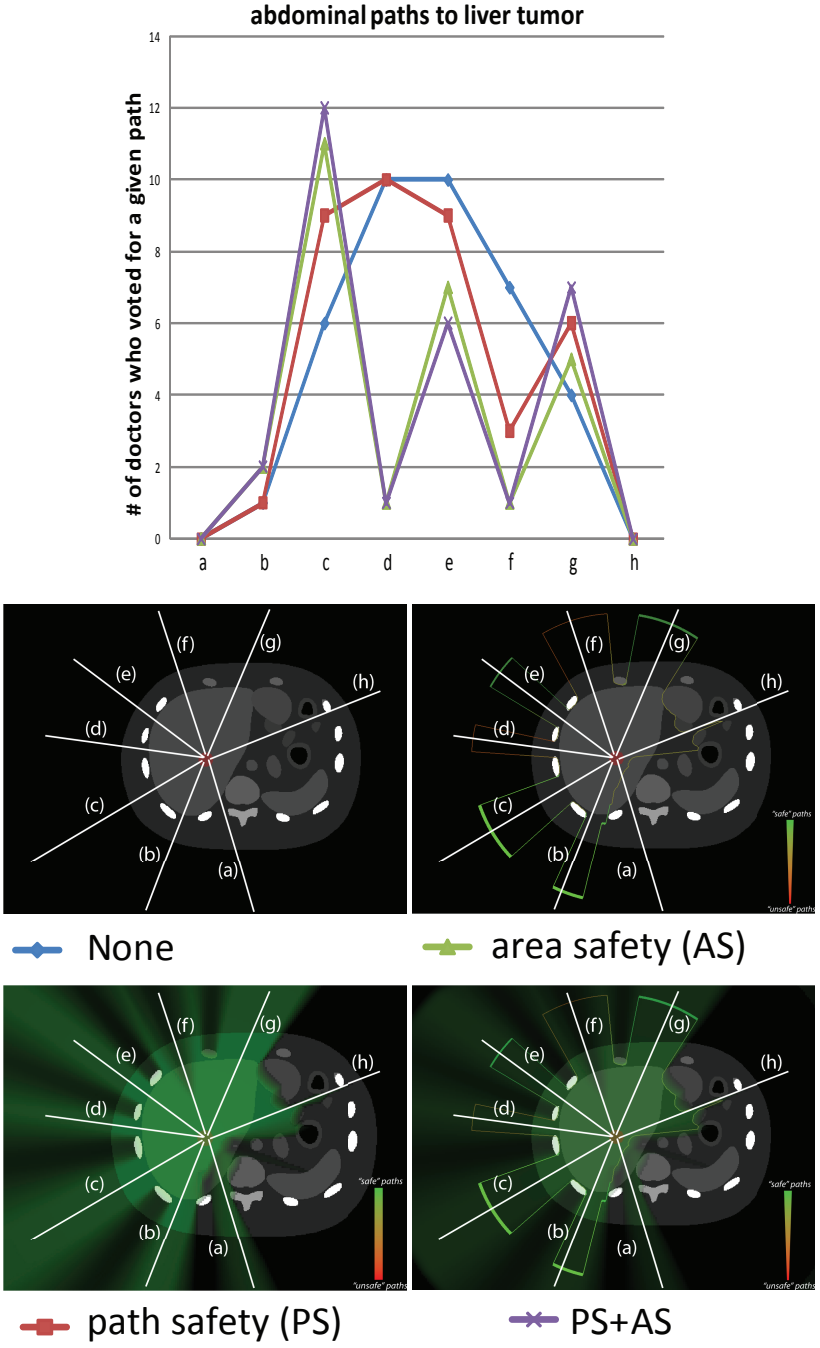


Figure 4.7: Impact of our visualization method on path choice for liver tumor case. **PS** refers to the path safety volume calculation and its projection onto 2D slices. **AS** refers to the geometry augmentation in 2D including the visualization of the leeway (area safety). Note that our method has a strong influence on certain path decisions.

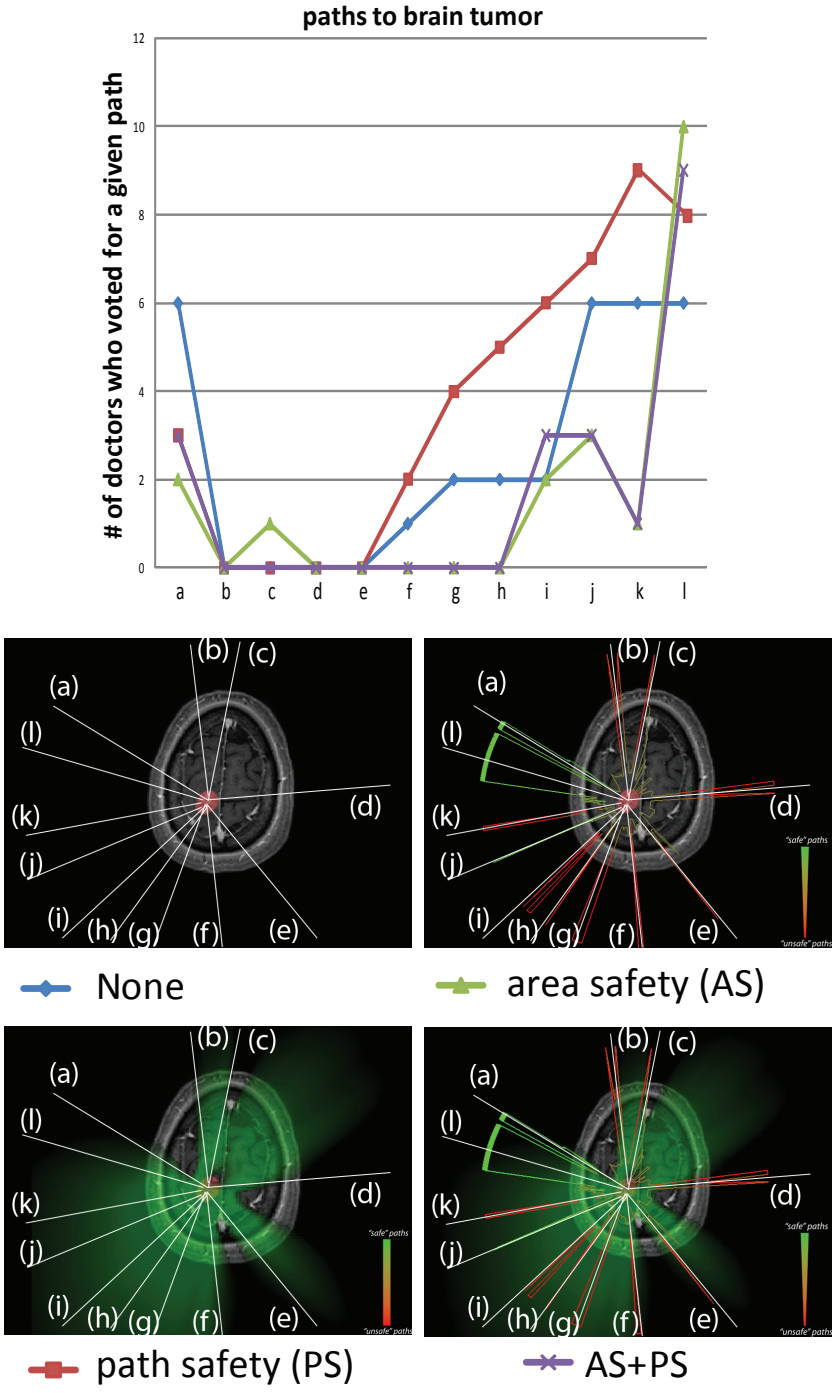


Figure 4.8: Impact of our visualization method on path choice for brain tumor case. **PS** refers to the path safety volume calculation and its projection onto 2D slices. **AS** refers to the geometry augmentation in 2D including the visualization of the leeway (area safety). Note that our method has a strong influence on certain path decisions.

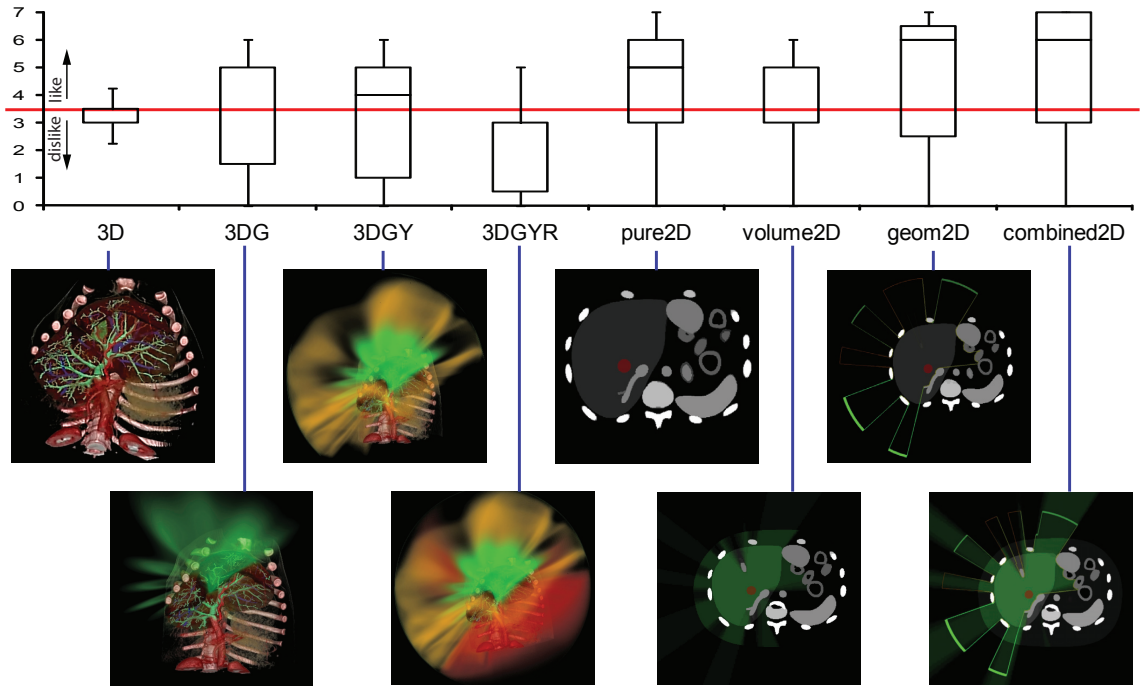


Figure 4.9: Evaluation of the acceptance of our methods for clinical practice. The participants were offered a choice in the form of an eight-value Likert scale after watching a video of our method. The 2D methods were preferred and are indicated with *pure2D*. The acceptance of 3D methods decreased with increasing amounts of displayed information. The option labeled *3DG* shows safe paths only; *3DGY* also shows medium-safe paths; *3DGYR* further adds impassable paths. 2D multi-planar methods were also evaluated with *area safety* geometry augmentation (*lines2D*), a 2D projection of the *path safety* volume (*volume2D*), and a combination of both (*combined2D*). For this box-plot, the ends of the whisker are set at 1.5× interquartile range above the third quartile and 1.5× interquartile range below the first quartile, which corresponds to the normal convention for box-plots. The thumbnails below each box show frames from the corresponding videos.

We considered the chosen paths as the gold standard because all of the performing physicians had more than 10 years of experience with minimally invasive abdominal interventions. Furthermore, all patients except one have not had complications after the intervention. The only case who suffered a complication (bleeding) also recovered after a short time. Several physicians, who were consulted after the intervention, were not able to find an explanation for this complication, which leads us to assume that the scan may have been of insufficient resolution to show all vulnerable structures for proper planning. We consider this case as valid for our study, because all rules to find an optimal access path were followed during the actual intervention. However, we excluded two cases from

the study. The first case was excluded, because the patient had an implanted medical pump and several aortic stents, and the intervention had to be specially adapted. The second case was not suitable for our study, because the respiratory motion in the scan showing the needle trajectory was too strong for proper registration. This patient had several partial organ resections because of multiple cancer metastases. Hence, we used 90% of the valid example interventions for this retrospective gold standard study.

For eight cases (47%), the needle trajectory coincided with a path for which our algorithm calculated a path safety of 100%, which means that, in these cases, there were no obstacles on the way to the tumor along this path. In most of these cases, the access area with the largest leeway was chosen. We also discussed these patients with our medical partners, who confirmed that the position of the tumor was suited very well for minimally invasive intervention. The remaining cases can be considered to be difficult, and they had an average gold-standard path safety of 77% using our method.

To calculate the relative area safety of the chosen paths, we rate the areas in proportion to the largest available safe area (thus, the largest area has a safety of 100%). In seven cases (41%), the experienced physician chose the access area with the largest leeway. The average area safety of the remaining 59% of cases was 70%, which shows clear evidence that areas with the most available leeway are always chosen by the experts and that our method reflects several decision criteria for real-world intervention planning.

4.6 Limitations

According to our user study, area safety geometry is a very good visual cue for understanding safety information in the 2D slice view. However, the extrusion of the vertices in normal directions may lead to some loss of the information stored in path safety volume even though the current geometry approximation is conservative (i.e., the geometric representation will not consider a point safe, when the path safety volume states the opposite). One approach is to use a solid fitting algorithm [55] on the path safety volume to extract the outer surface of the volume containing values in a safe interval of values.

4.7 Discussion

We effectively applied a natural metaphor to a difficult medical-accessibility decision problem. To exploit this metaphor, we calculate an additional volumetric dataset on the GPU that encodes the safety of all possible access paths as bright rays shining out of the body,

based on various replaceable segmentation procedures. Thus, the intensity values provide additional information on how much of the target structure can be reached from every position within the region of interest. Furthermore, we evaluate the available leeway for each ray bundle and display this information in 2D slices, which are widely used in medicine. With this method, a physician can quickly and reliably determine the possible tool trajectories. A combination of area and path safety augmented 2D MPR views was implemented in a medical visualization prototype, as shown in Figure 4.10. This figure also shows that our visualization approach can be easily used to indicate dangerous and impassable areas instead of safe areas only.

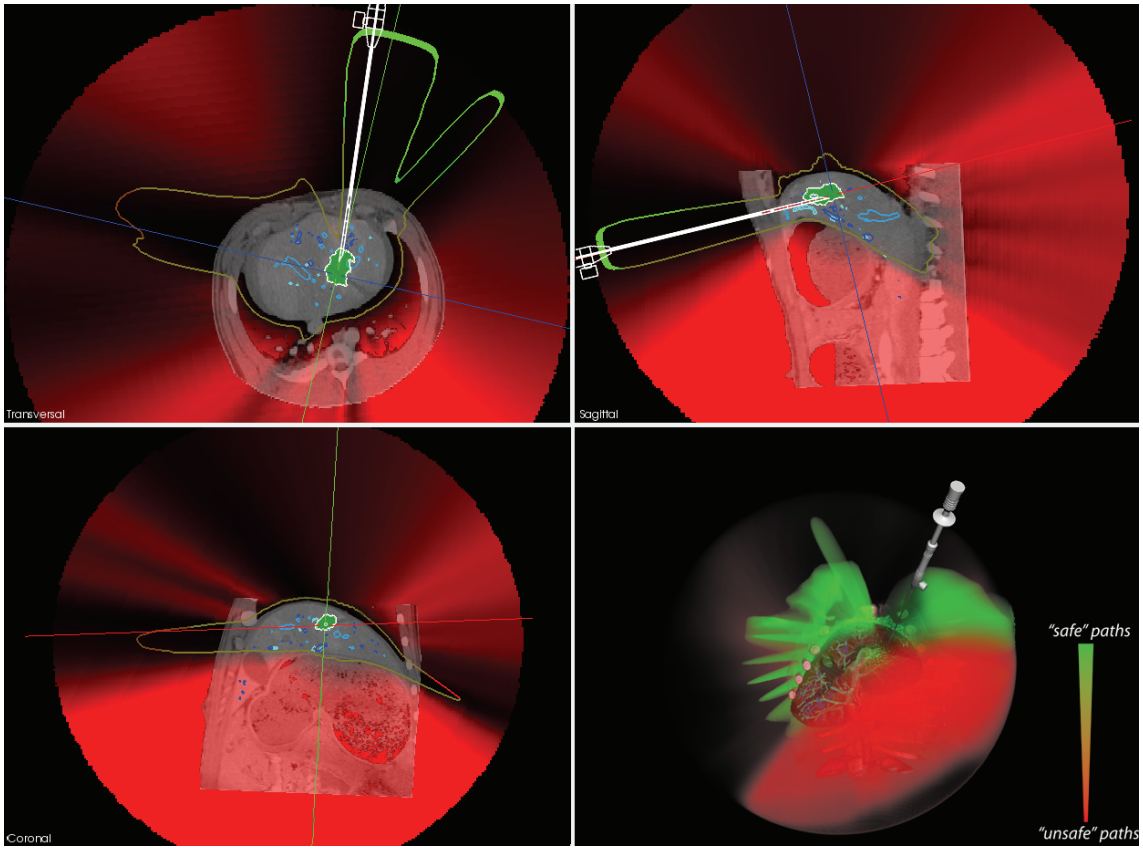


Figure 4.10: A screenshot of our medical visualization system with *area safety* and *path safety* augmentation. In this example, the path safety volume shows all dangerous and impassable paths in red. The area safety geometry shows all safe access areas. The 2D MPR views are aligned with the direction of the main axis of one safe access area. For datasets similar to this one, which show mainly large safe access areas, the area safety geometry can be smoothed and additionally displayed as translucent geometry in 3D, as shown here. The tool to be placed into the tumor (green) is a RFA needle. All vulnerable vessels are displayed in shades of blue.

We performed three different evaluations of our method. The first and most obvious one considered expert knowledge during the implementation process. We worked closely with medical experts and discussed all results during the various stages with them. However, in approximately 50% of the cases, our experts did not agree on a single optimal path. This disagreement was one of the main reasons we chose to visualize safe *areas* instead of direct path proposals as a single line.

Table 4.1: An overview of our accessibility evaluation system compared to the most similar systems proposed by Baegert *et al.* [8] and Schumann *et al.* [145]. The strengths of our method are highlighted with bold font.

	Baegert	**Schumann**	**Ours**
direct output	one path	path list	**all paths**
path visualization	3D[a]	2D[b]	**3D+2D**[c]
target structure	point[d]	point[d]	**full tumor**
computation space	2D	2D	3D
computation time	30s[e]	5s[f]	30s – 300s[g]
target application	RFA	RFA	**generic**
computation strategy	two-pass	multi-pass	single-pass
visualization strategy	cut outs	one path	volumetric

[a] areas and one path [b] one path [c] all paths [d] volumetric constraints [e] depends on polygon resolution [f] depends on numerical constraints [g] depends on tumor size

The online survey we conducted was mainly intended to evaluate the acceptance of the proposed visual enhancement by a broader medical audience. A full medical evaluation would also include a study on the intervention outcome of various medical procedures using our method compared to the classical planning approach based only on experience. Such a study is planned for our next project. The participants criticized the fact that the path selection took place in one selected axial plane only, although the plane was selected by an expert. In the future, we also plan to implement a scrollable medical-imaging interface for our online survey system.

Our retrospective gold-standard survey relies on the assumption of a perfect physician. As is shown by the available patient data, this assumption is of course not true. The

treatment outcome depends strongly on the difficulties inherent to the tumor position and also on further circumstances such as the overall patient history. Therefore, this evaluation gives an impression of the possible prospective benefit for unskilled physicians, training simulators, and complication investigation.

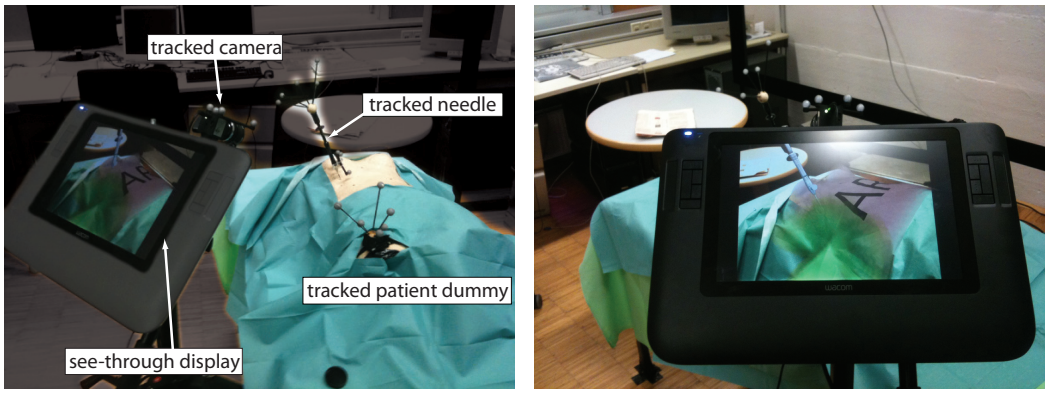
For the sake of completeness, we compare the most advanced access path planning systems of Schumann *et al.* [145] and Baegert *et al.* [8] with our approach in Table 4.1. Note that the main difference is that our system aims to provide a visualization to assist in planning and leaves the freedom of decision to the performing physician, instead of trying to find an optimal path in a fully automatic way.

4.7.1 Other applications

Respiration compensation: During abdominal and thoracic interventions, the respiratory motion of the organs cannot be prevented, which poses a severe problem for applying offline planning results to the actual intervention. A feasible approach to deal with this problem is to apply our method to every stage of a patient-specific respiration simulation sequence, as is possible, for example, with the XCat patient model [111]. We can then apply fast GPU-based registration methods between the time steps and between the patient scan and the model (e.g., Optical Flow motion fields [157]) and hence integrate the patient-specific anatomy into the model. We can thus warp a single scan of a patient to enough discrete sampling points of a whole respiration cycle (usually 10-20) and apply our method to each of the resulting volumes. The resulting path safety volumes are combined in a volume that only classifies paths as *safe* if no vulnerable structures are hit in any of the respiration states.

Interventional augmented reality: In addition to planning an optimal access path, finding that path during the intervention itself is also a challenging task. This stage can be defined as a third level of our method and can be configured to display either only the selected entrance segments or all paths of a selected safety level. We selected liver cancer RFA treatment for a first prototype of the system. The target clinical setup usually consists of a two-camera infrared tracking system and one or several monitors. The tracking system is small and movable to allow an optimal working area. Monitors can be provided as projections or, more commonly, as ceiling-mounted movable devices. Furthermore, additional monitors are provided outside the intervention room to control the imaging modality used. We use a small monitor mounted on a tripod to simulate the

intervention room's movable monitor and an additional monitor, away from the operating field, to simulate the movable monitor within the intervention room and an additional monitor, away from the operating field, to simulate the control monitor. The prototype of this system is shown in Figure 4.11.



(a) (b)

Figure 4.11: Prototype of an interventional Augmented Reality (AR) application, as a possible third level of our method. Using this system, planning decisions can be directly mapped to the real situation in the operating room.

Applications outside the medical domain: The applications of our method are not limited to medicine. For example, it can be directly applied to mechanical accessibility assistance. An example visualization can be seen in Fig 4.12.

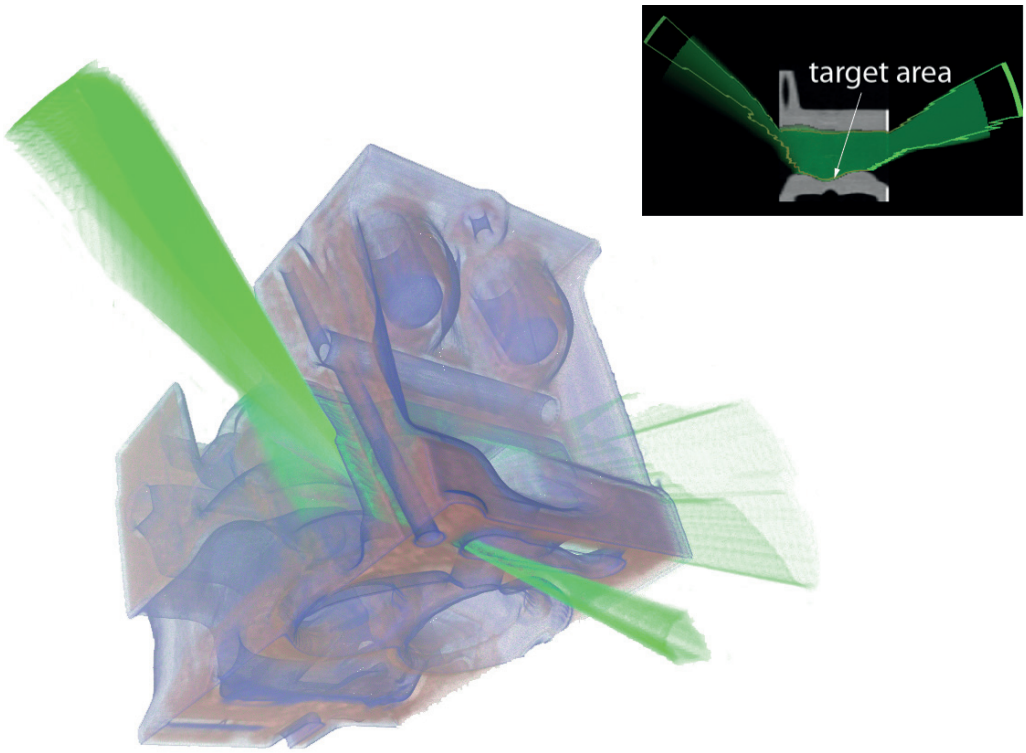


Figure 4.12: An example of the applicability of our method to other fields of research. We have chosen a spot within an engine block that is difficult to reach. Because the engine block itself is solid, only 100% *'safe'* paths are displayed. All other paths are impassable. Note the non-obvious access path at the bottom of the image.

Chapter 5

Smoke simulation

Contents

In this chapter, we use the visual appearance of smoke as an abstract metaphor to separate the available space between multiple variables. While the overall appearance of our visualization is similar to that of smoke, it is not driven by physical processes, but rather by the data that needs to be displayed. Nevertheless, we show that, despite a rather abstract interpretation of the phenomenon, the visualization is still perceived efficiently by the human visual system.

5.1 Motivation

While the representation of volumetric data sets that contain only scalar data has been well researched since the 1980-ies [105], multivariate visualization of multiple dependent or independent data values per volume element is still an active topic of research. Applications from medicine and engineering to meteorology and geology require the analysis of

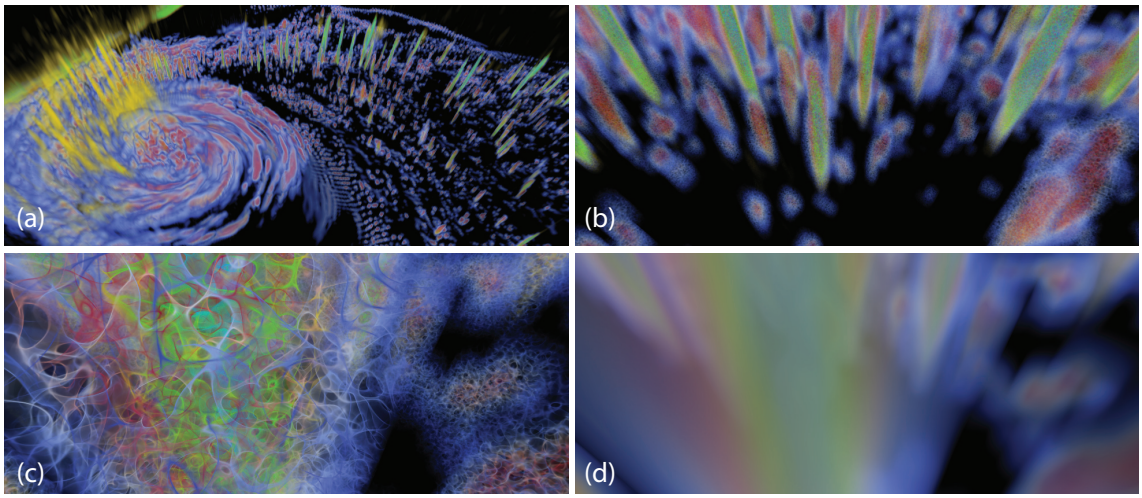


Figure 5.1: Visualization of hurricane Isabel simulation data: amount of cloud water (blue-red) and upwind strength (yellow-green). At large distances our method converges to normal direct volume rendering to avoid aliasing (a). However, at smaller distances the user is able to interpret the exact data values for both variables simultaneously (b, c). For example, it is possible to see the high amount of cloud water (red) in the middle of high-velocity upstream (cyan). This is impossible with normal direct volume rendering (d).

such multivariate data and are in need of comprehensive multivariate data visualization methods. However, currently data is usually mapped to a scalar dimension and evaluated separately with established state-of-the-art volume rendering methods. Furthermore, some data dimensions form secondary information, which should not distract a user from the primary information. Such secondary information can be data uncertainty or secondary sensor information. State-of-the-art volume rendering methods are neither able to visually differentiate between multiple data values nor between different information qualities.

The most straightforward approach to the visualization of spatially fixed scalar data is color mapping. A variety of mapping – or in case of 3D volume rendering – transfer functions have been developed to allow for accurate perception of displayed data, e.g., divergent color scales [118]. However, when multiple co-located continuous scalar values are to be displayed, the direct application of color mapping is not possible. This problem can be solved by blending the colors for different variables. However, Hagh-Shenas and colleagues have shown that color blending is inferior to *color weaving*, where each data attribute receives it's own dedicated portion of screen space. They show that the users were significantly more accurate in inferring individual values when the data were inter-weaved using a high-frequency texture pattern than when the colors were blended [60].

However, in 3D volume rendering, color blending is inherent to the algorithm itself even for single scalar dataset, as it uses transparency to reveal the internal structures. In this chapter, we propose a novel visualization method that allows simultaneous display of multiple 3D spatially fixed data sets (see Fig. 5.1 for an example visualization achieved with our method). We present the following technical contributions in this work:

- We propose an algorithm for creating an opacity redistribution pattern, inspired by smoke appearance, using procedural noise (Section 5.3.1). We set the frequency of the noise in a way that no increase in sampling rate is required and the information in the data is not lost. An independent noise function is used for every data variable, which allows simultaneous display of several data variables at the same location. We also propose a way of constructing an opacity mapping function, which maintains the visibility of internal structures dictated by an arbitrary transfer function used for data classification (Section 5.3.3).

- We propose a filtering approach for the opacity mapping function that allows to avoid aliasing. It does not require increasing the sampling distance along the ray or supersampling in screen space. We also show that our method converges to conventional direct volume rendering, when the noise cannot be represented on screen without aliasing (Section 5.3.4).

We have applied our method for the visualization of three-dimensional simulation data in combination with isosurface uncertainty information and for multivariate climate data (Section 5.4). Furthermore, we have evaluated our method by comparing it to other visualization techniques in a controlled user study (Section 5.6.1). The results of our study show that users are significantly more accurate in determining exact data values with our visualization method and subjectively prefer it over other methods commonly used for multivariate visualization.

While our method is qualitative in nature, it may be used by experts to derive hypotheses about dependencies between variables directly from 3D renderings. To verify these hypotheses, quantitative data probing methods such as picking or slicing can be used. Overall our method can be seen as an approach for *overview* and *zoom* tasks identified in the *Visual Information Seeking Mantra**. From this point of view, transfer functions (1D, 2D, and others) are responsible for the *filtering* task and direct picking or slicing for *details-on-demand*.

*Overview first, zoom and filter, then details-on-demand.

In addition, we extend our method for visualization of multivariate 2D data. While the approach for 2D is very similar in spirit to the 3D version, lack of perspective-related distortions in 2D data visualization allows us to display additional data variables using the orientation and frequency of the noise. Furthermore, by using a spatially varying noise, we adapt the resulting pattern to be alias free and easily interpretable at any zoom level. To produce the final image, the generated pattern is composed with underlying color information by modifying the value component using the HSV color model. We have conducted a preliminary user-study to find suitable contrast levels for our method, and we show strong evidence that our method is effective by providing the results of a second comparative user study, which shows that our method surpasses commonly used 2D multivariate visualization methods.

5.2 Background

Visualization of multivariate 3D data The methods for displaying multivariate 3D data can be roughly subdivided into two distinct classes: fusion-based methods which fuse the data at various stages of the rendering pipeline, and methods that employ different communication channels or rendering styles for each of the variables within the dataset. In this section we give an overview of these two groups of methods. For more in-depth analysis and categorization of methods for visualization of multivariate data, we refer the reader to the recent work of Kehrer and Hauser [82].

For fusion-based methods, the main problem is the exact choice of fusing of multiple values at each location. One common approach is to analyze the data to extract features of the dataset based on multiple variables. For example, Kniss et al. approach this problem by specifying a multidimensional transfer functions during the classification stage of the volume rendering pipeline for generic multivariate data [90] and special case of data uncertainty [91]. However, these approaches do not directly display multiple variables at the same location simultaneously.

Another approach is to apply different transfer functions to make the values of different variables visually distinct. Cai and Sakas compare three fusion methods applied at different stages of the rendering pipeline, namely image level intensity fusion, accumulation level opacity fusion, and illumination model level parameters fusion [26]. Rößler et al. use straightforward alpha blending in a texture-based volume rendering system to mix the colors acquired from distinct transfer functions to display functional brain data [138]. Similar approaches were also applied in the context of raycasting [58, 77]. Akiba et al.

additionally employ user-controlled weighted color blending to assign different importance levels to variables when displaying the turbulent combustion simulation data [1].

Other methods use different communication channels for the visualization of the variables within a multivariate dataset. For example, Crawfis and Max apply noise to communicate secondary information in volume data [35]. However, their method is only applicable for splatting-based volume visualization and has not been evaluated. Similarly, Djurcilov and colleagues propose to use noisy textures to convey the information about the uncertainty of volumetric data [41]. Yu et al. combine volume and particle rendering to display two variables for in-situ visualization of large-scale combustion simulation [175].

Note that our method is not intended to replace the fusion-based methods and may be used in conjunction with them to improve the readability of exact data values and increase the number of variables that can be displayed simultaneously. For instance, at least two variables may be displayed with our method, as opposed to one volume-rendered variable, and an additional one – using glyphs or particles.

Color weaving Our method is similar in spirit to color-weaving approaches that proved to be effective for 2D data. Urness et al. propose to weave the color-coded scalar variables using a line integral convolution texture instead of blending the colors [156]. Hagh-Shenas et al. have shown that weaving-based approaches are significantly better than blending-based ones in showing distinct values for multiple variables. Additionally, Livingston et al. have shown that methods which rely on subdivision of available screen space and visualization of each attribute with a distinct color scale are most effective [109].

Visualization of continuous 2D multivariate data Tang et al. propose to use natural textures for the visualization of weather data [153]. While this method is comparable to our method, the authors do not discuss the behavior of their method during user interaction. Given that the texture is generated once per dataset and not adapted to the zoom level, this method may lead to strong aliasing artifacts during data exploration.

The attribute blocks method by Miller subdivides the screen into a regular grid of blocks [117]. Each block corresponds to an array of visual representations (*lenses*). In each lens, a single attribute is displayed with color maps [156] (saturation of a distinct color). The author proposes two possibilities for the behavior of attribute blocks, when the user scales the map. The first possibility is to link the attribute blocks to the object space, i.e., the attribute blocks are scaled along with the map. This may lead to undesired results on large and small scales, especially if the size of a single attribute lens becomes

smaller than a pixel. The second possibility is to link the attribute block size to screen space. This leads to an effect resembling a 'screen door of lenses', as the map moves and the grid of attribute blocks stays static. In contrast, our method adds or removes the details smoothly in-place, which results in a more consistent visualization *during* the scaling process.

Healey et al. employ perceptually-based brush strokes for non-photorealistic visualizations [66]. The authors discuss the behavior of their method during scaling. In contrast to our work, they discuss the display of additional data rather than the adaptation of the approach for particular zoom level.

Kosara et al. refer to blur based methods as *semantic depth of field* (SDOF) [93]. Their main idea is to blur information which is not relevant for the current cognitive task. However, the authors showed in later work [94] that SDOF cannot be used as a full visualization dimension, because users are not able to distinguish between different levels of blur.

Noise-based 2D methods Noise-like textures were used for information display in the work by van Wijk based on 'spot noise' [159], which has been extended for multivariate data visualization by Botchen et al. [21]. However, these methods require a complex definition of a constructing element (i.e., spot shape), which should be adapted for every task at hand. In the work by Holten et al. [70], the authors propose a method to generate textures with desired properties for filling several disjoint areas (e.g., member states of the USA). The drawback of direct spectral methods of texture synthesis is that they lack local spatial control, and this almost prohibits their use for the visualization of continuous data. While these methods are very powerful, they are not zoom-independent and their performance may be compromised during the data exploration stage.

Coninx et al. propose to use animated Perlin noise in conjunction with classic color maps to convey the information about uncertainty of one scalar field displayed on 3D surfaces [32]. The authors use the intensity of Perlin noise multiplied by the amount of uncertainty as lookup offset in a color map. As a result, the data appears noisy in areas where data is uncertain. Thorough analysis of low-level perception of noise contrast allows to adjust the noise to ensure that the uncertain areas are visible in the resulting visualization. While our method shares the idea of using noise for visualization purposes, our method aims at visualizing arbitrary data and is not limited to uncertainty of scalar fields.

Stylized rendering Preserving texture appearance and scale is a common task in stylized rendering and animation. In this context, it is referred to as *flatness requirement* [12]. Two other requirements need to be met to assure high quality animation with stylized rendering. *Motion coherence* refers to the correlation between the motion of a 3D scene and the motion of stylization primitives in screen space, and *temporal continuity*, corresponds to the absence of popping and flickering artifacts. These three goals are inherently contradictory and therefore a tradeoff between them must be made. However, the motion coherence requirement is not relevant for our 2D application. Therefore, we only need to preserve flatness and temporal continuity. We use one of the methods that satisfies these two requirements well, namely random-phase Gabor noise, as proposed by Lagae et al. [104], as a basis for our visualization method.

5.3 Method

When developing our method for rendering multivariate 3D data, we have set two important goals that we wanted to meet:

- The color-coded scalar values should be easily interpretable by the user.
- The see-through properties of volume-rendering techniques should be maintained.

Unfortunately, these goals contradict each other, because the lower the opacity that allows better see-through capabilities, the lower is the discriminability of the color of a particular voxel. To approach this problem, we notice that at high zoom levels, when the user explores the details of a particular feature within a dataset, every voxel usually occupies a significant portion of the screen. Therefore, we aim at redistributing the opacity of a voxel by making parts of it more opaque, thus improving the readability of the value within this voxel, while making the other parts more transparent and maintaining the see-through capabilities.

To achieve these goals, we superimpose a high-frequency pattern which is used to redistribute the opacity within a voxel (see Fig. 5.2). This also allows us to avoid color blending when displaying multiple co-located data variables. In this section, we will describe how to create such a pattern (Section 5.3.1), how to ensure that the average opacity is maintained after redistribution (Section 5.3.3), and how to avoid the potential aliasing that may arise from high frequencies caused by the opacity redistribution (Section 5.3.4).

Extension for data 2D visualization For two-dimensional data, we employ the 2D version of the Gabor noise. However, because projection-related effects, such as perspective foreshortening, do not apply in 2D case, we modify our method to visualize at least three variables in two dimensions. †

5.3.1 Redistribution pattern

To avoid dependency on the viewing direction caused by regular redistribution patterns, such as shown in Fig, 5.3, we generate the opacity redistribution pattern using an isotropic noise function. More precisely, we use an isotropic version of random-phase Gabor noise [101], which offers precise control over its parameters and has a predictable intensity distribution as compared to other procedural noise functions [102].

Random-phase Gabor noise [101] is noise of a sparse convolution type. The value of the noise function $\mathcal{N}$ is computed as a convolution of impulses at random positions $\mathbf{x}_i$, which are defined by a Poisson impulse process, using a phase-augmented Gabor kernel:

$$g\left(\mathbf{x}; a, \omega, \phi\right) = e^{-\pi a^2 |\mathbf{x}|^2} \cos\left(2\pi \mathbf{x} \cdot \omega + \phi\right), \tag{5.1}$$

$$\mathcal{N}\left(\mathbf{x}; a, \omega\right) = \sum_i g\left(\mathbf{x} - \mathbf{x}_i; a, \omega, \phi_i\right), \tag{5.2}$$

where a is the bandwidth, ω is the frequency, and ϕ_i are random phases that are distributed according to an uniform distribution in the interval $[0, 2\pi)$.

As the support of a Gabor kernel is not limited, the computation of noise values would require computing an infinite sum. However, the exponential part of the Gabor kernel falls off very quickly, so it can be truncated without introducing recognizable artifacts. To compute the noise value, Lagae et al. have introduced a virtual grid of size G, which equals the radius of the Gabor kernel truncated at a threshold level t [103]. To compute the noise value for a point within a cell, only impulses within this cell and its neighbors are considered in the summation in Eq. (5.2).

We set the frequency magnitude of the noise to $|\omega| = 1/v$, where v is the minimum extent of a single voxel in the dataset. On the one hand, this frequency is sufficiently high so that a single voxel will have a wide variety of noise values, which allows the mapped opacity to have both opaque and transparent parts according to the opacity mapping function (see Section 5.3.3). An illustration of the behavior of the noise function within

†Notice that all the visual results related to 2D data visualization were achieved using a different, but visually equivalent opacity mapping function [86].

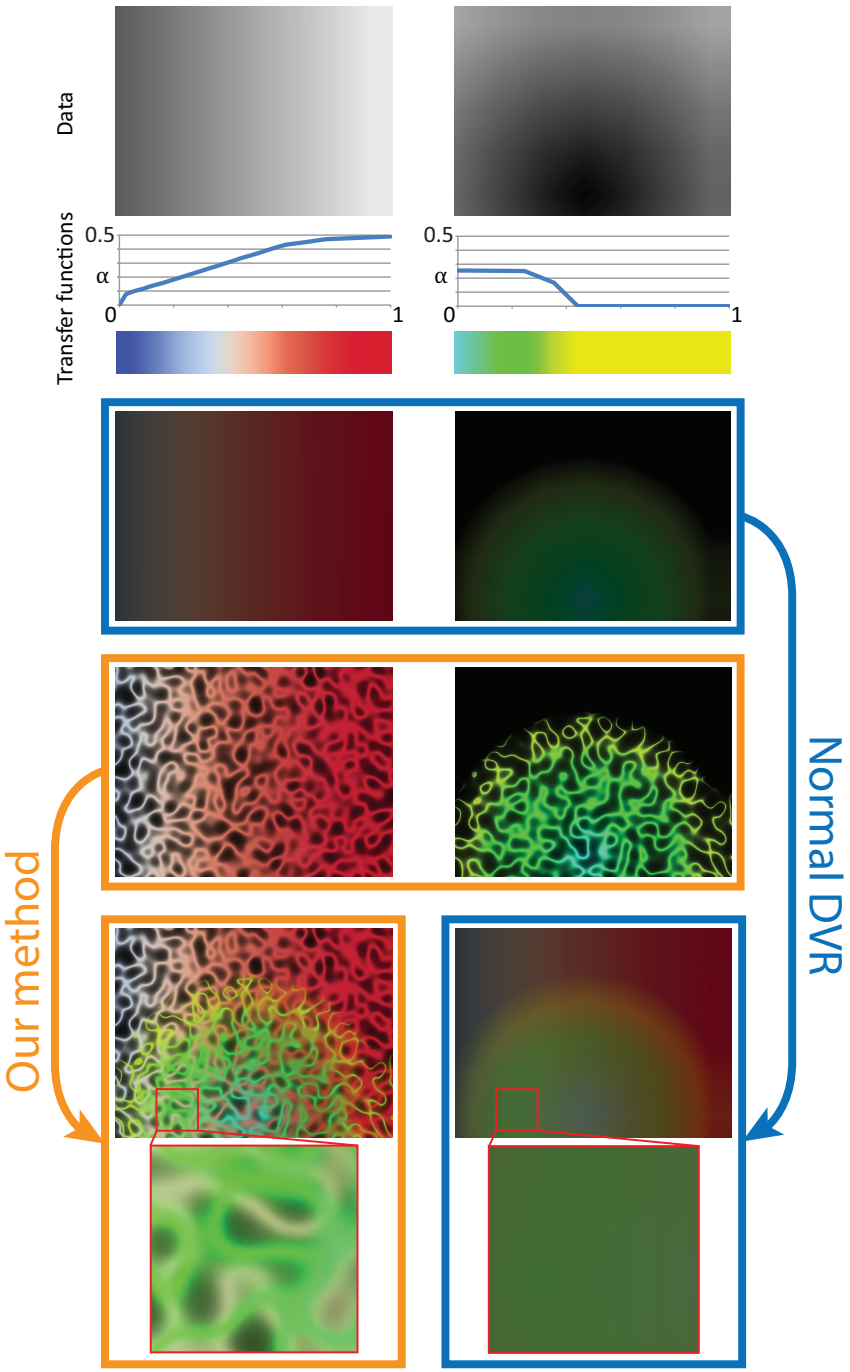


Figure 5.2: An artificial example showing the comparison of visualization of two variables with normal direct volume rendering to our method. Our method redistributes the opacity using a noise pattern. Notice that with our method the individual color-coded data values (gray and light green, bottom-left) can be clearly seen, while for DVR distinguishing these colors is impossible due to color mixing (bottom-right). The size of the enlarged areas in the bottom row is one voxel.

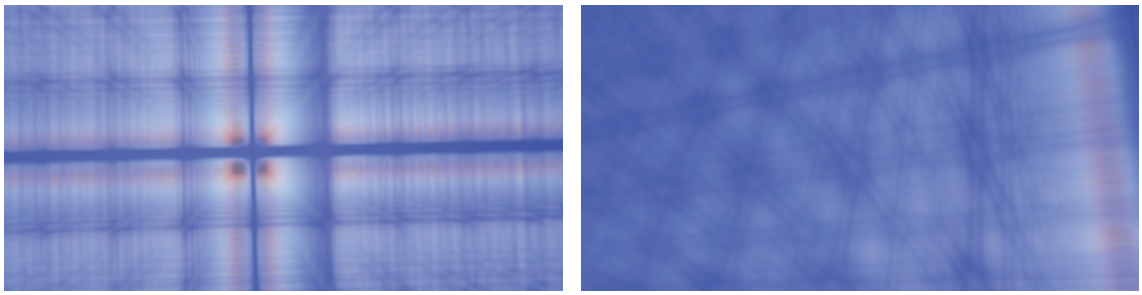


Figure 5.3: An illustration of problems with regular redistribution pattern in 3D when viewing along one of the axes and in oblique direction.

a single voxel is shown in the bottom-left panel of Fig. 5.2. On the other hand, this frequency is low enough so that the sampling frequency during raycasting does not need to be increased to maintain visual quality (Section 5.3.4).

The orientation of the frequency vector ω is randomized to obtain isotropic noise [101]. For convenience, we denote the frequency magnitude as $f = |\omega|$. The size of the virtual grid G is consequently set to [168, p. 164]:

$$G = {}^{1}/_{f}. \tag{5.3}$$

The parameter a is then set to:

$$a = \sqrt{\frac{-\ln(t)}{\pi}} \cdot f, \tag{5.4}$$

so that the size of a Gabor kernel truncated at the level t equals the grid size G [103].

5.3.2 2D redistribution pattern

For two-dimensional data, we encode the variables not only with color, but also with redistribution pattern orientation and frequency. To achieve this, we use a spatially varying version of the Gabor noise [104]. Below, we refer to this 2D opacity redistribution pattern as the *noise texture*.

Using spatially varying random-phase Gabor noise, we can convey three distinct scalar data attributes. To visualize the first attribute, we use local texture frequency. The second attribute is shown with local texture orientation. We then use the resulting texture to modulate the 'value' component of the color-coded third attribute using the HSV color model. As the considerations of color accumulation arising from volume rendering are not

applicable in case of 2D data, we display the third attribute in a dense manner.

Frequency control Without loss of generality, let us assume that the value x that has to be conveyed in the visualization lies in the interval $x \in [0, 1]$. To avoid sliding artifacts during panning and zooming, we compute the noise values using the coordinates in the original data space. However, to ensure that the screen space frequency of the resulting texture lies in the range perceived well by the human visual system, we adjust the noise parameters according to the current zoom level. To achieve that, we choose the grid size G in data space so that its screen space size lies within this range and the exact size corresponds to the value being visualized:

$$G = Z^{-1} \cdot s \cdot 2^{(1-x)\cdot \log_2(S/s)}, \tag{5.5}$$

where s and S are the desired minimum and maximum screen-space sizes of the noise, and Z is the current zoom level defined by the size in pixels of one data grid cell on the screen. The exponential dependence on the visualized value is due to the fact that repeatedly doubling the texture frequency results in equal perceptual steps, while adding a constant value does not [70]. The radial frequency f and the bandwidth a can then be computed using the equations (5.3) and (5.4).

The parameters for the opacity mapping function can be then computed using the local noise frequency f and the number of samples per unit-length segment N_v (see equation (5.13)):

$$N_v = Z \cdot U, \tag{5.6}$$

where Z is the zoom level, and U is the number of voxels per unit length. Finally, the variance of intensity distribution for two-dimensional random-phase Gabor noise can be computed as $\sigma_n^2 = {}^{-N}/_{4\ln(t)}$, where t is the level of truncation of Gabor kernels, and N is the average number of impulses per cell.

Orientation control As opposed to 3D redistribution pattern, we do not perform randomization of the angular frequency to obtain anisotropic noise. By choosing the angular component of $\{\omega_i\}$ according to one of the displayed data attributes, we are able to link the local orientation of the resulting texture to the value of that attribute.

5.3.3 Mapping noise values to opacity

Once the redistribution pattern is set up, it is necessary to define the mapping of its values to opacity, which are used in the final DVR step. To maintain the classification that is already defined by the classical DVR transfer function, we impose the following condition on the opacity mapping of the noise values:

$$\forall \alpha : \int_{-\infty}^{\infty} p(t) \cdot \mathcal{M}_\alpha(t) \mathrm{d}t = \alpha, \tag{5.7}$$

where $p(t)$ is the probability density function of the intensity distribution of the noise, and $\mathcal{M}_\alpha(t)$ is the mapping function that corresponds to a particular opacity value α. In other words, we maintain the average opacity value that is defined by the regular transfer function.

The opacity mapping function $\mathcal{M}_\alpha$ may be chosen arbitrarily, as long as it satisfies condition (5.7). We use Gaussian mapping functions, which allow us to achieve high quality rendering without requiring much additional computations (see Sec. 5.3.4 for more details). The family of mapping functions is defined as:

$$\mathcal{M}_\alpha(x) = C_\alpha \cdot e^{-\frac{x^2}{2\sigma_\alpha^2}}, \tag{5.8}$$

where the parameters C_α and σ_α are chosen to satisfy the condition of Eq. (5.7). The intensity distribution $p\,(t)$ of three-dimensional random-phase Gabor noise can be closely approximated by the normal distribution with zero mean [101]:

$$p\,(t) = \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{t^2}{2\sigma_n^2}}, \text{with} \tag{5.9}$$

$$\sigma_n^2 = \frac{\lambda}{2\left(\sqrt{2}a\right)^3}, \tag{5.10}$$

where λ is the Gabor noise impulse density. Substituting equations (5.8) and (5.9) to equation (5.7), we can derive that:

$$\sigma_\alpha = \frac{\sigma_n \alpha}{\sqrt{C_\alpha^2 - \alpha^2}}. \tag{5.11}$$

To ensure that $\sigma_\alpha \in \mathbb{R}_+$ and that $\mathcal{M}_\alpha(x) : \mathbb{R} \to [0, 1]$, the parameter C_α may be chosen arbitrarily from the interval $(\alpha, 1]$. However, an additional constraint on C_α is necessary due to the requirements of the filtering step as described in Section 5.3.4.

5.3.4 Filtering

Applying an opacity mapping function directly to the noise pattern introduces additional high frequencies, which may lead to aliasing during rendering. This aliasing may appear both in the dataset space during the accumulation of color for a single pixel, and in screen space (see Fig. 5.4, top).

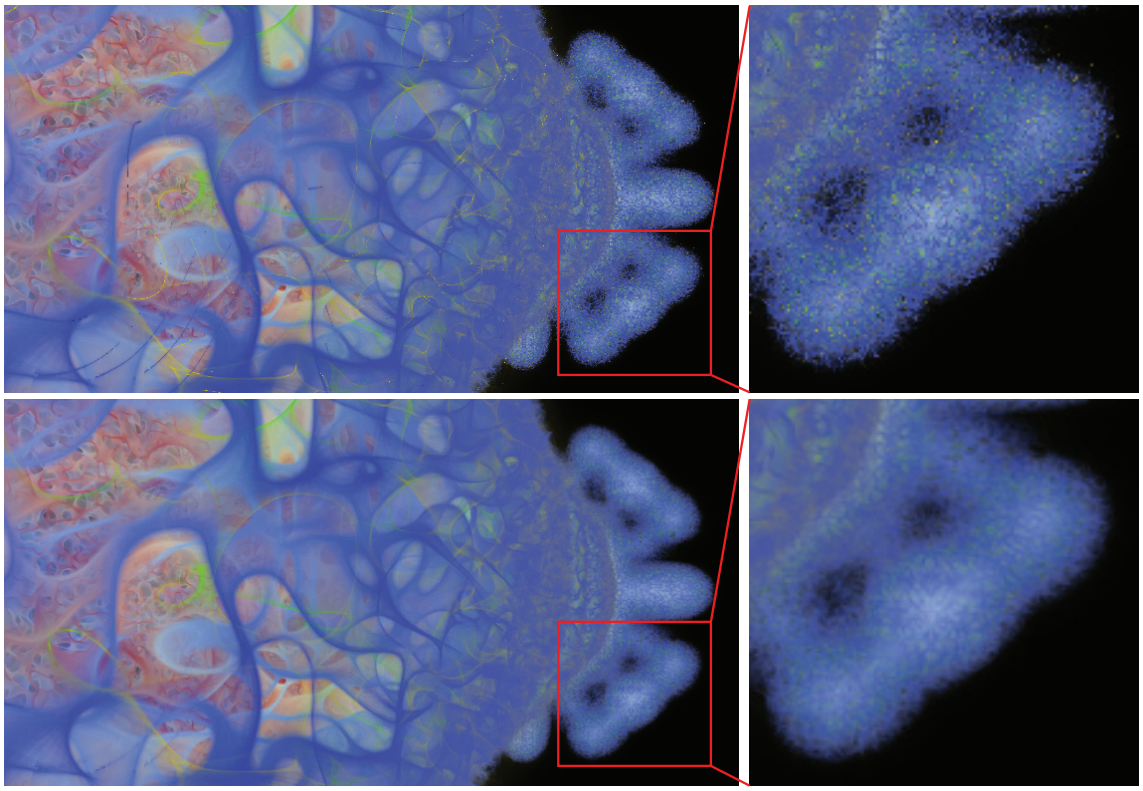


Figure 5.4: Aliasing that occurs due to high frequencies in the noise modified by the opacity mapping function (top). Proper filtering eliminates this aliasing (bottom).

To avoid the dataset-space aliasing, we use the properties of the opacity mapping function. As the opacity mapping function is a Gaussian function, we use the possibility for analytic integration of Gaussian transfer functions to reduce the aliasing [90]. Note that this analytic integration applies only to the opacity mapping function and does not impose any constraints on the transfer function used for data classification.

Aliasing in screen space occurs when the screen-space sampling frequency, defined by the image resolution, is not high enough to represent the data projected on the image plane. One way to deal with this undersampling is to use multisample anti-aliasing (MSAA) techniques. The drawback of the MSAA approach is that the number of rays that need to

be cast to achieve the final anti-aliased image is significantly increased. In order to avoid this performance penalty, we reduce the data frequency by modifying the opacity mapping function instead of increasing the screen-space sampling frequency.

The maximum frequency of a signal – the signal is the noise function in our case – is modified by our opacity mapping function as [13]:

$$f_{\mathcal{N}_\alpha} = \max_x \left|\mathcal{N}'(x)\right| \cdot f_{\mathcal{M}_\alpha}, \tag{5.12}$$

where $f_{\mathcal{N}_\alpha}$ is the maximum frequency of the resulting function, $f_{\mathcal{M}_\alpha}$ is the maximum frequency of the opacity mapping function, and $\mathcal{N}'(x)$ is the derivative of the noise function.

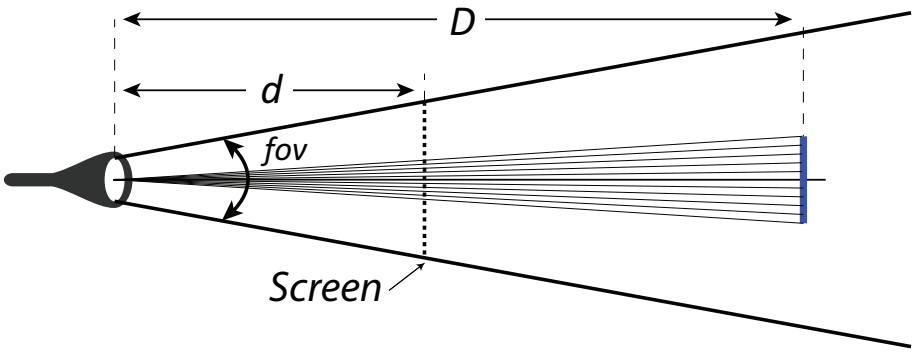


Figure 5.5: Sampling of a unit-length segment (blue) according to the pixels on the screen using one ray per pixel. d is the distance to the near plane, fov is the camera's field of view, and D is the distance from the camera to the segment.

The function with object-space frequency of $f_{\mathcal{N}_\alpha}$ is sampled on the screen, where every pixel is one sample. To ensure that the opacity-mapped noise function is sufficiently sampled, we have to ensure that the sampling frequency defined by the screen resolution is above the Nyquist rate for the opacity-mapped noise signal projected onto the screen. For convenience, we invert this problem and solve it in object space. Consider a unit-length segment at a distance D from the camera (see Fig. 5.5). The number of samples N_v along this segment can be computed as:

$$N_v = \frac{H}{2D\tan(fov/2)}, \tag{5.13}$$

where H is the screen resolution, fov is the camera's field of view, and D is the distance from the camera to the segment. N_v is exactly the frequency at which the projected signal at distance D is sampled by pixels. As we would like to avoid changing this sampling

frequency for performance reasons, it is necessary to modify the signal, so that:

$$f_{\mathcal{N}_\alpha} < \frac{N_v}{2} = \frac{H}{4D\tan(fov/2)}. \tag{5.14}$$

The first multiplier contributing to the frequency of the opacity-mapped noise function (Eq. (5.12)) is the derivative of the noise. As the noise function is the sum of Gabor kernels (see Eq. (5.2)), the conservative estimate for its maximum derivative can be computed as the sum of maximum derivatives of the Gabor kernels, given the impulse density λ. As only the cell of the virtual grid the point belongs to and the neighboring cells are considered in the computation of the noise due to negligible influence of impulses located further away, the maximum noise derivative can be conservatively estimated as:

$$\max_x \left|\mathcal{N}'(x)\right| < 27 \cdot \lambda G^3 \cdot \max_x \left|g'(x)\right| \tag{5.15}$$

By analyzing the structure of the Gabor kernel, we can write:

$$\begin{aligned} &\max_x |g'(x)| = 2\pi e^{-\pi a x_0^2} \cdot (f + a x_0), \\ &\text{where} \quad x_0 = -f/2a + \sqrt{(f/2a)^2 + \tfrac{1}{2\pi a}} \end{aligned} \tag{5.16}$$

The second multiplier contributing to the frequency of the opacity-mapped noise function (Eq. (5.12)) is the maximum frequency of the opacity mapping function. The frequency spectrum of the Gaussian opacity mapping function $\mathcal{M}_\alpha$, defined in Eq. (5.8), is another Gaussian:

$$\mathcal{F}\left\{\mathcal{M}_\alpha(x)\right\}(f) = C_\alpha \cdot \sqrt{2\pi}\sigma_\alpha e^{-2\sigma_\alpha^2 \pi^2 f^2}, \tag{5.17}$$

where $\mathcal{F}\{\cdot\}$ is the Fourier transform. Obviously, it contains arbitrarily high frequencies with non-zero amplitude, and, thus, the requirement given in Eq. (5.14) cannot be strictly satisfied. Therefore, we leave out of account the frequencies whose amplitude is less than a fraction τ of the maximum amplitude M in the frequency spectrum of the opacity mapping function $\mathcal{M}_\alpha$. This maximum amplitude M is reached at $f = 0$. As the frequency spectrum of the opacity mapping function is a zero-mean Gaussian function, we can define a cutoff frequency f_c, whose amplitude in the spectrum equals $\tau \cdot M$:

$$\begin{aligned} \mathcal{F}\left\{\mathcal{M}_\alpha(x)\right\}(f_c) &= \tau \cdot \max_f \mathcal{F}\left\{\mathcal{M}_\alpha(x)\right\}(f) = \\ &= \tau \cdot \mathcal{F}\left\{\mathcal{M}_\alpha(x)\right\}(0) = \tau \cdot M \end{aligned} \tag{5.18}$$

Consequently, all the frequencies with absolute value above f_c will have the amplitude

below $\tau \cdot M$:

$$\forall f, |f| > |f_c| : \mathcal{F}\{\mathcal{M}_\alpha(x)\}(f) < \tau \cdot M. \tag{5.19}$$

Combining equations (5.17) and (5.18) and substituting σ_α from Eq. (5.11), we can derive that:

$$f_c = \frac{\sqrt{-ln(\tau)}}{\sqrt{2}\pi\sigma_\alpha} = \frac{\sqrt{-ln(\tau)\cdot(C_\alpha^2 - \alpha^2)}}{\sqrt{2}\pi\sigma_n\alpha} \tag{5.20}$$

Finally, we consider that the maximum frequency of the opacity mapping function is $f_{\mathcal{M}_\alpha} = f_c$.

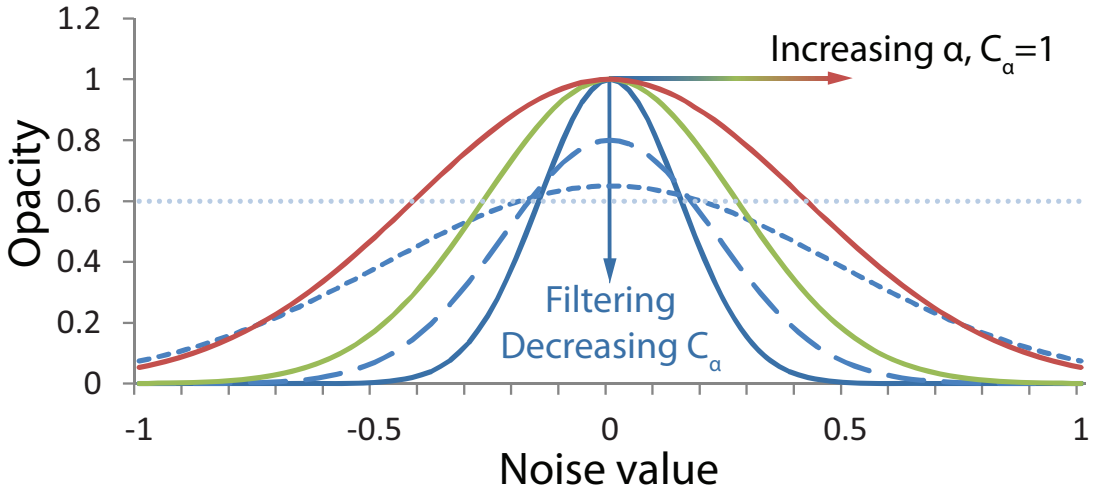


Figure 5.6: Several representative opacity mapping functions with varying parameters. The blue graphs show how the opacity mapping function for $\alpha = 0.6$ behaves, when additional filtering is required to avoid aliasing. Green and red graphs show the opacity mapping functions for $\alpha = 0.8$ and 0.9, respectively, for $C_\alpha = 1$. For all graphs, $\sigma_n = 0.2$.

From equations (5.12), (5.14), and (5.20), we get the following expression for C_α:

$$C_\alpha = \min\left(1, \alpha\sqrt{1 - \frac{\pi^2 H^2 \sigma_n^2}{8ln(\tau)\tan^2(^{fov}/_2)D^2 \max_x |\mathcal{N}'(x)|^2}}\right), \tag{5.21}$$

which gives exactly one solution for the parameters of the noise opacity mapping function at each sample in conjunction with equation (5.11). Please see Fig. 5.6 for an illustration of how the opacity mapping function changes with changing parameters C_α and σ_α. In our implementation, we set $\tau = 0.15$, which is sufficient to avoid recognizable screen-space aliasing. See Fig. 5.4 for the comparison of filtered and unfiltered results.

Note that when the noise function under certain viewing conditions has a frequency

that is too high to be represented on the screen without aliasing, the rendering results with our method will be equivalent to regular volume rendering. For example, when the distance to the sampling point increases, the term under the square root in equation (5.21) approaches 1, meaning that C_α approaches α, and σ_α approaches infinity. This means that $\forall x : \mathcal{M}_\alpha(x) = \alpha$, and the result of the DVR will be equivalent to conventional volume rendering (see Fig. 5.7, top row).

5.3.5 2D color blending

While for 3D volume rendering, we rely on conventional alpha blending for compositing the final image, blending the obtained 2D noise texture with the background color deserves special attention. According to Holten et al. [70], additive, multiplicative or alpha blending do not work well. They use a combination of additive and multiplicative blending depending on the value encoded in the texture. However, we observed that this method does not work well with textures obtained with our method, as too little of the original color stays visible. Therefore, in order to combine our texture with color, we convert the RGB color value to the HSV color space and modify the value component (v-component) according to the computed texture intensity at this point in space. This allows us, on the one hand, to maintain the hue and saturation of the original color and, on the other hand, to have enough contrast for the texture to be perceptible. Furthermore, we can keep a significant amount of the original v-component. There are two reasons for doing so. First, keeping a part of the original v-component helps us to maintain the resolution of the underlying color image, as zero values of the noise transfer function do not result in a black color. Second, we do not need the full value component range to make the texture perceptible. Overall, the value component of the color is computed as:

$$\hat{C}_v = \beta \cdot C_v + (1 - \beta) \cdot \alpha_n, \tag{5.22}$$

where β is a constant whose value controls the amount of the original v-component in the resulting image, C_v is the original, and $\hat{C}_v$ is the modified v-component. In our implementation, we set $\beta = 0.6$ (see Section 5.6.2).

5.4 Applications

5.4.1 3D Applications

In this section, we show two applications of our method. The first application deals with climate analysis. For this example, all the displayed variables are equally important The second application shows how our method can be applied to visualize primary information and secondary information concurrently with low visual distraction. For this example, we use fuel injection simulation data as primary information and the level crossing probability for one of the isovalues as uncertainty information, hence as secondary information. The interaction with these visualizations as well as comparison to conventional DVR are shown in the accompanying video [‡].

5.4.1.1 Climate data

Climate data obtained through simulation or measurement often contains multiple scalar attributes, such as air temperature, humidity, pressure and others. Understanding the relationship between these attributes is often crucial. In this example application, we apply our method to visualize the simulation of hurricane Isabel from the National Center for Atmospheric Research in the United States. We have selected two scalar attributes, namely the amount of cloud water and the strength of upwinds, which accompany cloud formation [43]. The resolution of the datasets is $500 \times 500 \times 100$ voxels. The datasets are mapped to color using two distinct color scales. The amount of cloud water in range $[0.0, 0.7]^{g}/_{m^3}$ is mapped to a red-to-blue diverging color scale [118] with constant opacity value of 0.03. The upwind velocity in range $[1.0, 4.7]^{m}/_{s}$ is mapped to a yellow-green-cyan color scale with opacity linearly rising from 0.01 for weak upwinds to 0.12 for strong ones. Figure 5.7 shows different zoom levels in comparison with conventional DVR. Our visualization allows reading the data values for both variables simultaneously, revealing the high amounts of cloud water deep within the strong upwind plume. With regular DVR, this interdependency of the data variables cannot be seen without additional effort.

5.4.1.2 Isosurface uncertainty

In some cases the information provided by an additional data attribute may carry contextual, less important information. To illustrate how our method is applied to such a use case, we visualize the results of a fuel-injection simulation simultaneously with the

[‡]http://www.computer.org/cms/Computer.org/dl/trans/tg/2013/12/extras/ttg2013122926s.mp4

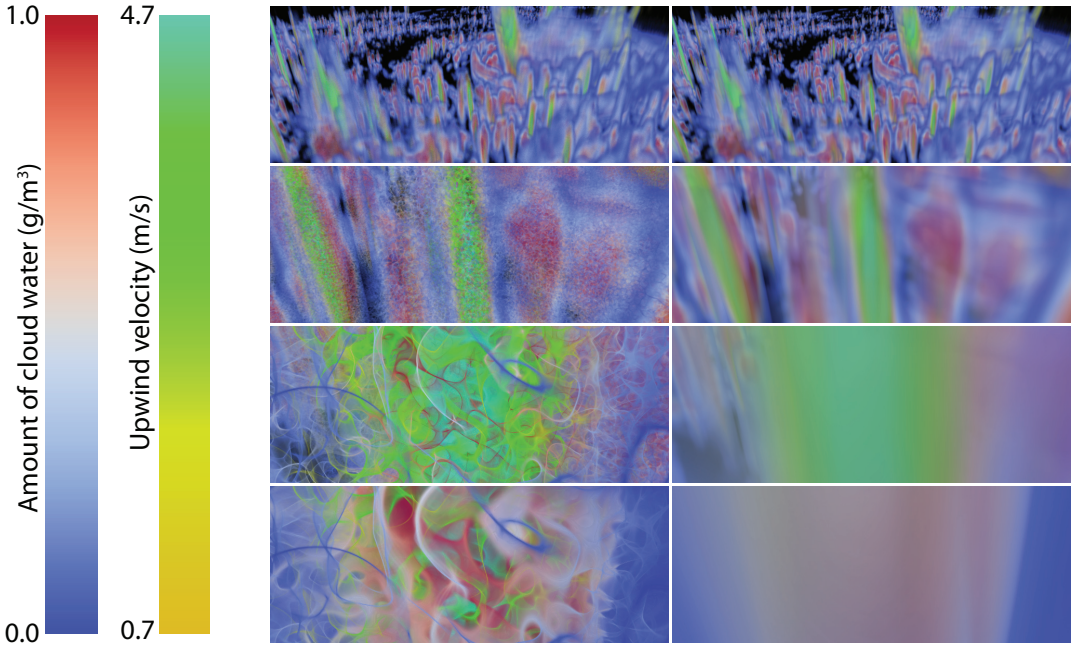


Figure 5.7: Climate data visualization with two attributes: amount of cloud water (blue-red) and upwind strength (yellow-green-cyan). Our method is shown on the left, conventional direct volume rendering on the right. The color scales are shown on the far left. Use-case scenario: An analyst at first tries to get an overview of the data (top row). At this zoom level, our method and DVR generate identical images, because the noise is automatically filtered out to avoid aliasing. When analyzing data details, our method shows that also within the plumes with high upwind velocity a high amount of cloud water is present, while this information is completely lost in DVR. Furthermore, if opacity values are chosen non-optimally (bottom row), our method is still able to convey information about all attributes, while DVR hides all information from the analyst.

information about positional uncertainty of one of the isosurfaces of this data. In order to minimize the impact of displaying additional information along with the main data, we set the opacity of the positional uncertainty information to a significantly lower value. In this example, the opacity of the uncertainty information is five to ten times lower than that of the main data.

To keep this example comprehensible, we employ the level-crossing probability (LCP) [128]. Although it overestimates the probabilities according to Pfaffelmoser and colleagues [126], it is sufficient for our use case. Figure 5.8 shows the resulting images for this use case. Note that our method can be used with any more sophisticated probability computation method [126, 129].

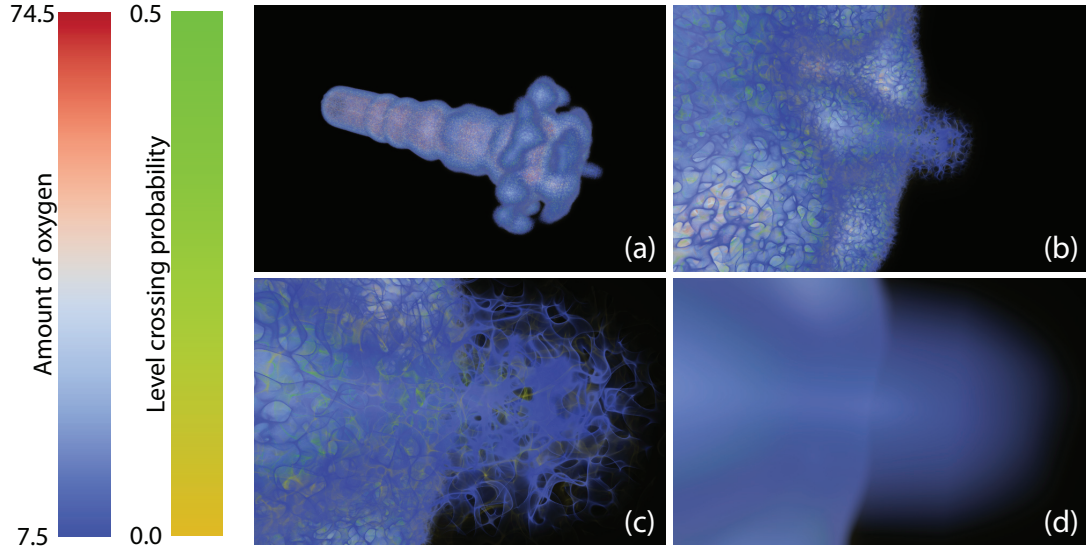


Figure 5.8: Volume rendering of uncertainty data with two attributes: amount of oxygen (blue-red diverging, constant opacity of 0.1) and level-crossing probability for the isosurface with value 55 (yellow-green, linear ramp opacity in range $[0.01, 0.02]$). Results of our method are shown in panels (a), (b), and (c). Panel (d) shows the result for conventional DVR. Note how the low-opacity uncertainty information is completely hidden by conventional DVR (bottom) due to color intermixing. The color scales are shown on the left.

5.4.2 2D applications

A common task in many application fields is to correlate multiple dimensions of a dataset with each other in order to draw conclusions from it. We have applied the proposed method to two 2D application scenarios. In both examples, we encode additional information on top of an existing base visualization. Additionally, we show that our method scales to three dimensions by providing a 3D use case.

5.4.2.1 Video data

The first example is concerned with the evaluation of data acquired during perceptual experiments performed with an eye-tracker [163]. In a user study, the authors evaluated in which way and how much the visual bottom-up saliency [71] of a scene can be altered to steer the attention of a subject to a certain piece of information without noticeable changes for the subject. In order to evaluate the success of their method, the authors needed to compare three videos in a side-by-side fashion, as shown in Fig. 5.9 (top row). The first video is the original source video. The second video visualizes the user's gaze

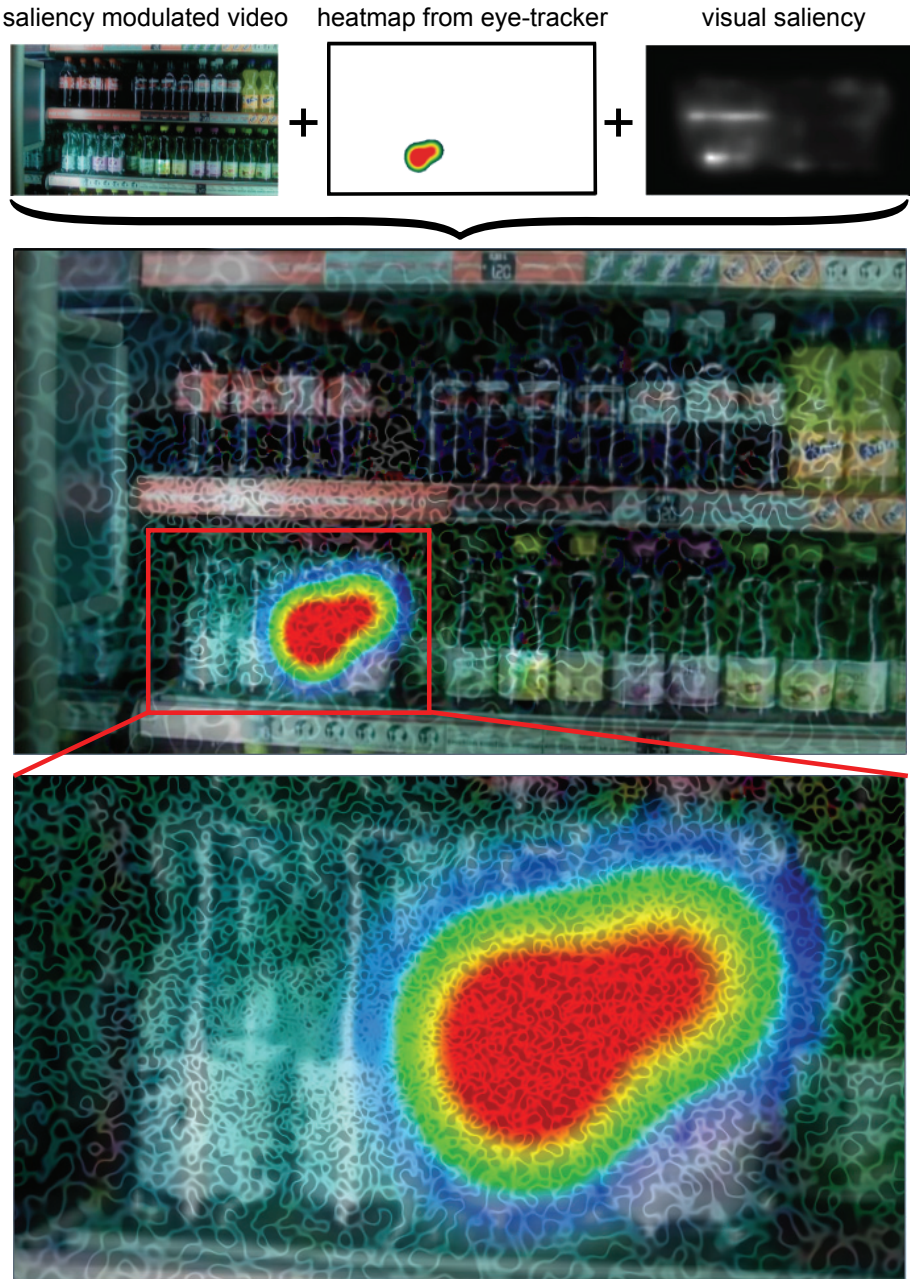


Figure 5.9: This figure shows our approach during the analysis of an eye-tracker experiment with visual saliency modulated videos [163]. The visual saliency is visualized as texture frequency and texture intensity. Highly salient regions are encoded with high texture frequency and intensity. Concurrently, we show the time the user looked at a specific region with a color-coded heatmap.

while watching the first video during the experiment. The time a subject was looking at a certain spot in the shown scene is color-coded. The third video shows the calculated visual saliency for the original source video. In this evaluation scenario, the visual variable *color* is already used by the gaze heatmap and therefore cannot be used for visualizing the modulated saliency on top of the same video. We applied our method to integrate the visual saliency as well as the user's gaze concurrently on top of the video. We set the v-component of the user's gaze color map to 1 so that the texture does not affect the values read from it. The rest of the frame is only context information and therefore v-component changes are not required. Figure 5.9 (middle and bottom rows) shows the combined result for a single sample video frame. By inspecting the combined image, the expert who evaluates the experiment can identify that users tend to look more on saliency-modulated spots – which supports the hypothesis of this work [163].

5.4.2.2 Geospatial data

A second example is the visualization of weather data stemming from various sources. We have selected freely available global data from NOAA (http://www.noaa.gov/) at a resolution of approximately 11 km/pixel (4096 x 2048 pixels). We chose to combine color-coded sea temperature with the amount of precipitation over the sea and the precipitation's optical flow, which encodes the direction of movement of precipitation areas. Precipitation is encoded with texture frequency and texture intensity. The optical flow of the precipitation density is encoded as texture orientation. As we obtained data for several days, we are able to display an animated sequence of the changes in time for these values. Figure 5.10 illustrates different zoom levels and possible findings from a map encoding precipitation and sea temperature from the 14^{th} of September 2011.

5.5 Implementation and Performance

We implemented our method using NVIDIA CUDA [124]. The parallelization of the evaluation procedure is straightforward because there are no dependencies between the pixels. The CUDA kernel, which is executed for every pixel, is outlined in Algorithm 2. The datasets are stored in GPU texture memory. We use the GPU texture units to allow hardware accelerated trilinear interpolation and fast access via the texture cache. The implementation is based on traditional ray-casting to perform DVR. It should be noted that our method can also be implemented for slice-based volume rendering, similarly to

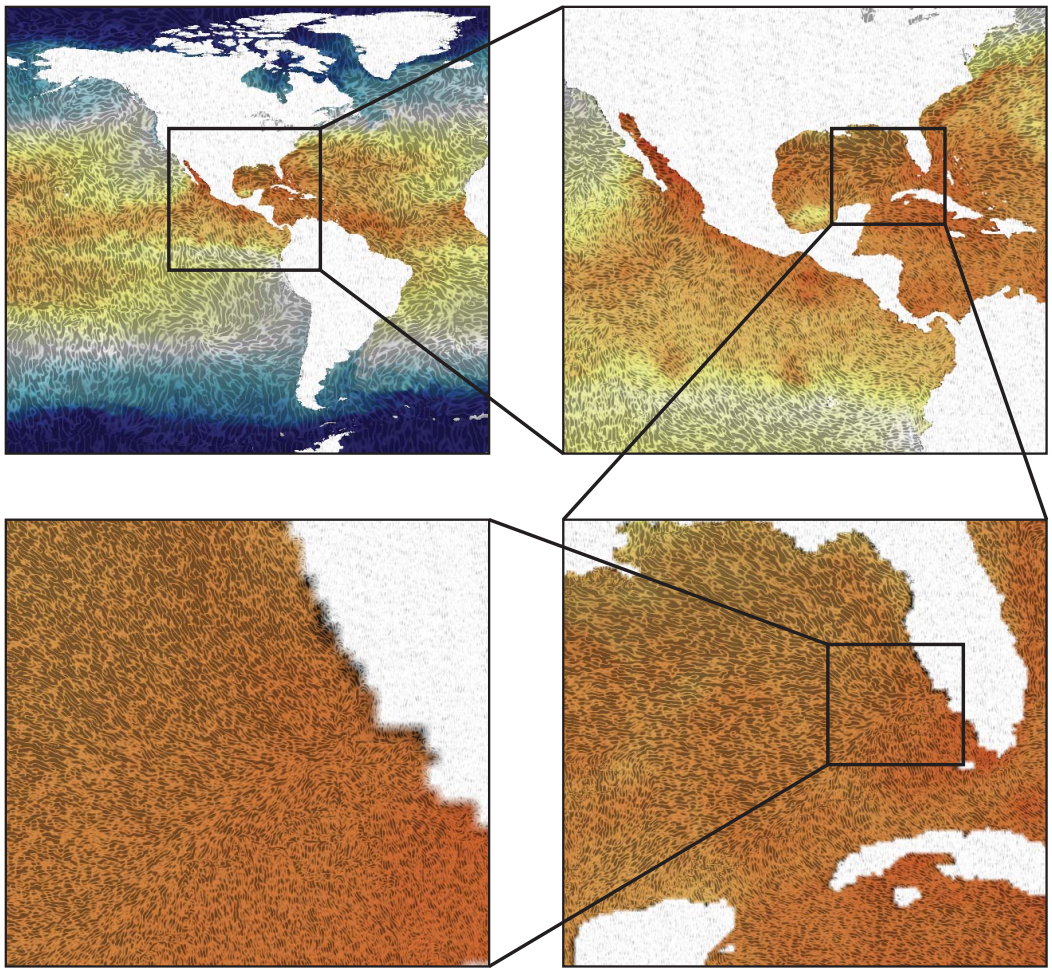


Figure 5.10: The shown world map color codes the average sea temperature from 14^{th} of September 2011. On top, the procedural noise encodes the amount of precipitation (noise frequency and texture intensity) with the optical flow of the precipitation between 14^{th} and 15^{th} of September 2011 (texture orientation). Encoding the optical-flow value between two points in time as second visualization channel, allows the user to estimate the evolution of the precipitation value as well. Note the increasing amount of details with increasing zoom level.

implementing preintegrated transfer functions [49].

The noise values were precomputed and stored in a 256^3 texture with $32\times$ oversampling to ensure high visual quality of the output (i.e., 32 noise voxels per one dataset voxel). The texture wrapping mode was set to *mirror* mode to cover the whole dataset. To obtain high-quality noise, we chose the impulse density λ such that we have 16 impulses per cell of the virtual grid: $\lambda = {}^{16}/_{G^3}$. The noise values were converted to 2-byte fixed-point representation to reduce the memory requirements and to improve texture cache hit rate.

The maximum noise derivative (Eq. (5.15)) is then computed using a Sobel operator.

While it is possible to reduce the size of the precomputed noise patch, it would lead to either decreased visual quality of the noise (in case the number of noise voxels per dataset voxel is decreased) or to a noticeable repetitive pattern on the edges between noise blocks. To avoid these problems and still reduce the memory footprint, the noise function can be computed directly as described by Lagae et al. [101]. However, for Gabor noise, this becomes prohibitively slow, reducing the frame rate up to 10 times even when using only three impulses per cell.

Algorithm 2 Noise-based direct volume rendering CUDA kernel. The noise functions $\mathcal{N}_i$ are separate instances of random-phase Gabor noise with different seeds for random number generator.

```
SetupRay()
pixelColor ← 0
for all sampling points x do
    sampleColor ← 0
    for all datasets do
        value ← SampleDataset(dataset, x)
        color, α ← Classify(value)
        C_α, σ_α ← ComputeParameters(α, x)
        n ← N_i(x)
        α ← IntegrateGaussian(M_α(C_α, σ_α), n, prevN)
        sampleColor ← Blend(sampleColor, color, α)
        prevN ← n
    end for
    pixelColor ← Composite(sampleColor)
end for
```

We tested the overall performance on an Intel Core i7-870, 8GB RAM PC equipped with one nVidia GeForce GTX 680 graphics card. The noise-based volume rendering runs at frame rates between between 5 and 15 frames per second in a 1680×1000 viewport for two datasets with resolution of $500 \times 500 \times 100$, which is 0 to 20% slower than conventional volume rendering.

For the 2D approach, the implementation is based on CUDA and the Visualization Toolkit (VTK) [143]. We use VTK as the framework for basic interaction, data loading and rendering of color-coded data. For every frame, we compute a viewport-size overlay with our procedurally generated texture. The texture itself is generated using CUDA. We compute a transformation matrix from screen space to data-texture space and pass this matrix to the texture-synthesis procedure, so it is able to sample the values of the data

textures. The acquired values, along with the screen-space requirements for noise, are then used to control the noise parameters.

The performance of our method depends mostly on the performance of the procedural noise evaluation. As all the noise parameters are computed on-the-fly, no additional heap or global memory in CUDA is required. On our test system, the synthesis of a 2D texture with a resolution of 1680x988 pixels takes on average 17ms, in case anisotropic noise is used, and 21ms, if isotropic noise is used. Overall, the frame rate of the whole system does not fall below 30 frames per second.

5.6 User studies

To evaluate the effectiveness our methods, we have conducted three controlled experiments. The first experiment was aimed at comparing our 3D approach with other multivariate visualization methods. The second experiment was used to determine the optimal color blending of the 2D noise texture with the background color-coded data. Finally, the third experiment compared the 2D noise-based approach with 2D visualization methods with similar goal using the parameters obtained from the second experiment.

5.6.1 3D methods comparison

We recruited 20 participants (aged 22 to 35, 19 males, 1 female) from a local university with self-reported normal or corrected-to-normal vision. The participants were from the fields of computer graphics and augmented reality. The goal of the study was to test the influence of redistributing the opacity within a voxel on the ability of the users to estimate the values of two variables within this voxel.

We used the data from the Twentieth Century Reanalysis V2 data, provided by the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA, from their Web site [31]. We use daily mean air temperature and specific humidity fields. Longitude, latitude and pressure level are used as dataset dimensions. The resolution of the datasets (after upsampling of the pressure level dimension) is $180 \times 91 \times 61$ voxels. The datasets are mapped to color using two distinct diverging color scales [118]. The mapped temperature ranges are $[-7, 7]°C$ and $[0.0025, 0.0045]$ for air temperature and specific humidity respectively. The opacity for these ranges was set to a constant value of 0.03. This data exhibits various useful patterns, which we used to avoid favoring any particular technique (see Section 5.6.1.1 for more details).

We displayed the data using four techniques for multivariate data analysis, which we compare in our user study (see Figure 5.11):

- **Noise-based** - our algorithm.
- **Switching** - the datasets were visualized separately using conventional volume rendering. The users could switch between visualization of temperature and relative humidity by pressing the space bar.
- **Mixture** - the datasets were visualized with conventional volume rendering and colors obtained for two data values were alpha-blended at each sampling point.
- **Isosurfaces** - the datasets were visualized with combination of color-coded isosurfaces for the first variable (air temperature) and volume rendering for the second one (specific humidity).

5.6.1.1 Task and Procedure

The participants were provided with a computer with a 22" monitor with 1680×1050 resolution at the viewing distance of approximately 60 cm. The study was conducted as a within-subjects experiment with four experimental conditions (technique) and one task per condition. The task was to determine the values of two variables within a voxel-sized 3D box. We selected twelve boxes in different locations, where both variables had non-zero opacity. To avoid favoring any particular method, we have chosen the locations of four types depending on the amount of the empty space surrounding the location. Two locations had empty space on roughly three quarters of the solid angle (i.e., both variables exhibit a peak, Fig. 5.13a), three locations had empty space on a half of the solid angle (i.e., on the side of the dataset, Fig. 5.13b), four locations had empty space on roughly a quarter of the solid angle (Fig. 5.13c), and three had no empty space in the surroundings (i.e., deep inside the data, Fig. 5.13d). The boxes were rendered in wireframe with a distinct color (fully saturated yellow). The users were encouraged to interact with the visualization in a fixed-size 512×512 pixels window by using the mouse to rotate, pan and zoom. To enter the values, the participants used a separate dialog box with two sliders (see Fig. 5.12). Task completion time was measured automatically.

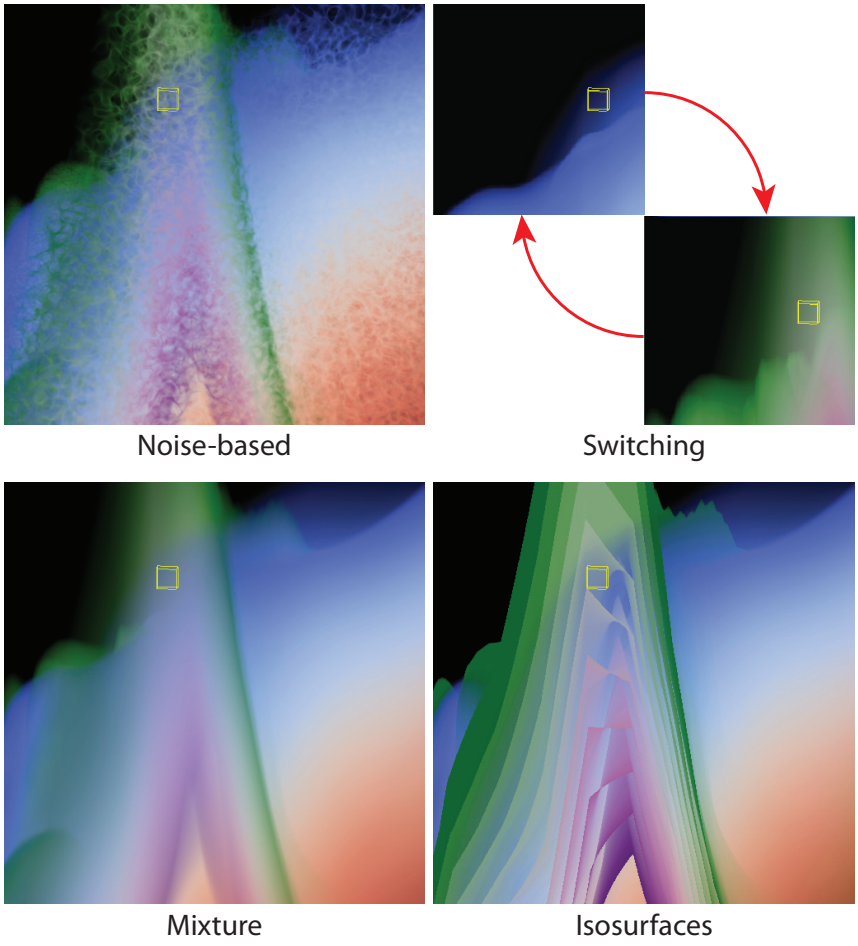


Figure 5.11: Example views for the four visualization techniques compared in our study. For the "switching" method the users were able to switch between the visualization of single variables with a press of space bar. The yellow box denotes one of the locations where the participants were asked to read values from.

Figure 5.12: The user interface which was used by participants to enter the values during the user study. This figure also shows the color maps that were used for temperature and specific humidity.

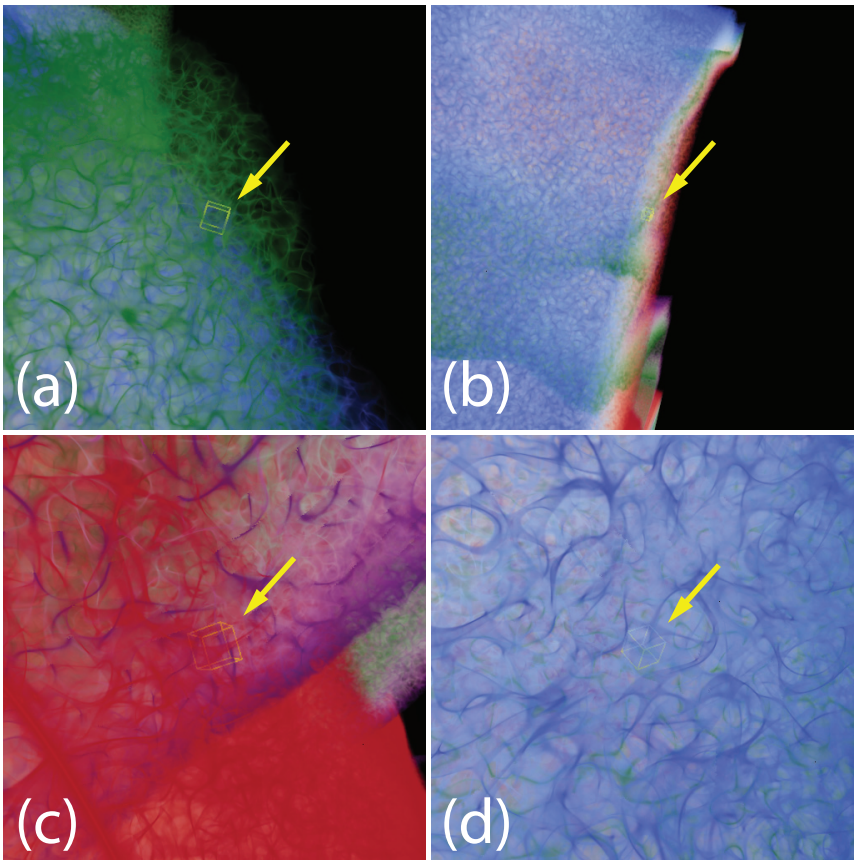


Figure 5.13: Four types of locations used during the user study. (a) peak for both variables, (b) on the side of the dataset, (c) close to the empty space for both variables, (d) deep inside the datasets.

5.6.1.2 Results

The participants started each condition with a warm-up phase, where they familiarized themselves with the visualization method and the task. The answers and the time measured during the warm-up phase were not included into the final statistics. The participants were encouraged to rest between the conditions. To reduce the influence of learning effects, the sequence of the conditions was counter-balanced and the sequence of the locations was randomized.

Upon completion of the experiment, the users filled out a questionnaire, where they were asked to assess their subjective satisfaction by agreeing or disagreeing with the following statements on a five-point bipolar Likert scale:

- **Overview** - It was easy to gain a good overview of the data.

- **Understanding** - It was easy to understand the information displayed with this method.
- **Interpretation** - It was easy to interpret the data values within the boxes.
- **Confidence** - I am confident that my answers were correct when using this method.

Additionally, the participants were asked to rank all four methods by their own impression on how accurate and efficient they were in determining the values and by their overall personal preference.

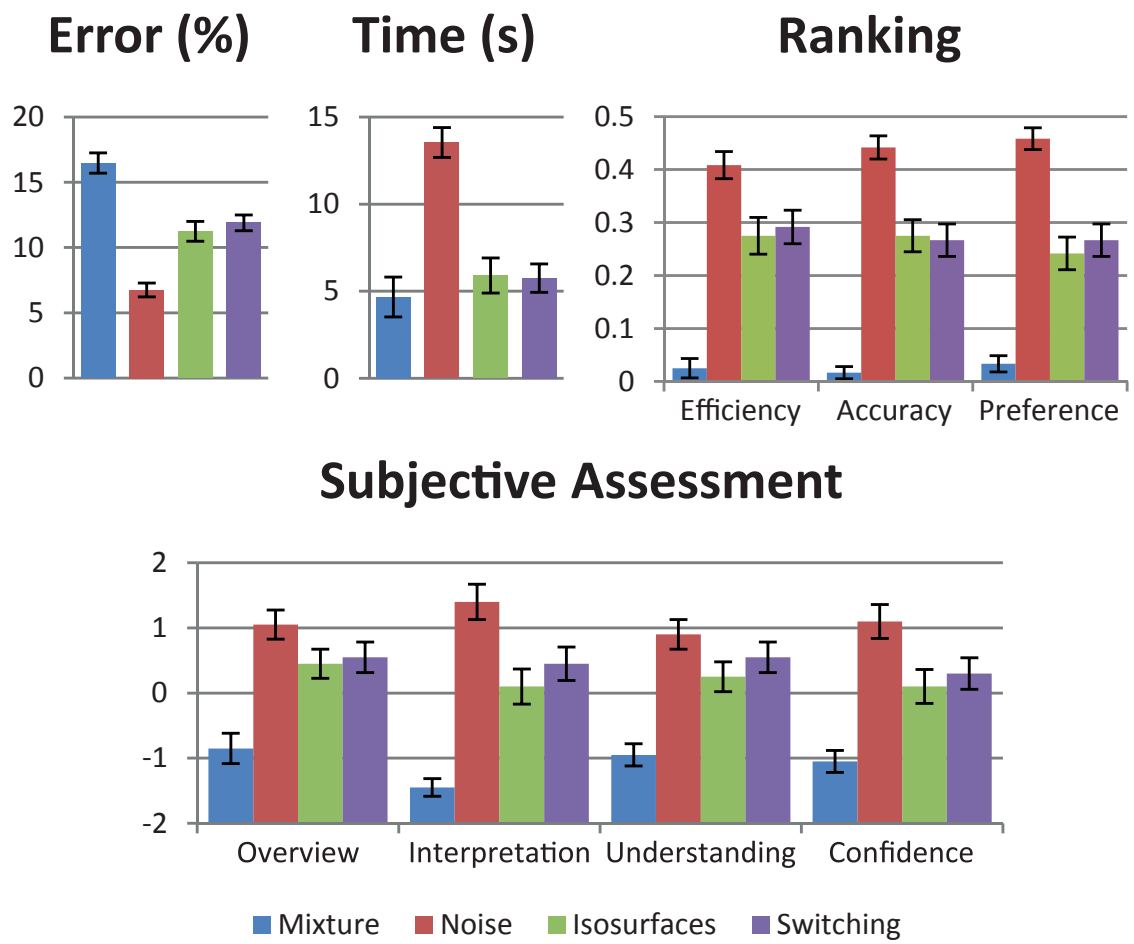


Figure 5.14: The results of our user study. Top, from left to right: average absolute difference to the real value, average task completion time, and overall preference of the users for choosing a certain visualization method. Bottom: a qualitative evaluation of the methods. For error and time graphs – lower value is better, for ranking and subjective assessment – higher value is better.

Hypothesis: Our hypothesis was that the noise-based volume rendering performs better than the other techniques in both quantitative and qualitative assessment.

Error measures, timing and questionnaire answers were analyzed using Friedman non-parametric tests ($\alpha = .05$) for main effects and post-hoc comparisons using Wilcoxon Signed Rank tests with Bonferroni adjustments. A non-parametric test was used in all cases, because a Shapiro-Wilk test rejected normality in all cases. The evaluated measurements are summarized in the following:

- *Error measures* correspond to the absolute difference to the real value.
- *Timing* corresponds to the time from the data set appearing on screen until the participants report the determined values.
- *Subjective assessment* is evaluated by the questionnaire answers on a five-point bipolar Likert scale.
- *Preference* was assessed by assigning points to the techniques according to the order in which they were arranged by the participants.

We found a significant main effect for error ($\chi^2(3) = 38.04$, $p < 0.01$). Post-hoc comparison revealed that the error was lower for **Noise-based** than for any other technique and higher for **Mixture** than for all other techniques. No significant difference was found between the methods **Isosurfaces** and **Switching**.

We also found a significant main effect for timing ($\chi^2(3) = 31.98$, $p < 0.01$). Post-hoc comparison showed that the timing was higher for **Noise-based** than for any other method. No difference was found between **Mixture**, **Isosurfaces** or **Switching**.

There was a significant main effect for the qualitative questions *Overview* ($\chi^2(3) = 26.77$, $p < 0.01$), *Interpretation* ($\chi^2(3) = 38.13$, $p < 0.01$), *Understanding* ($\chi^2(3) = 28.00$, $p < 0.01$), and *Confidence* ($\chi^2(3) = 34.73$, $p < 0.01$). Post-hoc comparison revealed that **Mixture** was rated significantly lower than all other techniques for all questions asked. *Interpretation* and *Confidence* were rated significantly higher for the **Noise-based** method than for all other techniques. Although the mean score of the **Noise-based** method was also higher for *Overview* and *Understanding*, the difference to **Isosurfaces** and **Switching** was not significant. We found no significant difference between **Isosurfaces** and **Switching** for any question asked.

We also found a significant main effect for the rankings accuracy ($\chi^2(3) = 39.78$, $p < 0.01$), efficiency ($\chi^2(3) = 33.72$, $p < 0.01$), and preference ($\chi^2(3) = 39.18$, $p < 0.01$). Post-hoc comparison revealed that **Mixture** was again rated lower than all other techniques

in all three categories. **Noise-based** was rated higher than **Switching** in accuracy and higher than other techniques in preference. The other pair-wise comparisons did not reveal a significant difference.

5.6.1.3 User study analysis

The study shows strong evidence that our method is superior compared to the other methods in terms of data reading accuracy. However, reading the values took participants longer with our method than with any other. This fact can be explained by two factors. Firstly, the participants have never used noise-based volume rendering before, unlike conventional volume rendering or isosurfaces. Therefore, even though there was a warm-up task before each condition, the unfamiliarity with the method might increase the time until the results were entered. Secondly, as the participants reported in the qualitative evaluation, they were much more confident that their answers were correct using the noise-based volume rendering. This confidence shows that the participants were continuously interacting with the visualization method to confirm their answer, while for other methods, they have given a rough answer and then realized that further interaction will not be effective. This was also confirmed during an informal interview with each user study participant, after all tasks were completed.

Additionally, the participants' answers for noise-based volume rendering has a lower error than conventional volume rendering even for univariate data. This conclusion can be made by comparing the error rates that were achieved using noise-based and switching methods. This is due to the fact that the users are able to judge the color in a thin nearly opaque band for noise-based DVR, while even for volume rendering of a single scalar field, the color of each pixel accumulates contribution from many sample points along the ray due to semi-transparency. Therefore, it may be beneficial to substitute simple volume rendering with noise-based visualization in cases where qualitative understanding of data values from the visualization is crucial for the user's task. Consequently, our method can be combined with methods requiring additional interaction such as slicing or picking, in case the user needs the exact data values.

5.6.2 2D optimal color blending

In order to evaluate the influence of the parameter β (Eq. 5.22) on the ability to read data values, we conducted an experiment with 79 participants. In a pilot study, we identified a β range of 0.5 to 0.9 to be suitable. For the main experiment, we tested the β values

0.5, 0.6, 0.7, 0.8, and 0.9, each creating one group. Participants were randomly assigned to these groups.

As test data, we used 20 triplets of data values drawn from uniform distributions with a minimum value of 0 and a maximum of 1. These triplets were encoded using *texture frequency*, a *color gradient* from red to blue, and *texture orientation*. Using the different β values, we created 20 images of 64×64 pixels for each group (see Fig. 5.15). The participants were shown these 20 images in a randomized order. With the help of a legend, they tried to read the encoded variables and reported their results numerically. The entire task took them approximately 10 minutes.

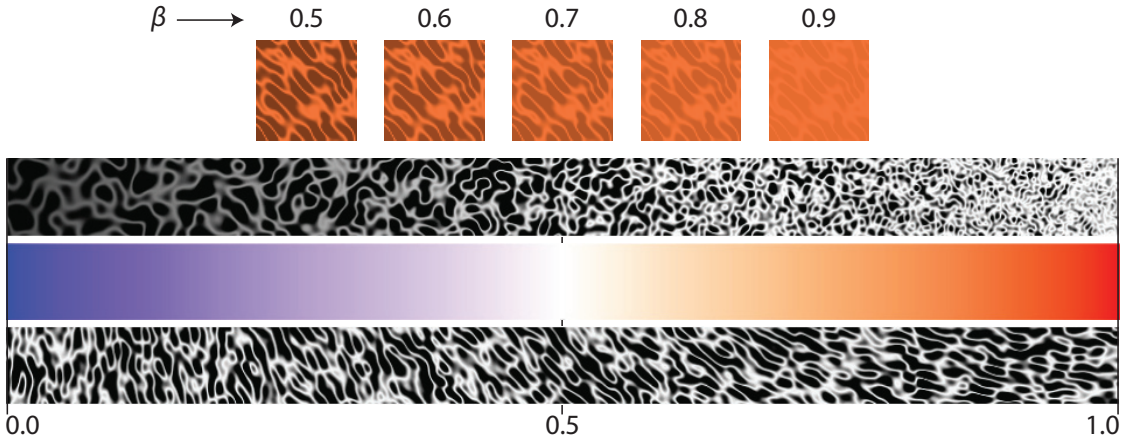


Figure 5.15: 1^{st} row: an example data triplet (0.521, 0.633, 0.312) for the five tested β values. $2^{nd}, 3^{rd}, 4^{th}$ rows: the legend for our method that was used in both our user studies.

We compared the obtained results to the original data values computing the mean squared error (MSE), as depicted in Fig. 5.16. After outlier removal, we computed a one-way ANOVA. We found no statistically significant difference between the groups for frequency ($F_{4,73} = 1.713, p = .156$), or color gradient ($F_{4,71} = 0.696, p = .597$). There was a main effect for orientation ($F_{4,73} = 4.114, p = .005$). A Tukey post-hoc test revealed that the MSE for $\beta = 0.6$ and $\beta = 0.7$ was statistically significantly lower than for $\beta = 0.9$.

The results indicate that a β value of 0.6 or 0.7 works well, as the MSEs are low for all three attributes. For our further experiments, we thus chose $\beta = 0.6$.

5.6.3 2D methods comparison

We recruited 18 participants (aged 22 to 35, 15 males, 3 females) from a local university with self-reported normal or corrected-to-normal vision. The participants were from the

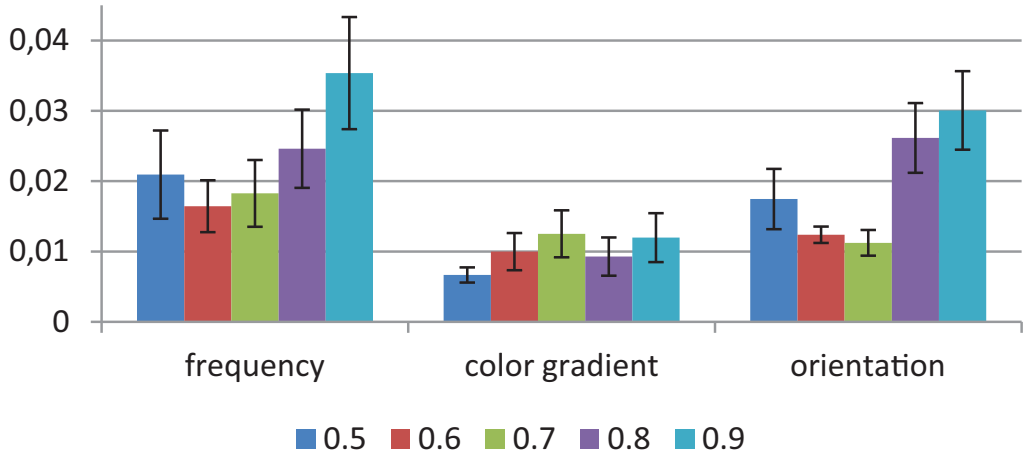


Figure 5.16: Average mean squared error for different β values between 0.5 and 0.9. Choosing β between 0.6 and 0.7 seems to create the best results, yielding an average mean squared error below 2%.

fields of computer science and economics. Fourteen participants indicated that they had experience with visual data analysis.

We compared three techniques for multivariate data analysis in our user study (Fig. 5.17):

- **Noise-based procedural texture (N)** - our algorithm
- **Attribute blocks (A)** algorithm [117]
- **Labeled contours (C)** with one variable mapped to color and others mapped to labeled contour lines with different contrast colors.

5.6.3.1 Task and Procedure

We designed the tasks relevant for an exploration of geographic data according to Hagh-Shenas et al. [61]. The visualization methods displayed the Climate Research Unit (CRU) high resolution climate data, obtained via Intergovernmental Panel on Climate Change (IPCC). We used temperature, precipitation, and amount of water vapor for January, averaged over 30 years (1961 - 1990). In these datasets, the Earth's surface is sampled with 0.5 ° intervals, which results in images with a resolution of 720 × 360 pixels. The exact mapping of variables for each method is shown in Table 5.1. The diverging color scale was used for all methods (attribute blocks required only half of it).

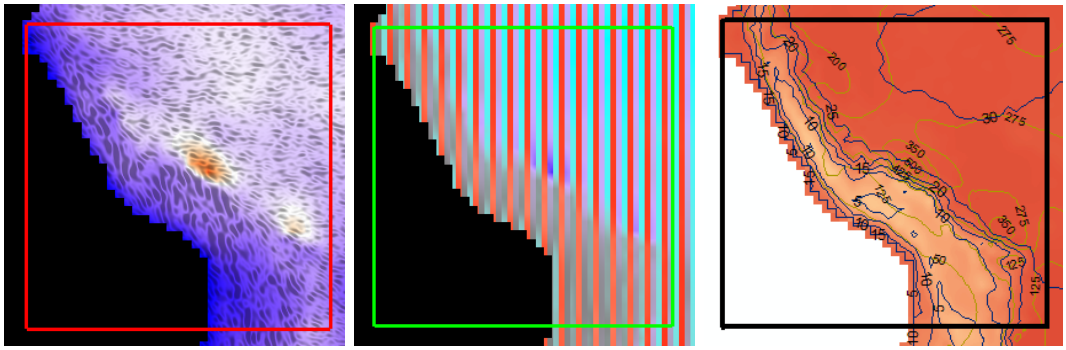


Figure 5.17: Examples of the visualizations compared in our user study (from left to right): our method (noise-based texture), attribute blocks [117], and labelled contours (produced with ArcGIS 10).

	Our method	Attr. blocks	Contours
Temperature	Frequency	Grey-red	Blue-red
Precipitation	Blue-red	Grey-blue	Contour 1
Water vapor	Orientation	Grey-cyan	Contour 2

Table 5.1: Mapping of variables for the techniques used in our user study. For all the methods, the diverging color scale was used (attribute blocks require only half of it).

The users were provided with a computer with two 22" monitors, one showing a maximized window with the data and the other displayed a corresponding legend. The viewing distance was approximately 60 cm. The study was conducted as a within-subjects experiment with three experimental conditions (technique) and three tasks per condition:

- **'global correlate'** - correlate all pairs of variables on the full map: "If 'a' is small, is 'b' then also small? (yes / no / not clear)"
- **'region comparison'** - determine the location of highest/lowest value for each variable
- **'detail comparison'** - correlate all pairs of variables for a certain region of the map: "Is the value of 'a' generally greater or smaller as the value of 'b' in the pointed area? (yes / no / not clear)".

We selected three rectangular regions (North-East of South America, a region from India to Vietnam, and Papua New Guinea) for the execution of the 'detail comparison' task. The users were encouraged to interact with the visualization by using the mouse to pan and zoom.

The participants started each condition with a warm-up phase reading out values on random locations on the map. The inter-task correctness was averaged to yield one correctness value per task and condition per participant. After each condition, the users completed a questionnaire assessing subjective satisfaction. Upon completion of the experiment, they were asked to assess their overall preference. To reduce the influence of learning effects, the sequence of the conditions was counter-balanced and the sequence of the locations in the 'detail comparison' task was randomized.

Hypothesis Our hypothesis was that the noise-based procedural texture performs better than the other techniques in both quantitative and qualitative assessment.

5.6.3.2 Results

Correctness measures were evaluated using repeated measures ANOVA ($\alpha = .05$) with Bonferroni adjusted post-hoc comparisons. Questionnaire answers, which were given on a seven-point Likert scale, were analyzed using Friedman non-parametric tests for main effects and post-hoc comparisons using Wilcoxon Signed Rank tests with Bonferroni adjustments. The questionnaire items concerning preference were analyzed using repeated measures non-parametric Chi Square tests with Bonferroni adjusted post-hoc comparisons. The results are illustrated in Fig. 5.18.

Correctness We found a significant main effect for correctness on 'global correlate' ($F_2 = 29.759$, $p < .001$) and 'region comparison' ($F_2 = 7.108$, $p = .003$). Post-hoc comparison revealed that correctness was higher for N than A and C in both cases. We did not find a significant main effect on 'detail comparison' ($F_{1.366} = 2.771$, $p = .077$).

Subjective Assessment The questionnaire items assessing readability ($\chi^2(2) = 16.687$, $p < .001$), ability to correlate different data values ($\chi^2(2) = 12.091, p = .002$), ability to find a certain point in the data ($\chi^2(2) = 13.059$, $p = .001$), and overall acceptance of the technique ($\chi^2(2) = 20.086, p < .001$) were rated significantly higher for N and A than for C. The ability to gain an overview of the data ($\chi^2(2) = 26.226$, $p < .001$) revealed an order of the techniques with N > A > C. The questionnaire item assessing visual clutter ($\chi^2(2) = 5.848$, $p > .05$) was not significant.

While the preference counts for reading a concrete data value ($\chi^2(2) = 3.000$, $p > .05$) was not significant, preferences for getting an overview ($\chi^2(2) = 16.333$, $p < .001$) were significantly higher for N than for A and C. The preference counts for correlating two data values ($\chi^2(2) = 16.333$, $p < .01$), and correlating more than two data values ($\chi^2(2) = 13.000$, $p = .002$) were significantly higher for N than for C, but the differences between

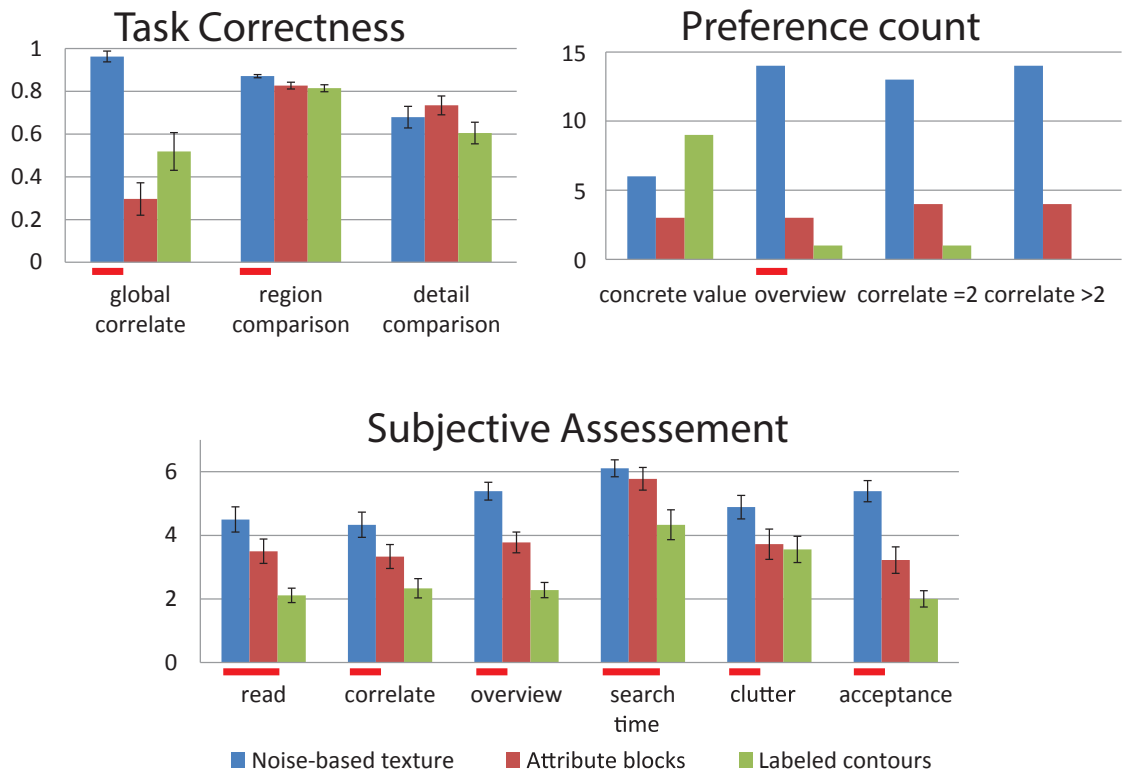


Figure 5.18: The results of our user study. Top, from left to right: results of correctness of completing the quantitative tasks by the users, and overall preference of the users for choosing a certain visualization method. Bottom: a qualitative evaluation of the methods. A higher value is better for all the cases. The red underline shows the methods that were superior in the cases where the results are statistically significant.

N and A and between A and C were not significant.

5.6.3.3 User study analysis

Our hypothesis was supported by the results of the user study. We found that for global correlation of data and for region comparisons the noise-based procedural texture method performs significantly better than the other two methods. In the subjective assessment, our method achieved the highest score in all questioned asked, with four being statistically significant. Furthermore, the personal preference count shows a clear support for our method. Only for reading concrete values from the map labelled contours achieved a higher rating, which was not statistically significant. Reading exact data values is not the goal for any of the tested methods. Other techniques, such as interactive data probing,

should be used to complement them. Please refer to Fig. 5.18 for more details.

5.7 Limitations

5.7.1 3D method limitations

We have conducted an informal interview with a meteorology expert to find potential limitations of our 3D approach for its use in the practice. Overall, the expert was impressed by the basic idea and noted that our method can be very useful for getting an impression of 3D multivariate data. However, he has also expressed a concern about the additional structure introduced by the remapping of the opacity using a noise pattern and about possible information loss by changing the opacity of the pattern. Firstly, our method may be confusing for the user without proper training. Secondly, additional structure can potentially distract the user from analyzing the actual data.

Additional measures could be taken to avoid or at least reduce the negative impact of introducing additional structure to the data. An example of such measures is the use of silhouettes to convey the structure of the data itself (see Fig. 5.19). Such an effect can be achieved by smoothly modulating the noise value towards zero in the vicinity of a set of isosurfaces. These silhouettes provide additional cues about the structure of the data and do not introduce aliasing, because their opacity is filtered along with the opacity of the noise.

Another interesting approach to the issue of additional structure is the use of lighting-related effects, such as specular highlights and shadows that work directly on the original opacity and not on the values gathered from the noise opacity mapping function. Overall, this limitation needs to be considered when applying it to a particular visualization task or should be mitigated with additional algorithms and proper training. However, development of such algorithms and evaluation of their efficiency are out of scope of this paper. We will investigate them in future work.

5.7.2 2D method limitations

While our method provides a way to display both an overview and details at different zoom levels, it is prone to some data hiding similar to other methods that employ texture-based visualization. This problem is not given much attention in the related work. Urness et al. propose to upsample the data in order to not hide any information in the original data [156]. Shenas and Interrante propose to choose a texture such that its frequency

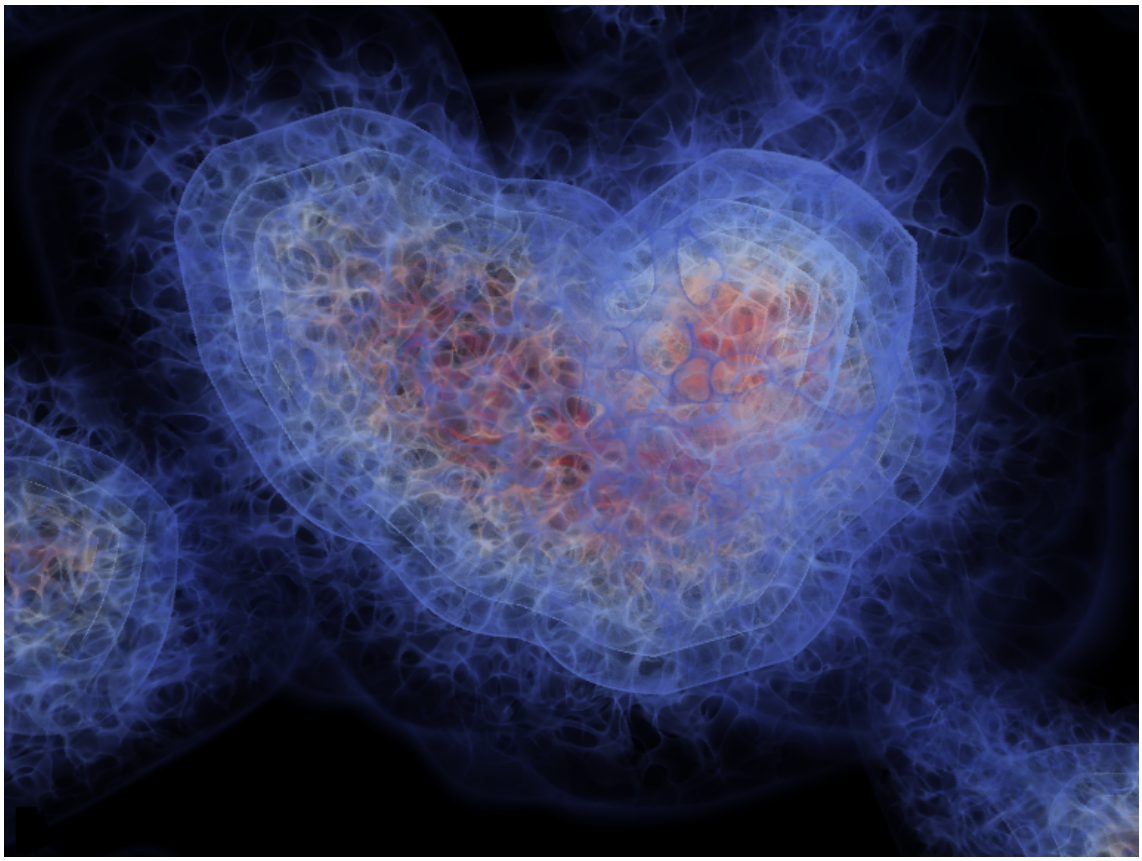


Figure 5.19: An example of using silhouettes to provide additional information about the structure of the data that might be suppressed by introducing the noise pattern.

exceeds the data frequency [148]. Taylor also points out the same problem and relies on user interaction with the visualization to see the details [154]. Our method is also limited in this respect, i.e., our method cannot show data variations whose frequency exceeds that of the texture. However, as the texture is distinguishable at any scale, we can at least show the general trends within the dataset and rely on the user to zoom in the areas of high interest. For our method, mapping the most important and/or most varying data attribute to the underlying color simplifies the task of finding regions of interest, because the perceivable resolution is higher for the color-mapped attribute than for the others.

5.8 Discussion

We have presented and evaluated a visualization method which is well suited for qualitative analysis of the data and which can be used to understand the dependencies between the

data values directly from 3D visualizations. Our method shows significantly lower error for reading data values and higher subjective ranking than common multivariate visualization techniques. This implies that with attention to the limitations of our method, it have a high chance of being accepted for solving the tasks involving the analysis of multivariate 3D data.

Furthermore, we have shown how our method can be extended to visualization of 2D multivariate data. The main advantage of our method in this case is that it gives a possibility to adjust the texture in a zoom-independent manner to avoid aliasing at low scales and keep the data readable at large scales. With our user study, we have shown that our method outperforms other methods that are commonly used for similar tasks.

5.9 Possible extensions

There are several directions for extending and improving our method. Firstly, it is important to test how well our 3D visualization method scales with the number of variables that are visualized simultaneously. Even though the extension of our 3D method for more than two variables is straightforward, the perception of the concurrent visualization of three or more 3D data fields may prove to be too difficult for the users. Secondly, it is necessary to analyze how our method can be combined with other multivariate visualization approaches, such as 2D transfer functions. For example, a 2D transfer function can be used to specify the color and opacity mapping for two variables, which can be visualized with one noise pattern. It can then be combined with another noise pattern for the third variable or another pair of variables using one more 2D transfer function. And, thirdly, some of the approaches shown to work well for 2D noise-based textures may be used in 3D as well. This includes, for example, non-uniform and anisotropic noise. One option is to adapt the frequency of the noise according to data features in order to avoid over-filtering in areas where the data is relatively homogeneous. Another option is to use the direction of the noise pattern to display additional variables or vector data. However, we expect that additional measures will have to be taken for the 3D texture pattern to be well perceived by the users. Usage of time-varying noise may also be of interest.

In addition, it is important to test the applicability of noise-based volume rendering to other application areas. One such possibility is presented by the medical tumor-ablation simulation data, where it is necessary to communicate a predicted cell-death area together with the uncertainty of the simulation, because this may help a doctor to decide on a specific ablation protocol, so that all tumor cells including all *possibly* tumorous areas are

killed without harming the surrounding healthy tissue too much. Furthermore, our method may be of use for the emerging field of hyper-spectral imaging. This imaging method allows to distinguish different chemical materials on-the-fly after a thorough manual adjustment. For this application area, our method can be used to visualize the probabilities for different materials and therefore ease the manual adjustment process.

Chapter 6

Conclusions

Contents

6.1 Summary

In this thesis, we sought to derive efficient data visualization metaphors from natural phenomena occurring in the real world. Even though we did not cover the vast spectrum of all the natural phenomena, we have presented three approaches that help the users understand the data at hand. These approaches, however, represent three distinct levels of the depth abstraction of the visualization metaphor from the actual natural phenomenon.

The first approach shows almost direct transfer of the real world interaction of light with scattering media to direct volume rendering, with the goal of increasing the realism and subsequently improving the perception of the rendered data.

The second approach employs crepuscular rays in a way similar to how they are actually formed in nature. However, the context within which these crepuscular rays appear in our applications would not be feasible in nature. Furthermore, even though conceptually and visually similar to the natural ones, our crepuscular rays in fact do not share the attenuation behavior with natural light, but carry additional information through the use of various accumulation strategies.

And finally, our third approach, albeit being inspired by the visual appearance of smoke, does not share physical properties or logical rationale with real smoke. It is only used to provide the concept of space being separated into transparent and relatively opaque

partitions, which proved to be useful for both 2D and 3D multivariate data visualization.

Despite their differences in levels of abstraction, we showed that all three metaphors are useful in the context of data visualization. This gives reason to believe that, even though our understanding of the processes occurring in the human visual system is still incomplete and thus we cannot design perceptually optimal, in the full extent of this word, visualizations, we can imitate the visual appearance of the natural phenomena to create efficient and accessible methods for visualizing data.

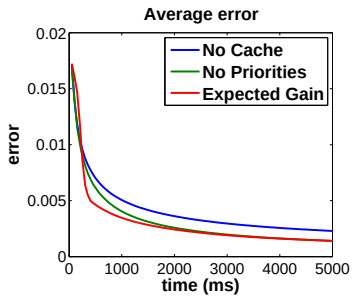
6.2 Directions for future work

In the future, we are excited to see more natural phenomena-based metaphors for effective visualization of different types of data. For example, many natural phenomena inherently evolve with time. Be it fast changes in appearance of fire or relatively slow cloud formation, they might be useful for visualization of time-dependent data. Alternatively, the methods that we proposed in this thesis may be altered to be used in different scientific disciplines and extended for more complex cases within the application areas we applied them to. And, even though we outlined some of the potential directions for the development at the end of respective chapters, we are also excited to see other researchers to create efficient, effective and aesthetically pleasing visualization approaches employing the exceptional variety of tools given to us by the nature itself.

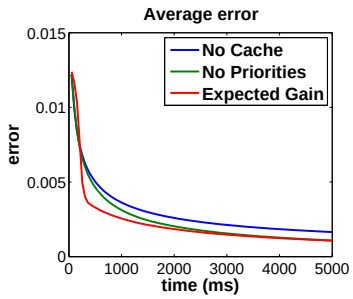
Appendix A

Convergence measurements

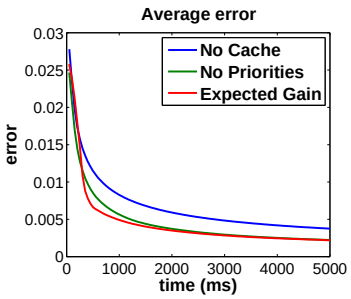
The following figures describe our extensive experiments and recorded quantities described in Section 3.7.



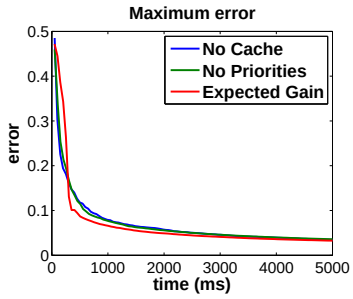
(a) Bonsai Average Error



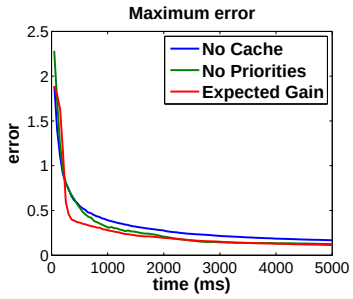
(b) Manix Average Error



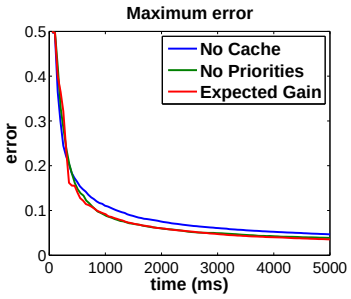
(c) Macoessix Average Error



(d) Bonsai Maximum Error

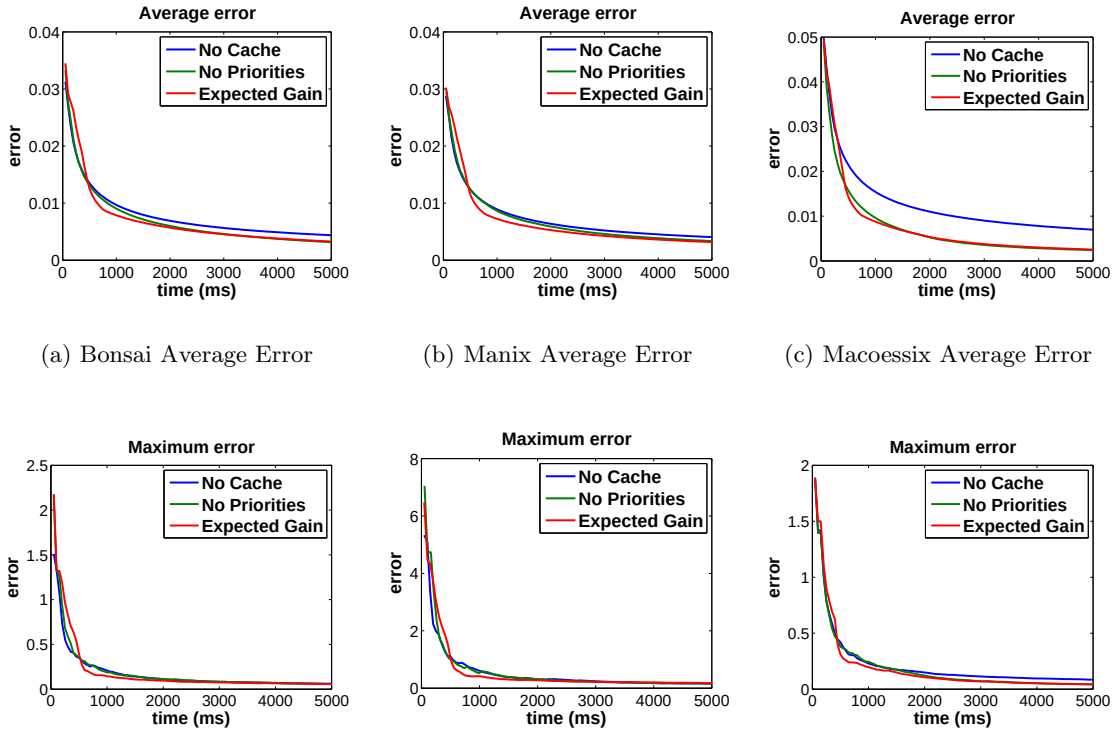


(e) Manix Maximum Error



(f) Macoessix Maximum Error

Figure A.1: Testrun 1: Static Scene, Convergence for full scale volumes at 1280x720, light situation 1.



(a) Bonsai Average Error

(b) Manix Average Error

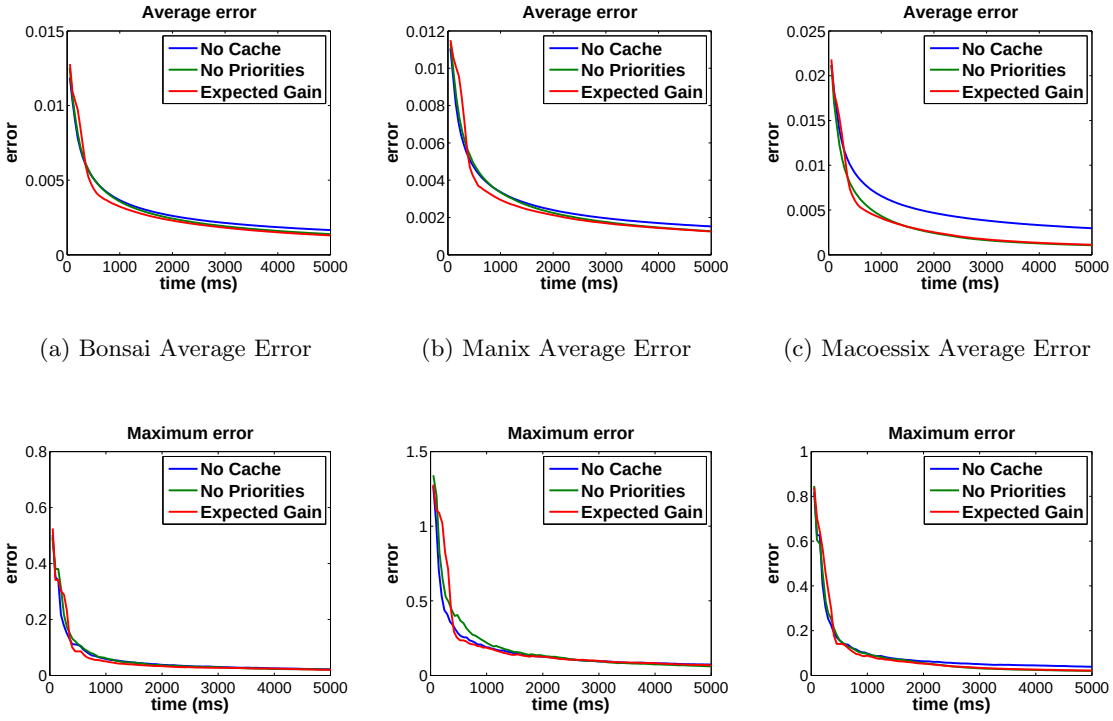
(c) Macoessix Average Error

(d) Bonsai Maximum Error

(e) Manix Maximum Error

(f) Macoessix Maximum Error

Figure A.2: Testrun 1: Static Scene, Convergence for full scale volumes at 1280x720, light situation 2.



(a) Bonsai Average Error

(b) Manix Average Error

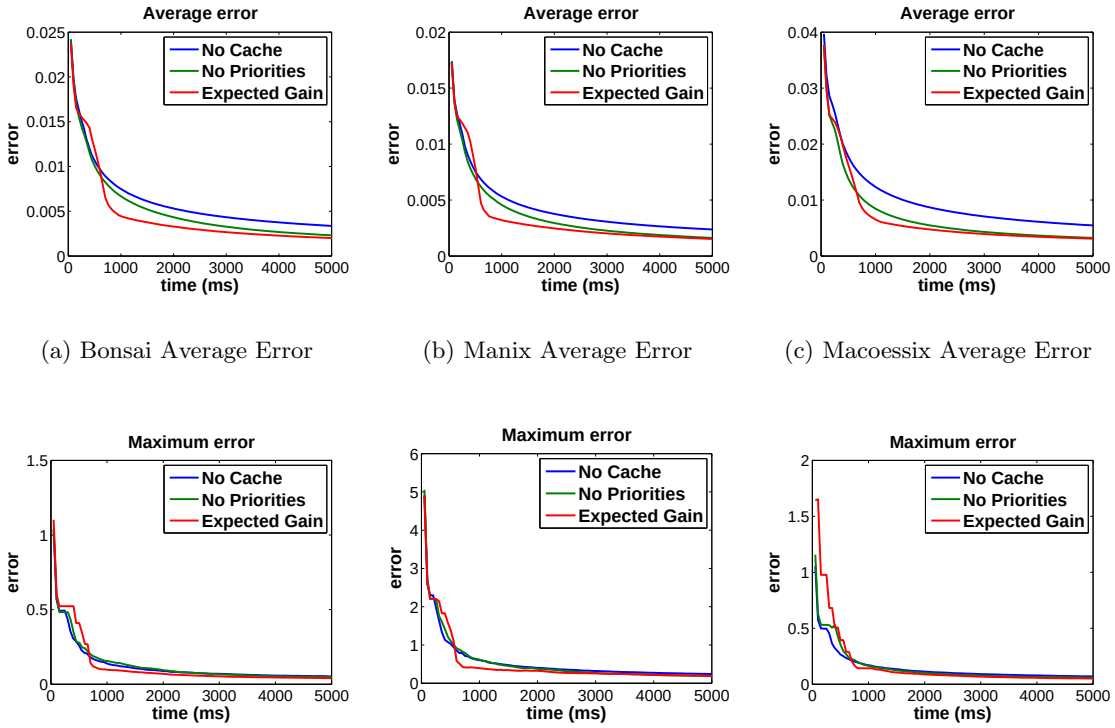
(c) Macoessix Average Error

(d) Bonsai Maximum Error

(e) Manix Maximum Error

(f) Macoessix Maximum Error

Figure A.3: Testrun 1: Static Scene, Convergence for full scale volumes at 1280x720, light situation 3.



(a) Bonsai Average Error

(b) Manix Average Error

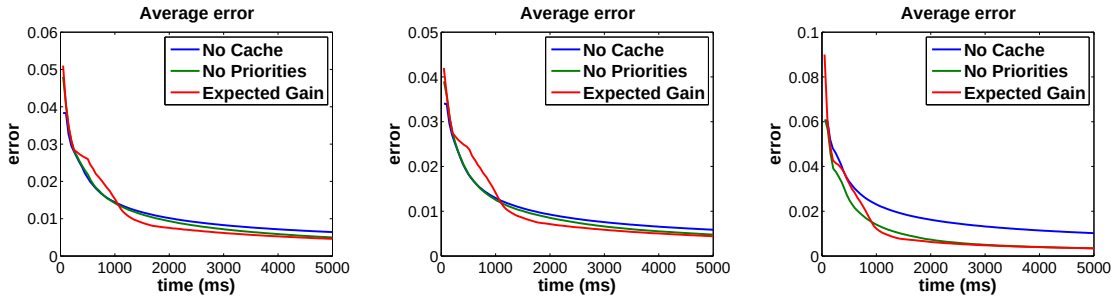
(c) Macoessix Average Error

(d) Bonsai Maximum Error

(e) Manix Maximum Error

(f) Macoessix Maximum Error

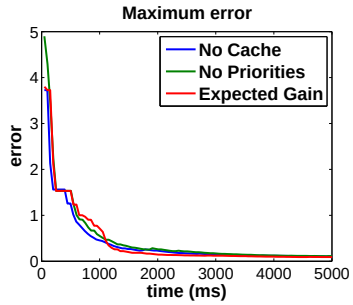
Figure A.4: Testrun 2: Static Scene, Convergence for full scale volumes at 1920x1080, light situation 1.



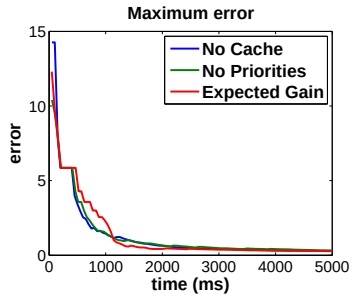
(a) Bonsai Average Error

(b) Manix Average Error

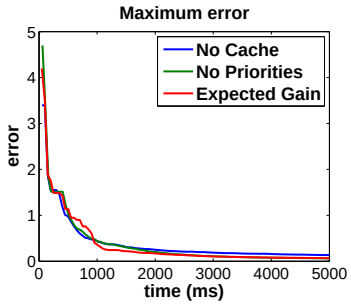
(c) Macoessix Average Error



(d) Bonsai Maximum Error

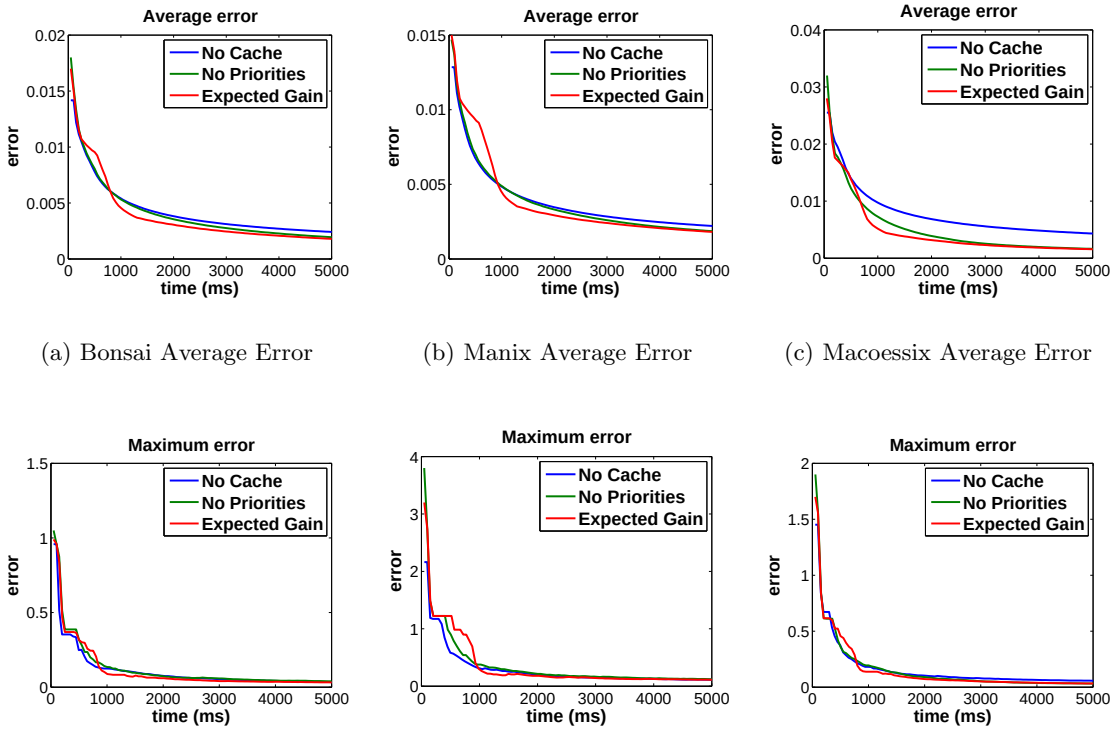


(e) Manix Maximum Error



(f) Macoessix Maximum Error

Figure A.5: Testrun 2: Static Scene, Convergence for full scale volumes at 1920x1080, light situation 2.



(a) Bonsai Average Error

(b) Manix Average Error

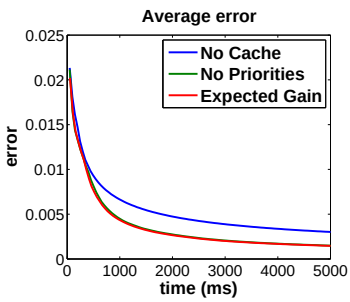
(c) Macoessix Average Error

(d) Bonsai Maximum Error

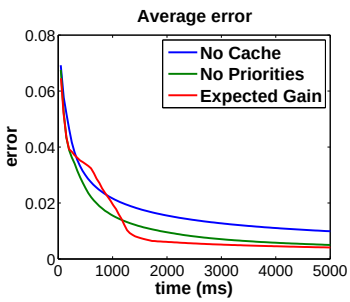
(e) Manix Maximum Error

(f) Macoessix Maximum Error

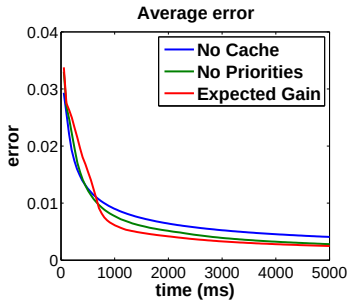
Figure A.6: Testrun 2: Static Scene, Convergence for full scale volumes at 1920x1080, light situation 3.



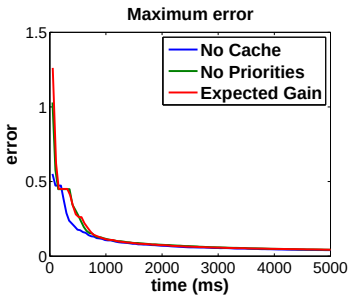
(a) Bonsai Average Error



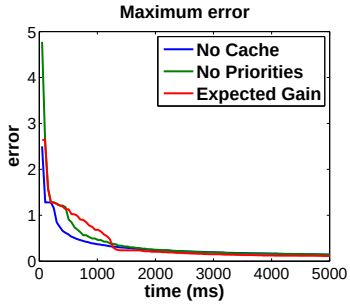
(b) Manix Average Error



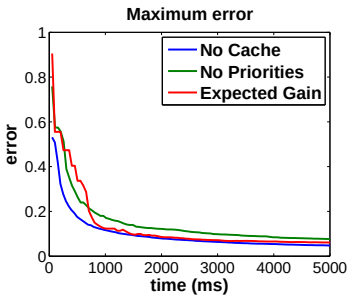
(c) Macoessix Average Error



(d) Bonsai Maximum Error

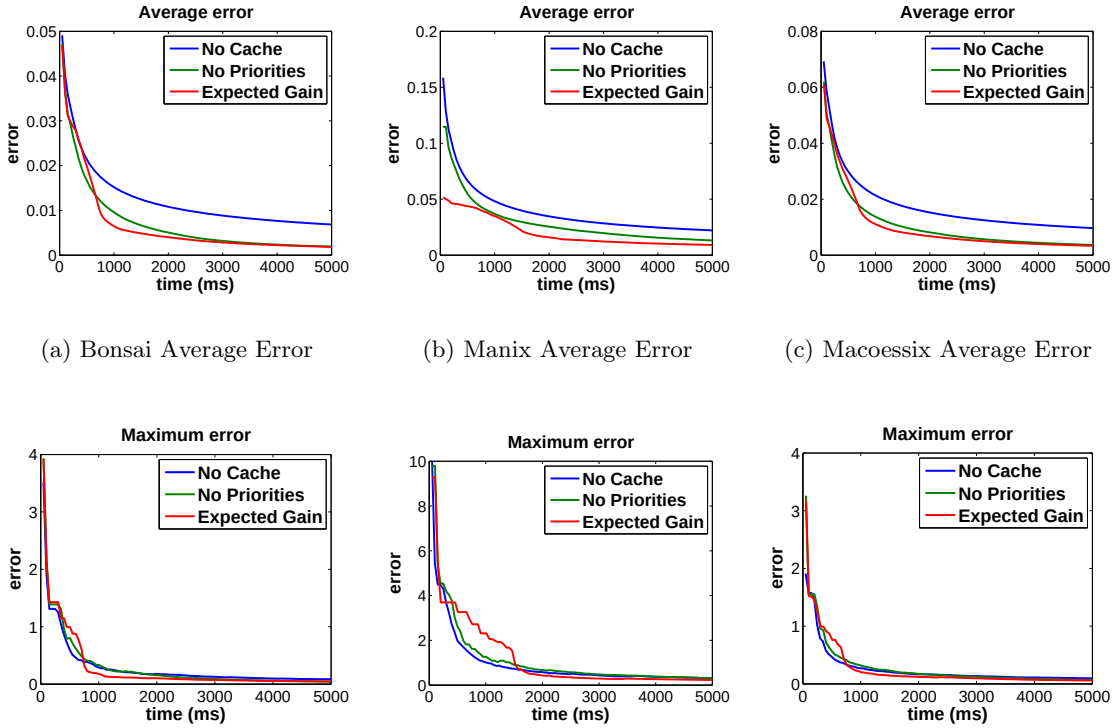


(e) Manix Maximum Error



(f) Macoessix Maximum Error

Figure A.7: Testrun 3: Static Scene, Convergence for half scale volumes at 1920x1080, light situation 1.



(a) Bonsai Average Error

(b) Manix Average Error

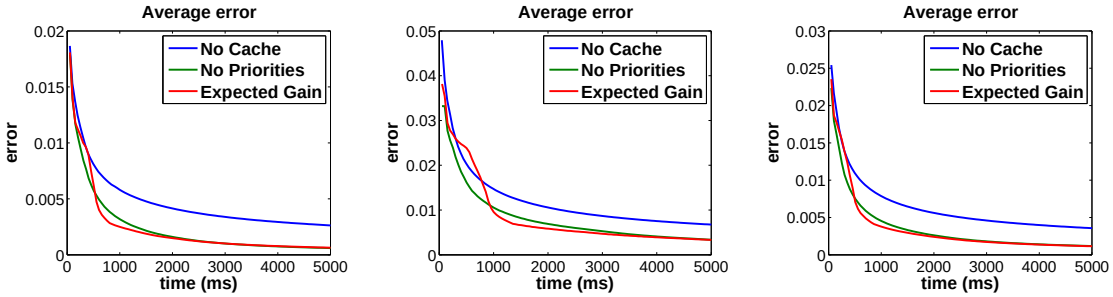
(c) Macoessix Average Error

(d) Bonsai Maximum Error

(e) Manix Maximum Error

(f) Macoessix Maximum Error

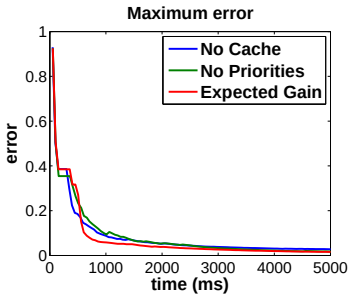
Figure A.8: Testrun 3: Static Scene, Convergence for half scale volumes at 1920x1080, light situation 2.



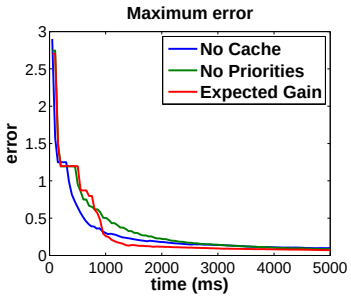
(a) Bonsai Average Error

(b) Manix Average Error

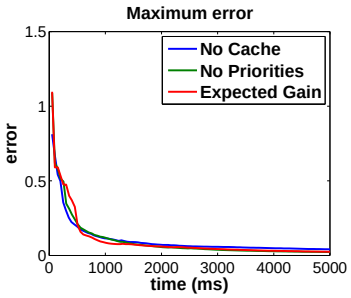
(c) Macoessix Average Error



(d) Bonsai Maximum Error

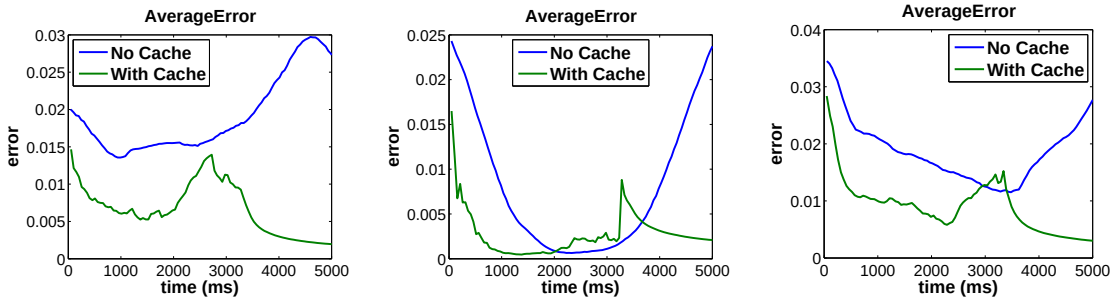


(e) Manix Maximum Error



(f) Macoessix Maximum Error

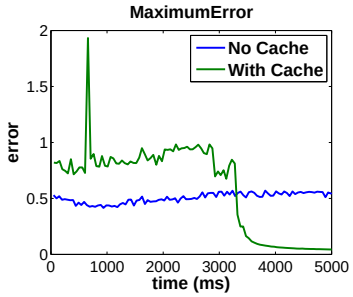
Figure A.9: Testrun 3: Static Scene, Convergence for half scale volumes at 1920x1080, light situation 3.



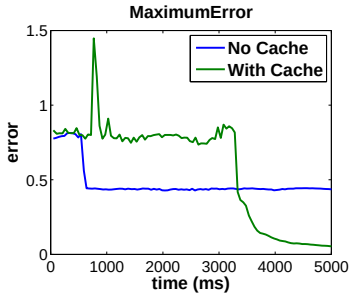
(a) Bonsai Average Error

(b) Manix Average Error

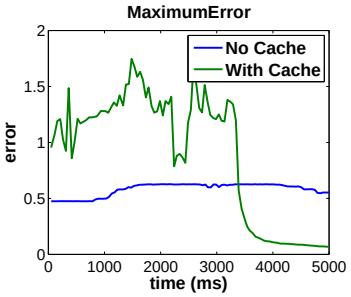
(c) Macoessix Average Error



(d) Bonsai Maximum Error

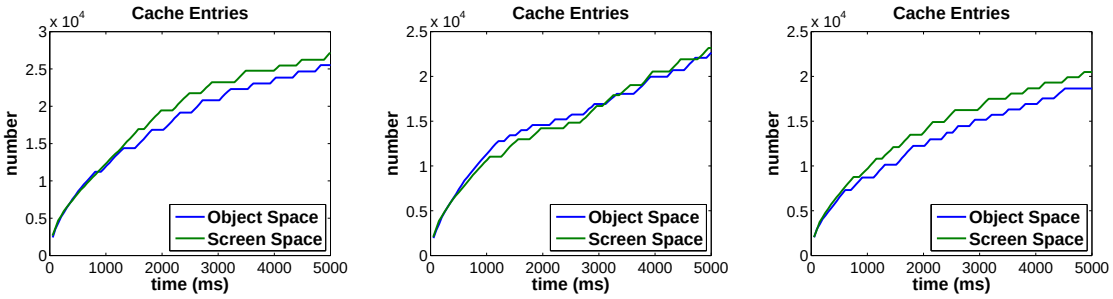


(e) Manix Maximum Error

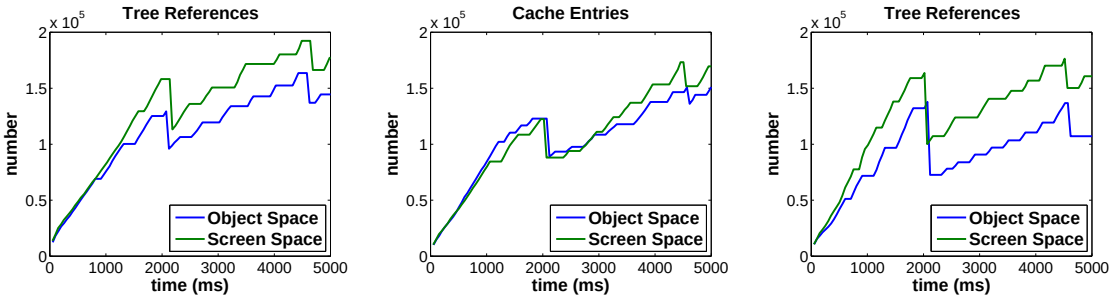


(f) Macoessix Maximum Error

Figure A.10: Testrun 4: Rotating Scene, Convergence for half scale volumes at 1920x1080, light situation 1.



(a) Bonsai Cache Entries (b) Manix Cache Entries (c) Macoessix Cache Entries



(d) Bonsai Tree References (e) Manix Tree References (f) Macoessix Tree References

Figure A.11: Testrun 5: Rotating Scene, Behaviour of cache entry count & reference count.

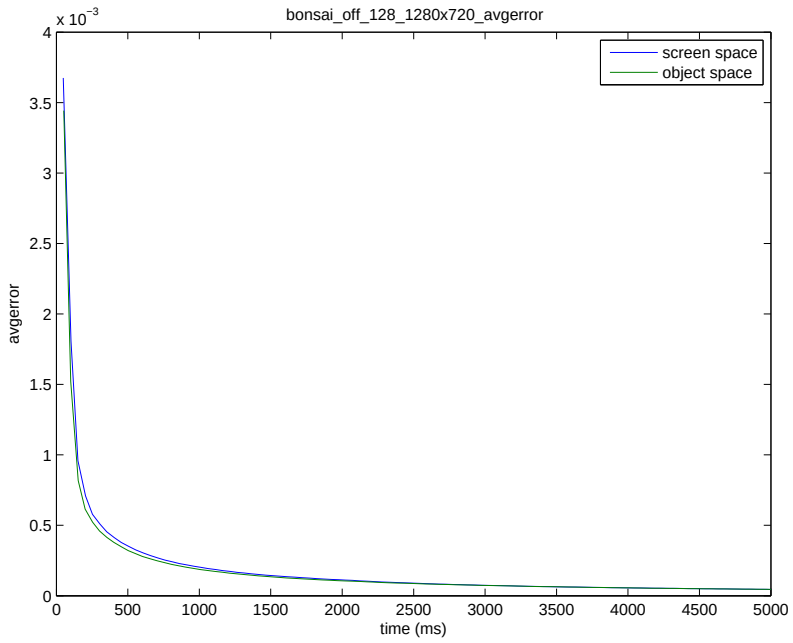


Figure A.12: screen space vs object space: average error bonsai, background off, volume res = 128, screen res = 1280x720, average error

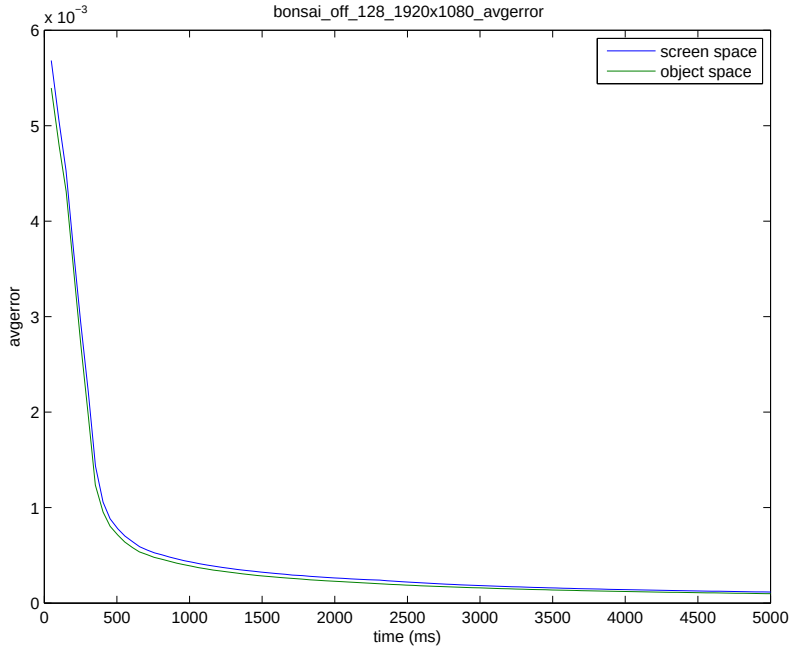


Figure A.13: screen space vs object space: average error bonsai, background off, volume res = 128, screen res = 1920x1080, average error

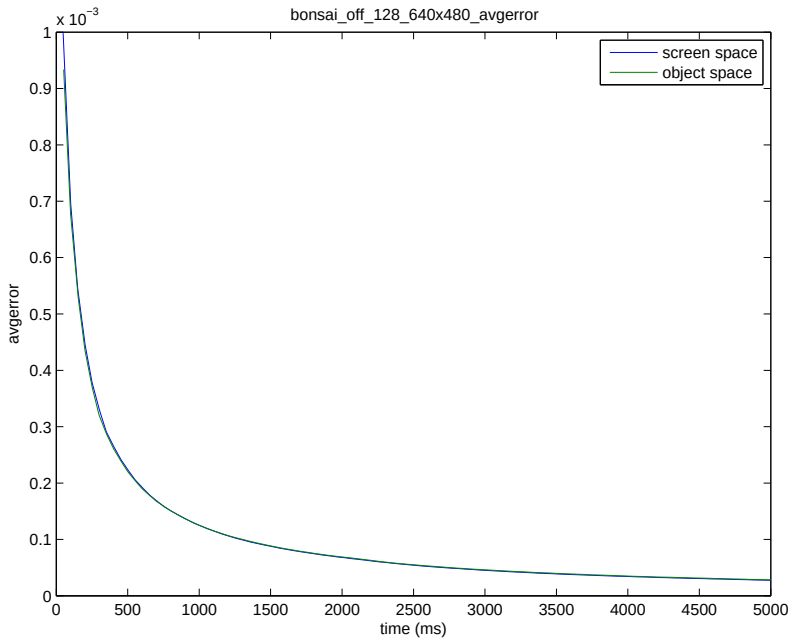


Figure A.14: screen space vs object space: average error bonsai, background off, volume res = 128, screen res = 640x480, average error

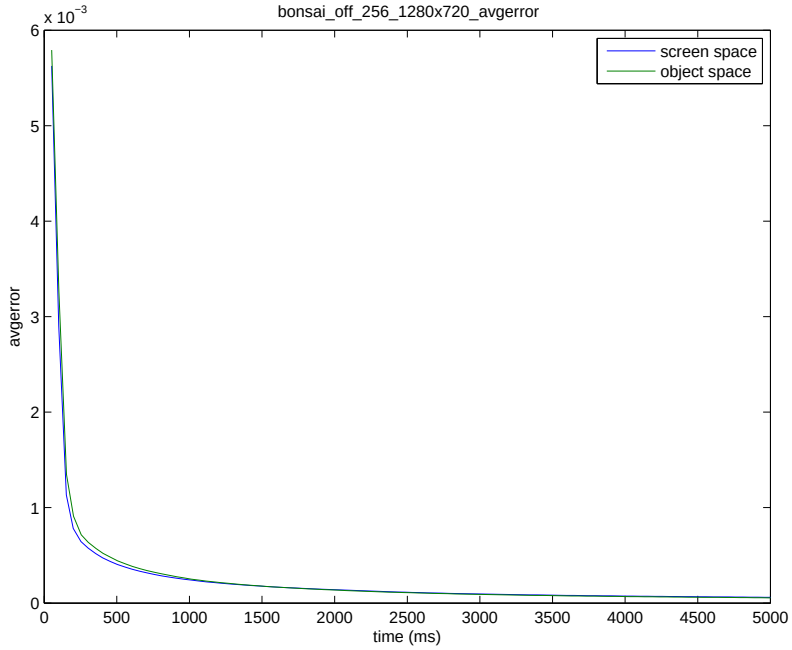


Figure A.15: screen space vs object space: average error bonsai, background off, volume res = 256, screen res = 1280x720, average error

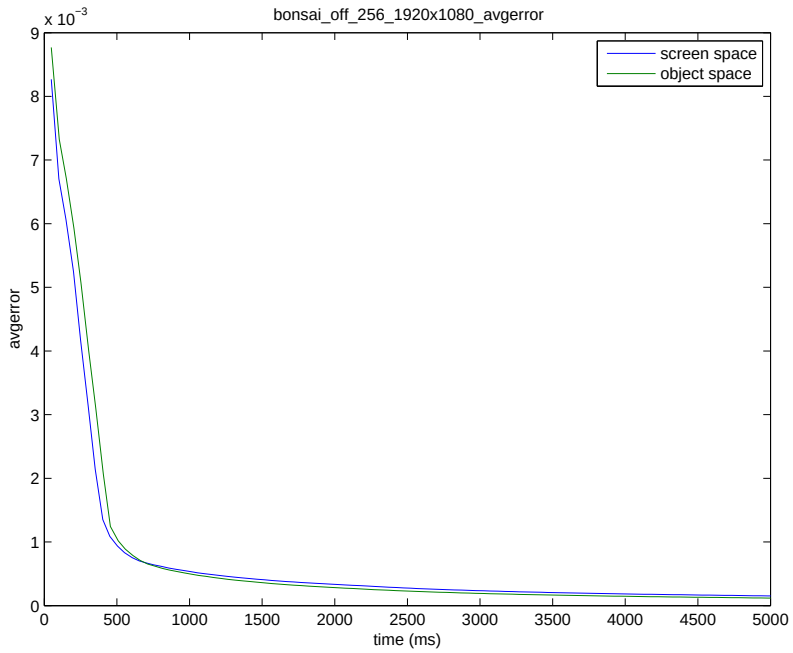


Figure A.16: screen space vs object space: average error bonsai, background off, volume res = 256, screen res = 1920x1080, average error

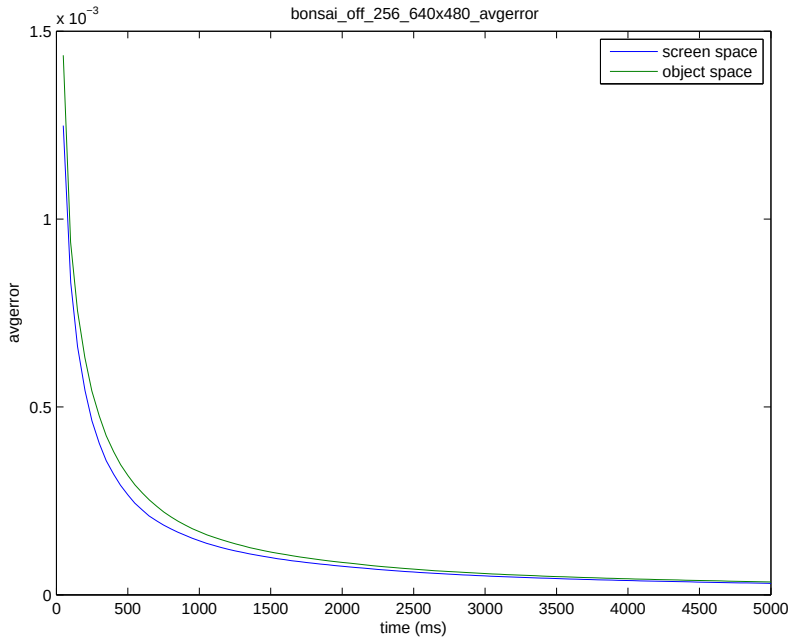


Figure A.17: screen space vs object space: average error bonsai, background off, volume res = 256, screen res = 640x480, average error

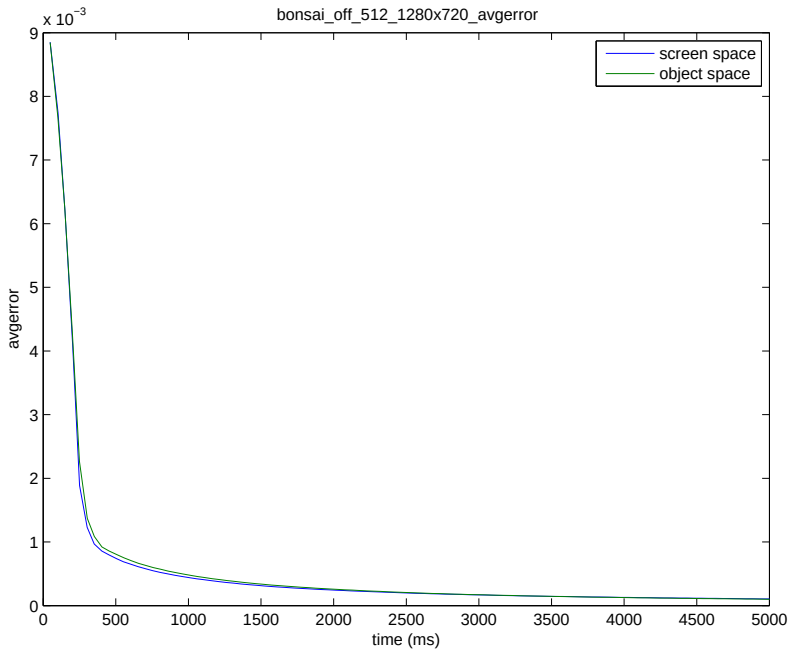


Figure A.18: screen space vs object space: average error bonsai, background off, volume res = 512, screen res = 1280x720, average error

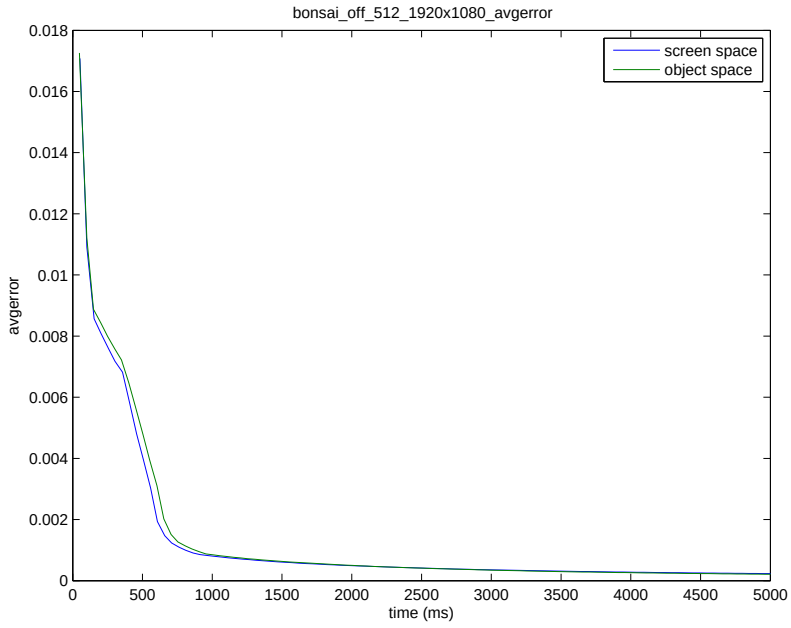


Figure A.19: screen space vs object space: average error bonsai, background off, volume res = 512, screen res = 1920x1080, average error

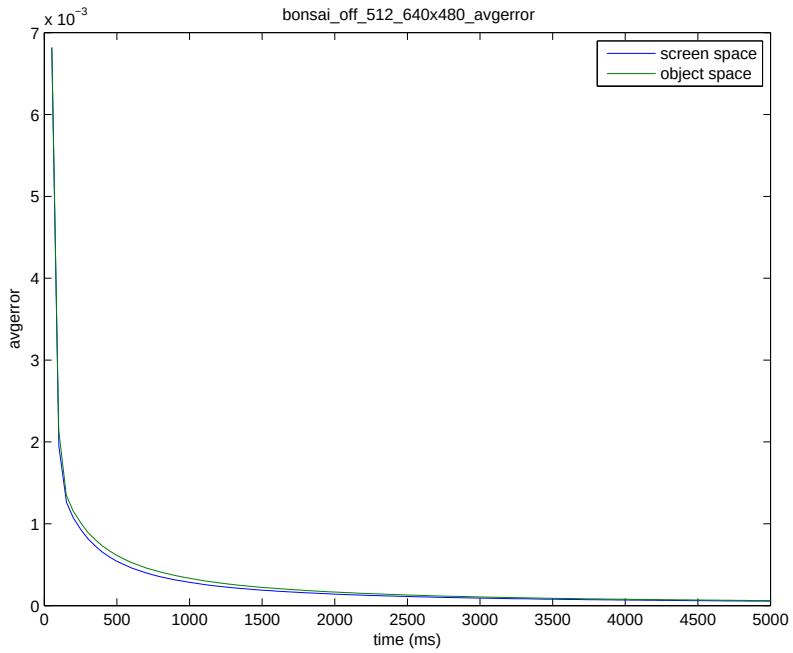


Figure A.20: screen space vs object space: average error bonsai, background off, volume res = 512, screen res = 640x480, average error

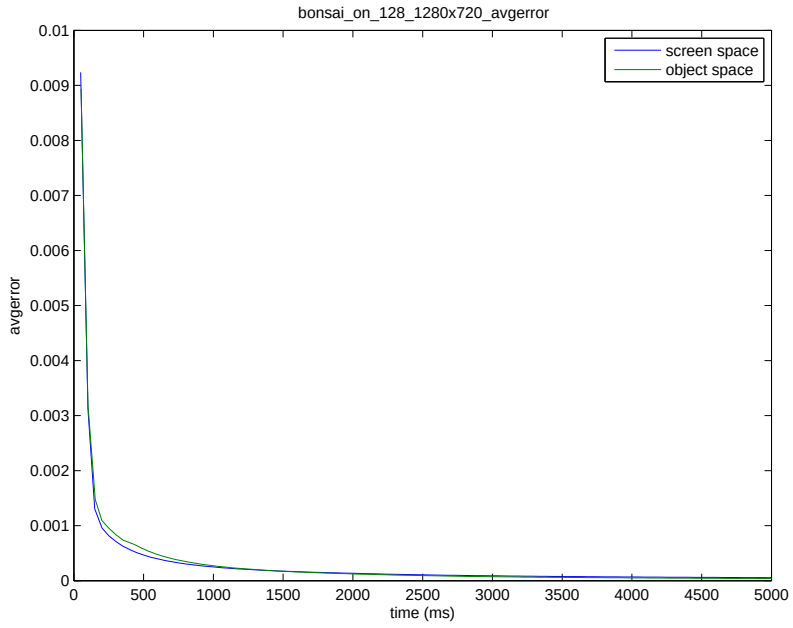


Figure A.21: screen space vs object space: average error bonsai, background on, volume res = 128, screen res = 1280x720, average error

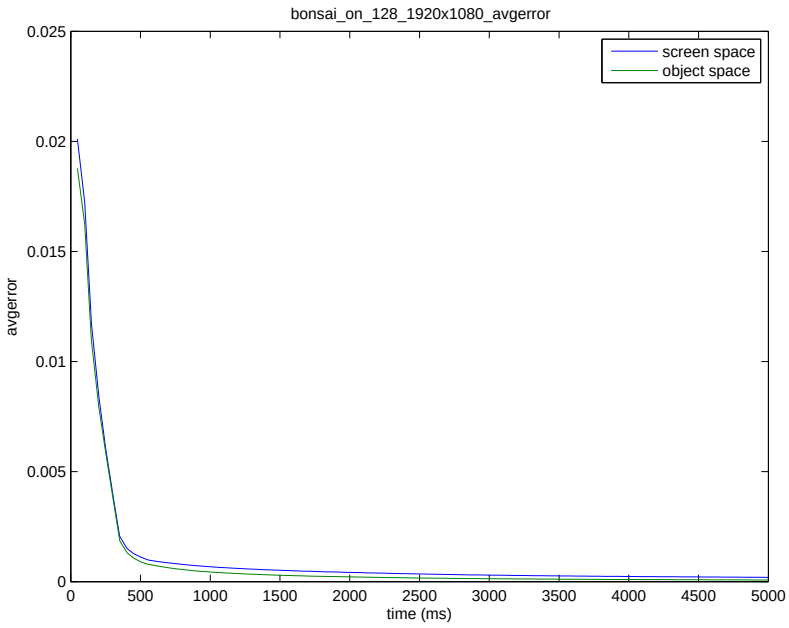


Figure A.22: screen space vs object space: average error bonsai, background on, volume res = 128, screen res = 1920x1080, average error

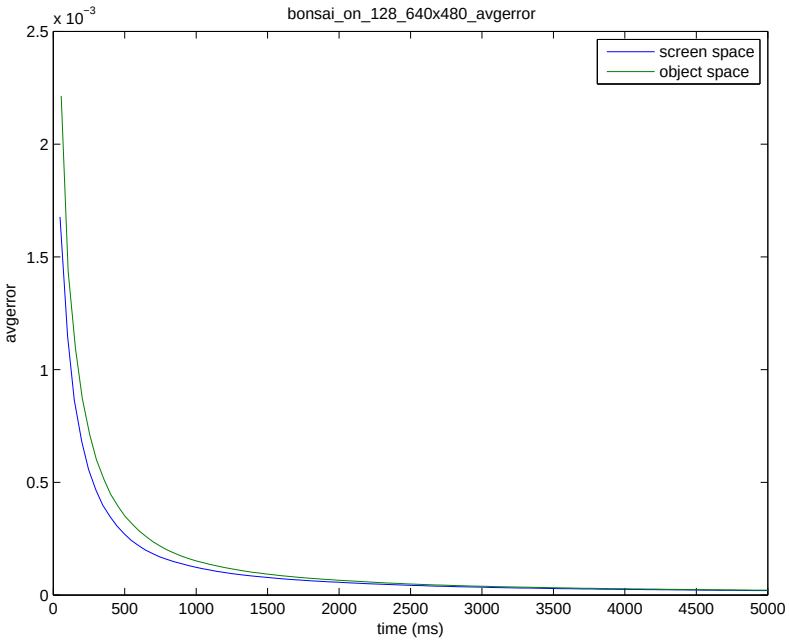


Figure A.23: screen space vs object space: average error bonsai, background on, volume res = 128, screen res = 640x480, average error

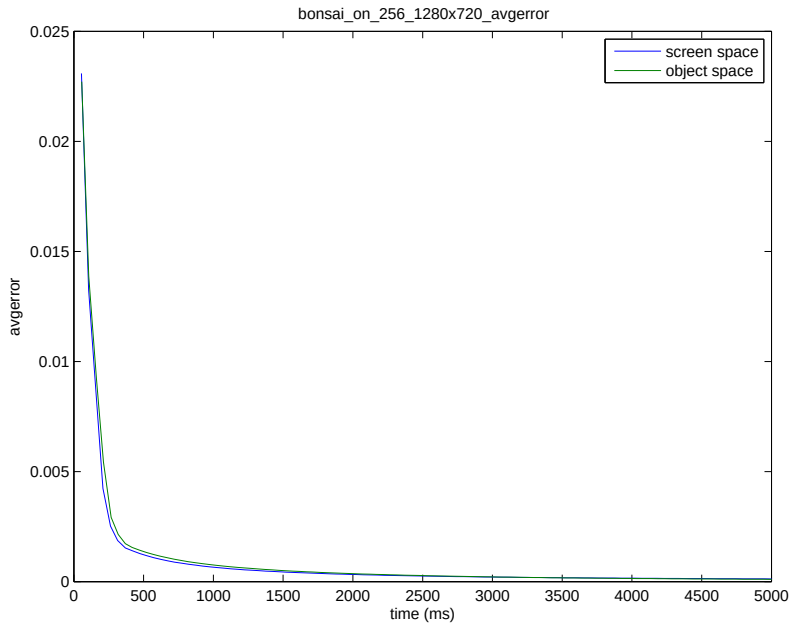


Figure A.24: screen space vs object space: average error bonsai, background on, volume res = 256, screen res = 1280x720, average error

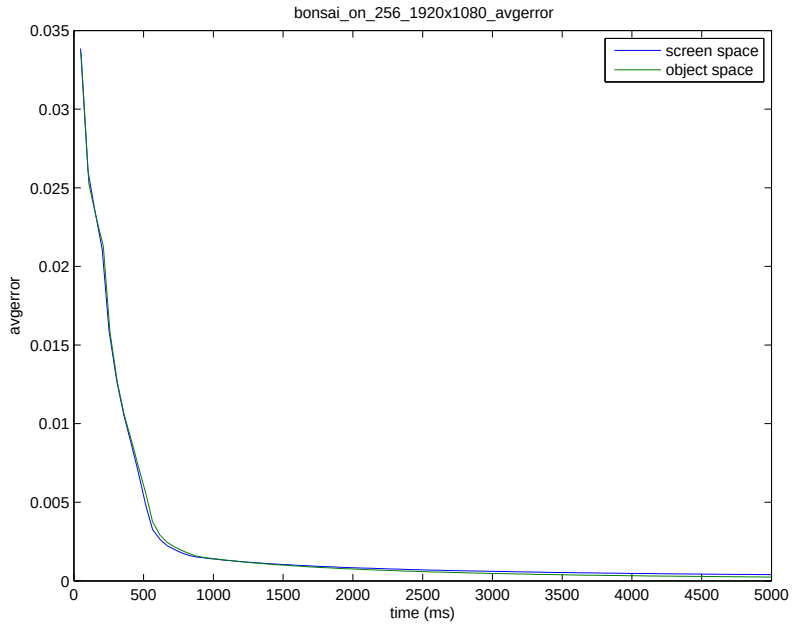


Figure A.25: screen space vs object space: average error bonsai, background on, volume res = 256, screen res = 1920x1080, average error

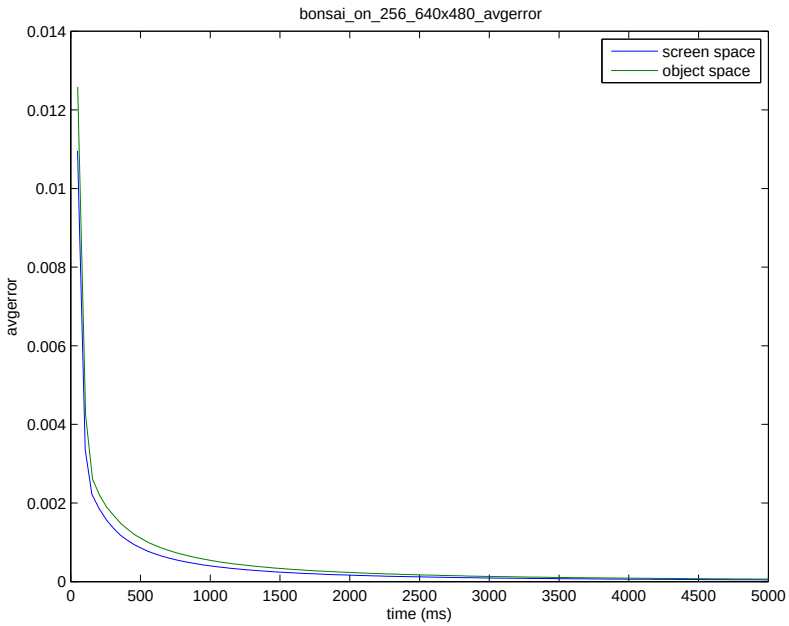


Figure A.26: screen space vs object space: average error bonsai, background on, volume res = 256, screen res = 640x480, average error

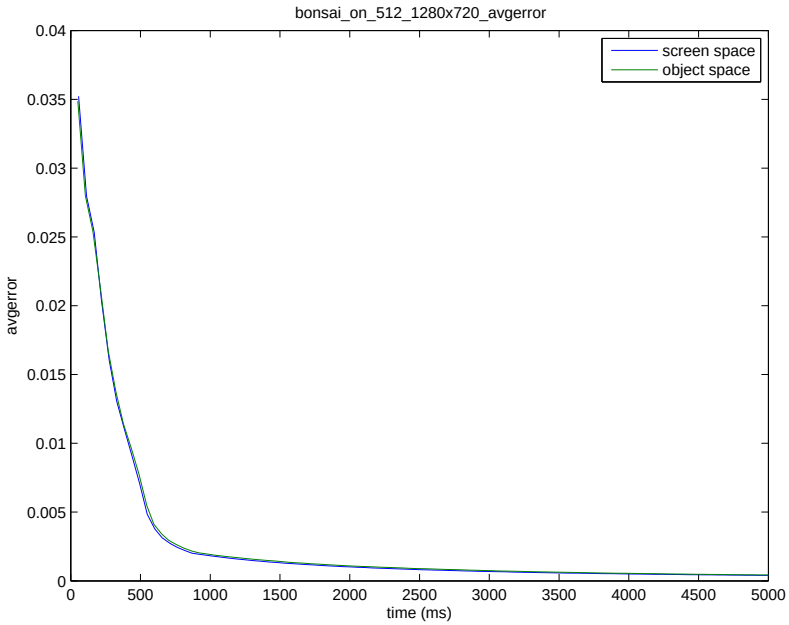


Figure A.27: screen space vs object space: average error bonsai, background on, volume res = 512, screen res = 1280x720, average error

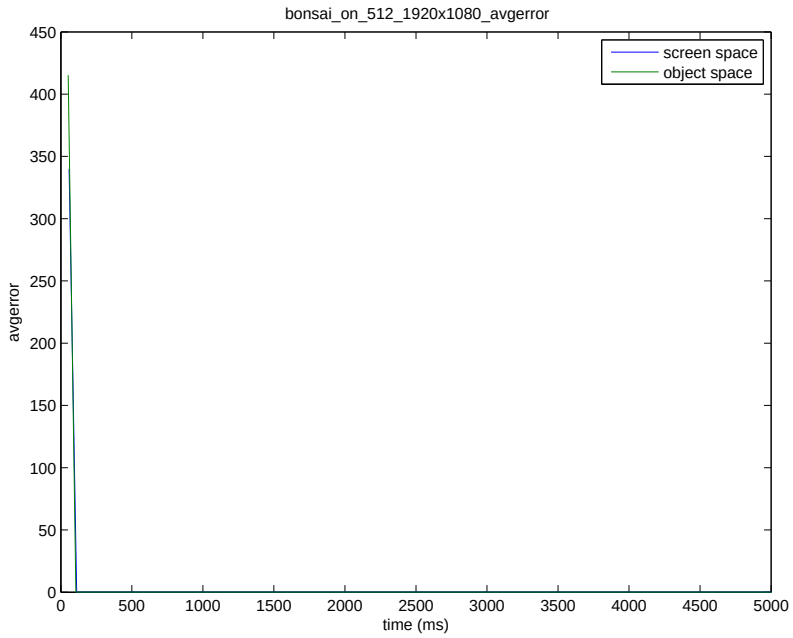


Figure A.28: screen space vs object space: average error bonsai, background on, volume res = 512, screen res = 1920x1080, average error

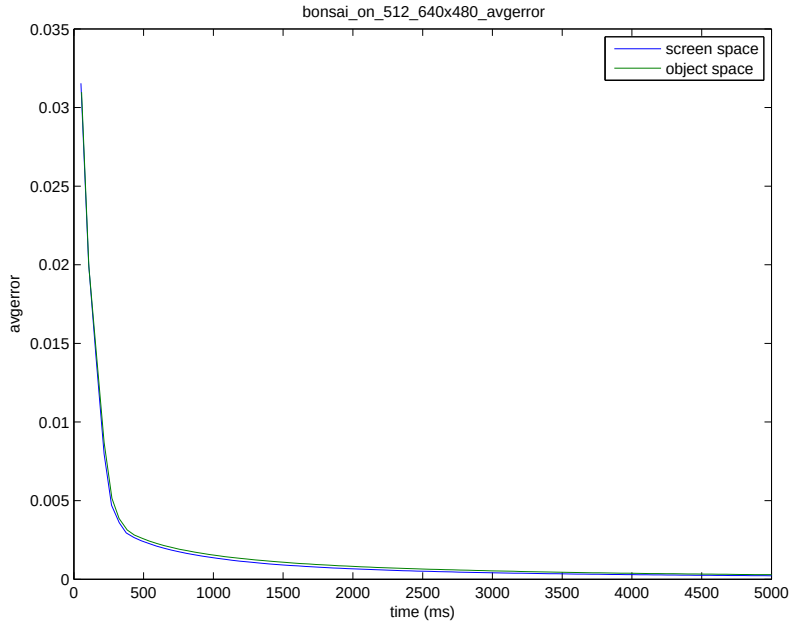


Figure A.29: screen space vs object space: average error bonsai, background on, volume res = 512, screen res = 640x480, average error

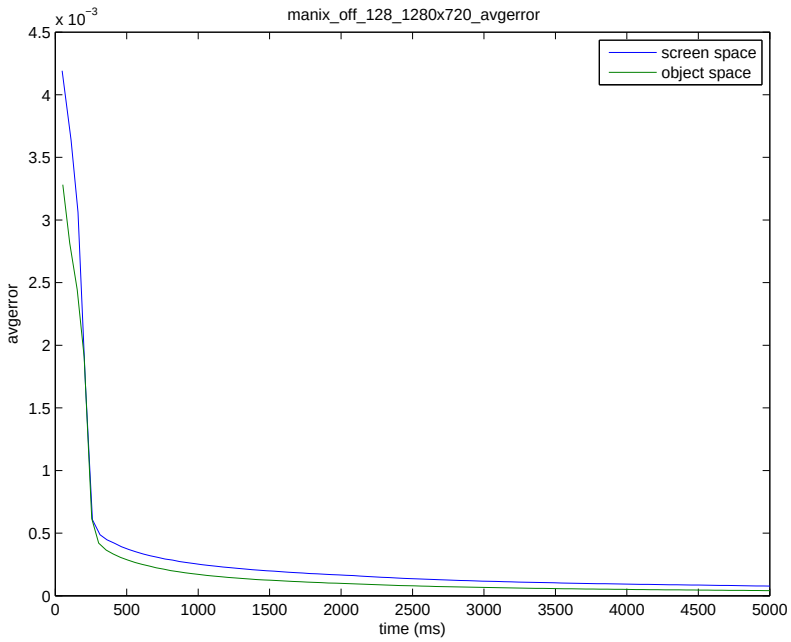


Figure A.30: screen space vs object space: average error manix, background off, volume res = 128, screen res = 1280x720, average error

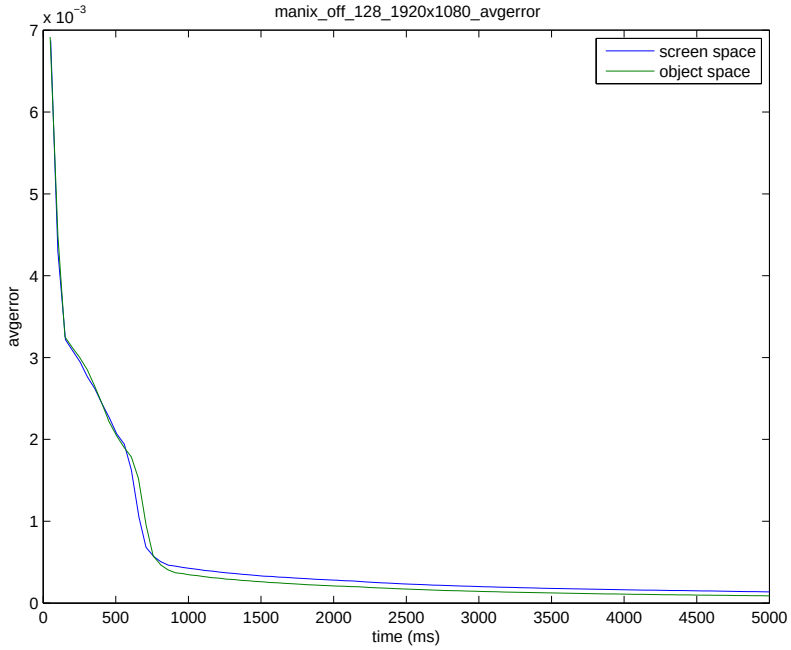


Figure A.31: screen space vs object space: average error manix, background off, volume res = 128, screen res = 1920x1080, average error

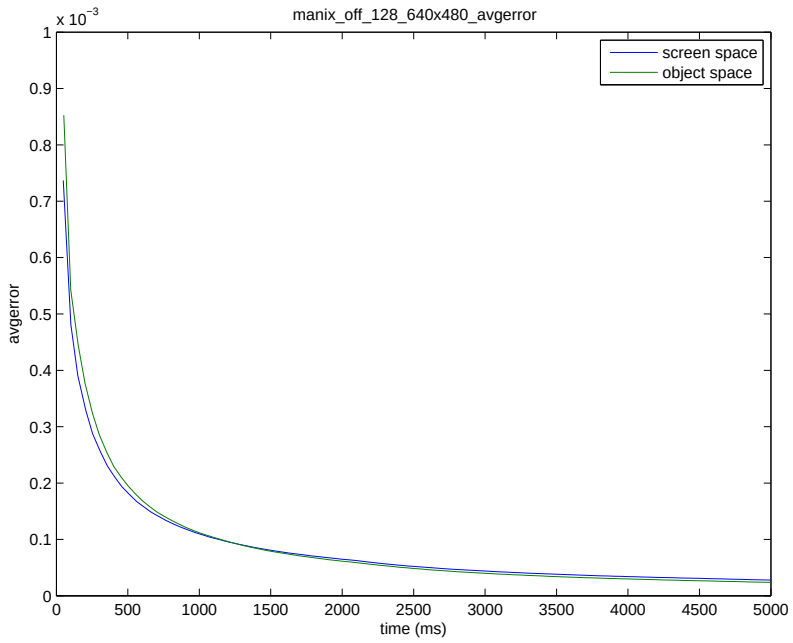


Figure A.32: screen space vs object space: average error manix, background off, volume res = 128, screen res = 640x480, average error

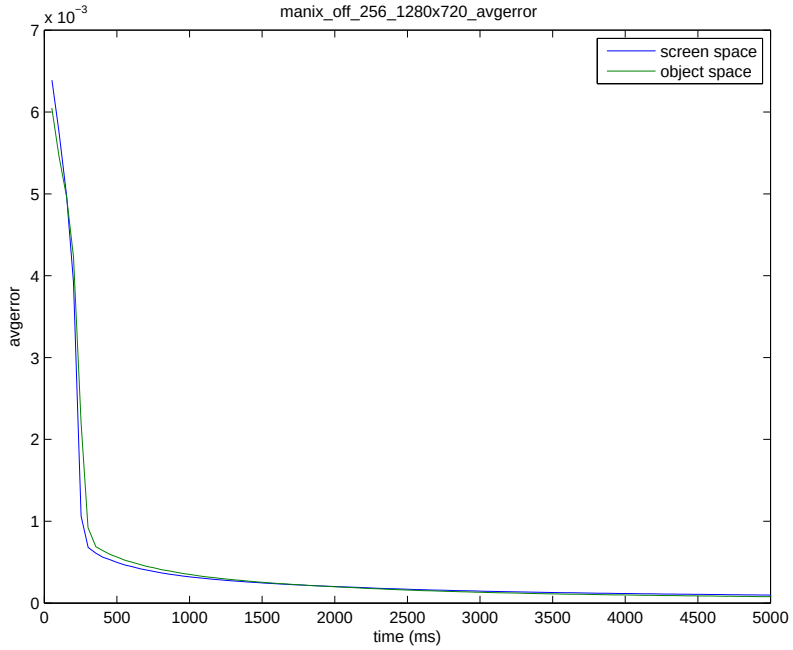


Figure A.33: screen space vs object space: average error manix, background off, volume res = 256, screen res = 1280x720, average error

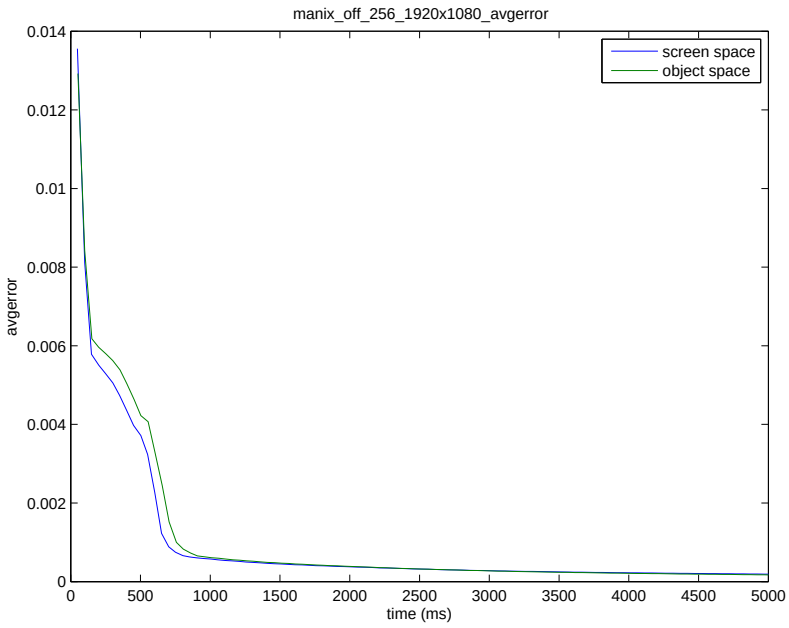


Figure A.34: screen space vs object space: average error manix, background off, volume res = 256, screen res = 1920x1080, average error

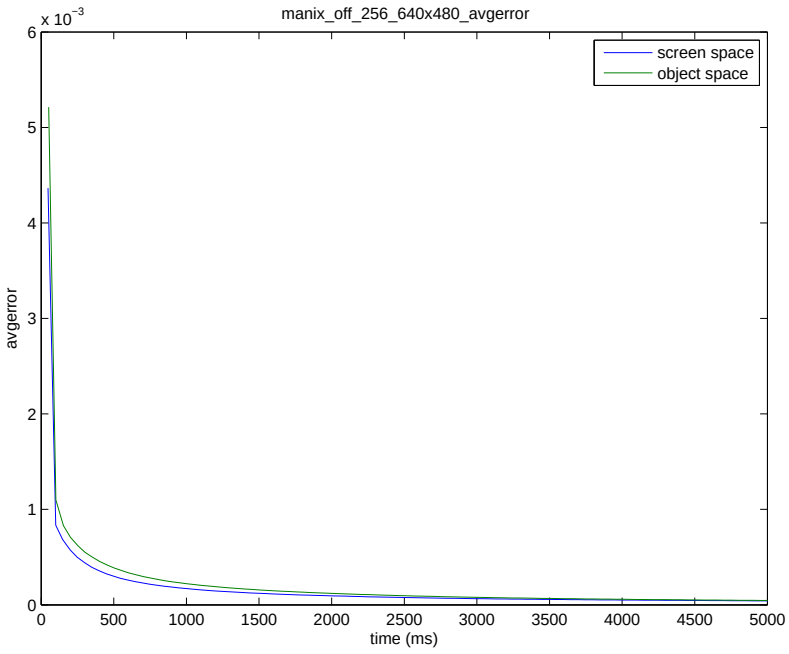


Figure A.35: screen space vs object space: average error manix, background off, volume res = 256, screen res = 640x480, average error

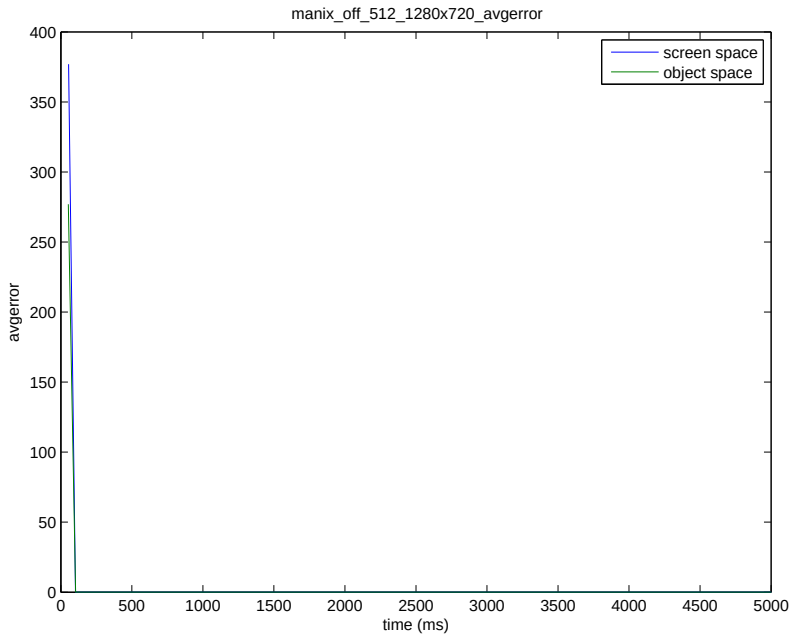


Figure A.36: screen space vs object space: average error manix, background off, volume res = 512, screen res = 1280x720, average error

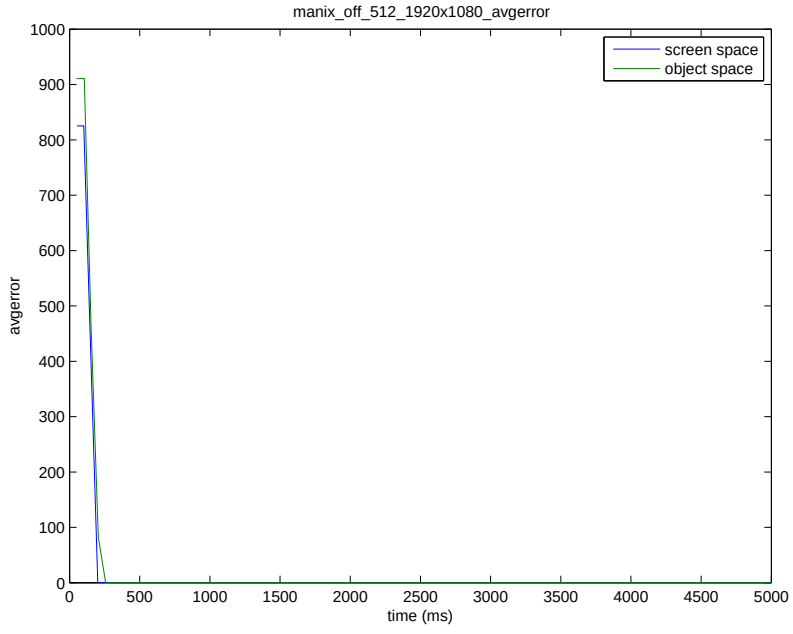


Figure A.37: screen space vs object space: average error manix, background off, volume res = 512, screen res = 1920x1080, average error

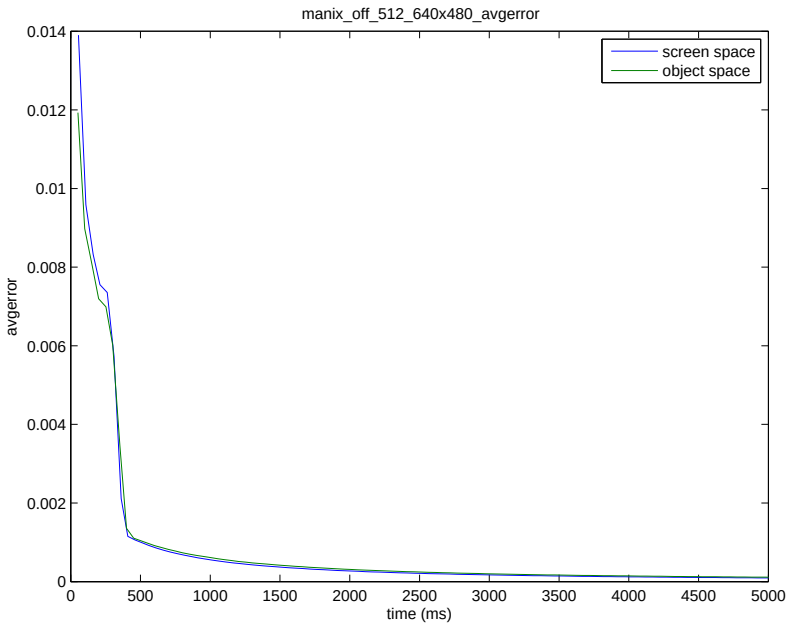


Figure A.38: screen space vs object space: average error manix, background off, volume res = 512, screen res = 640x480, average error

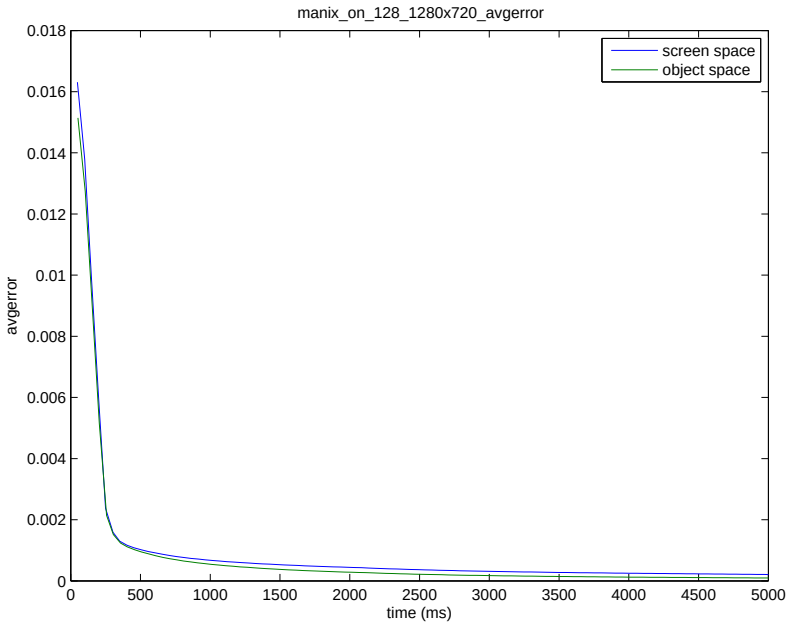


Figure A.39: screen space vs object space: average error manix, background on, volume res = 128, screen res = 1280x720, average error

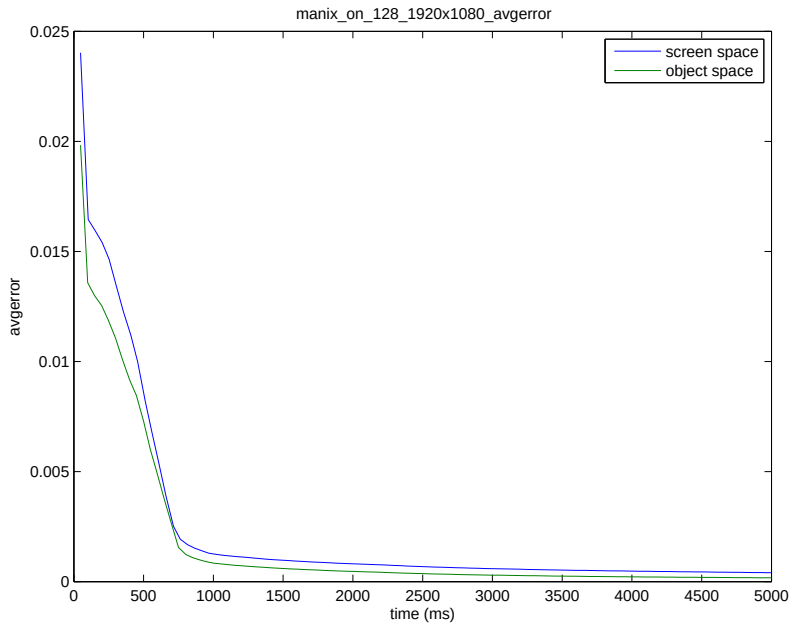


Figure A.40: screen space vs object space: average error manix, background on, volume res = 128, screen res = 1920x1080, average error

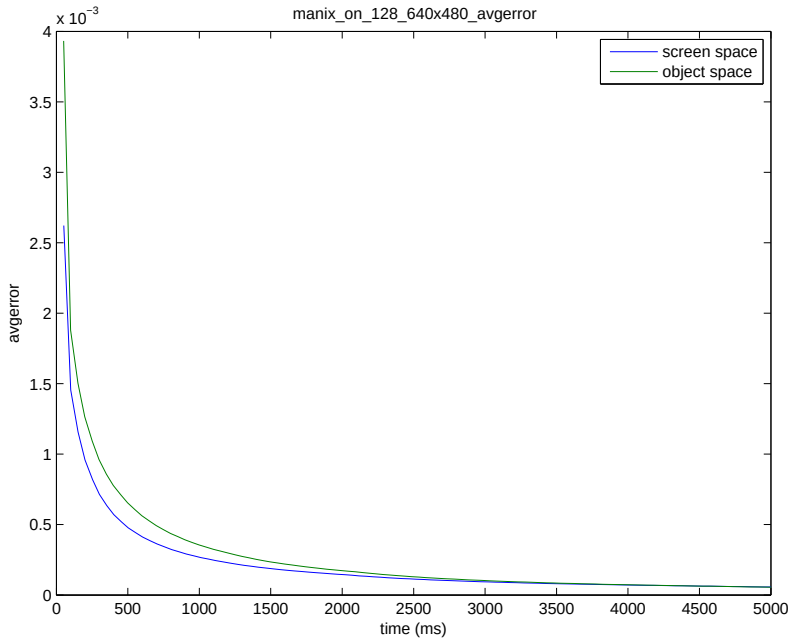


Figure A.41: screen space vs object space: average error manix, background on, volume res = 128, screen res = 640x480, average error

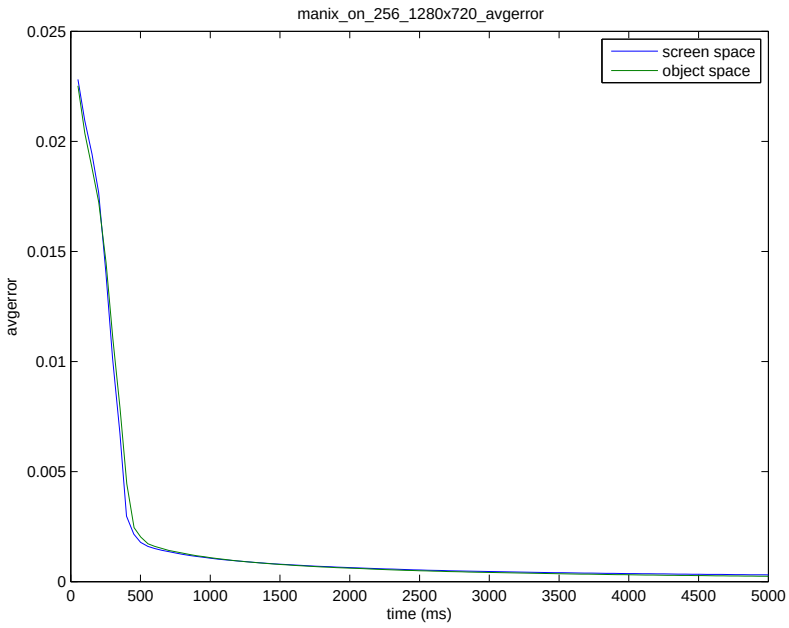


Figure A.42: screen space vs object space: average error manix, background on, volume res = 256, screen res = 1280x720, average error

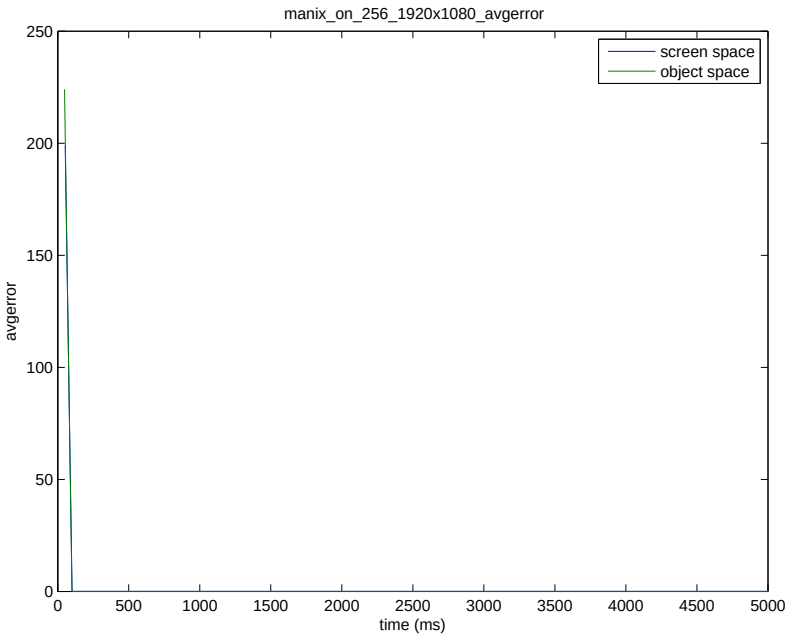


Figure A.43: screen space vs object space: average error manix, background on, volume res = 256, screen res = 1920x1080, average error

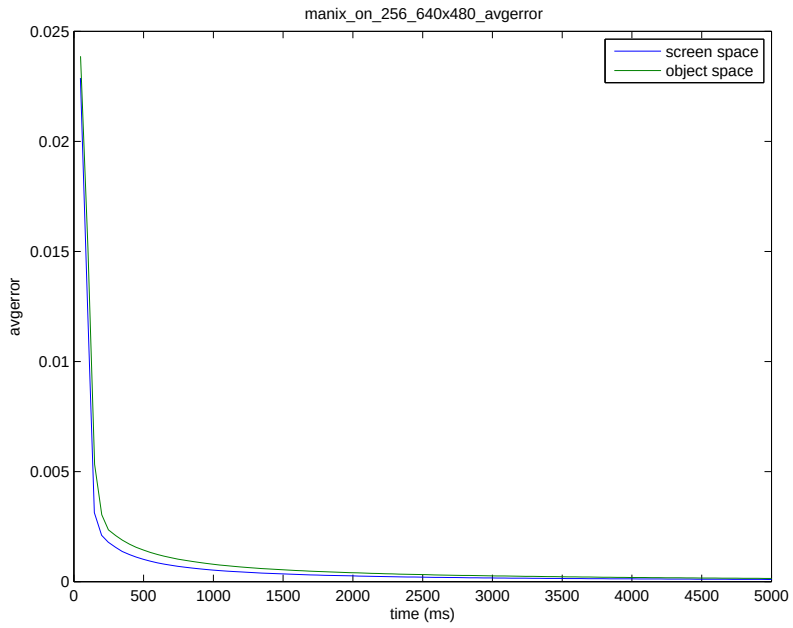


Figure A.44: screen space vs object space: average error manix, background on, volume res = 256, screen res = 640x480, average error

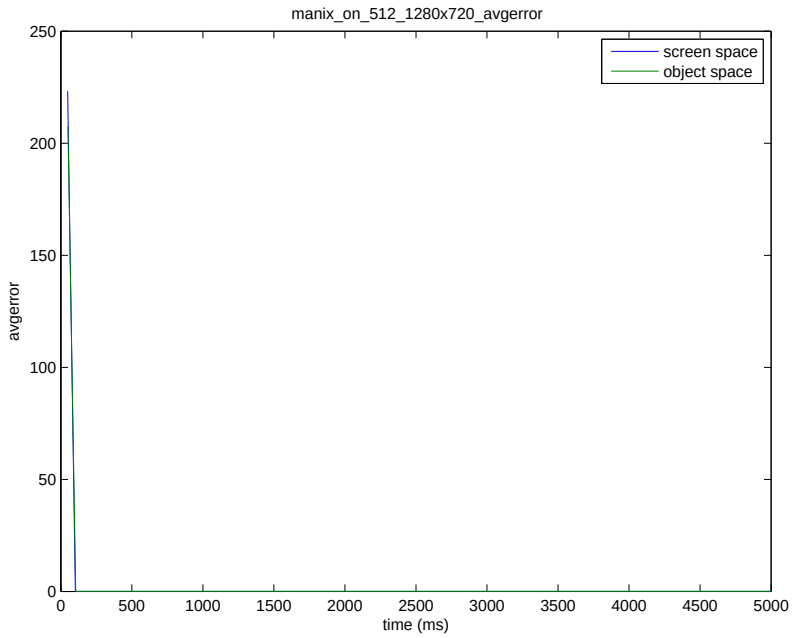


Figure A.45: screen space vs object space: average error manix, background on, volume res = 512, screen res = 1280x720, average error

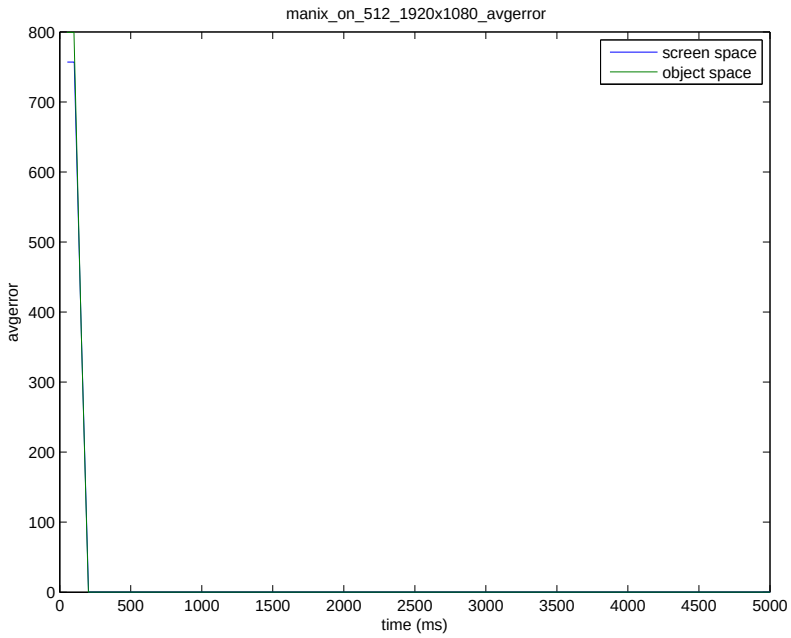


Figure A.46: screen space vs object space: average error manix, background on, volume res = 512, screen res = 1920x1080, average error

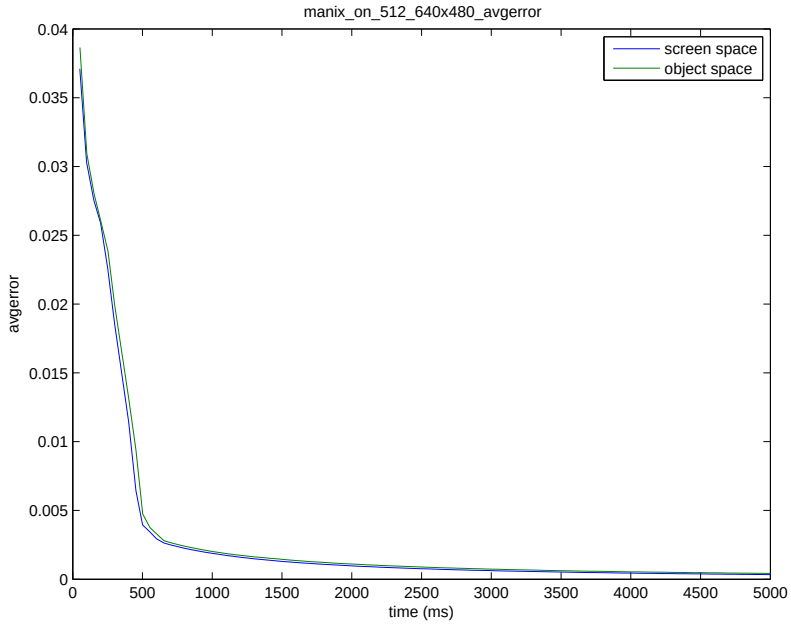


Figure A.47: screen space vs object space: average error manix, background on, volume res = 512, screen res = 640x480, average error

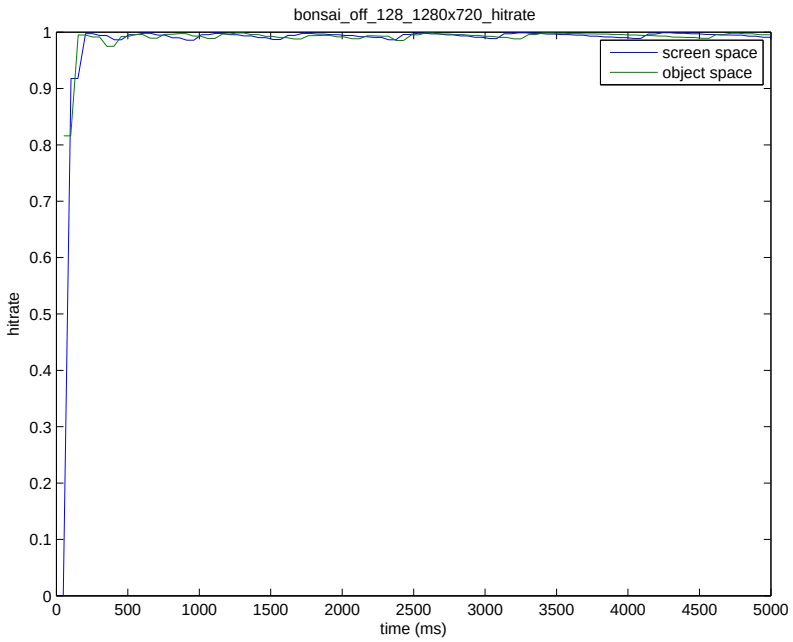


Figure A.48: screen space vs object space: cache hit rate bonsai, background off, volume res = 128, screen res = 1280x720, cache hit rate

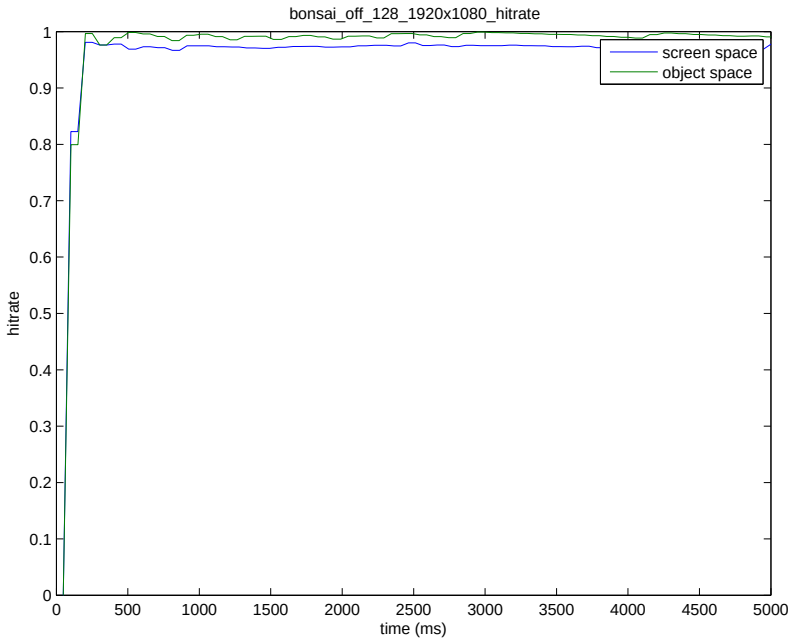


Figure A.49: screen space vs object space: cache hit rate bonsai, background off, volume res = 128, screen res = 1920x1080, cache hit rate

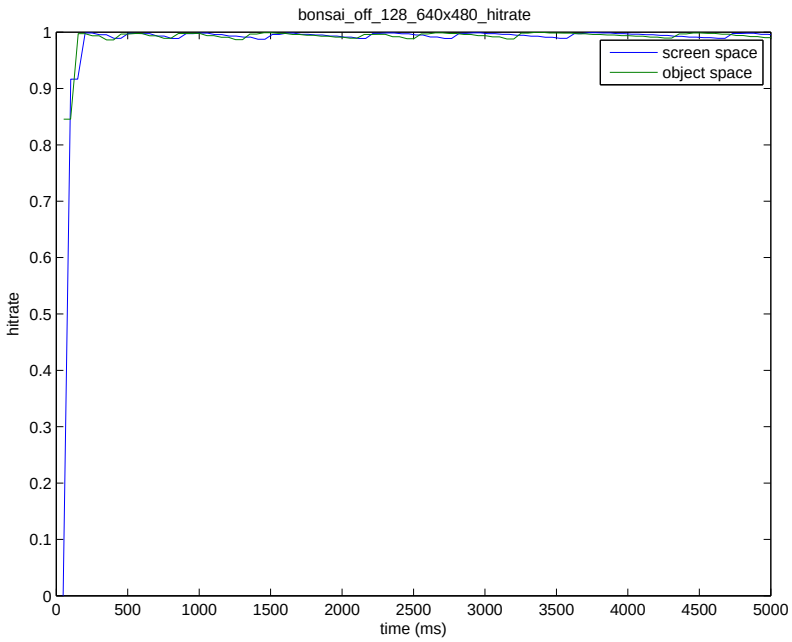


Figure A.50: screen space vs object space: cache hit rate bonsai, background off, volume res = 128, screen res = 640x480, cache hit rate

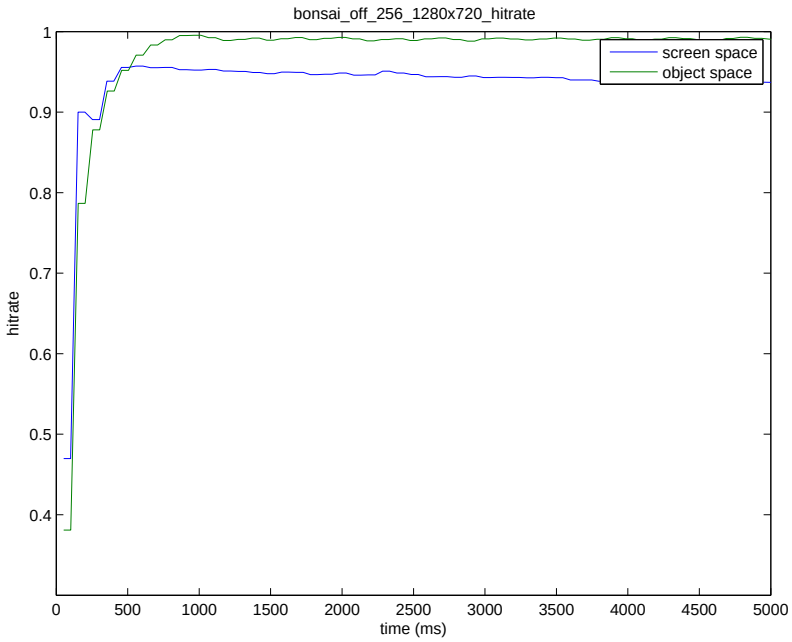


Figure A.51: screen space vs object space: cache hit rate bonsai, background off, volume res = 256, screen res = 1280x720, cache hit rate

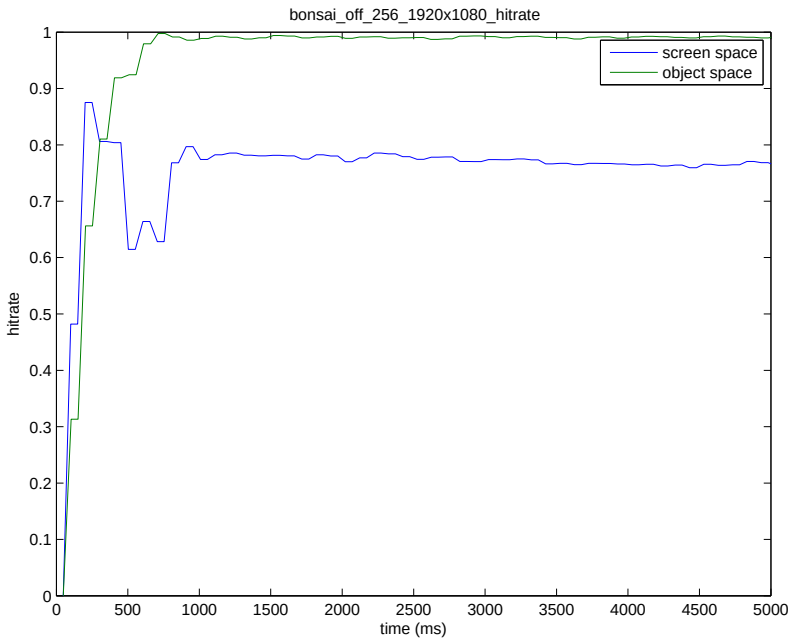


Figure A.52: screen space vs object space: cache hit rate bonsai, background off, volume res = 256, screen res = 1920x1080, cache hit rate

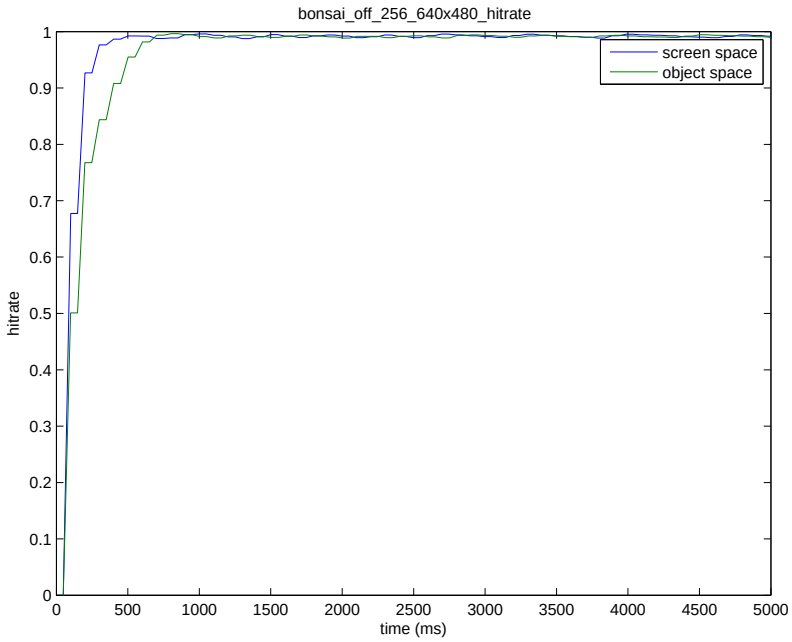


Figure A.53: screen space vs object space: cache hit rate bonsai, background off, volume res = 256, screen res = 640x480, cache hit rate

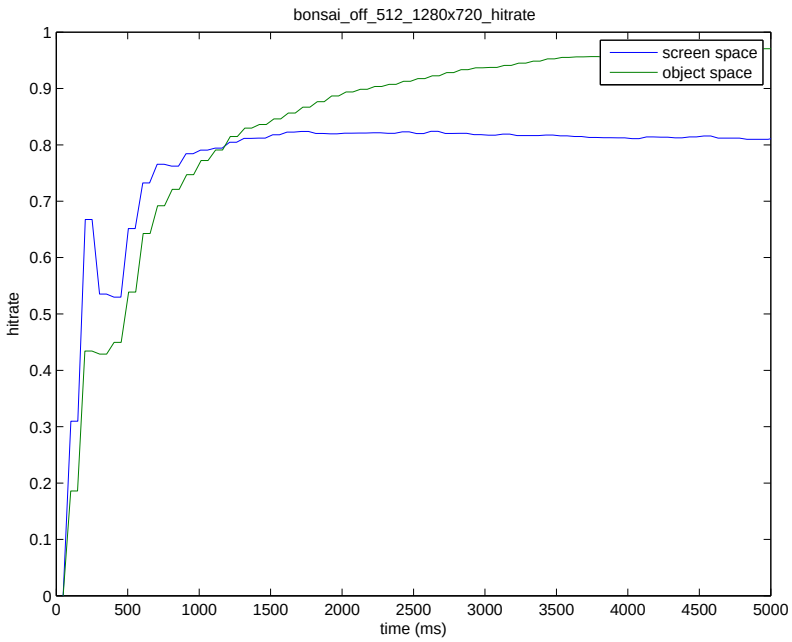


Figure A.54: screen space vs object space: cache hit rate bonsai, background off, volume res = 512, screen res = 1280x720, cache hit rate

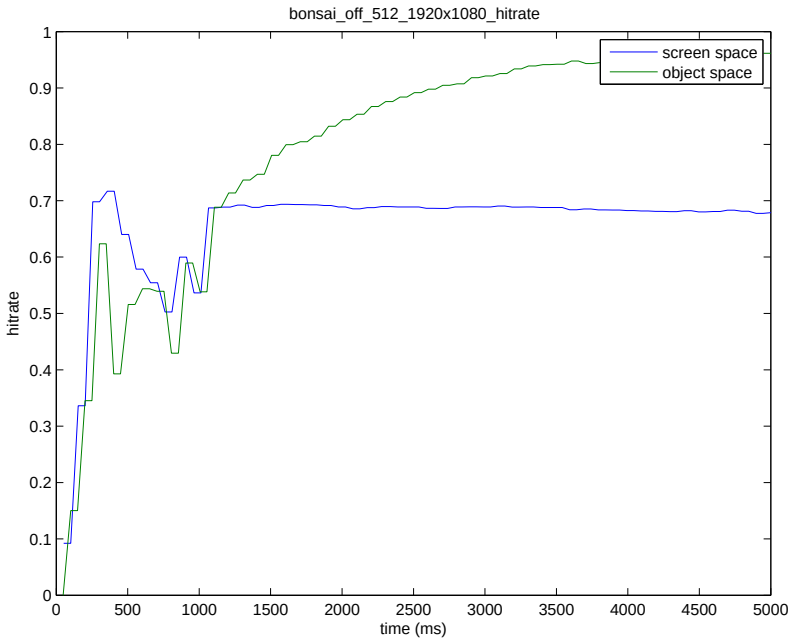


Figure A.55: screen space vs object space: cache hit rate bonsai, background off, volume res = 512, screen res = 1920x1080, cache hit rate

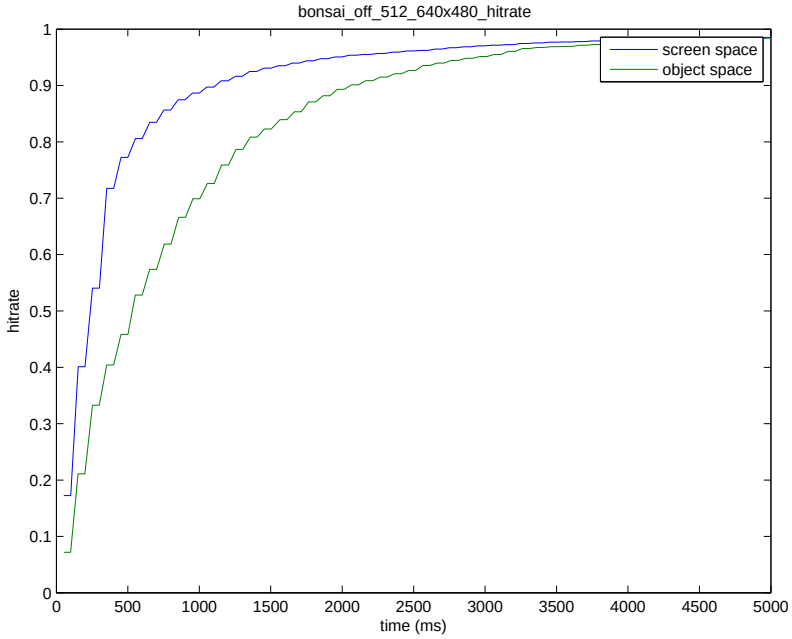


Figure A.56: screen space vs object space: cache hit rate bonsai, background off, volume res = 512, screen res = 640x480, cache hit rate

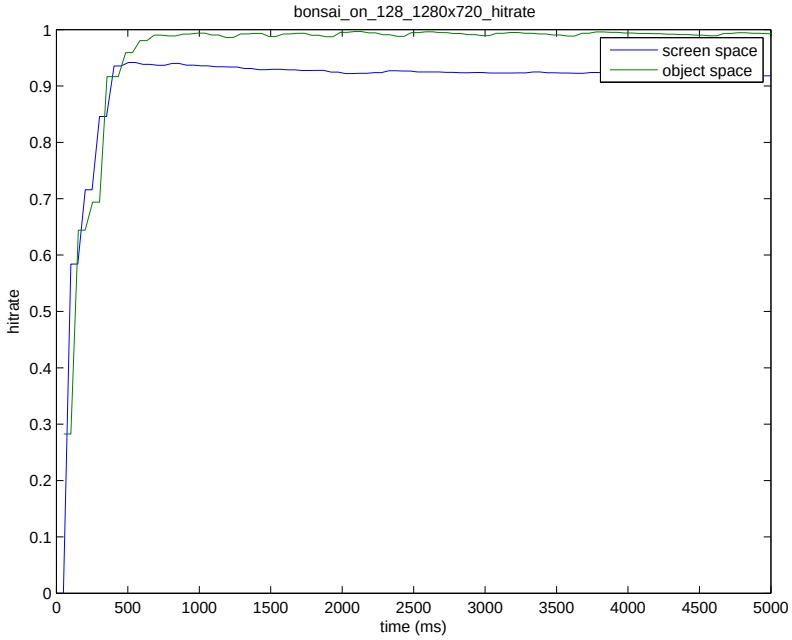


Figure A.57: screen space vs object space: cache hit rate bonsai, background on, volume res = 128, screen res = 1280x720, cache hit rate

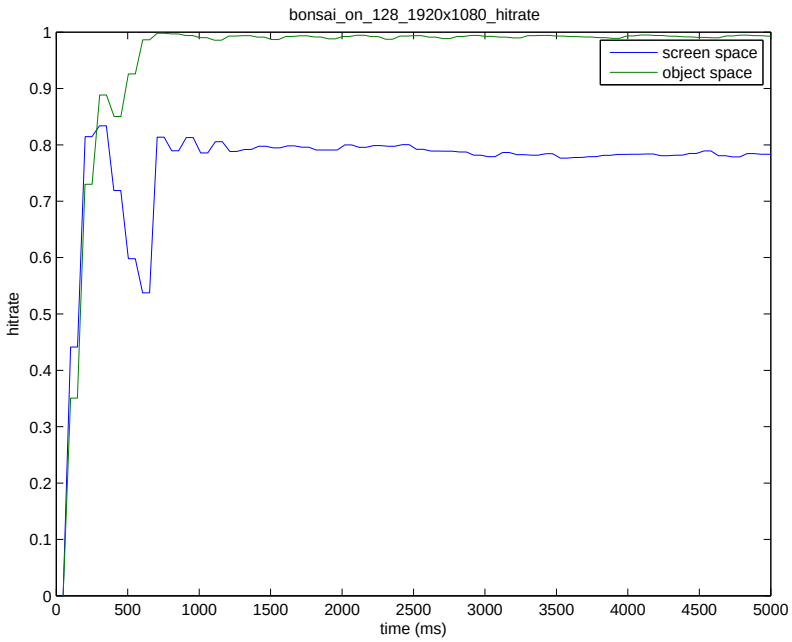


Figure A.58: screen space vs object space: cache hit rate bonsai, background on, volume res = 128, screen res = 1920x1080, cache hit rate

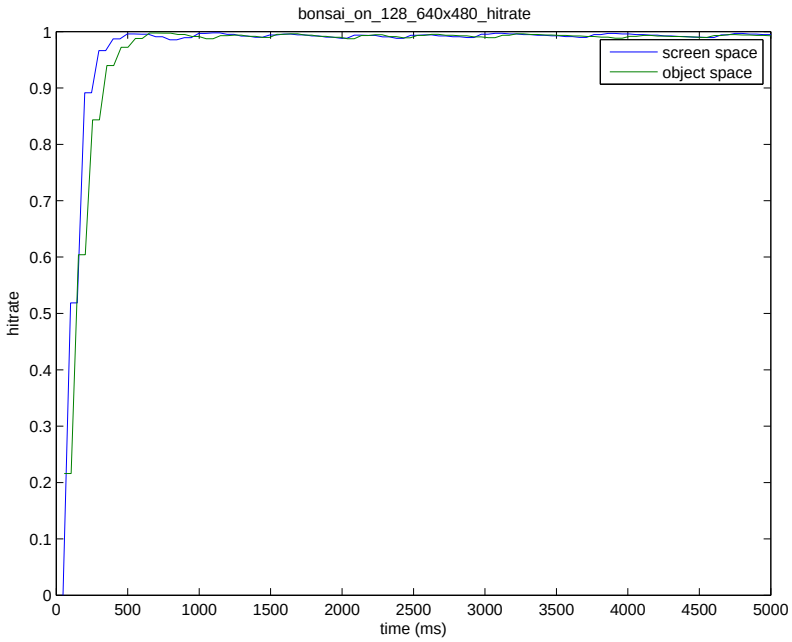


Figure A.59: screen space vs object space: cache hit rate bonsai, background on, volume res = 128, screen res = 640x480, cache hit rate

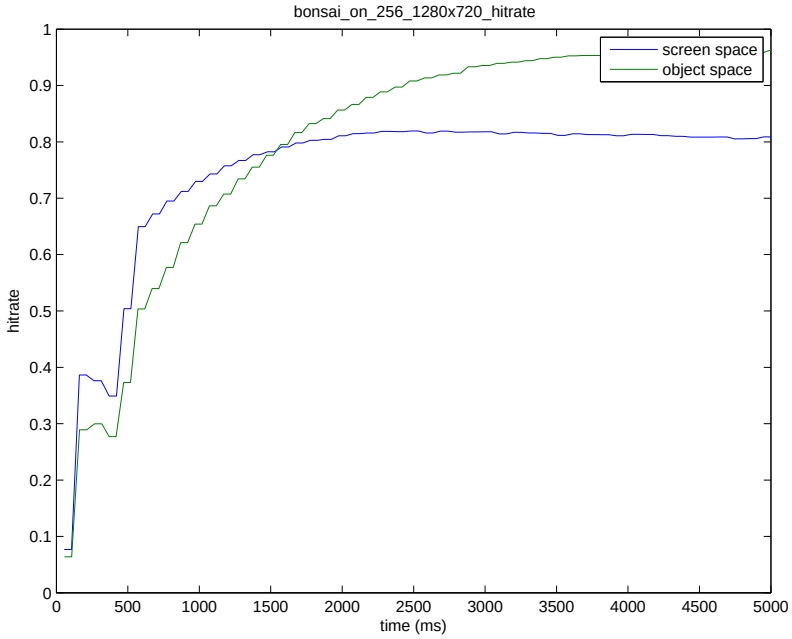


Figure A.60: screen space vs object space: cache hit rate bonsai, background on, volume res = 256, screen res = 1280x720, cache hit rate

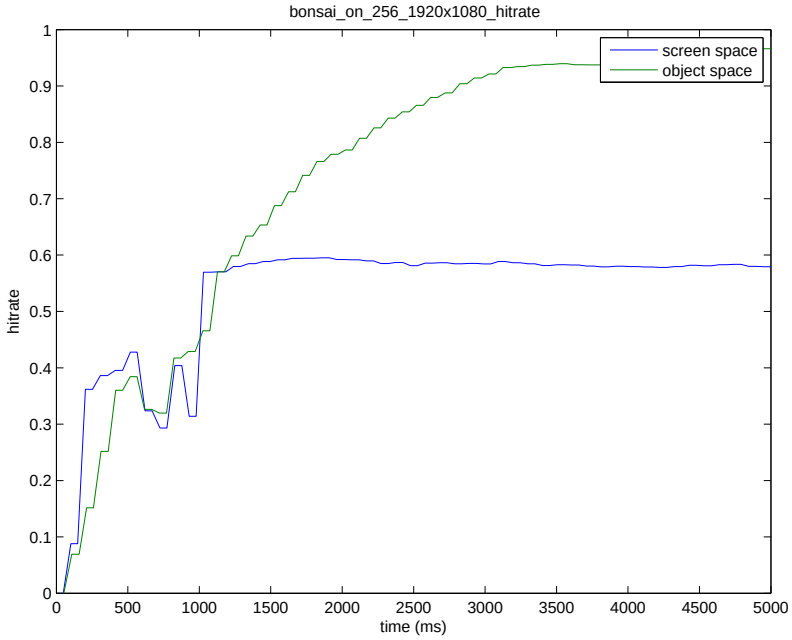


Figure A.61: screen space vs object space: cache hit rate bonsai, background on, volume res = 256, screen res = 1920x1080, cache hit rate

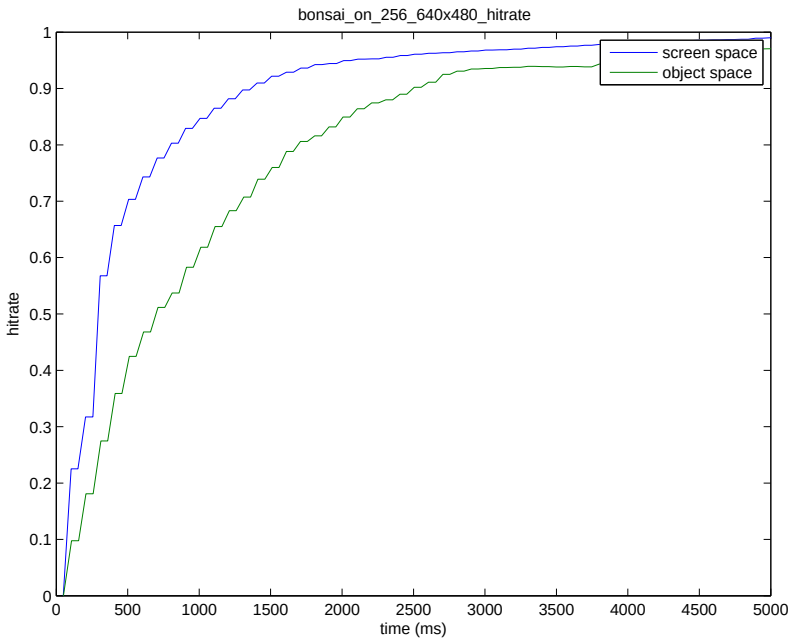


Figure A.62: screen space vs object space: cache hit rate bonsai, background on, volume res = 256, screen res = 640x480, cache hit rate

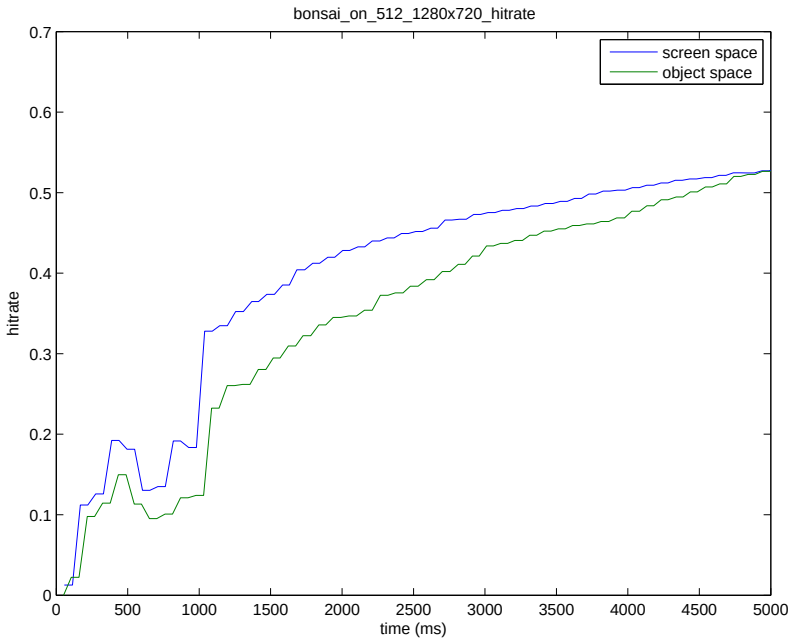


Figure A.63: screen space vs object space: cache hit rate bonsai, background on, volume res = 512, screen res = 1280x720, cache hit rate

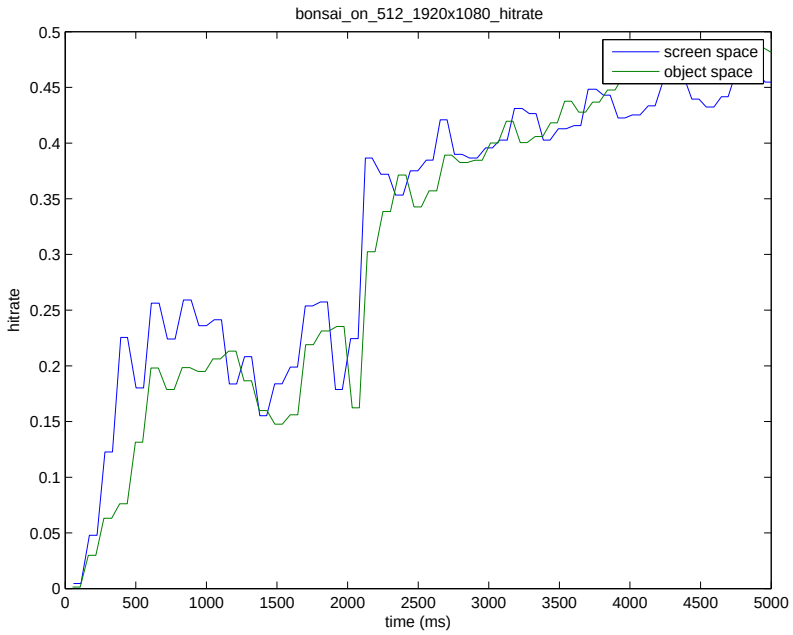


Figure A.64: screen space vs object space: cache hit rate bonsai, background on, volume res = 512, screen res = 1920x1080, cache hit rate

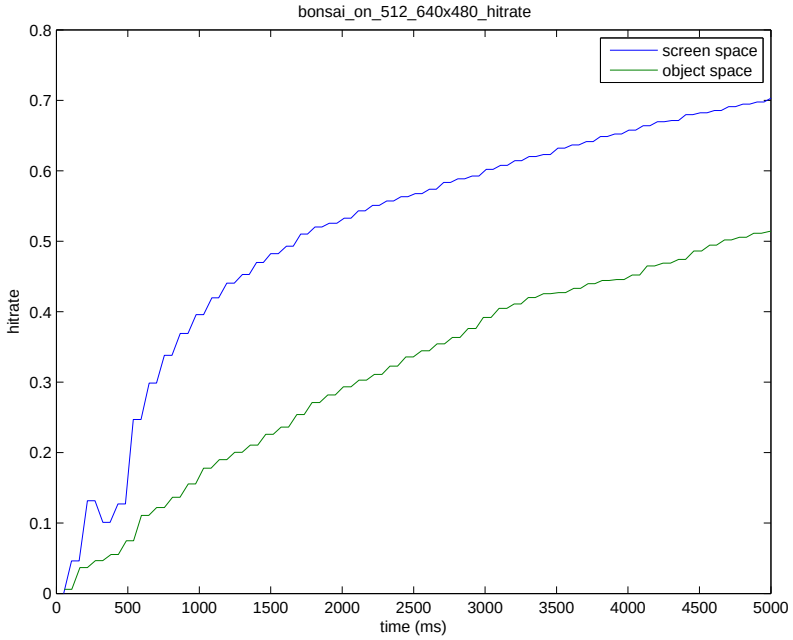


Figure A.65: screen space vs object space: cache hit rate bonsai, background on, volume res = 512, screen res = 640x480, cache hit rate

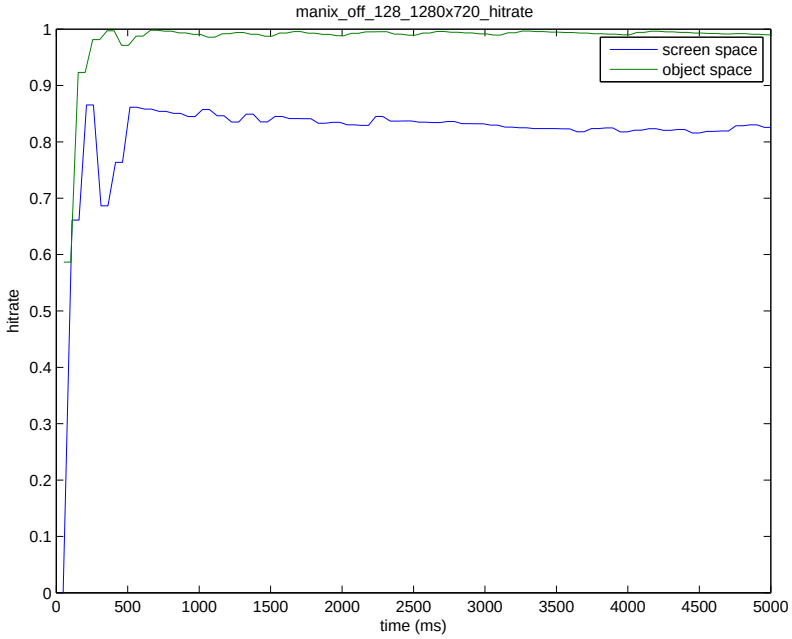


Figure A.66: screen space vs object space: cache hit rate manix, background off, volume res = 128, screen res = 1280x720, cache hit rate

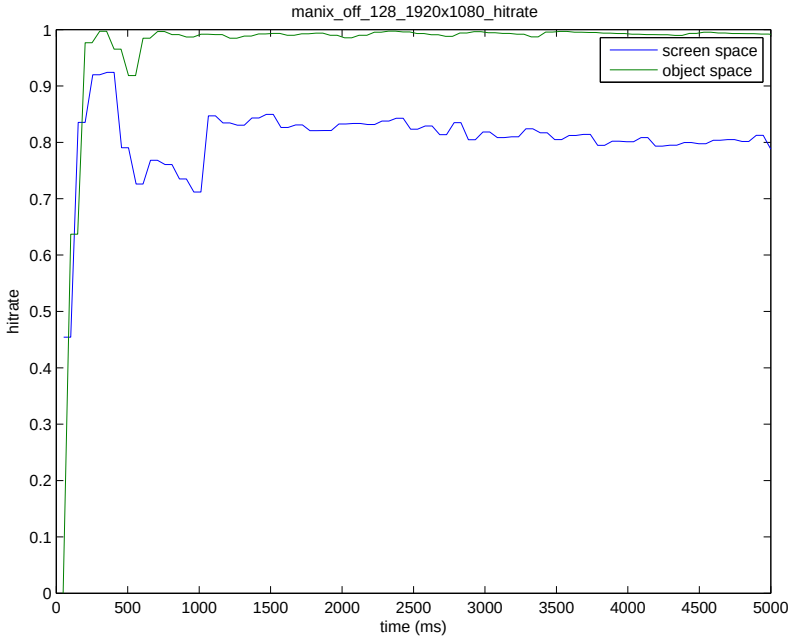


Figure A.67: screen space vs object space: cache hit rate manix, background off, volume res = 128, screen res = 1920x1080, cache hit rate

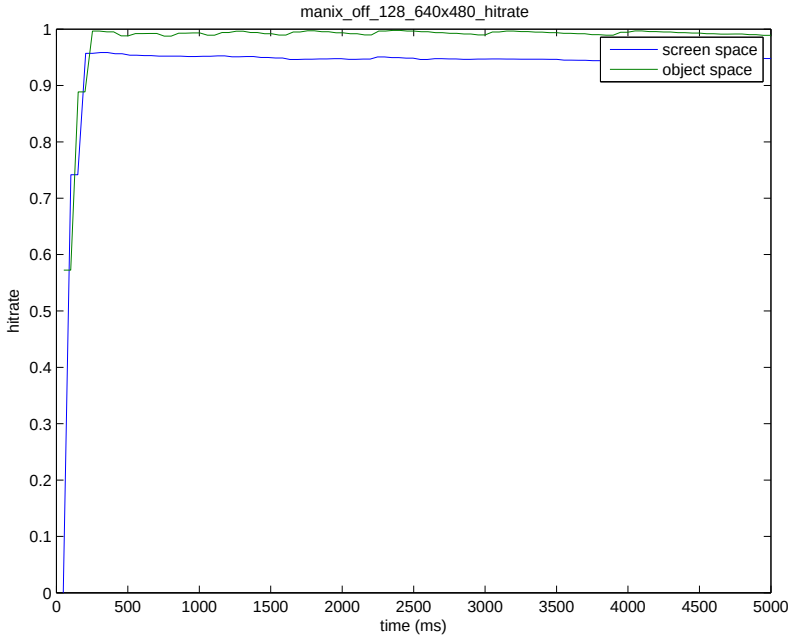


Figure A.68: screen space vs object space: cache hit rate manix, background off, volume res = 128, screen res = 640x480, cache hit rate

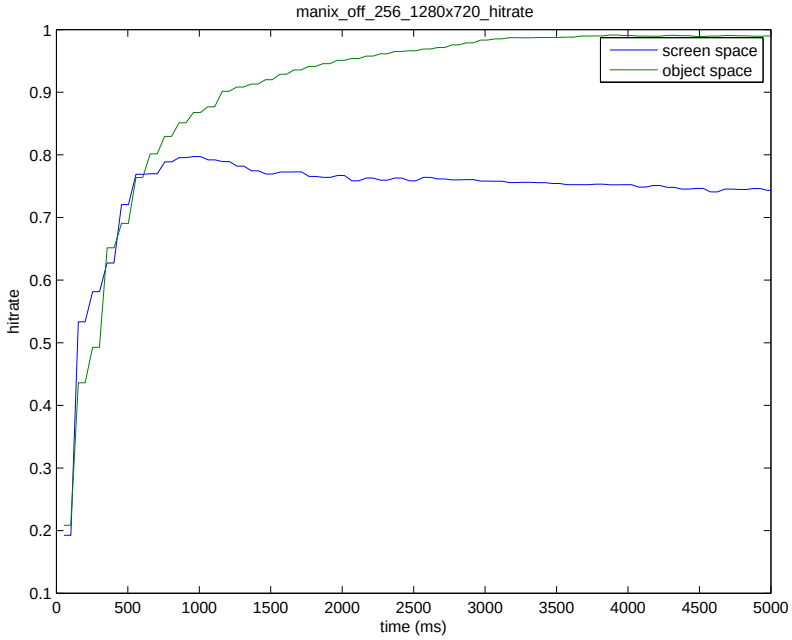


Figure A.69: screen space vs object space: cache hit rate manix, background off, volume res = 256, screen res = 1280x720, cache hit rate

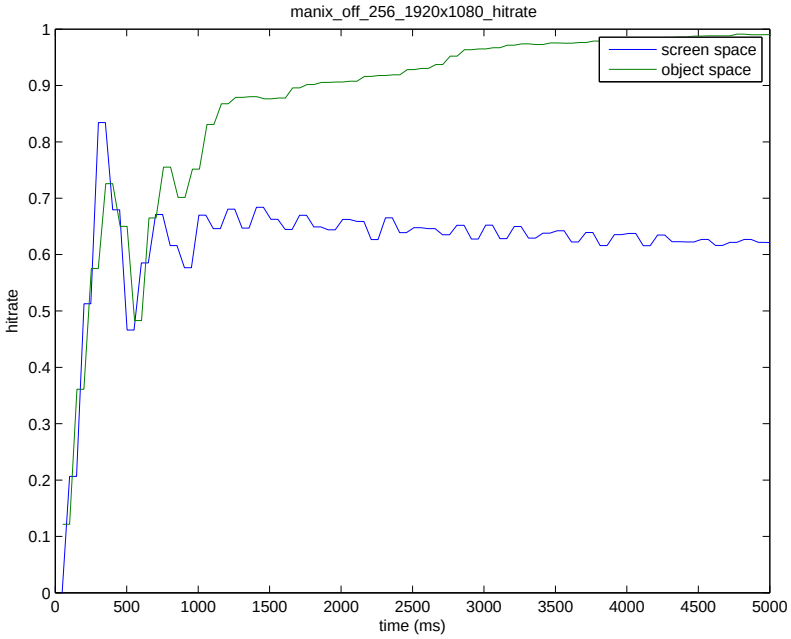


Figure A.70: screen space vs object space: cache hit rate manix, background off, volume res = 256, screen res = 1920x1080, cache hit rate

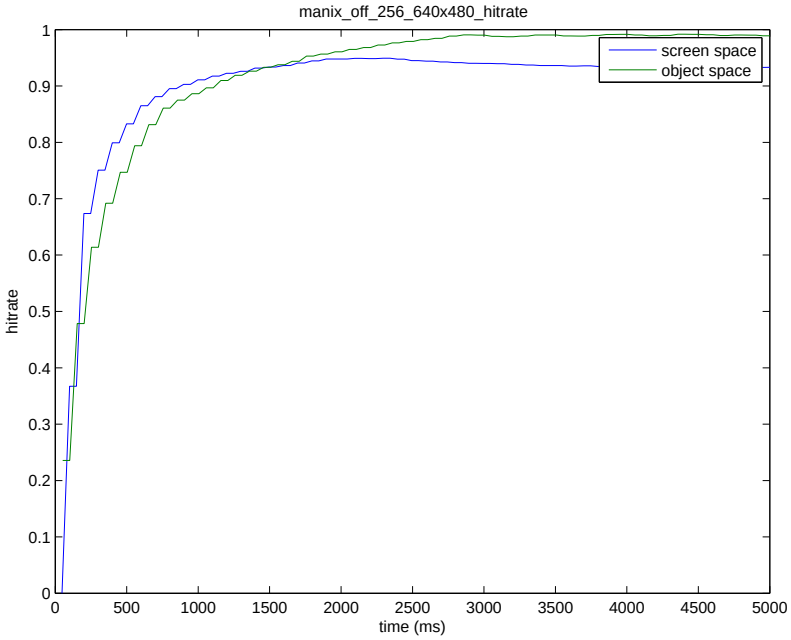


Figure A.71: screen space vs object space: cache hit rate manix, background off, volume res = 256, screen res = 640x480, cache hit rate

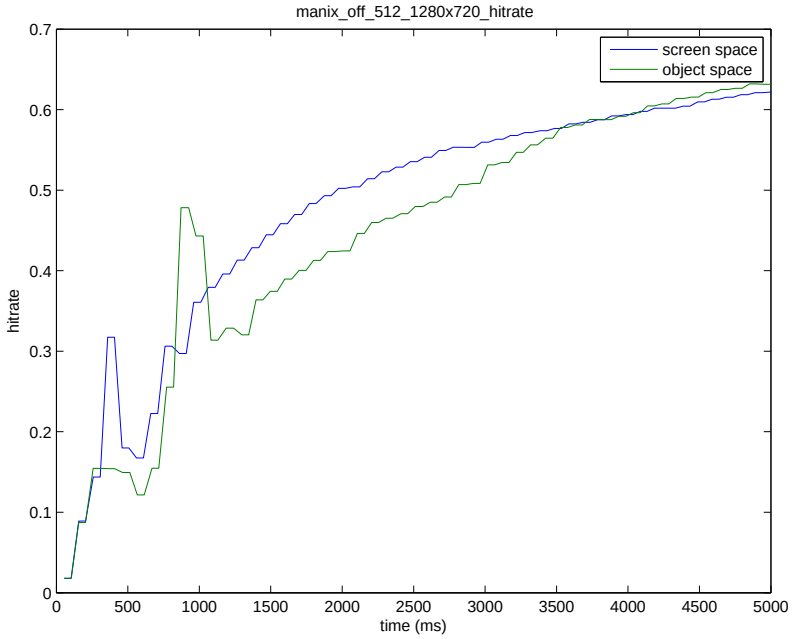


Figure A.72: screen space vs object space: cache hit rate manix, background off, volume res = 512, screen res = 1280x720, cache hit rate

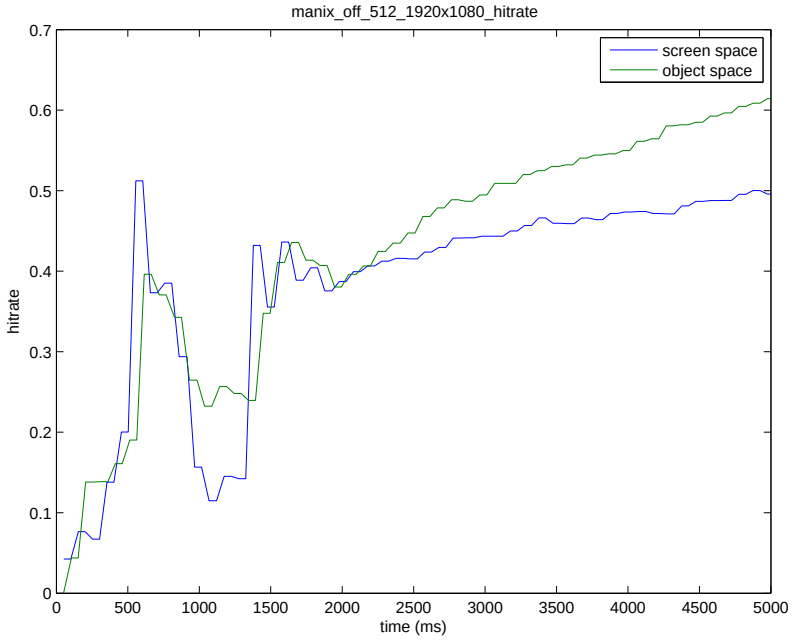


Figure A.73: screen space vs object space: cache hit rate manix, background off, volume res = 512, screen res = 1920x1080, cache hit rate

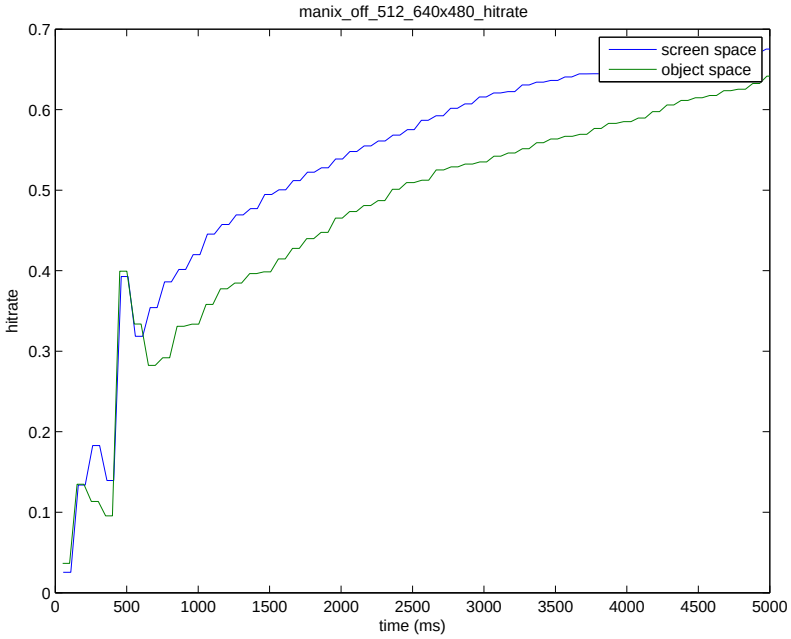


Figure A.74: screen space vs object space: cache hit rate manix, background off, volume res = 512, screen res = 640x480, cache hit rate

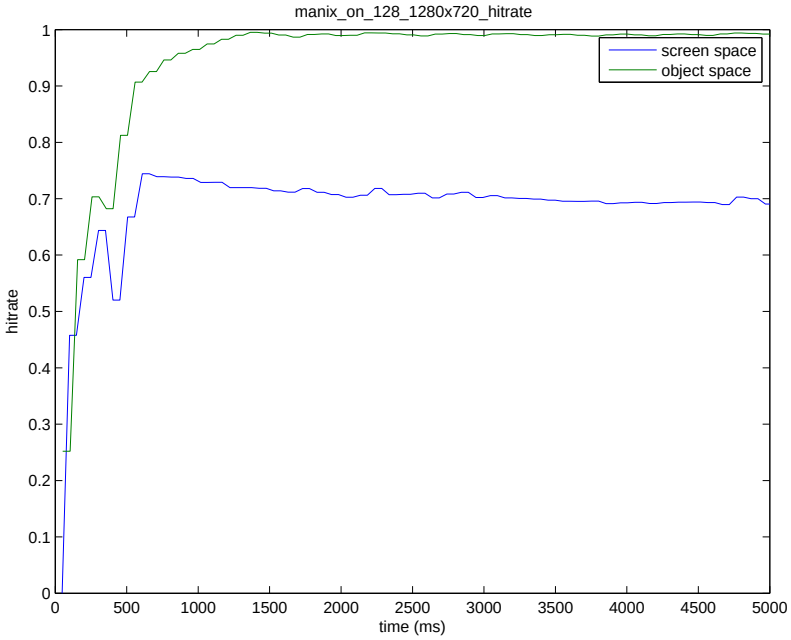


Figure A.75: screen space vs object space: cache hit rate manix, background on, volume res = 128, screen res = 1280x720, cache hit rate

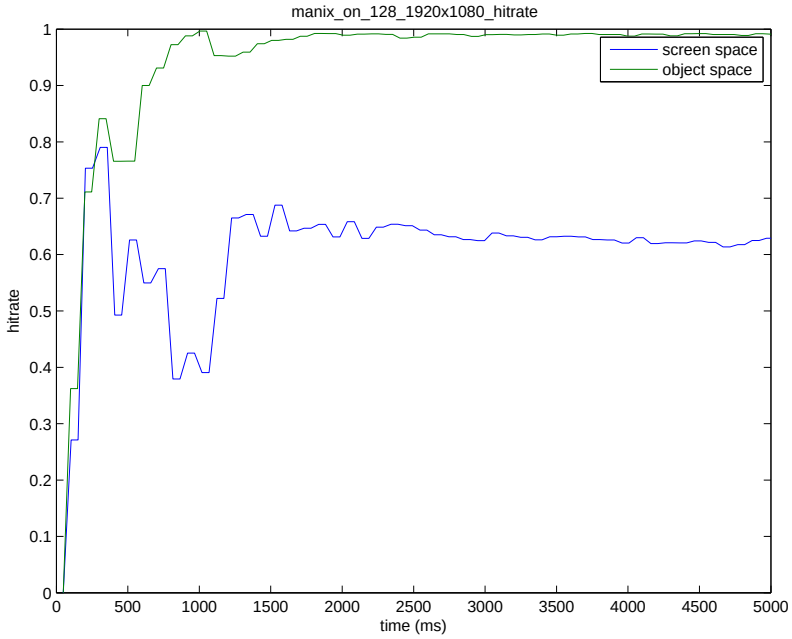


Figure A.76: screen space vs object space: cache hit rate manix, background on, volume res = 128, screen res = 1920x1080, cache hit rate

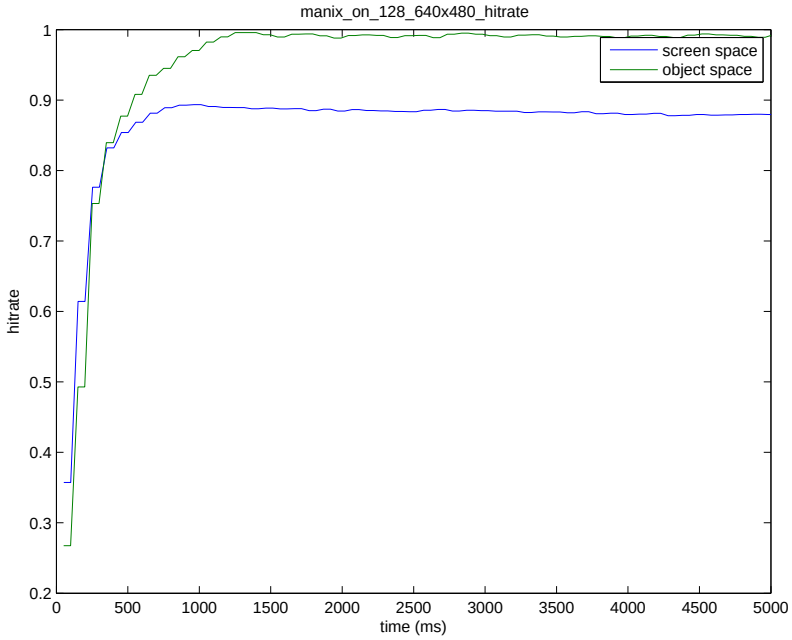


Figure A.77: screen space vs object space: cache hit rate manix, background on, volume res = 128, screen res = 640x480, cache hit rate

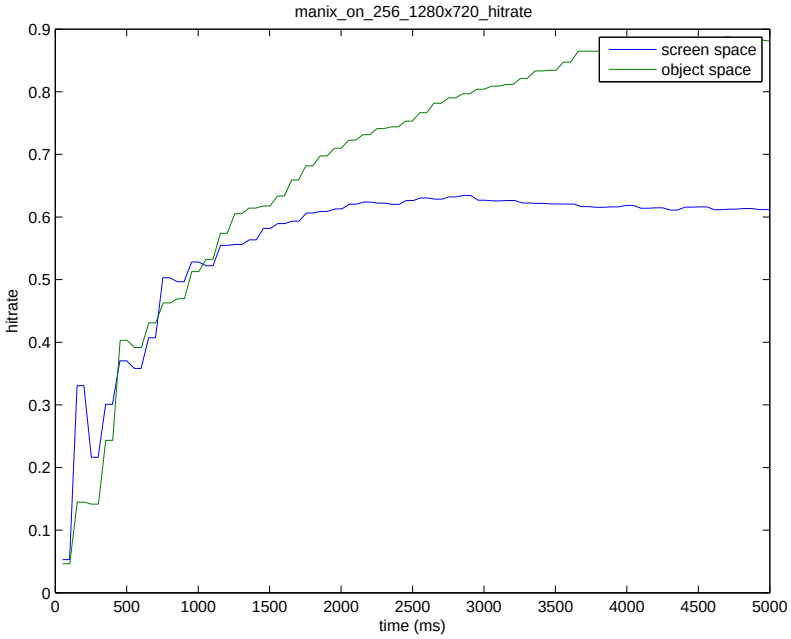


Figure A.78: screen space vs object space: cache hit rate manix, background on, volume res = 256, screen res = 1280x720, cache hit rate

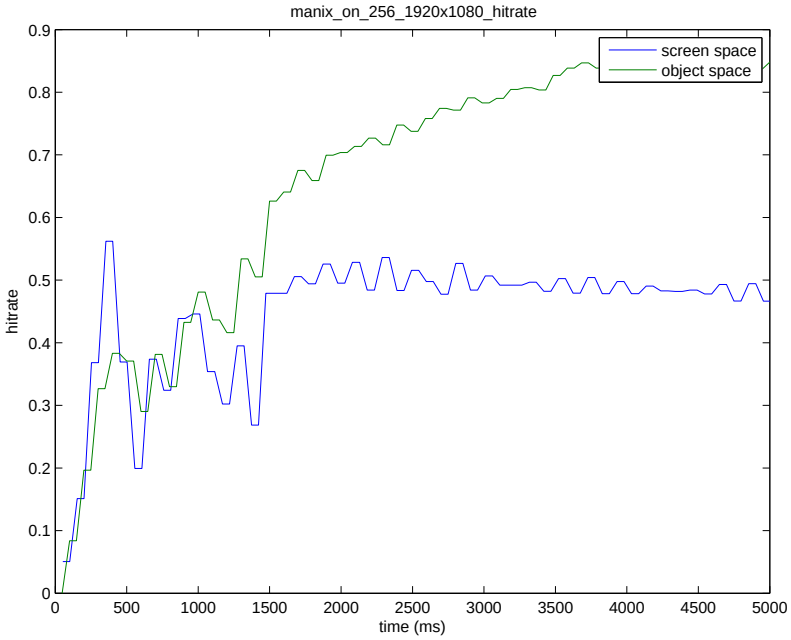


Figure A.79: screen space vs object space: cache hit rate manix, background on, volume res = 256, screen res = 1920x1080, cache hit rate

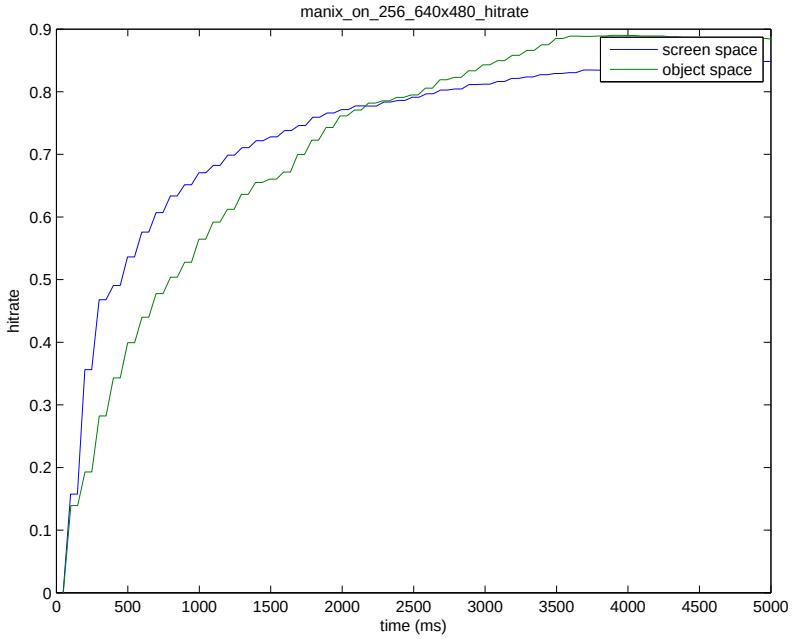


Figure A.80: screen space vs object space: cache hit rate manix, background on, volume res = 256, screen res = 640x480, cache hit rate

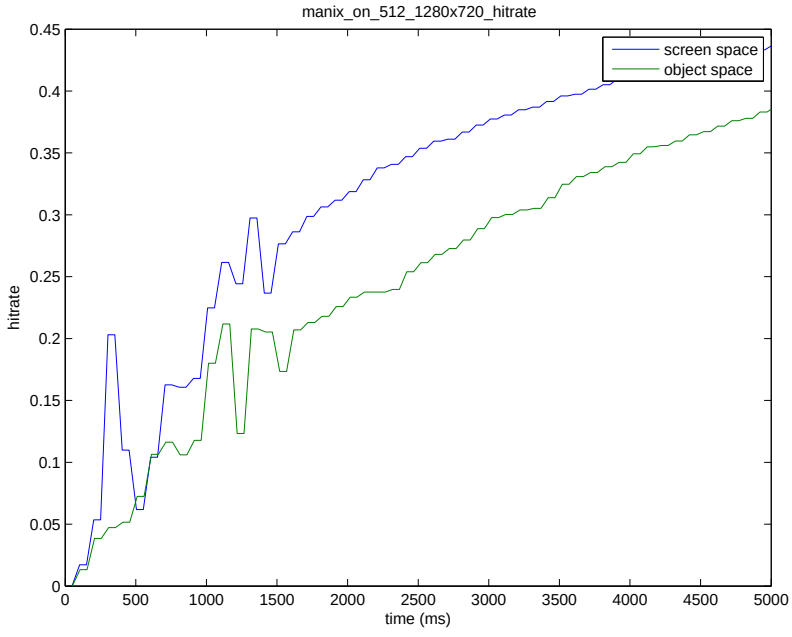


Figure A.81: screen space vs object space: cache hit rate manix, background on, volume res = 512, screen res = 1280x720, cache hit rate

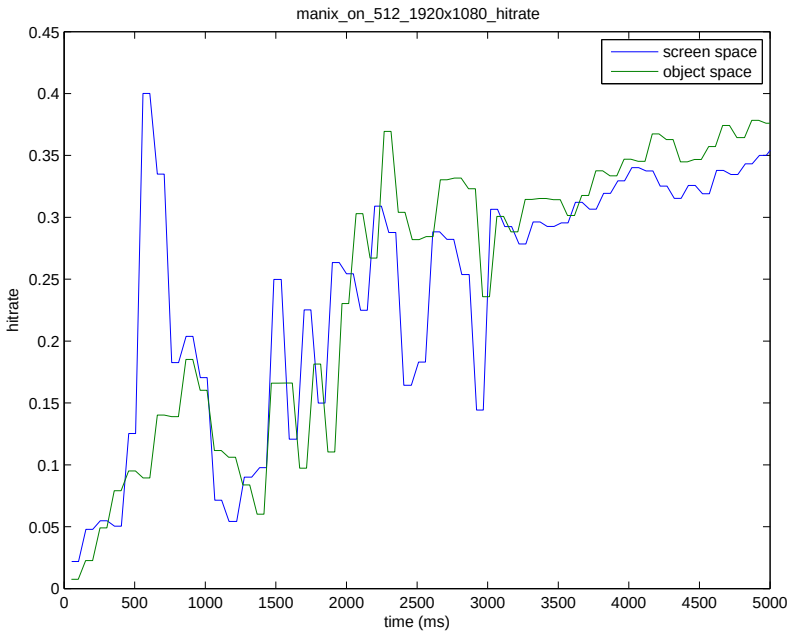


Figure A.82: screen space vs object space: cache hit rate manix, background on, volume res = 512, screen res = 1920x1080, cache hit rate

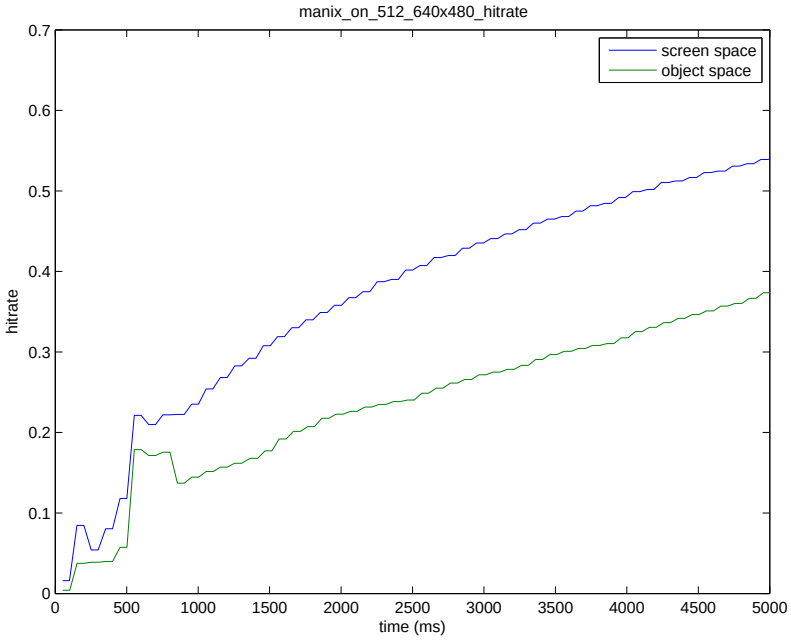


Figure A.83: screen space vs object space: cache hit rate manix, background on, volume res = 512, screen res = 640x480, cache hit rate

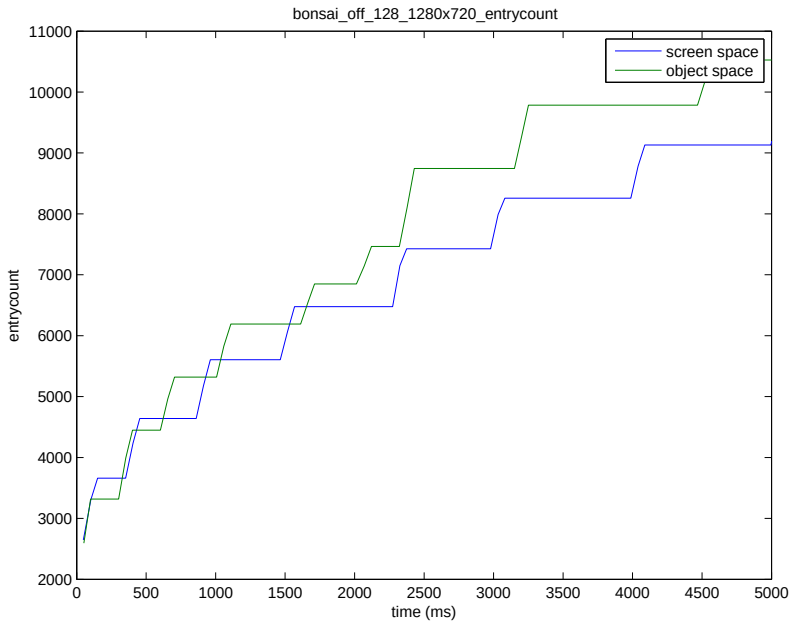


Figure A.84: screen space vs object space: number of entries bonsai, background off, volume res = 128, screen res = 1280x720, entrycount

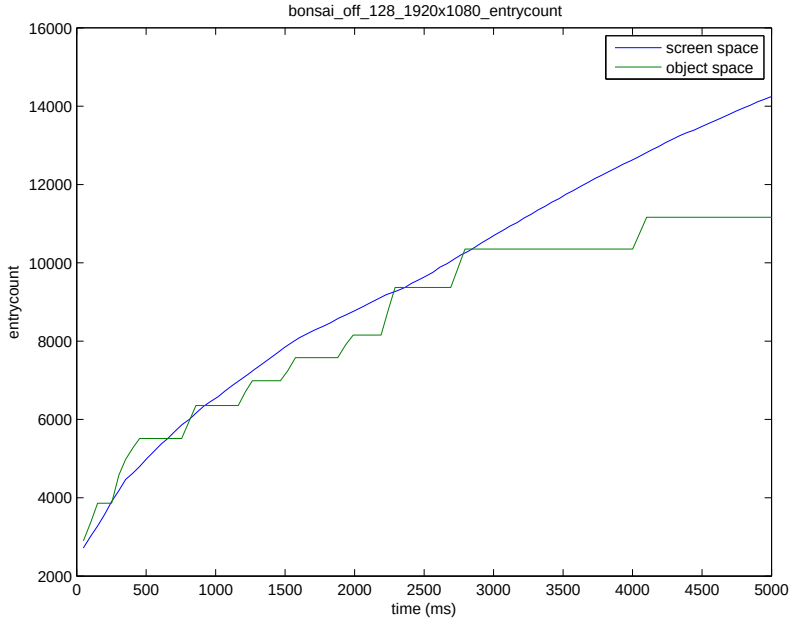


Figure A.85: screen space vs object space: number of entries bonsai, background off, volume res = 128, screen res = 1920x1080, entrycount

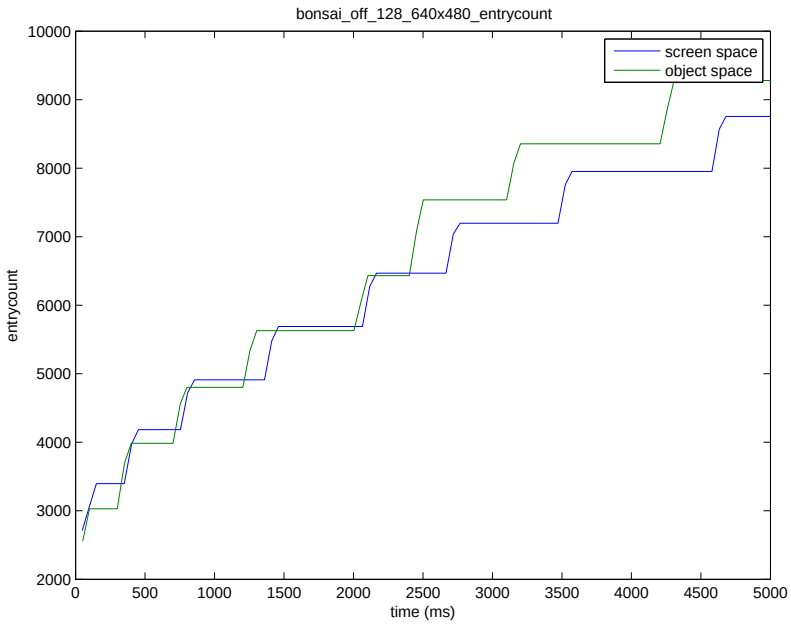


Figure A.86: screen space vs object space: number of entries bonsai, background off, volume res = 128, screen res = 640x480, entrycount

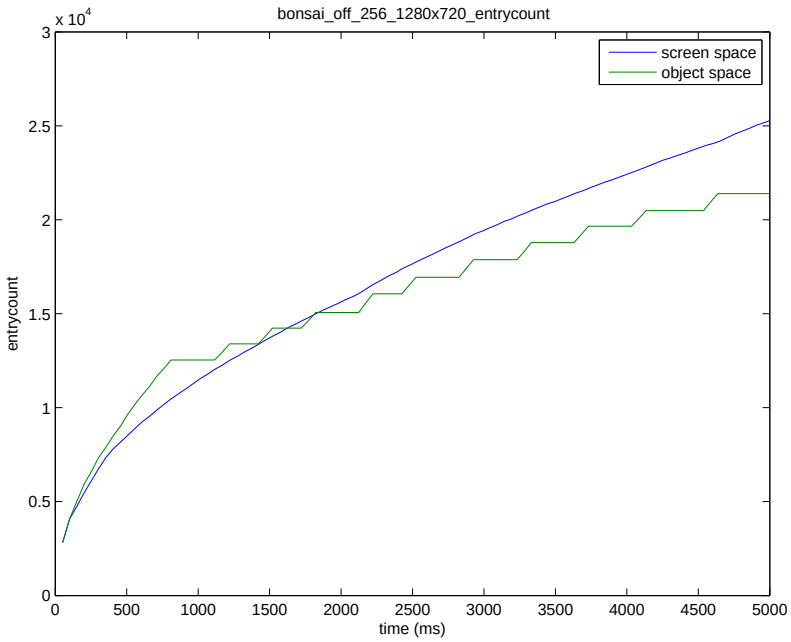


Figure A.87: screen space vs object space: number of entries bonsai, background off, volume res = 256, screen res = 1280x720, entrycount

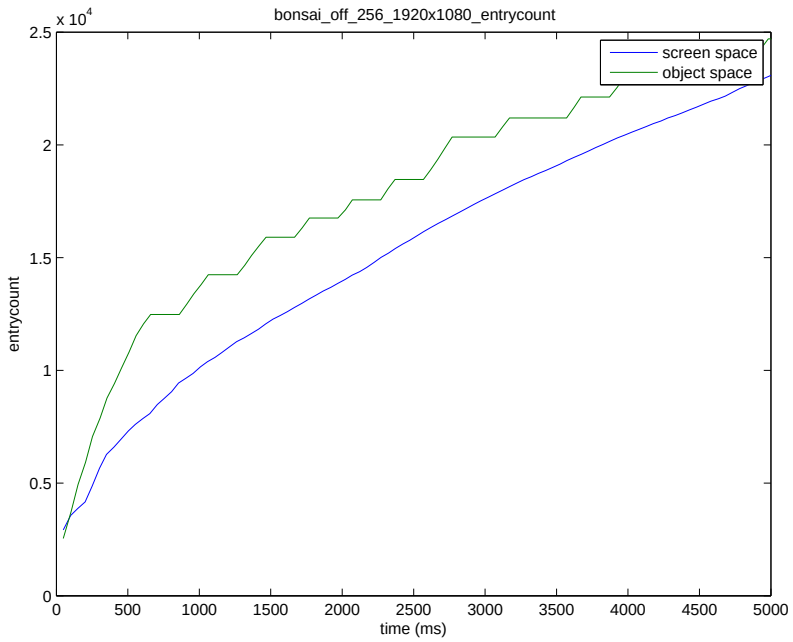


Figure A.88: screen space vs object space: number of entries bonsai, background off, volume res = 256, screen res = 1920x1080, entrycount

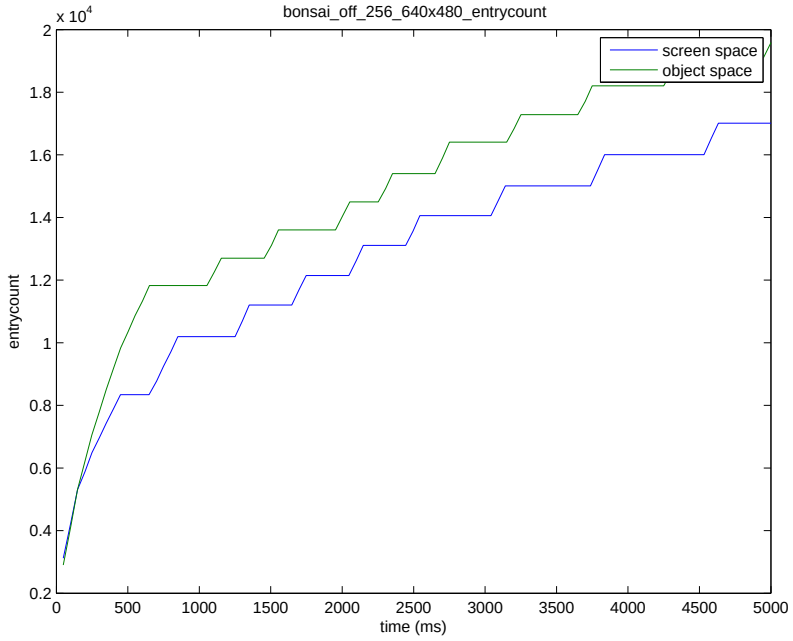


Figure A.89: screen space vs object space: number of entries bonsai, background off, volume res = 256, screen res = 640x480, entrycount

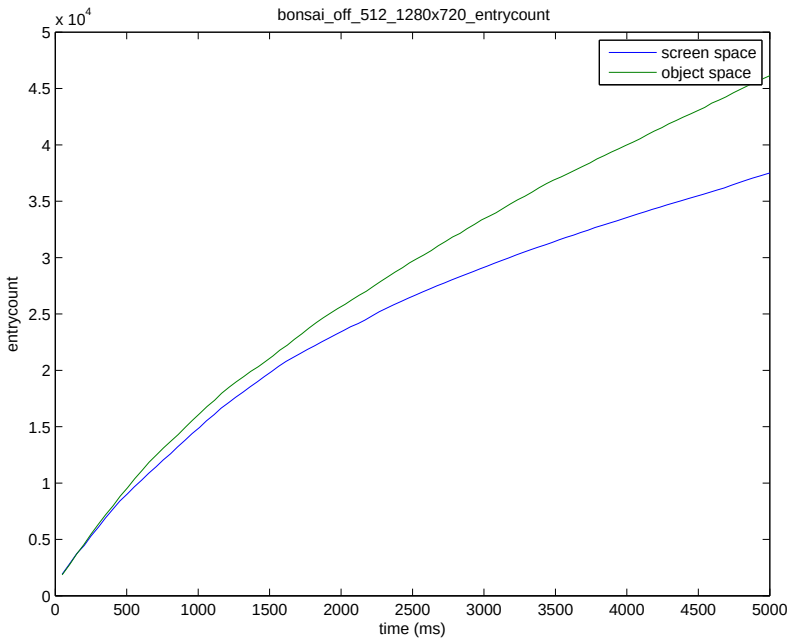


Figure A.90: screen space vs object space: number of entries bonsai, background off, volume res = 512, screen res = 1280x720, entrycount

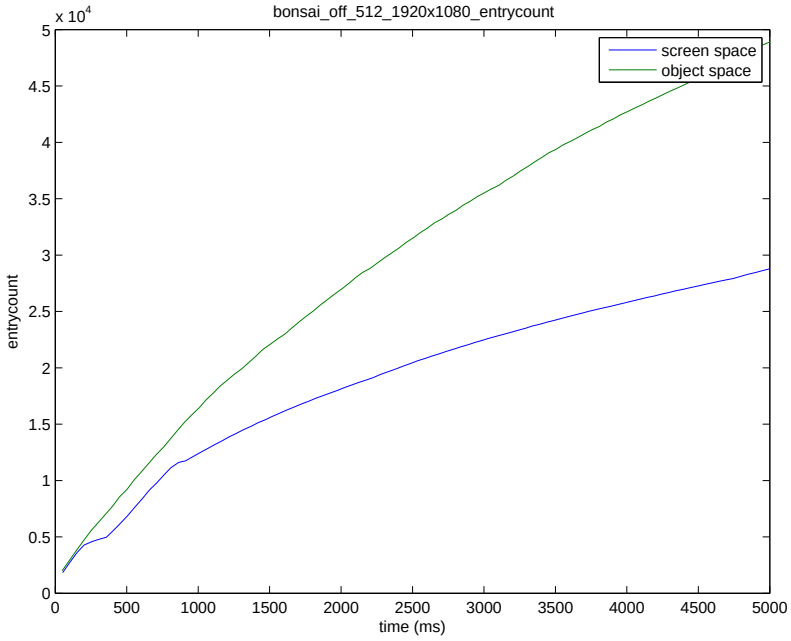


Figure A.91: screen space vs object space: number of entries bonsai, background off, volume res = 512, screen res = 1920x1080, entrycount

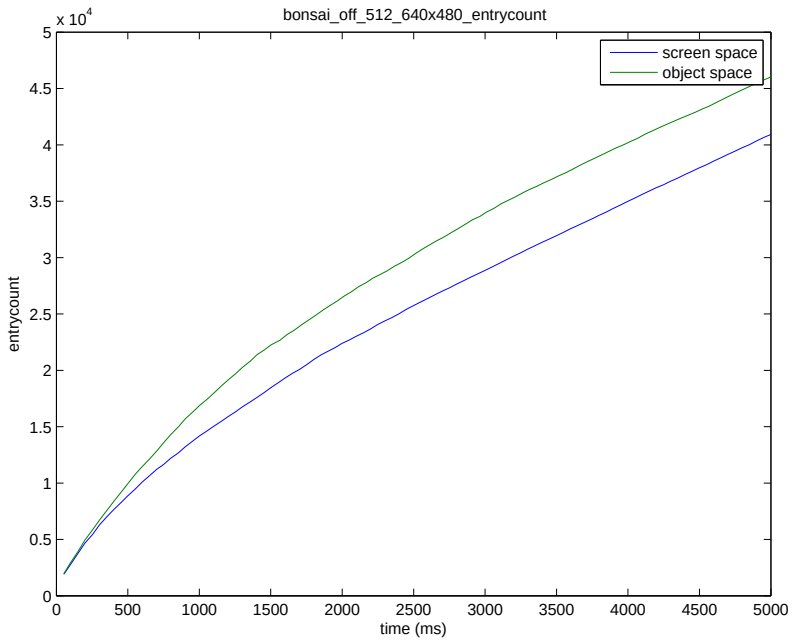


Figure A.92: screen space vs object space: number of entries bonsai, background off, volume res = 512, screen res = 640x480, entrycount

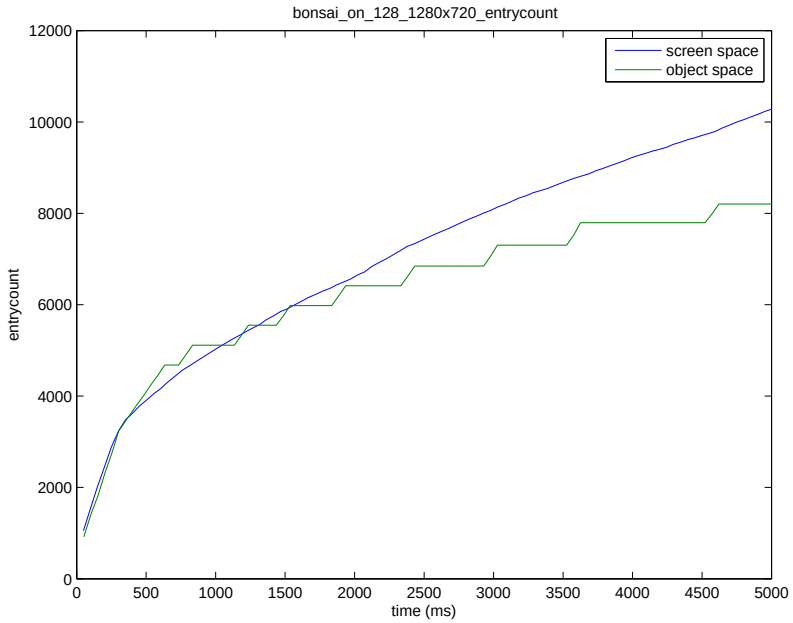


Figure A.93: screen space vs object space: number of entries bonsai, background on, volume res = 128, screen res = 1280x720, entrycount

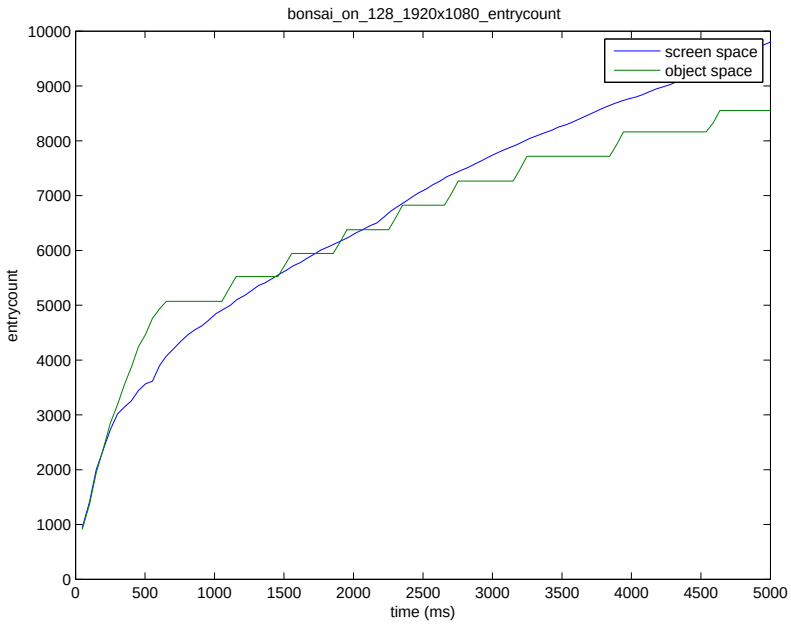


Figure A.94: screen space vs object space: number of entries bonsai, background on, volume res = 128, screen res = 1920x1080, entrycount

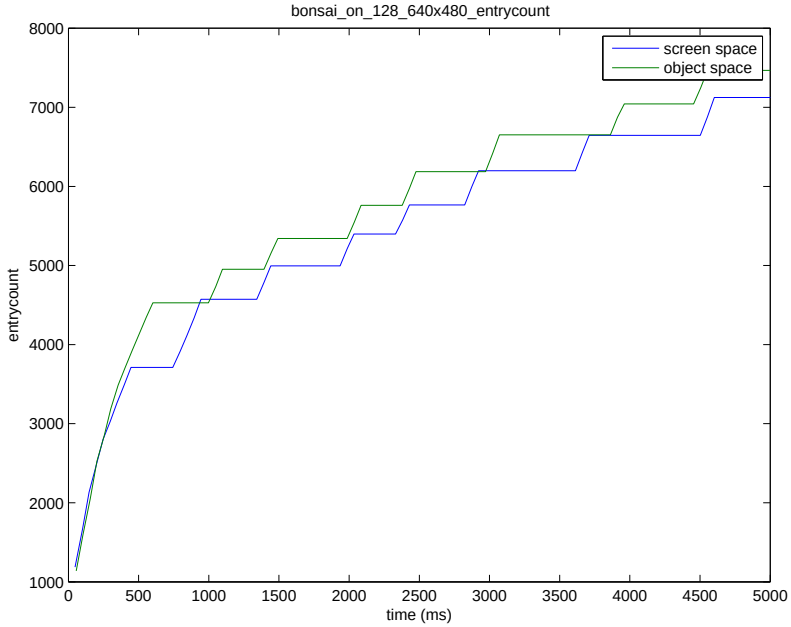


Figure A.95: screen space vs object space: number of entries bonsai, background on, volume res = 128, screen res = 640x480, entrycount

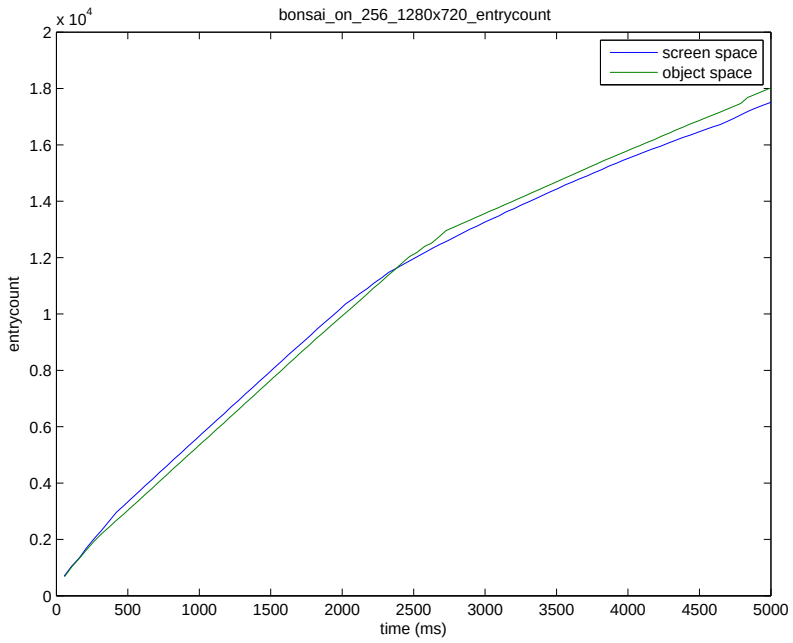


Figure A.96: screen space vs object space: number of entries bonsai, background on, volume res = 256, screen res = 1280x720, entrycount

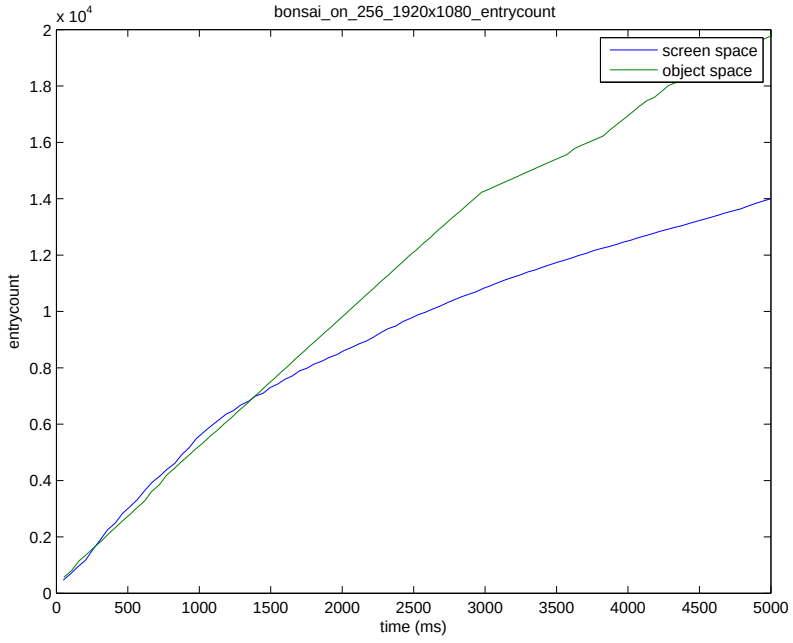


Figure A.97: screen space vs object space: number of entries bonsai, background on, volume res = 256, screen res = 1920x1080, entrycount

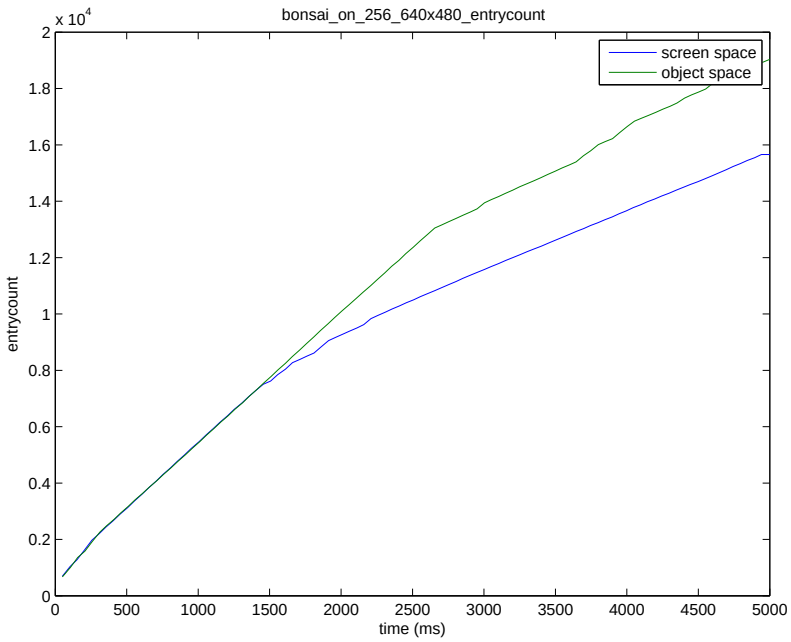


Figure A.98: screen space vs object space: number of entries bonsai, background on, volume res = 256, screen res = 640x480, entrycount

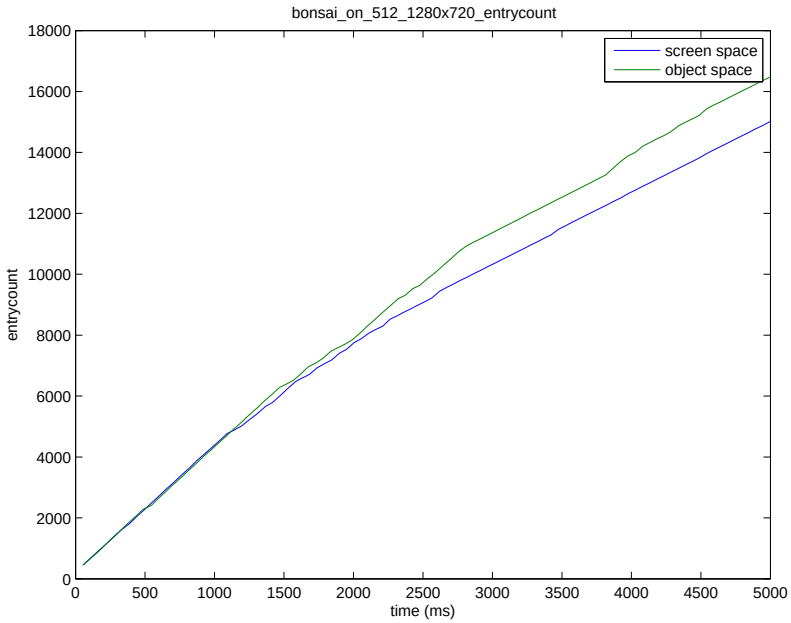


Figure A.99: screen space vs object space: number of entries bonsai, background on, volume res = 512, screen res = 1280x720, entrycount

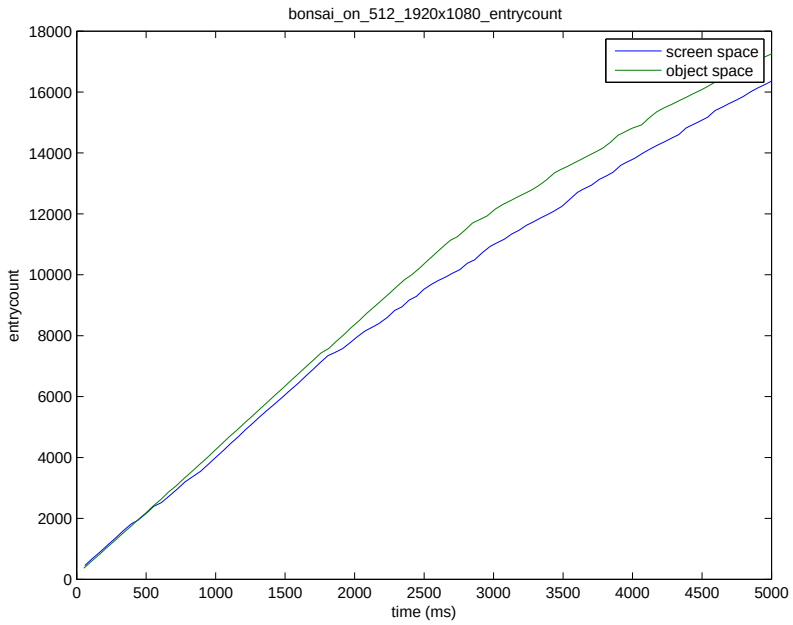


Figure A.100: screen space vs object space: number of entries bonsai, background on, volume res = 512, screen res = 1920x1080, entrycount

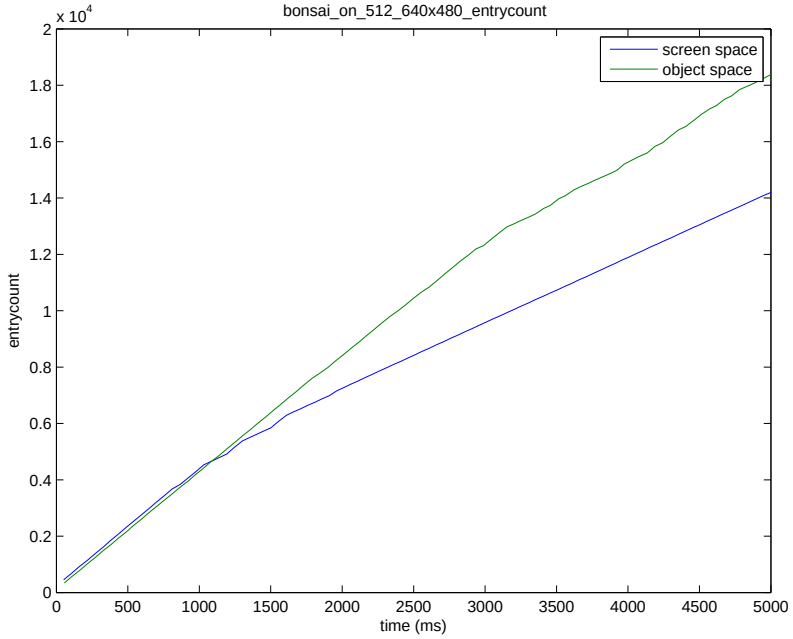


Figure A.101: screen space vs object space: number of entries bonsai, background on, volume res = 512, screen res = 640x480, entrycount

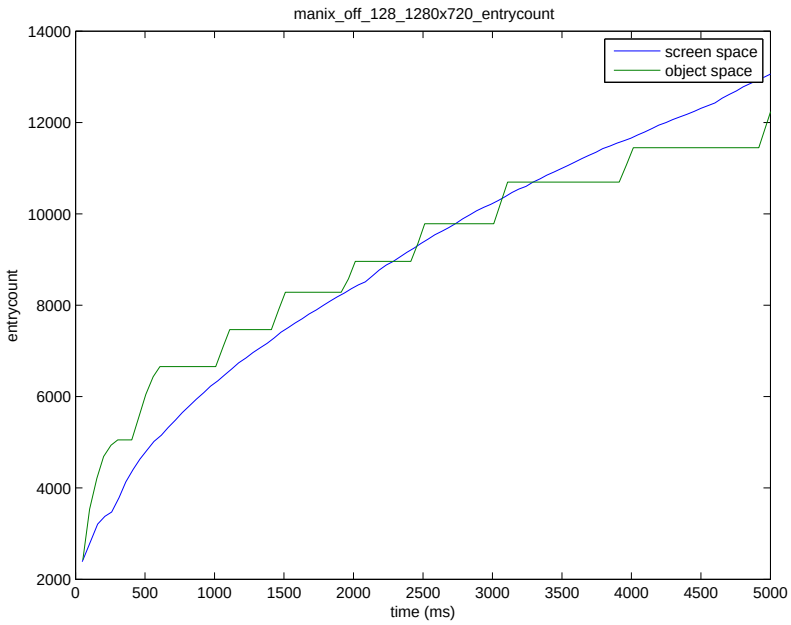


Figure A.102: screen space vs object space: number of entries manix, background off, volume res = 128, screen res = 1280x720, entrycount

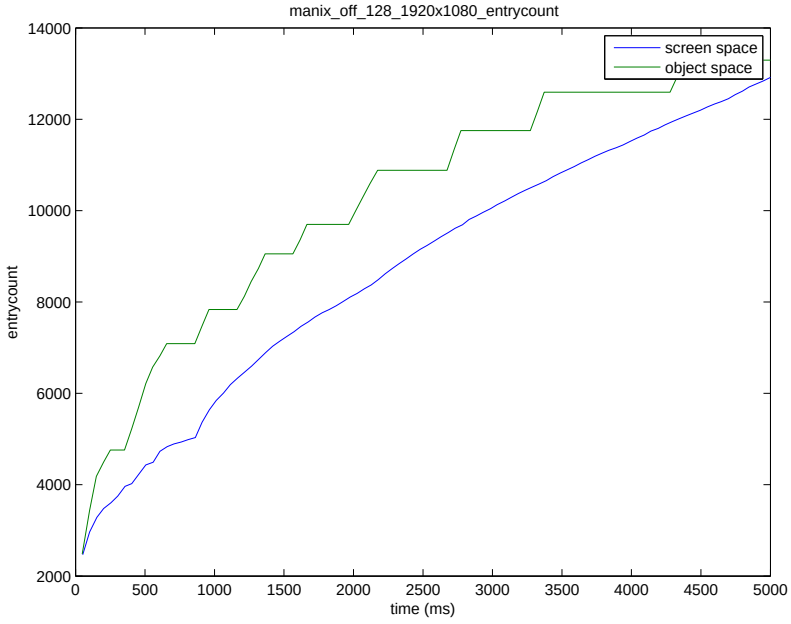


Figure A.103: screen space vs object space: number of entries manix, background off, volume res = 128, screen res = 1920x1080, entrycount

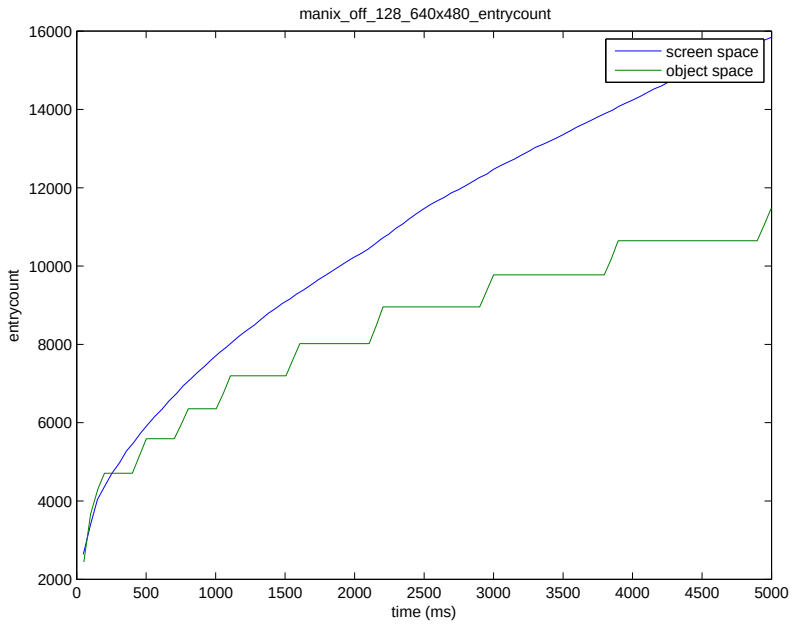


Figure A.104: screen space vs object space: number of entries manix, background off, volume res = 128, screen res = 640x480, entrycount

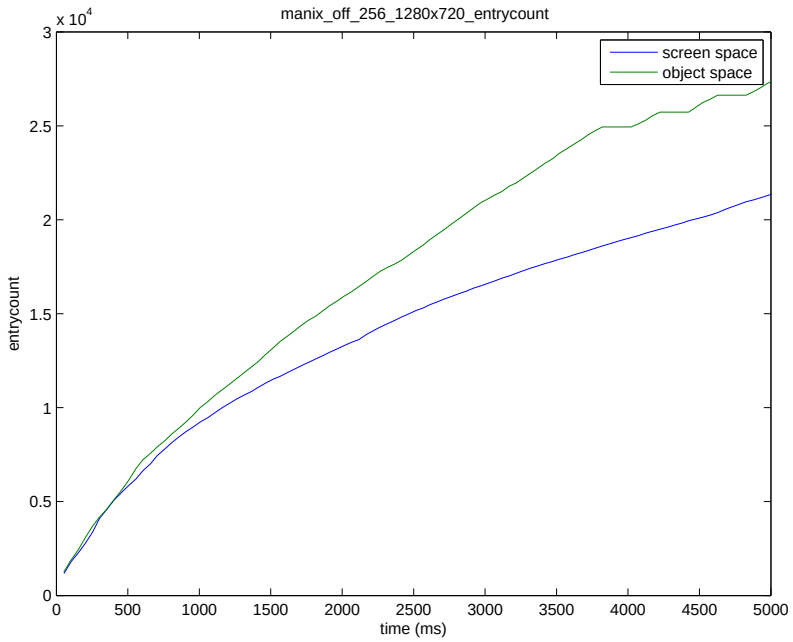


Figure A.105: screen space vs object space: number of entries manix, background off, volume res = 256, screen res = 1280x720, entrycount

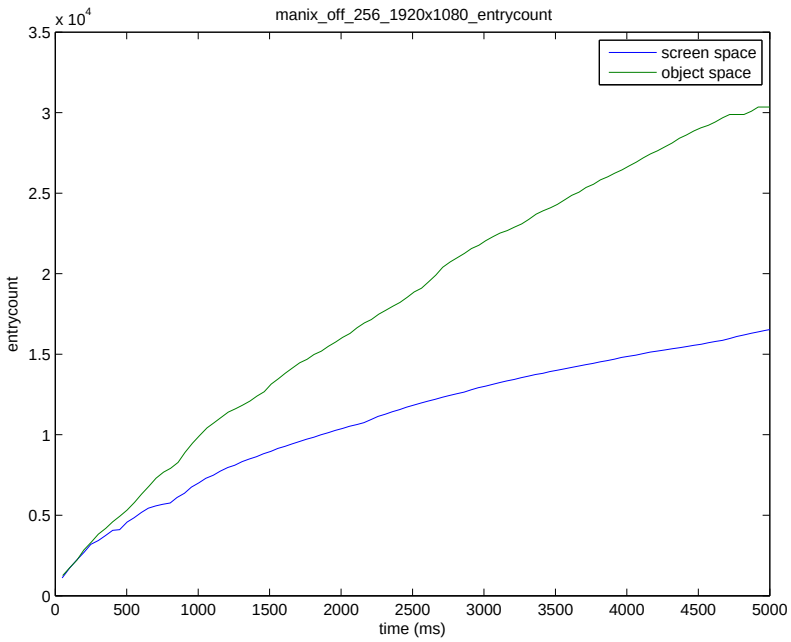


Figure A.106: screen space vs object space: number of entries manix, background off, volume res = 256, screen res = 1920x1080, entrycount

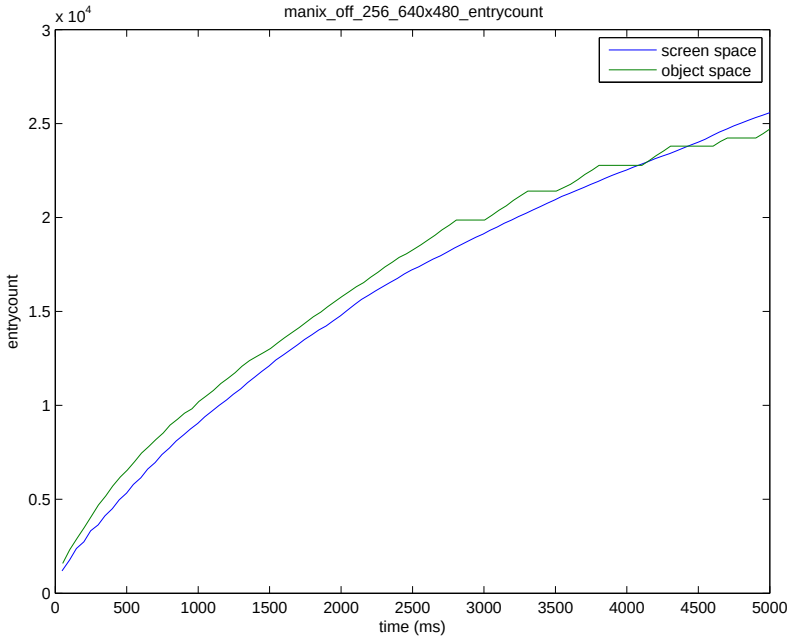


Figure A.107: screen space vs object space: number of entries manix, background off, volume res = 256, screen res = 640x480, entrycount

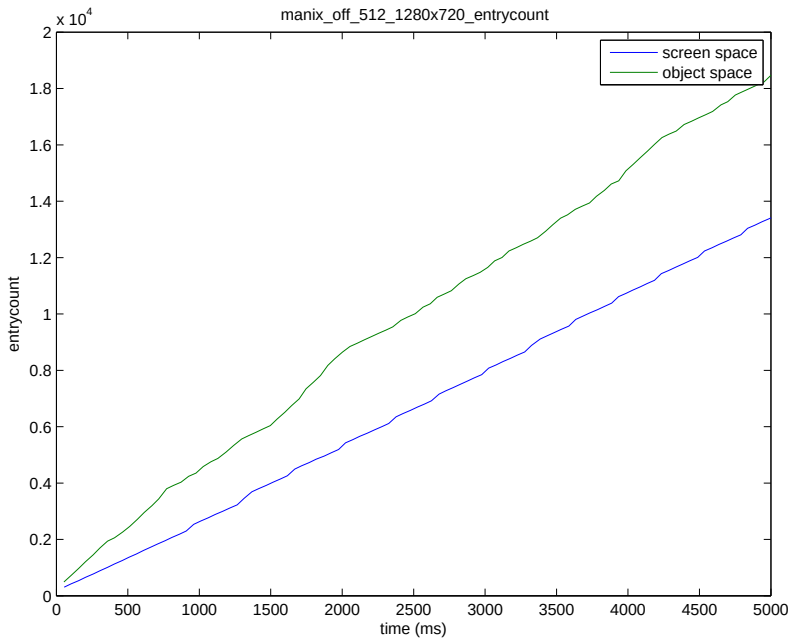


Figure A.108: screen space vs object space: number of entries manix, background off, volume res = 512, screen res = 1280x720, entrycount

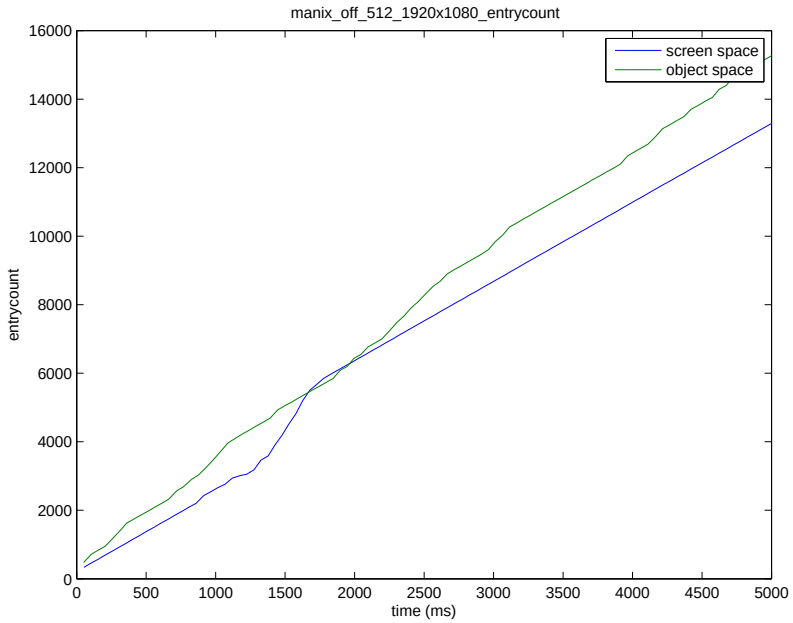


Figure A.109: screen space vs object space: number of entries manix, background off, volume res = 512, screen res = 1920x1080, entrycount

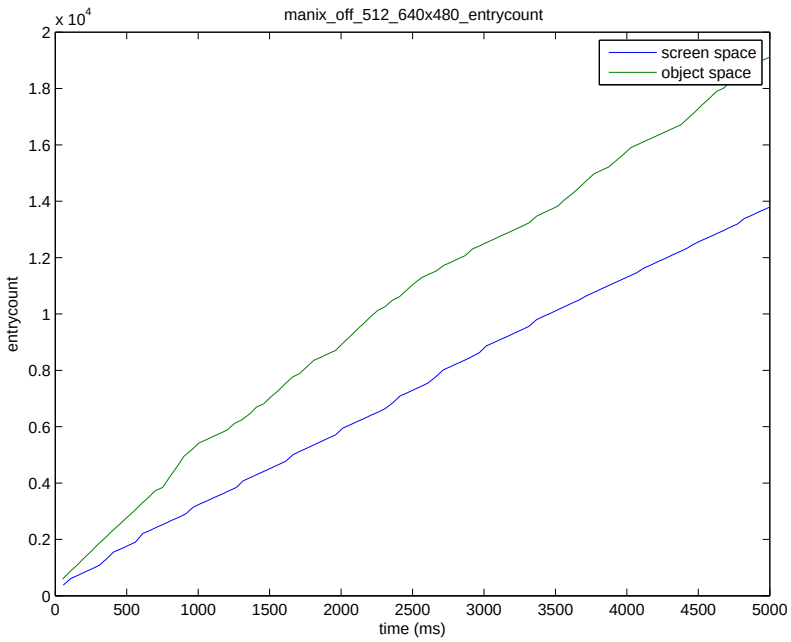


Figure A.110: screen space vs object space: number of entries manix, background off, volume res = 512, screen res = 640x480, entrycount

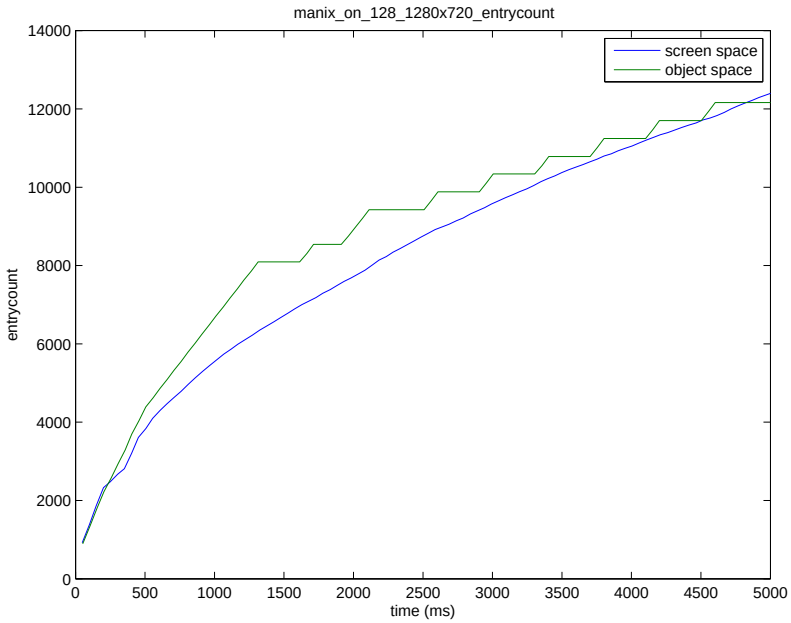


Figure A.111: screen space vs object space: number of entries manix, background on, volume res = 128, screen res = 1280x720, entrycount

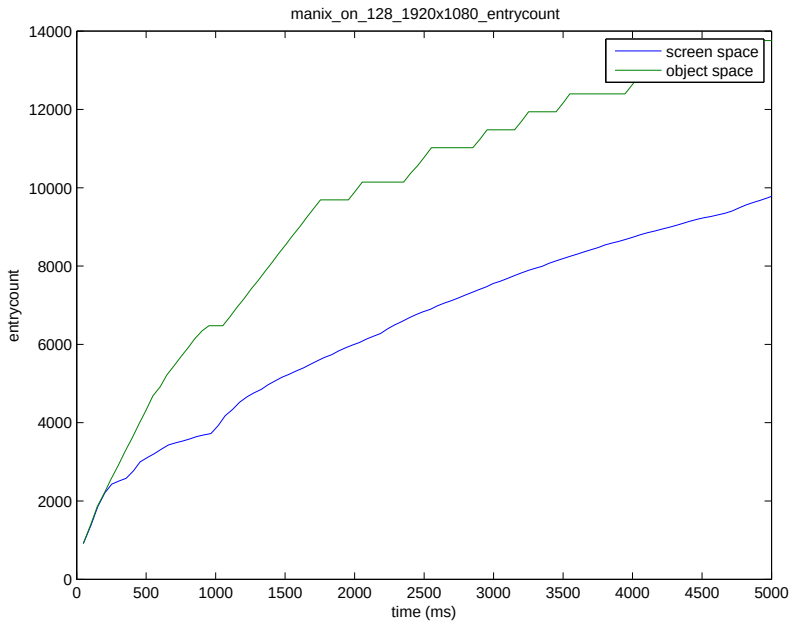


Figure A.112: screen space vs object space: number of entries manix, background on, volume res = 128, screen res = 1920x1080, entrycount

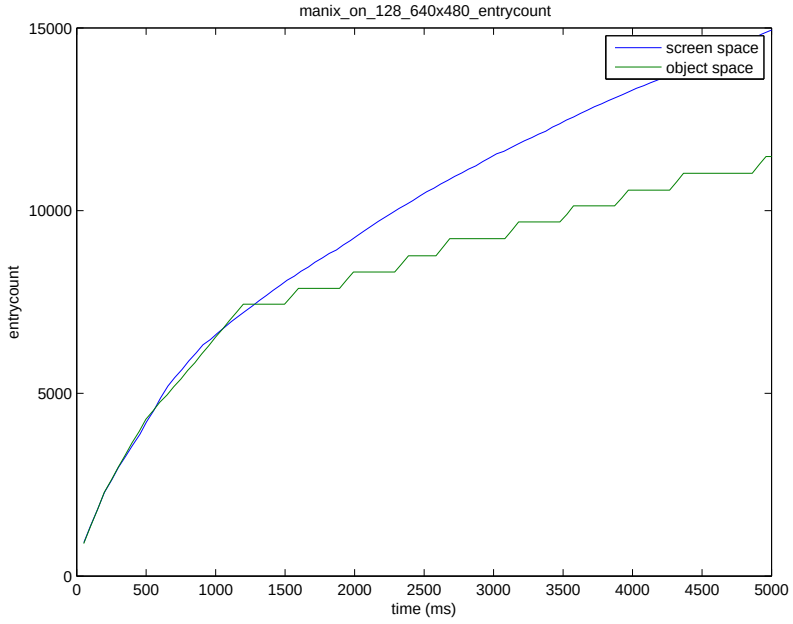


Figure A.113: screen space vs object space: number of entries manix, background on, volume res = 128, screen res = 640x480, entrycount

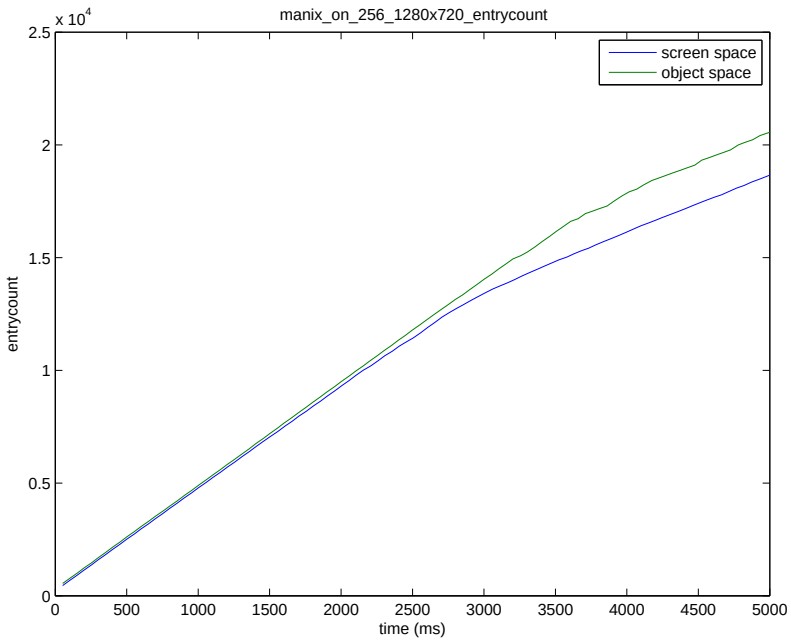


Figure A.114: screen space vs object space: number of entries manix, background on, volume res = 256, screen res = 1280x720, entrycount

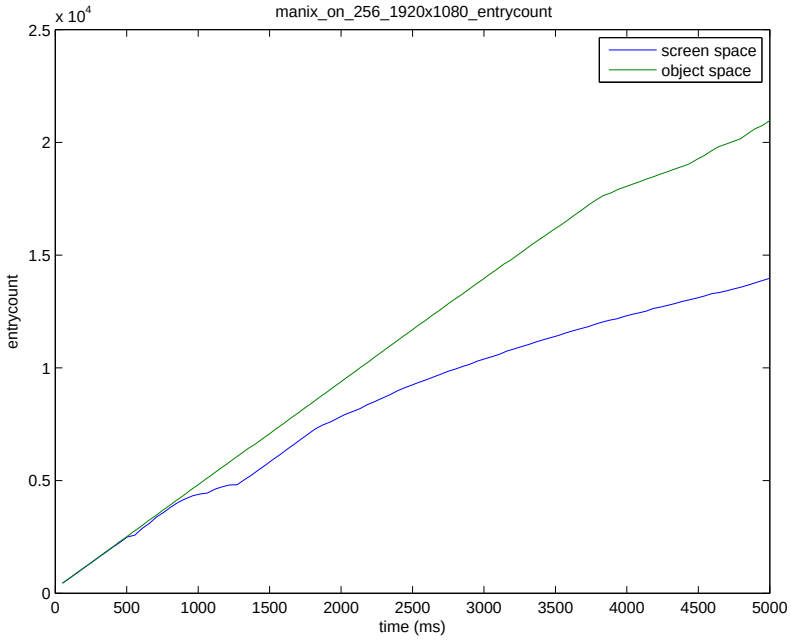


Figure A.115: screen space vs object space: number of entries manix, background on, volume res = 256, screen res = 1920x1080, entrycount

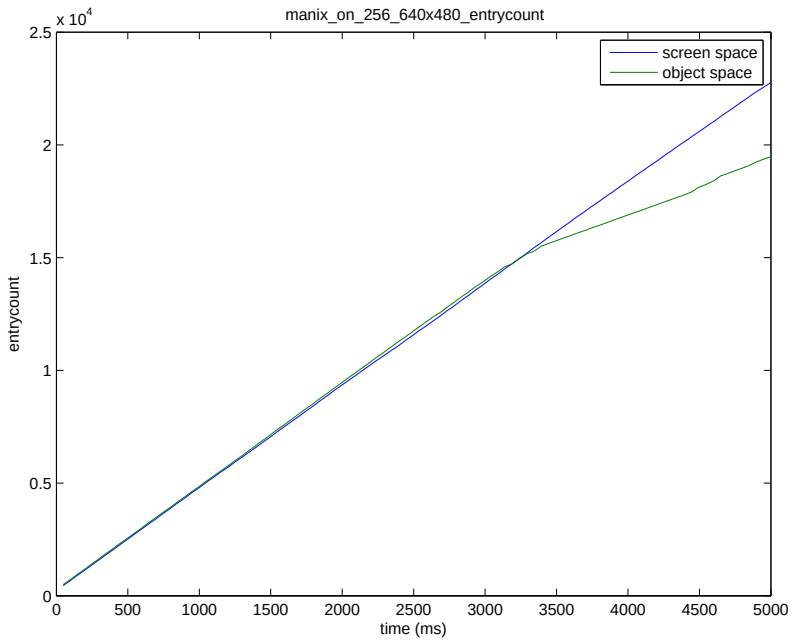


Figure A.116: screen space vs object space: number of entries manix, background on, volume res = 256, screen res = 640x480, entrycount

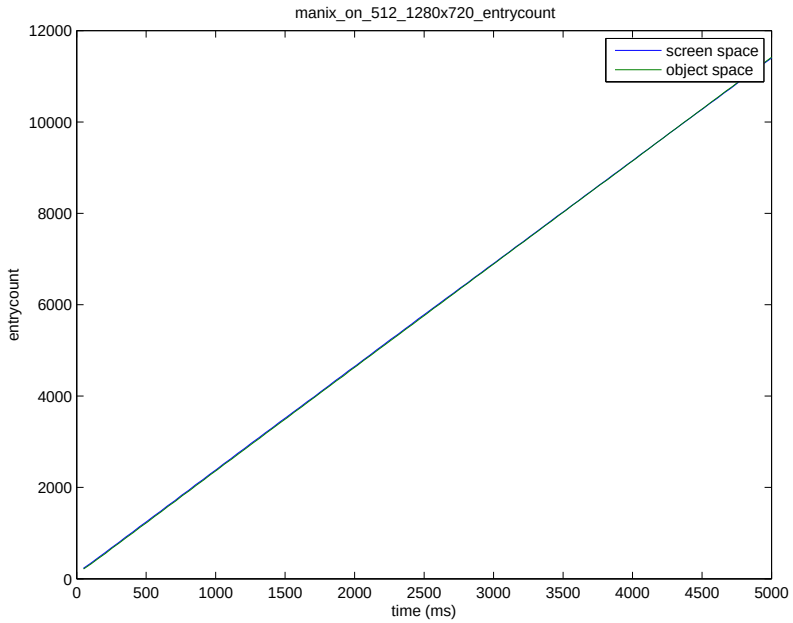


Figure A.117: screen space vs object space: number of entries manix, background on, volume res = 512, screen res = 1280x720, entrycount

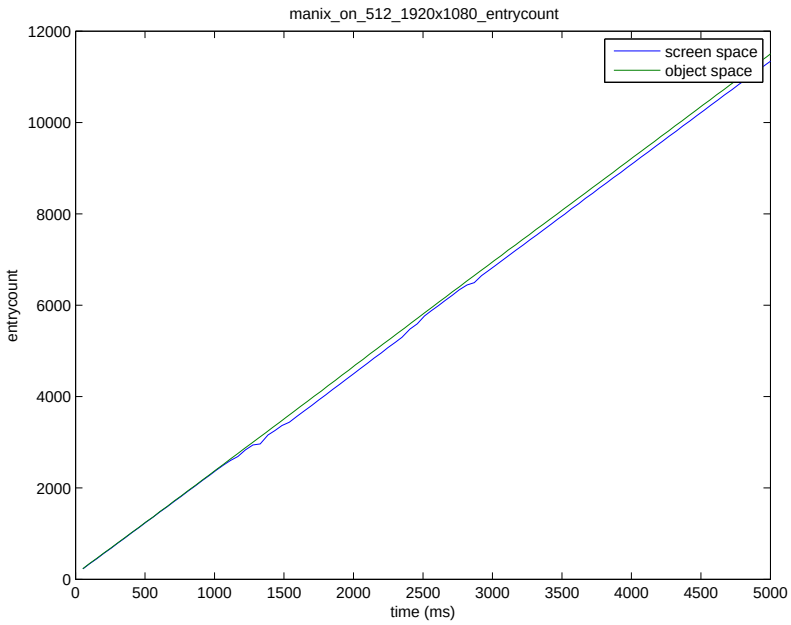


Figure A.118: screen space vs object space: number of entries manix, background on, volume res = 512, screen res = 1920x1080, entrycount

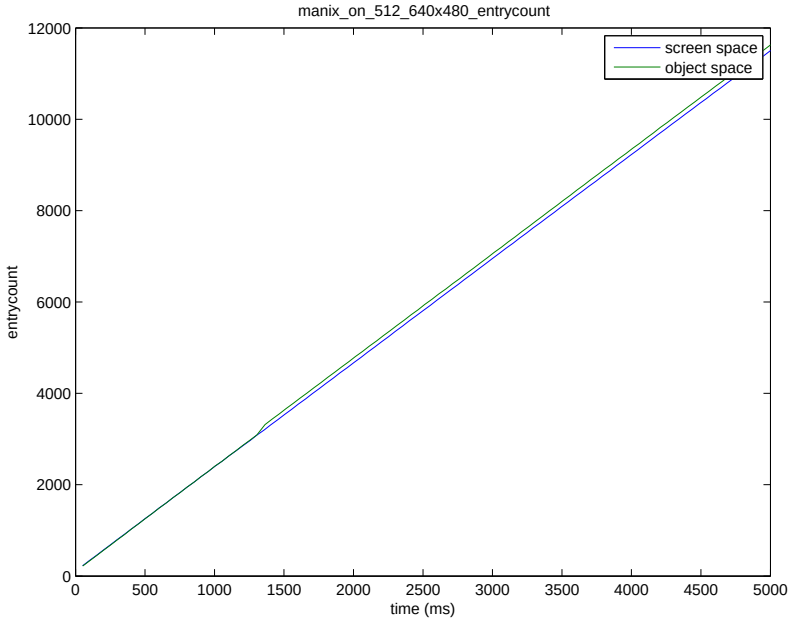


Figure A.119: screen space vs object space: number of entries manix, background on, volume res = 512, screen res = 640x480, entrycount

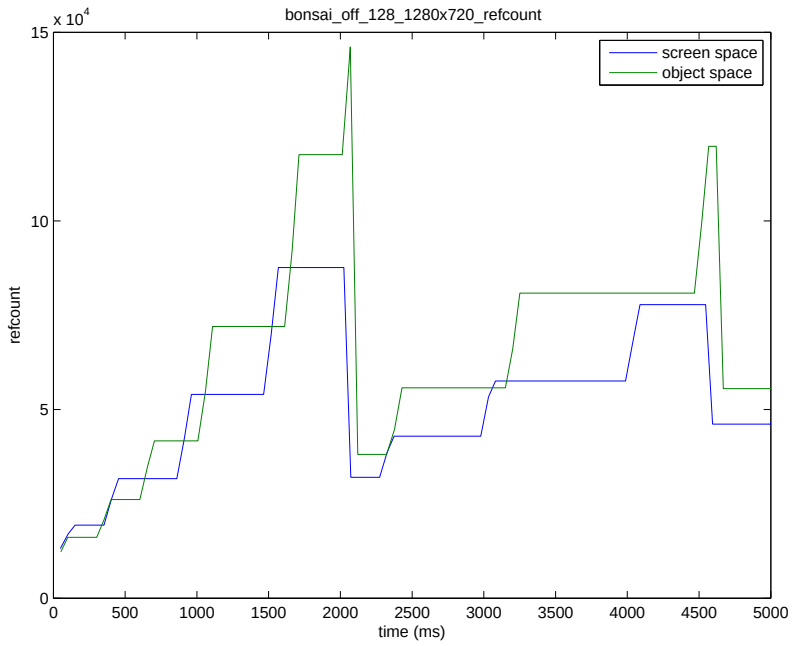


Figure A.120: screen space vs object space: number of references bonsai, background off, volume res = 128, screen res = 1280x720, number of references

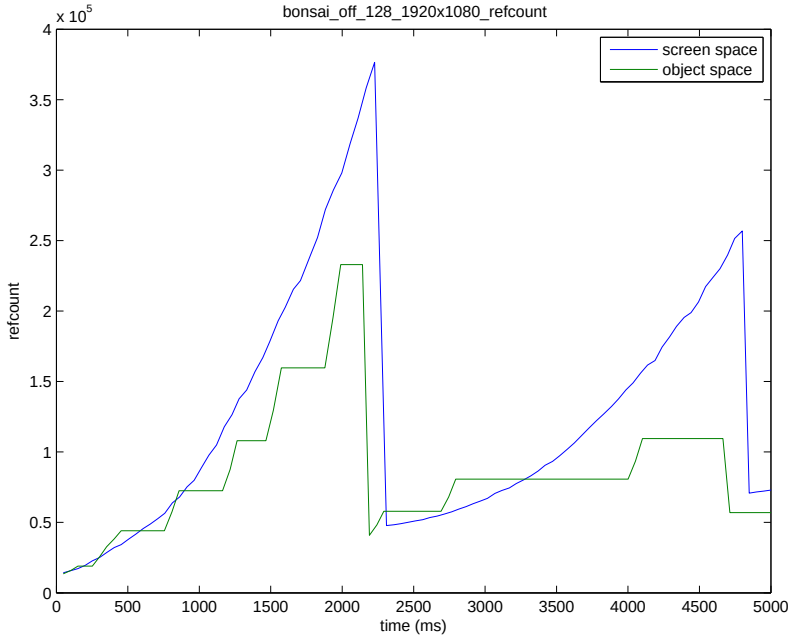


Figure A.121: screen space vs object space: number of references bonsai, background off, volume res = 128, screen res = 1920x1080, number of references

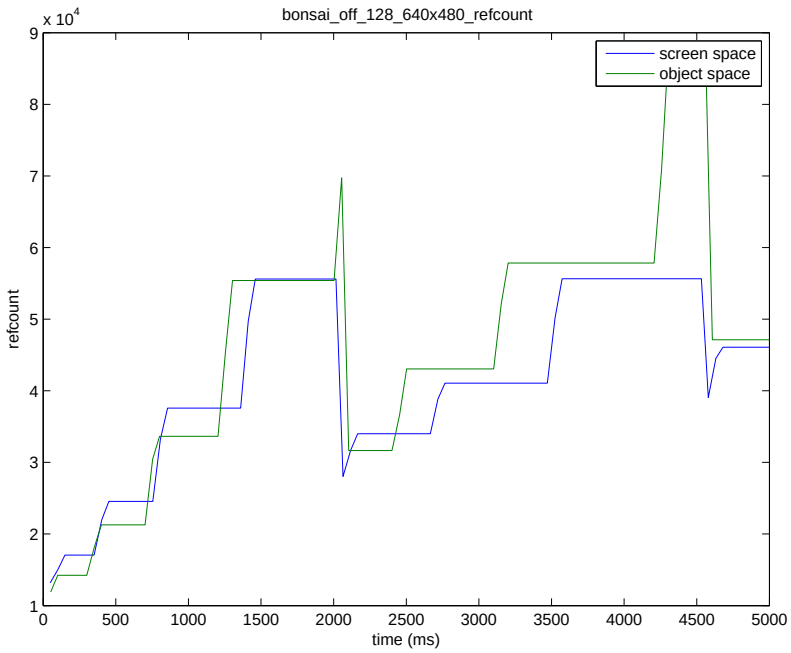


Figure A.122: screen space vs object space: number of references bonsai, background off, volume res = 128, screen res = 640x480, number of references

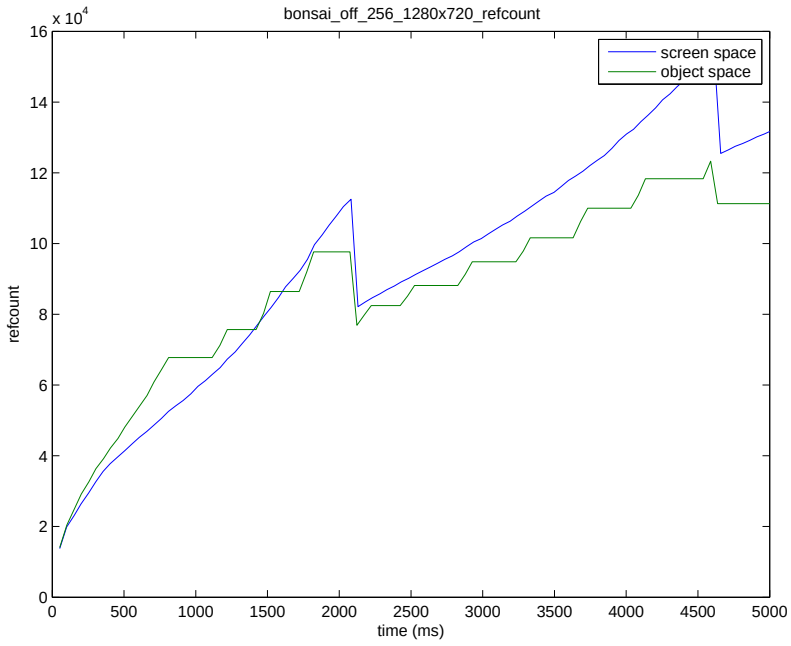


Figure A.123: screen space vs object space: number of references bonsai, background off, volume res = 256, screen res = 1280x720, number of references

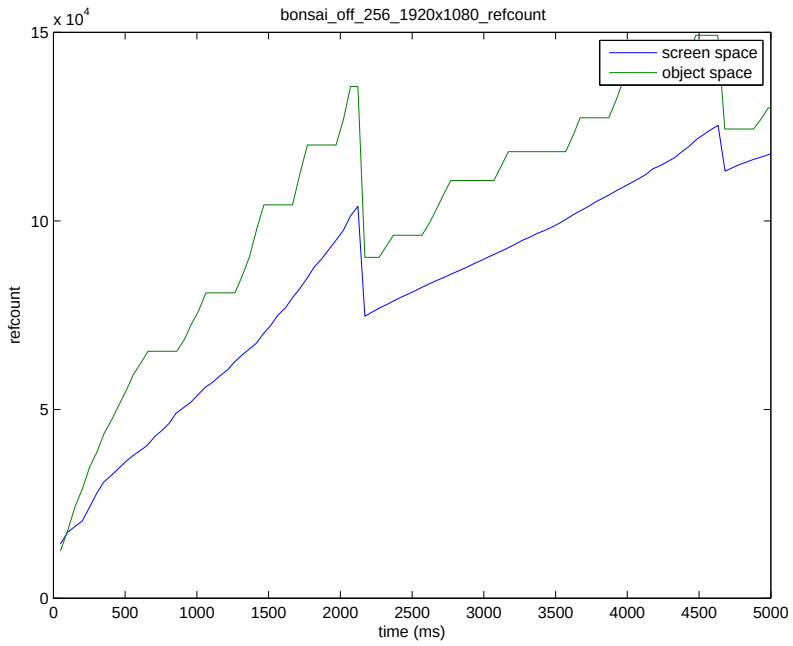


Figure A.124: screen space vs object space: number of references bonsai, background off, volume res = 256, screen res = 1920x1080, number of references

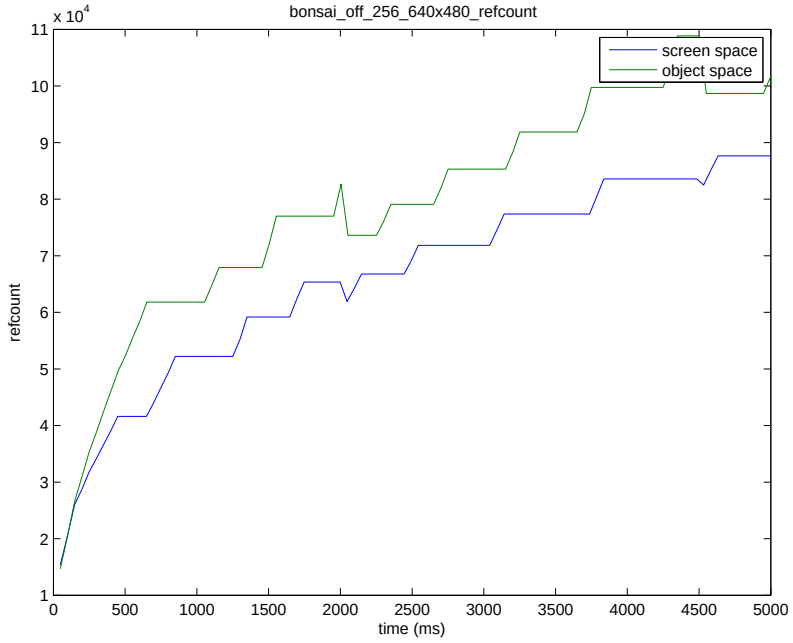


Figure A.125: screen space vs object space: number of references bonsai, background off, volume res = 256, screen res = 640x480, number of references

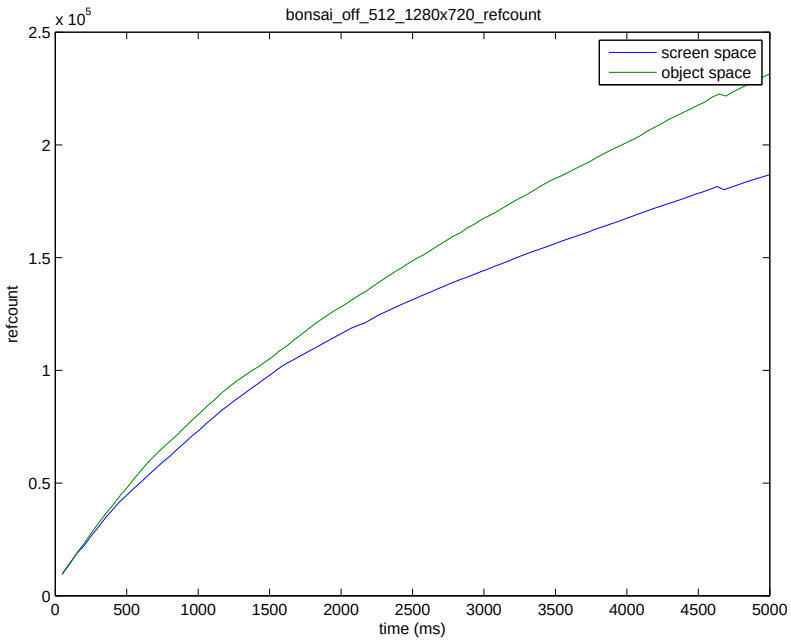


Figure A.126: screen space vs object space: number of references bonsai, background off, volume res = 512, screen res = 1280x720, number of references

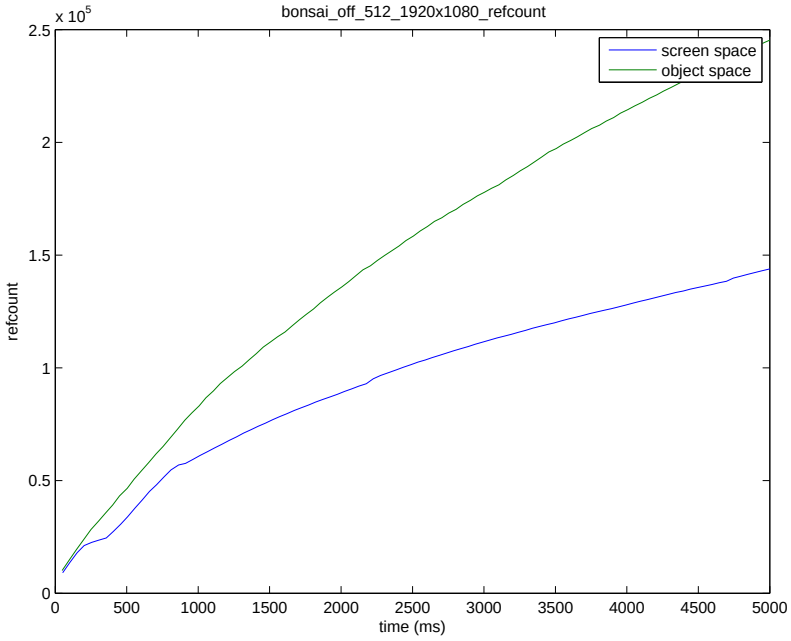


Figure A.127: screen space vs object space: number of references bonsai, background off, volume res = 512, screen res = 1920x1080, number of references

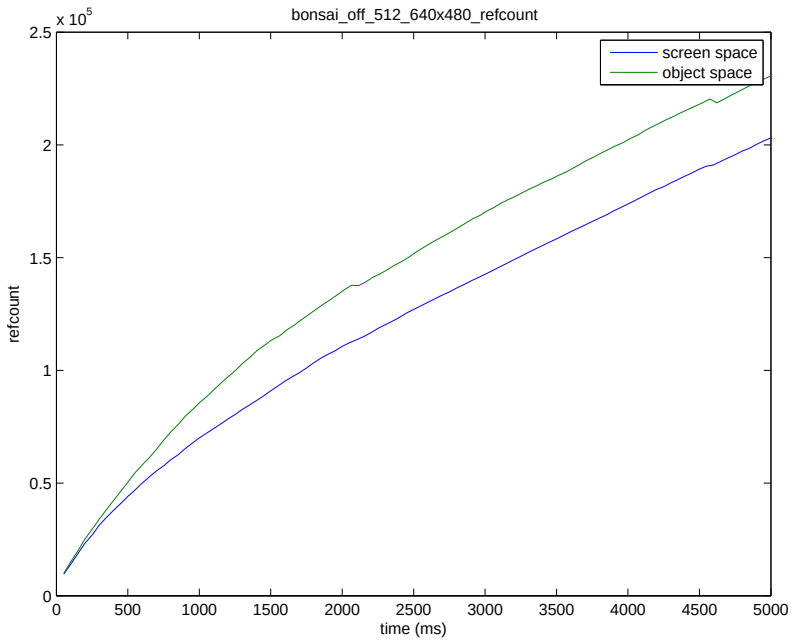


Figure A.128: screen space vs object space: number of references bonsai, background off, volume res = 512, screen res = 640x480, number of references

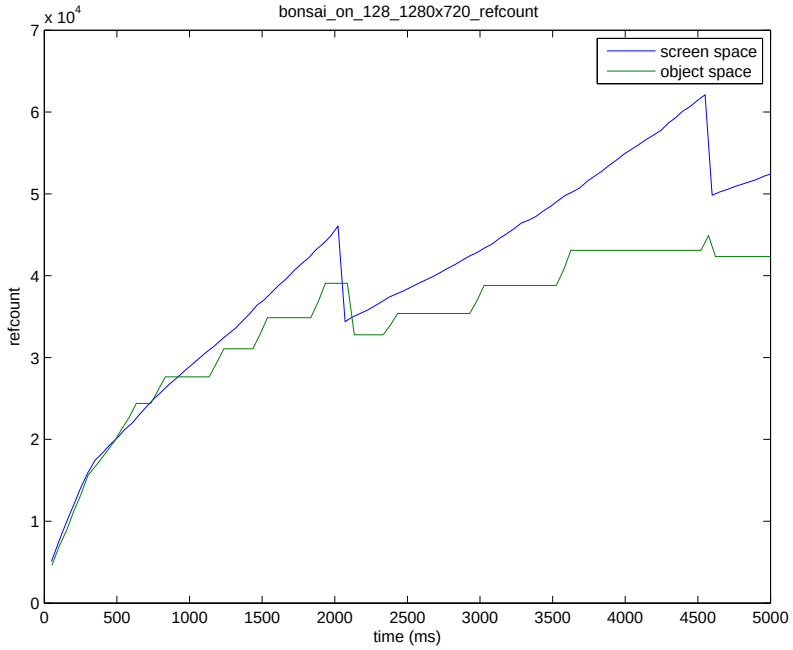


Figure A.129: screen space vs object space: number of references bonsai, background on, volume res = 128, screen res = 1280x720, number of references

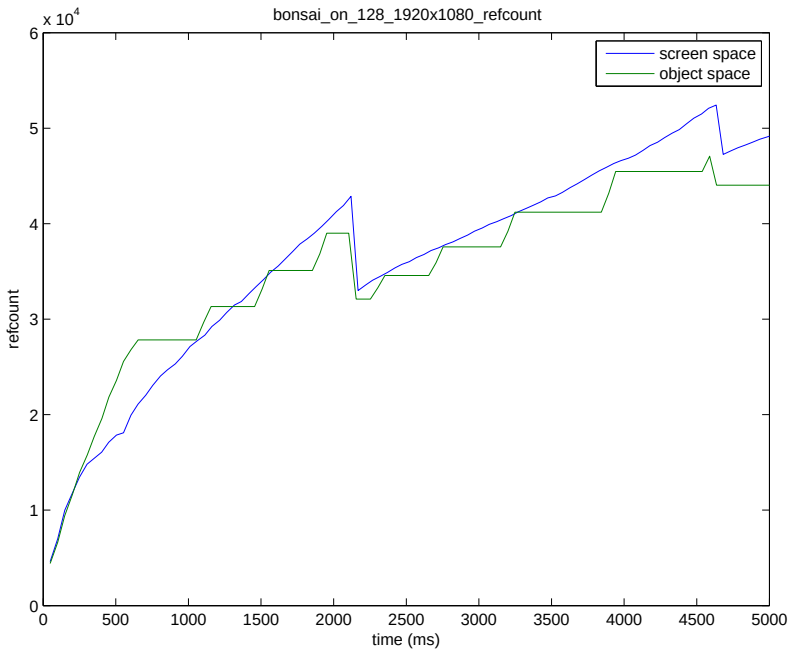


Figure A.130: screen space vs object space: number of references bonsai, background on, volume res = 128, screen res = 1920x1080, number of references

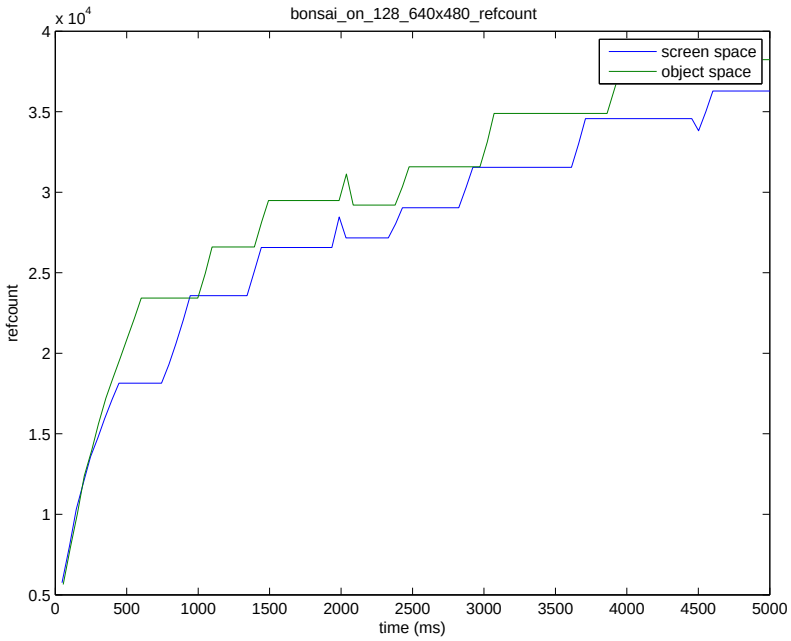


Figure A.131: screen space vs object space: number of references bonsai, background on, volume res = 128, screen res = 640x480, number of references

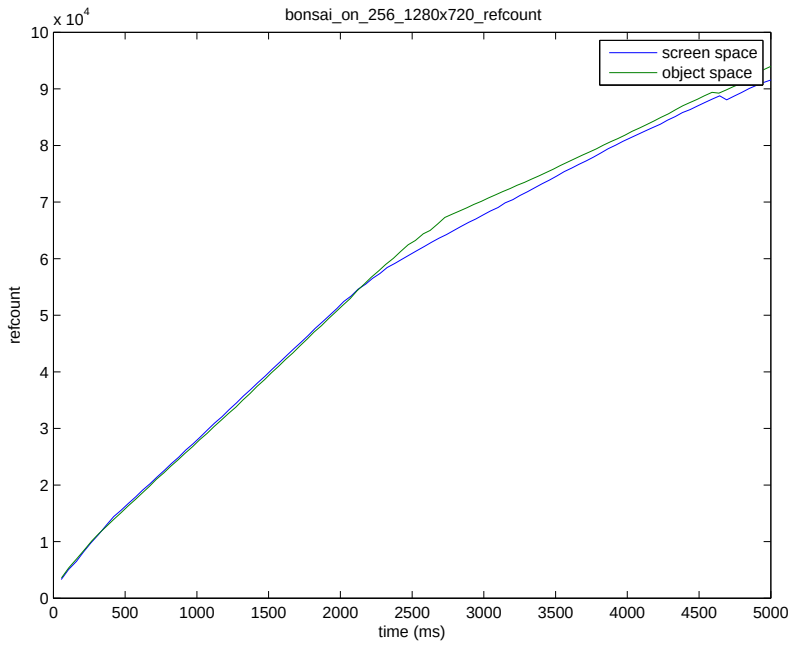


Figure A.132: screen space vs object space: number of references bonsai, background on, volume res = 256, screen res = 1280x720, number of references

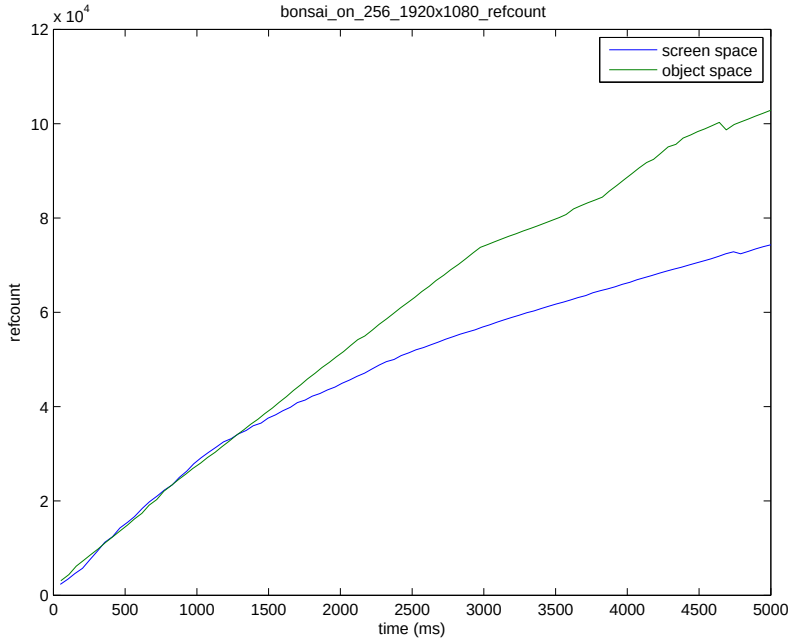


Figure A.133: screen space vs object space: number of references bonsai, background on, volume res = 256, screen res = 1920x1080, number of references

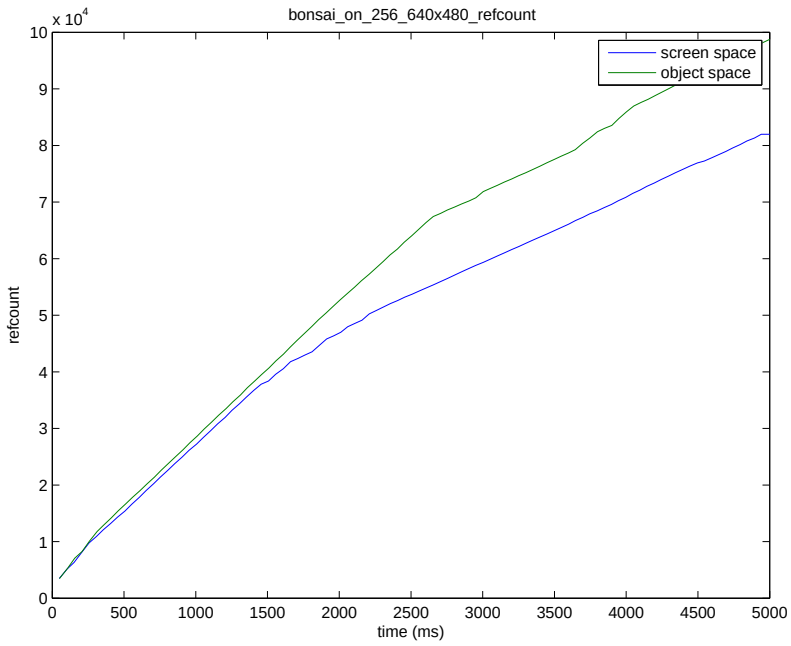


Figure A.134: screen space vs object space: number of references bonsai, background on, volume res = 256, screen res = 640x480, number of references

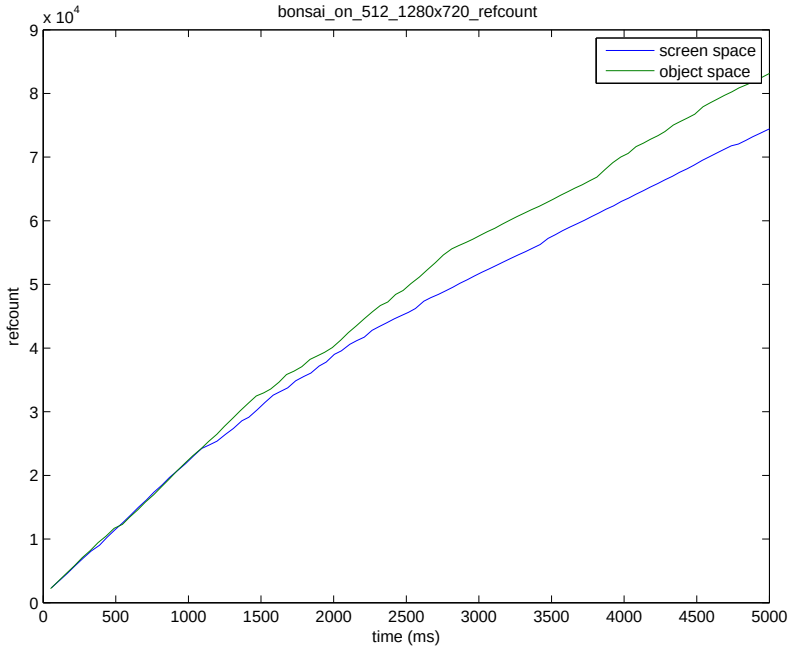


Figure A.135: screen space vs object space: number of references bonsai, background on, volume res = 512, screen res = 1280x720, number of references

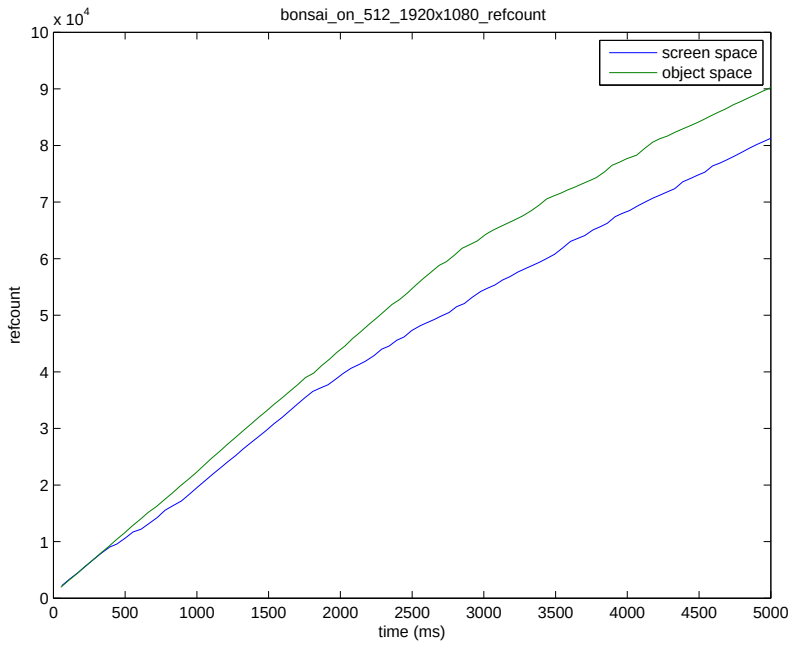


Figure A.136: screen space vs object space: number of references bonsai, background on, volume res = 512, screen res = 1920x1080, number of references

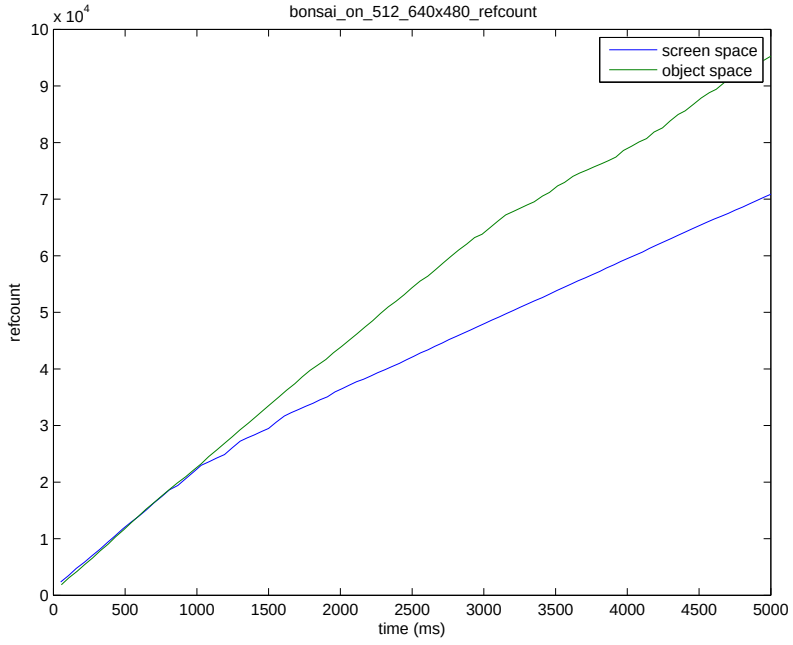


Figure A.137: screen space vs object space: number of references bonsai, background on, volume res = 512, screen res = 640x480, number of references

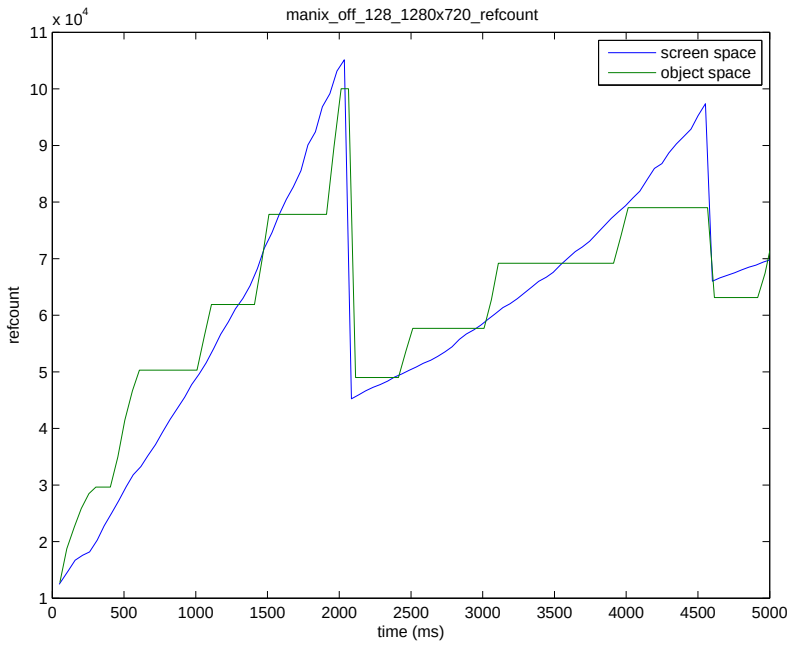


Figure A.138: screen space vs object space: number of references manix, background off, volume res = 128, screen res = 1280x720, number of references

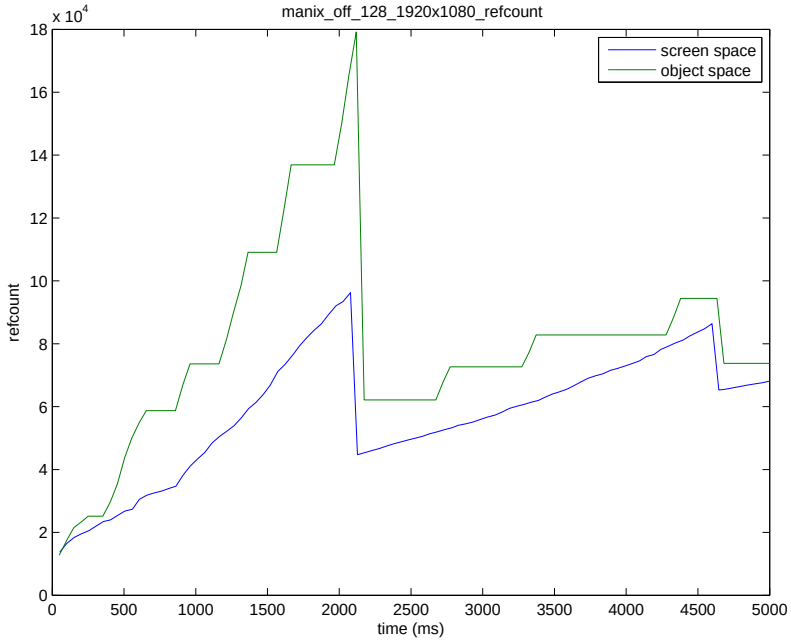


Figure A.139: screen space vs object space: number of references manix, background off, volume res = 128, screen res = 1920x1080, number of references

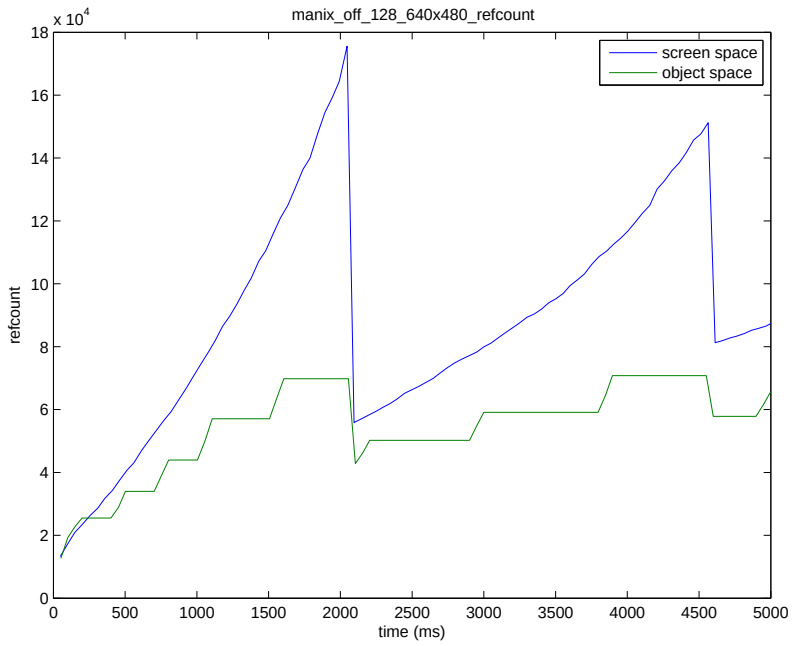


Figure A.140: screen space vs object space: number of references manix, background off, volume res = 128, screen res = 640x480, number of references

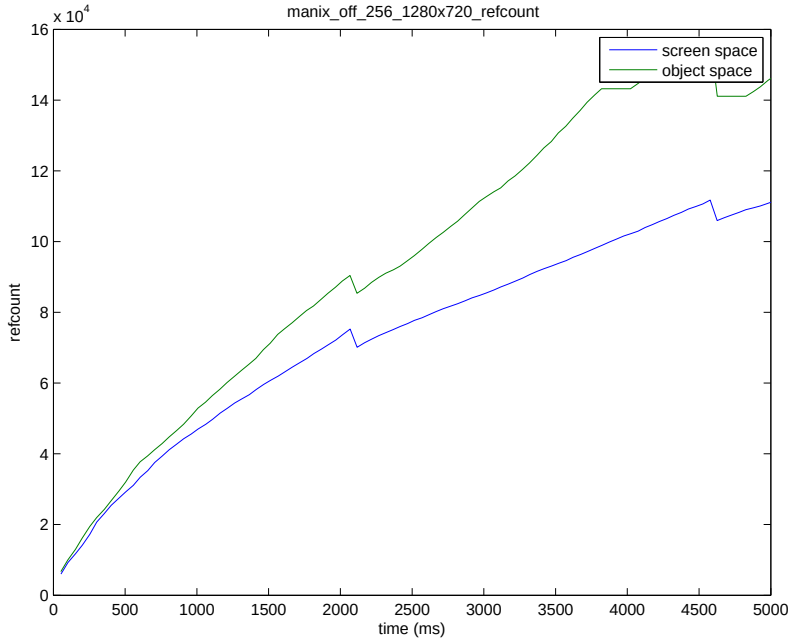


Figure A.141: screen space vs object space: number of references manix, background off, volume res = 256, screen res = 1280x720, number of references

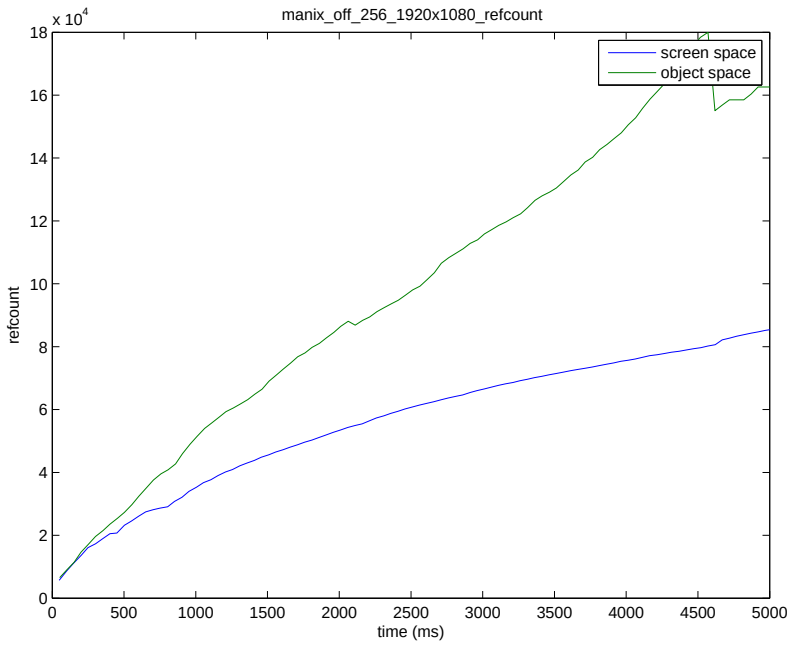


Figure A.142: screen space vs object space: number of references manix, background off, volume res = 256, screen res = 1920x1080, number of references

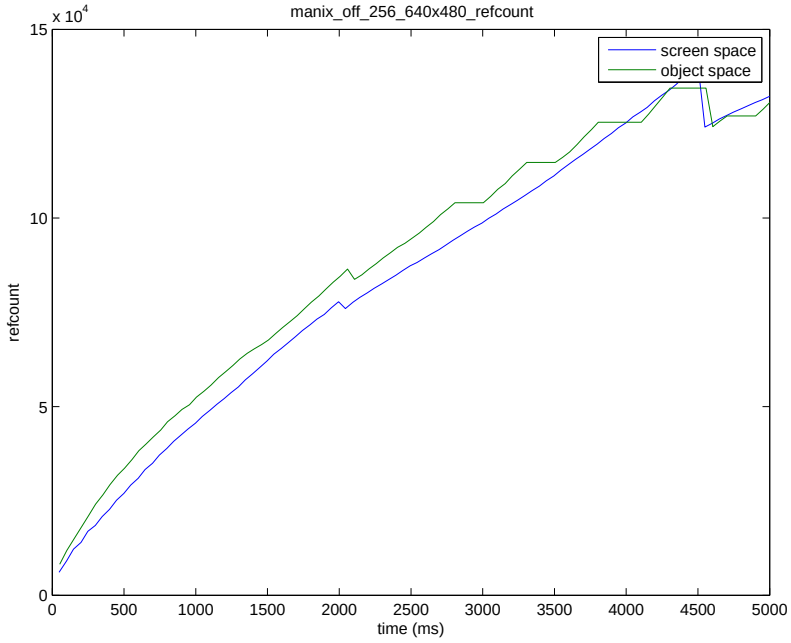


Figure A.143: screen space vs object space: number of references manix, background off, volume res = 256, screen res = 640x480, number of references

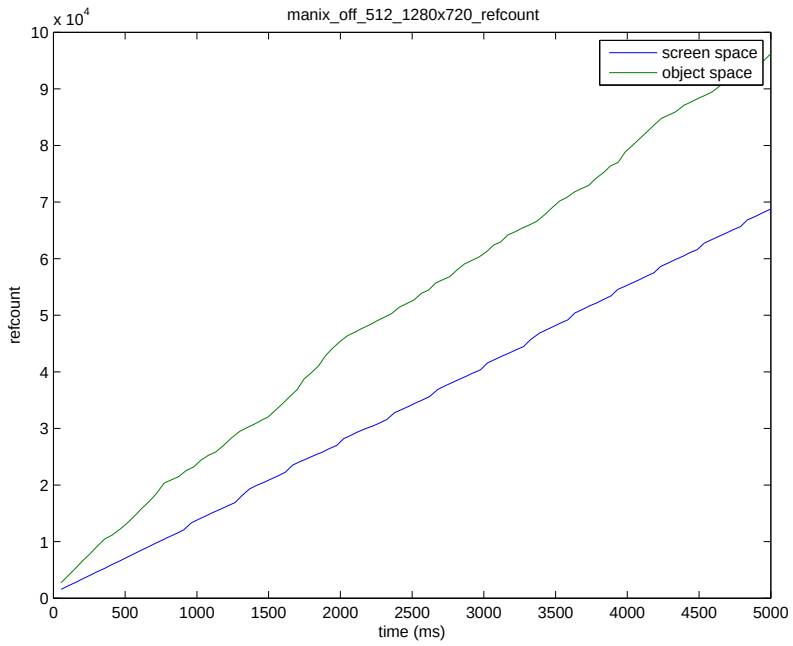


Figure A.144: screen space vs object space: number of references manix, background off, volume res = 512, screen res = 1280x720, number of references

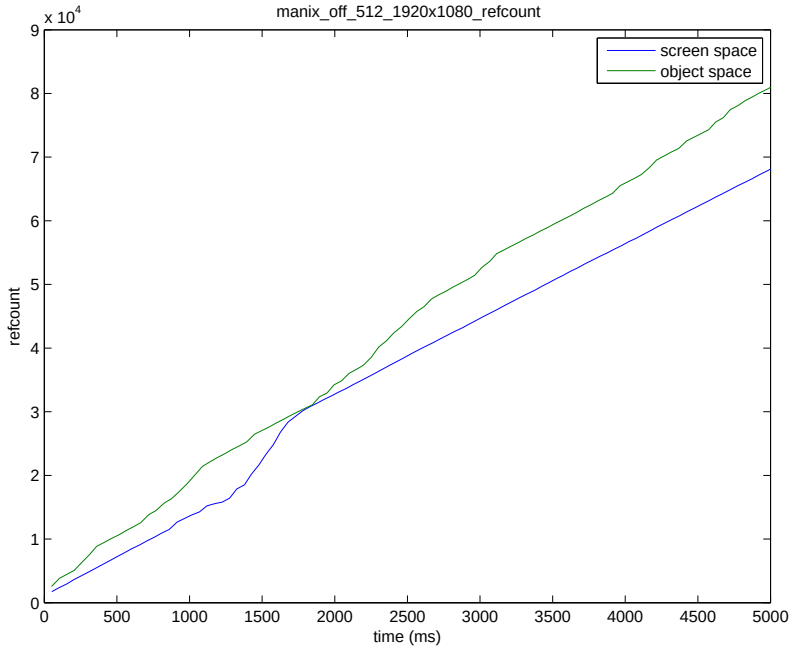


Figure A.145: screen space vs object space: number of references manix, background off, volume res = 512, screen res = 1920x1080, number of references

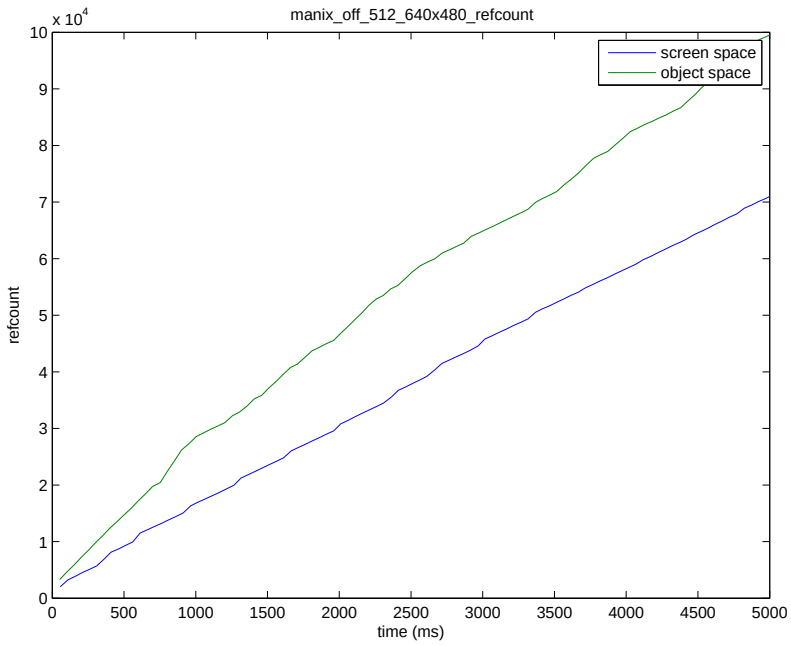


Figure A.146: screen space vs object space: number of references manix, background off, volume res = 512, screen res = 640x480, number of references

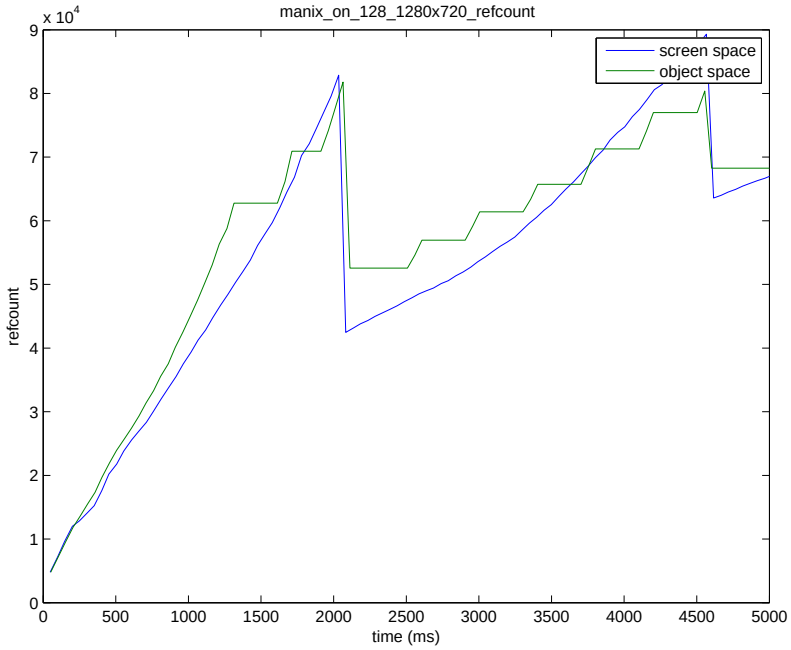


Figure A.147: screen space vs object space: number of references manix, background on, volume res = 128, screen res = 1280x720, number of references

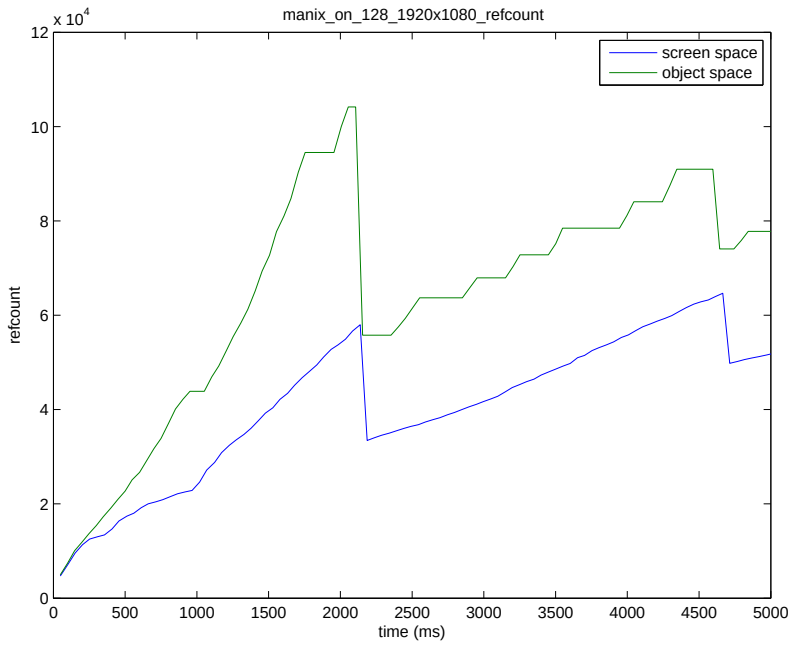


Figure A.148: screen space vs object space: number of references manix, background on, volume res = 128, screen res = 1920x1080, number of references

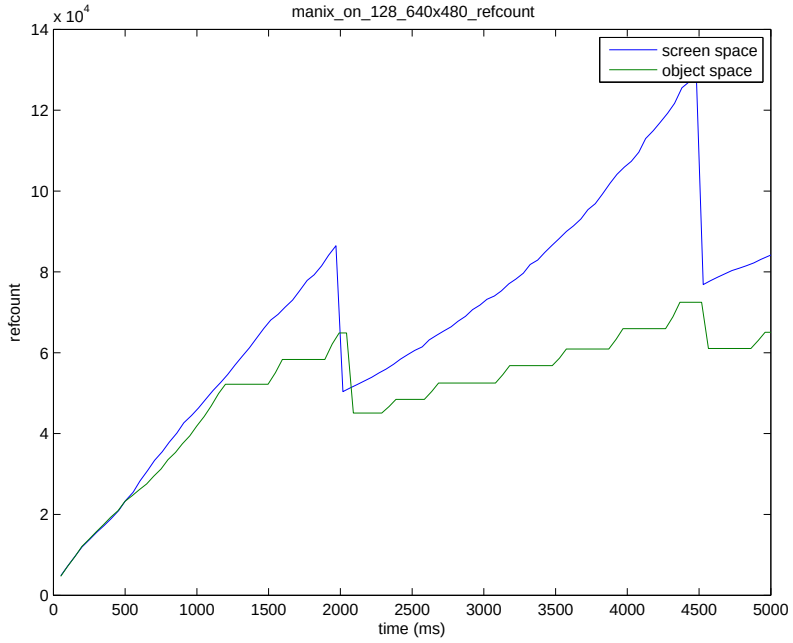


Figure A.149: screen space vs object space: number of references manix, background on, volume res = 128, screen res = 640x480, number of references

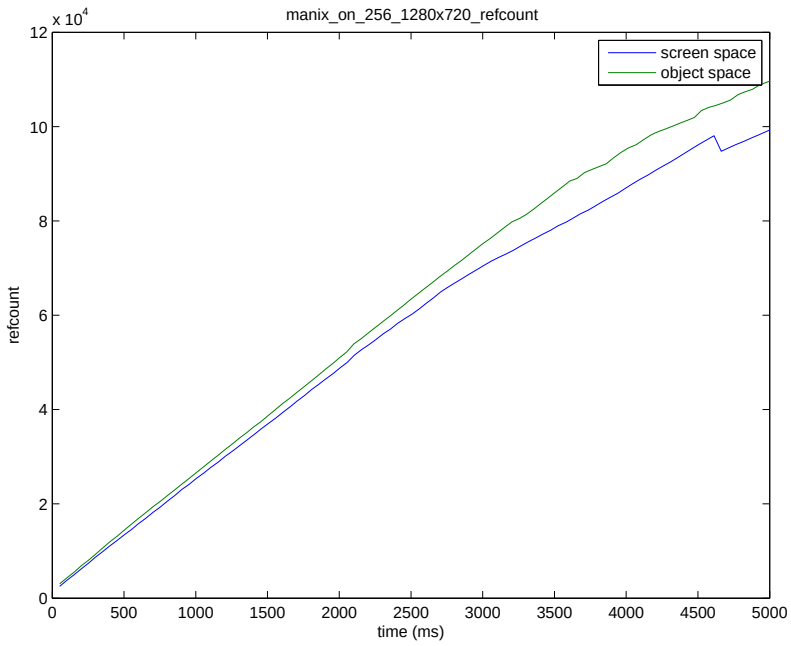


Figure A.150: screen space vs object space: number of references manix, background on, volume res = 256, screen res = 1280x720, number of references

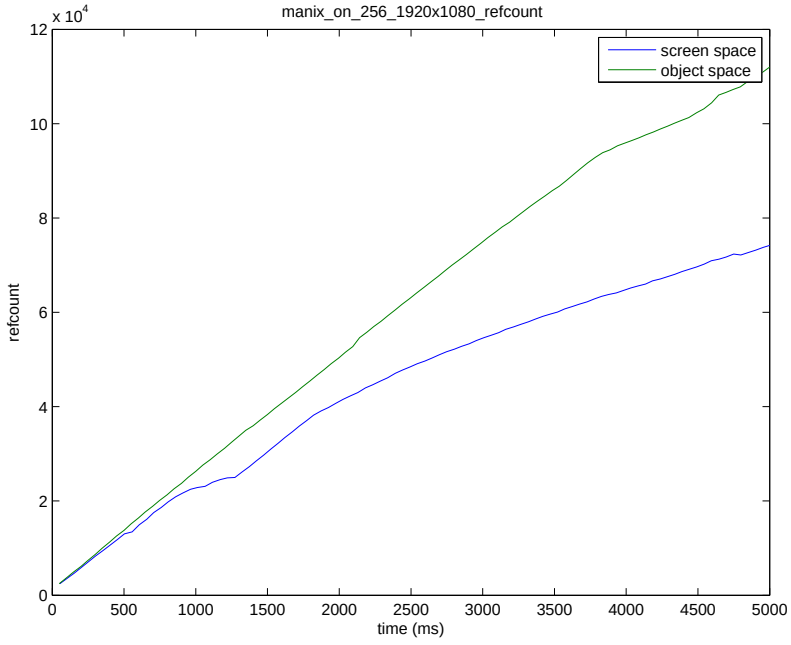


Figure A.151: screen space vs object space: number of references manix, background on, volume res = 256, screen res = 1920x1080, number of references

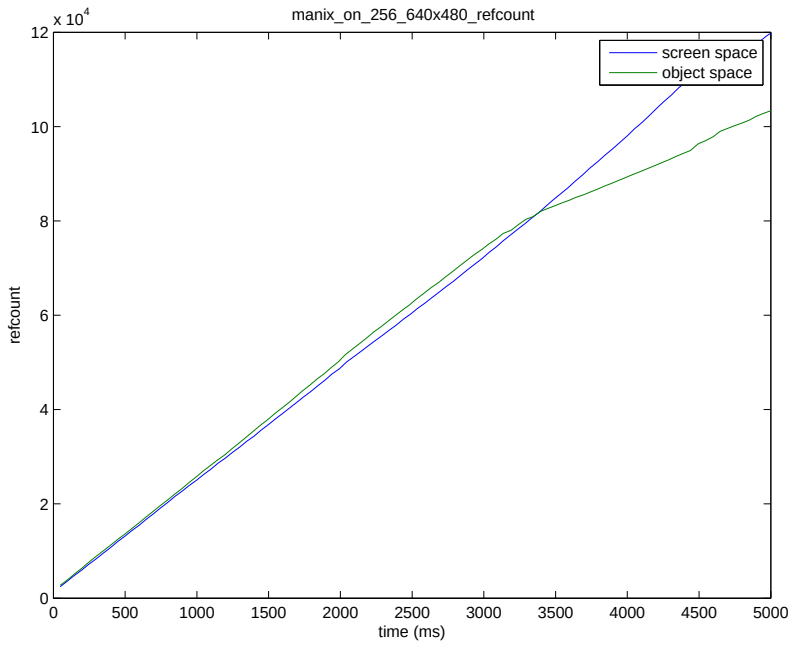


Figure A.152: screen space vs object space: number of references manix, background on, volume res = 256, screen res = 640x480, number of references

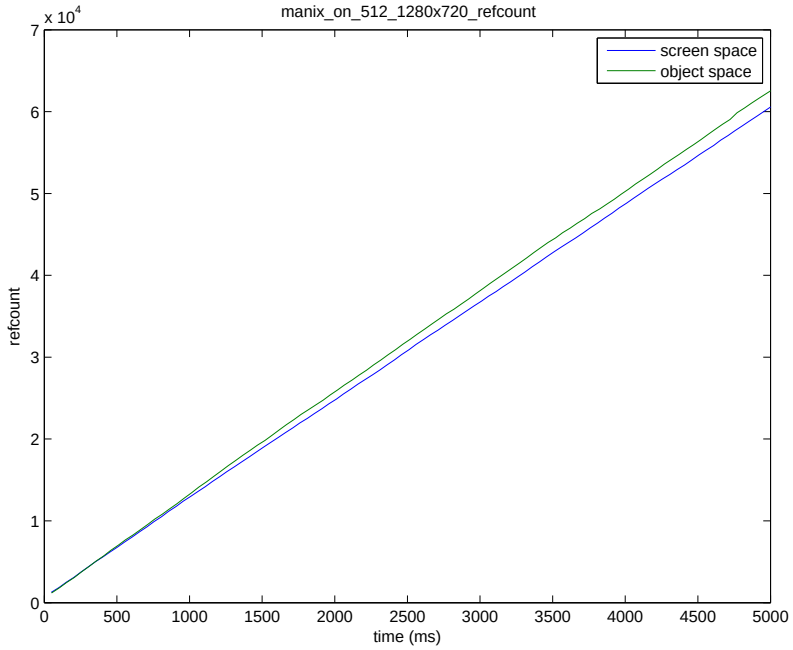


Figure A.153: screen space vs object space: number of references manix, background on, volume res = 512, screen res = 1280x720, number of references

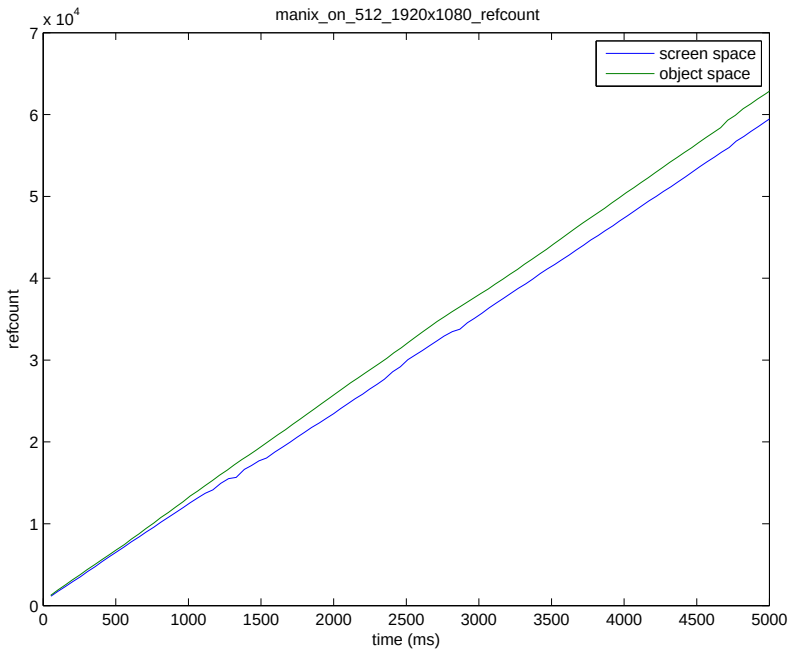


Figure A.154: screen space vs object space: number of references manix, background on, volume res = 512, screen res = 1920x1080, number of references

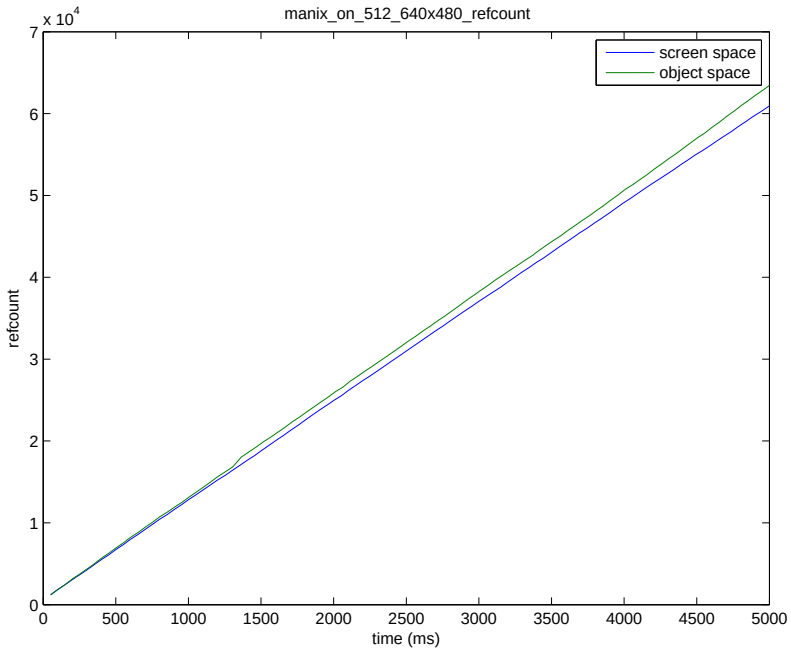


Figure A.155: screen space vs object space: number of references manix, background on, volume res = 512, screen res = 640x480, number of references

Bibliography

[1] Akiba, H., Ma, K.-L., Chen, J. H., and Hawkes, E. R. (2007). Visualizing multivariate volume data from turbulent combustion simulations. *Computing in Science Engineering*, 9(2):76–83.

[2] Alhonnoro, T., Pollari, M., Lilja, M., Flanagan, R., Kainz, B., Muehl, J., Mayrhauser, U., Portugaller, H., Stiegler, P., and Tscheliessnigg, K. (2010). Vessel segmentation for ablation treatment planning and simulation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 6361 of *LNCS*, pages 45–52. Springer.

[3] Altrogge, I., Kröger, T., Preusser, T., Büskens, C., Pereira, P. L., Schmidt, D., Weihusen, A., and Peitgen, H. O. (2006). Towards optimization of probe placement for radio-frequency ablation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 4190 of *LNCS*, pages 486–493. Springer.

[4] Alwin, D. F. and Krosnick, J. A. (1991). The Reliability of Survey Attitude Measurement. *Sociological Methods & Research*, 20(1):139–181.

[5] Alyassin, A. M., Lancaster, J. L., Downs, J. H., and Fox, P. T. (1994). Evaluation of new algorithms for the interactive measurement of surface area and volume. *Med. Phys.*, 6:741–52.

[6] Andrews, F. M. (1984). Construct validity and error components of survey measures: A structural modeling approach. *Public Opinion Quarterly*, 48(2):409–442.

[7] Arens, S. and Domik, G. (2010). A survey of transfer functions suitable for volume rendering. In *Proceedings of the 8th IEEE/EG international conference on Volume Graphics*, VG'10, pages 77–83, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[8] Baegert, C., Villard, C., Schreck, P., and Soler, L. (2007a). Multi-criteria trajectory planning for hepatic radiofrequency ablation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 4792/200 of *LNCS*, pages 676–684.

[9] Baegert, C., Villard, C., Schreck, P., and Soler, L. (2007b). Precise determination of regions of interest for hepatic RFA planning. In *SPIE Medical Imaging 2007: Visualization and Image-Guided Procedures*, volume 6509, pages 650923–650923–8, San Diego, CA, USA.

[10] Baran, I., Chen, J., Ragan-Kelley, J., Durand, F., and Lehtinen, J. (2010). A hierarchical volumetric shadow algorithm for single scattering. *ACM Trans. Graph.*, 29:178:1–178:10.

[11] Bavoil, L. and Myers, K. (2008). Order independent transparency with dual depth peeling. Technical report, NVIDIA Developer SDK 10.

[12] Bénard, P., Bousseau, A., and Thollot, J. (2011). State-of-the-art report on temporal coherence for stylized animations. *Computer Graphics Forum*, 30(8):2367–2386.

[13] Bergner, S., Möller, T., Weiskopf, D., and Muraki, D. (2006). A spectral analysis of function composition and its implications for sampling in direct volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1353–1360.

[14] Bernardon, F. F., Pagot, C. A., Comba, J. L. D., and Silva, C. T. (2006). Gpu-based tiled ray casting using depth peeling. *Journal of Graphics Tools*, 11.3(ISSN 1086-7651):23–29.

[15] Bethell, T. J. and Bergin, E. A. (2011). The propagation of ly? in evolving protoplanetary disks. *The Astrophysical Journal*, 739(2):78.

[16] Beyer, J., Hadwiger, M., Al-Awami, A., Jeong, W.-K., Kasthuri, N., Lichtman, J., and Pfister, H. (2013). Exploring the connectome: Petascale volume visualization of microscopy data streams. *Computer Graphics and Applications, IEEE*, 33(4):50–61.

[17] Bichlmeier, C., Sandro Michael, H., Mohammad, R., and Nassir, N. (2007). Laparoscopic Virtual Mirror for Understanding Vessel Structure: Evaluation Study by Twelve Surgeons. In *Proceedings of the 6th International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 125–128, Nara, Japan.

[18] Bishop, G., Fuchs, H., McMillan, L., and Zagier, E. J. S. (1994). Frameless rendering: Double buffering considered harmful. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 175–176, New York, NY, USA. ACM.

[19] Blasi, P., Le Saec, B., and Schlick, C. (1993). A rendering algorithm for discrete volume density objects. *Computer Graphics Forum*, 12(3):201–210.

[20] Blinn, J. (1988). Me and my (fake) shadow. *IEEE Comput. Graph. Appl.*, 8(1):82–86.

[21] Botchen, R. P., Weiskopf, D., and Ertl, T. (2005). Texture-based visualization of uncertainty in flow fields. In *In Proceedings of the IEEE Symposium on Visualization*, pages 647–654.

[22] Bouthors, A., Neyret, F., and Lefebvre, S. (2006). Real-time realistic illumination and shading of stratiform clouds. In *Proceedings of the Second Eurographics Conference on Natural Phenomena*, NPH'06, pages 41–50, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[23] Brecheisen, R., Platel, B., Bartroli, A. V., and ter Haar Romenij, B. (2008). Flexible gpu-based multi-volume ray-casting. In Oliver Deussen, Dietmar Saupe, D. K., editor, *Proceedings of Vision, Modelling and Visualization 2008, 13th International Fall Workshop*, pages 1–6.

[24] Brunenberg, E. J. L., Vilanova, A., Visser-Vandewalle, V., Temel, Y., Ackermans, L., Platel, B., and Romeny, B. M. T. H. (2007). Automatic trajectory planning for deep brain stimulation: a feasibility study. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, LNCS, pages 584–592. Springer.

[25] Cabral, B., Cam, N., and Foran, J. (1994). Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 symposium on Volume visualization*, VVS '94, pages 91–98, New York, NY, USA. ACM.

[26] Cai, W. and Sakas, G. (1999). Data intermixing and multi-volume rendering. *Computer Graphics Forum*, 18(3):359–368.

[27] Cerezo, E., Pérez, F., Pueyo, X., Seron, F. J., and Sillion, F. X. (2005). A survey on participating media rendering techniques. *The Visual Computer*, 21(5):303–328.

[28] Chandrasekhar, S. (1960). *Radiative Transfer*. Dover Books on Intermediate and Advanced Mathematics. Dover Publications.

[29] Chentanez, N., Alterovitz, R., Ritchie, D., Cho, L., Hauser, K. K., Goldberg, K., Shewchuk, J. R., and O'Brien, J. F. (2009). Interactive simulation of surgical needle insertion and steering. In *ACM Trans. Graph.*, pages 88:1–88:10. ACM.

[30] Colchester, A. C., Zhao, J., Holton-Tainter, K. S., Henri, C. J., Maitland, N., Roberts, P. T., Harris, C. G., and Evans, R. J. (1996). Development and preliminary evaluation of VISLAN, a surgical planning and guidance system using intra-operative video imaging. *Med. Image Analysis*, 1(1):73–90.

[31] Compo, G. P., Whitaker, J. S., Sardeshmukh, P. D., Matsui, N., Allan, R. J., Yin, X., Gleason, B. E., Vose, R. S., Rutledge, G., Bessemoulin, P., BrÃ¶nnimann, S., Brunet, M., Crouthamel, R. I., Grant, A. N., Groisman, P. Y., Jones, P. D., Kruk, M. C., Kruger, A. C., Marshall, G. J., Maugeri, M., Mok, H. Y., Nordli, ., Ross, T. F., Trigo, R. M., Wang, X. L., Woodruff, S. D., and Worley, S. J. (2011). The twentieth century reanalysis project. *Quarterly Journal of the Royal Meteorological Society*, 137(654):1–28.

[32] Coninx, A., Bonneau, G.-P., Droulez, J., and Thibault, G. (2011). Visualization of uncertain scalar data fields using color scales and perceptually adapted noise. In *Applied Perception in Graphics and Visualization*.

[33] Correa, C. and Ma, K.-L. (2009). The occlusion spectrum for volume classification and visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1465–1472.

[34] Correa, C. and Ma, K.-L. (2011). Visibility histograms and visibility-driven transfer functions. *Visualization and Computer Graphics, IEEE Transactions on*, 17(2):192–204.

[35] Crawfis, R. and Max, N. (1996). Multivariate volume rendering. Technical Report UCRL-JC-123623, LLNL.

[36] Csonka, F., Szirmay-Kalos, L., and Antal, G. (2001). Cost-driven multiple importance sampling for monte-carlo rendering. Technical Report TR-186-2-01-19, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria. human contact: technical-report@cg.tuwien.ac.at.

[37] Debattista, K., Dubla, P., Peixoto dos Santos, L., and Chalmers, A. (2011). Wait-free shared-memory irradiance caching. *Computer Graphics and Applications, IEEE*, 31(5):66–78.

[38] Debattista, K., Santos, L. P., and Chalmers, A. (2006). Accelerating the Irradiance Cache through Parallel Component-Based Rendering . pages 27–34, Braga, Portugal. Eurographics Association.

[39] Diaz, J., Vazquez, P.-P., Navazo, I., and Duguet, F. (2010). Real-time ambient occlusion and halos with summed area tables. *Computers and Graphics*, 34(4):337 – 350.

[40] DiMaio, S., Archip, N., Hata, N., Talos, I.-F., Warfield, S., Majumdar, A., Mcdannold, N., Hynynen, K., Morrison, P., III, W. W., Kacher, D., Ellis, R., Golby, A., Black, P., Jolesz, F., and Kikinis, R. (2006). Image-guided Neurosurgery at Brigham and Women's Hospital. 25(5):67–73.

[41] Djurcilov, S., Kim, K., Lermusiaux, P., and Pang, A. (2002). Visualizing scalar volumetric data with uncertainty. *Computers and Graphics*, 26:239–248.

[42] Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., and Nishita, T. (2000). A simple, efficient method for realistic animation of clouds. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 19–28, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[43] Doleisch, H., Muigg, P., and Hauser, H. (2004). Interactive visual analysis of hurricane Isabel with SimVis. Technical Report TR-VRVis-2004-058, VRVis Research Center, Vienna, Austria.

[44] Drebin, R. A., Carpenter, L., and Hanrahan, P. (1988). Volume rendering. *SIGGRAPH Comput. Graph.*, 22(4):65–74.

[45] Du, Z., Chiang, Y.-J., and Shen, H.-W. (2009). Out-of-core volume rendering for time-varying fields using a space-partitioning time (spt) tree. In *Proceedings of the 2009 IEEE Pacific Visualization Symposium*, PACIFICVIS '09, pages 73–80, Washington, DC, USA. IEEE Computer Society.

[46] Eisemann, E., Assarsson, U., Schwarz, M., Valient, M., and Wimmer, M. (2013). Efficient real-time shadows. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, pages 18:1–18:54, New York, NY, USA. ACM.

[47] Enderton, E., Sintorn, E., Shirley, P., and Luebke, D. (2010). Stochastic transparency. In *I3D '10: Proceedings of the 2010 symposium on Interactive 3D graphics and games*, pages 157–164, New York, NY, USA.

[48] Engel, K. and Ertl, T. (2002). Interactive high-quality volume rendering with flexible consumer graphics hardware. In *Eurographics '02 State of the Art Reports*.

[49] Engel, K., Kraus, M., and Ertl, T. (2001). High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware (HWWS '01)*, pages 9–16. ACM.

[50] Ernst, M., Akenine-Möller, T., and Jensen, H. W. (2005). Interactive rendering of caustics using interpolated warped volumes. In *Proceedings of Graphics Interface 2005*, GI '05, pages 87–96, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society.

[51] Essert, C., Haegelen, C., and Jannin, P. (2010). Automatic computation of electrodes trajectory for deep brain stimulation. In *Proceedings of the 5th international conference on Medical imaging and augmented reality*, MIAR'10, pages 149–158, Berlin, Heidelberg. Springer-Verlag.

[52] Estalella, P., Martin, I., Drettakis, G., and Tost, D. (2006). A GPU-driven Algorithm for Accurate Interactive Reflections on Curved Objects. In Akenine-Moller, T. and Heidrich, W., editors, *Eurographics Symposium on Rendering*, page 7, Nicosie, Chypre. Eurographics / ACM SIGGRAPH, ACM.

[53] Estalella, P., Martin, I., Drettakis, G., Tost, D., Devillers, O., and Cazals, F. (2005). Accurate Interactive Specular Reflections on Curved Objects. In Greiner, G., editor, *Vision Modeling and Visualization (VMV 2005)*, page 8, Erlangen, Allemagne. Berlin : Akademische Verl.-Ges. Aka, 2005.

[54] Finch, T. (2009). Incremental calculation of weighted mean and variance. *University of Cambridge*.

[55] Fujishiro, I. and Takeshima, Y. (1999). Solid fitting: Field interval analysis for effective volume exploration. In *Dagstuhl '97, Scientific Visualization*, pages 65–70, Washington, DC, USA. IEEE Computer Society.

[56] Gautron, P., Křivánek, J., Bouatouch, K., and Pattanaik, S. (2005). Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, EGSR'05, pages 55–64, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[57] Gobbetti, E., Marton, F., and Iglesias Guitián, J. (2008). A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets. *The Visual Computer*, 24(7-9):797–806. Proc. CGI 2008.

[58] Grimm, S., Bruckner, S., Kanitsar, A., and Gröller, M. E. (2004). Flexible direct multi-volume rendering in interactive scenes. In *Vision, Modeling, and Visualization (VMV)*, pages 386–379.

[59] Hadwiger, M., Kratz, A., Sigg, C., and Bühler, K. (2006). Gpu-accelerated deep shadow maps for direct volume rendering. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, GH '06, pages 49–52, New York, NY, USA. ACM.

[60] Hagh-Shenas, H., Kim, S., Interrante, V., and Healey, C. (2007). Weaving versus blending: a quantitative assessment of the information carrying capacities of two alternative methods for conveying multivariate data with color. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1270 –1277.

[61] Hagh-Shenas, H., Kim, S., and Tateosian, L. (2009). Multivariate visualization of continuous datasets, a user study. In *Proceedings of the IEEE Symposium on Information Visualization*.

[62] Häme, Y. (2008). Liver tumor segmentation using implicit surface evolution. *The Midas Journal*.

[63] Hansen, C., Wieferich, J., Ritter, F., Rieder, C., and Peitgen, H.-O. (2010). Illustrative visualization of 3d planning models for augmented reality in liver surgery. *Int. Journal of Computer Assisted Radiology and Surgery*, 5:133–141.

[64] Hasenfratz, J.-M., Lapierre, M., Holzschuch, N., and Sillion, F. (2003). A survey of real-time soft shadows algorithms. *Computer Graphics Forum*, 22(4):753–774.

[65] Hauswiesner, S., Khlebnikov, R., Steinberger, M., Straka, M., and Reitmayr, G. (2012). Multi-GPU image-based visual hull rendering. In Childs, H., Kuhlen, T., and Marton, F., editors, *EGPGV*, pages 119–128. Eurographics Association.

[66] Healey, C. G., Tateosian, L., Enns, J. T., and Remple, M. (2004). Perceptually based brush strokes for nonphotorealistic visualization. *ACM Trans. Graph.*, 23:64–96.

[67] Henyey, L. G. and Greenstein, J. L. (1940). Diffuse radiation in the galaxy. *Astrophysical Journal*, 93:70–83.

[68] Hildebrand, P., Leibecke, T., Kleemann, M., Mirow, L., Birth, M., Bruch, H., and Burk, C. (2006). Influence of operator experience in radiofrequency ablation of malignant liver tumours on treatment outcome. *European Journal of Surgical Oncology*, 32(4):430–434.

[69] Holmes, M., Gray, A., and Isbell, C. (2008). Ultrafast monte carlo for statistical summations. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 673–680. MIT Press, Cambridge, MA.

[70] Holten, D., van Wijk, J. J., and Martens, J.-B. (2006). A perceptually based spectral model for isotropic textures. *ACM Trans. Appl. Percept.*, 3:376–398.

[71] Itti, L. (2005). Models of Bottom-Up Attention and Saliency. In Itti, L., Rees, G., and Tsotsos, J. K., editors, *Neurobiology of Attention*, pages 576–582. Elsevier.

[72] Jarosz, W., Donner, C., Zwicker, M., and Jensen, H. W. (2008). Radiance caching for participating media. *ACM Transactions on Graphics (Presented at ACM SIGGRAPH 2008)*, 27(1):7:1–7:11.

[73] Jensen, H. W. and Christensen, P. H. (1998). Efficient simulation of light transport in scences with participating media using photon maps. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 311–320, New York, NY, USA. ACM.

[74] Jönsson, D., Kronander, J., Ropinski, T., and Ynnerman, A. (2012). Historygrams: Enabling interactive global illumination in direct volume rendering using photon mapping. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2364–2371.

[75] Jönsson, D., Sundén, E., Ynnerman, A., and Ropinski, T. (2012). Interactive Volume Rendering with Volumetric Illumination. In *Eurographics STAR program*.

[76] Jönsson, D., Sundén, E., Ynnerman, A., and Ropinski, T. (2013). A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering. *Computer Graphics Forum (conditionally accepted)*.

[77] Kainz, B., Grabner, M., Bornik, A., Hauswiesner, S., Muehl, J., and Schmalstieg, D. (2009). Ray casting of multiple volumetric datasets with polyhedral boundaries on manycore gpus. *ACM Trans. Graph.*, 28(5):Article No. 152.

[78] Kainz, B., Steinberger, M., Hauswiesner, S., Khlebnikov, R., Kalkofen, D., and Schmalstieg, D. (2011a). Using perceptual features to prioritize ray-based image generation. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 215–215, New York, NY, USA. ACM.

[79] Kainz, B., Steinberger, M., Hauswiesner, S., Khlebnikov, R., and Schmalstieg, D. (2011b). Stylization-based ray prioritization for guaranteed frame rates. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, NPAR '11, pages 43–54, New York, NY, USA. ACM.

[80] Kajiya, J. T. and Von Herzen, B. P. (1984). Ray tracing volume densities. *SIGGRAPH Comput. Graph.*, 18(3):165–174.

[81] Karlik, O. (2011). Data structures for interpolation of illumination with radiance and irradiance caching. Master's thesis, Czech Technical University in Prague.

[82] Kehrer, J. and Hauser, H. (2013). Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):495–513.

[83] Khlebnikov, R., Kainz, B., Muehl, J., and Schmalstieg, D. (2011a). Crepuscular rays for tumor accessibility planning. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2163–2172. *Karl-Heinz Höhne Award for Medical Visualization 2012, 3^{rd} place.*

[84] Khlebnikov, R., Kainz, B., Roth, B., Muehl, J., and Schmalstieg, D. (2011b). GPU based on-the-fly light emission-absorption approximation for direct multi-volume rendering. In Laramee, R. and Lim, I. S., editors, *Eurographics 2011 - Posters*, pages 11–12.

[85] Khlebnikov, R., Kainz, B., Steinberger, M., and Schmalstieg, D. (2013). Noise-based volume rendering for the visualization of multivariate volumetric data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2926–2935.

[86] Khlebnikov, R., Kainz, B., Steinberger, M., Streit, M., and Schmalstieg, D. (2012). Procedural texture synthesis for zoom-independent visualization of multivariate data. *Computer Graphics Forum*, 31(3):1355–1364.

[87] Khlebnikov, R. and Muehl, J. (2010). Effects of needle placement inaccuracies in hepatic radiofrequency tumor ablation. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 716–721.

[88] Khlebnikov, R., Voglreiter, P., Steinberger, M., Kainz, B., and Schmalstieg, D. (2014). Parallel irradiance caching for interactive Monte-Carlo direct volume rendering. *Computer Graphics Forum. In press.*

[89] Kniss, J., premoze, S., Hansen, C., Shirley, P., and McPherson, A. (2003a). A model for volume lighting and modeling. *Visualization and Computer Graphics, IEEE Transactions on*, 9(2):150–162.

[90] Kniss, J., Premoze, S., Ikits, M., Lefohn, A., Hansen, C., and Praun, E. (2003b). Gaussian transfer functions for multi-field volume visualization. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 497–504, Washington, DC, USA. IEEE Computer Society.

[91] Kniss, J., Van Uitert, R., Stephens, A., Li, G.-S., Tasdizen, T., and Hansen, C. (2005). Statistically quantitative volume visualization. In *Proceedings of the 16th IEEE Visualization 2005 (VIS'05)*, VIS '05, Washington, DC, USA. IEEE Computer Society.

[92] Koholka, R., Mayer, H., and Goller, A. (1999). MPI-parallelized radiance on sgi cow and smp. In Zinterhof, P., VajterÅ¡ic, M., and Uhl, A., editors, *Parallel Computation*, volume 1557 of *Lecture Notes in Computer Science*, pages 549–558. Springer Berlin Heidelberg.

[93] Kosara, R., Miksch, S., and Hauser, H. (2001). Semantic depth of field. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 97–104. IEEE Computer Society.

[94] Kosara, R., Miksch, S., Hauser, H., Schrammel, J., Giller, V., and Tscheligi, M. (2002). Useful properties of semantic depth of field for better F+C visualization. In *Proceedings of the Symposium on Data Visualisation*, pages 205–210. Eurographics Association.

[95] Kroes, T., Post, F. H., and Botha, C. P. (2012). Exposure render: An interactive photo-realistic volume rendering framework. *PLoS ONE*, 7(7):e38586.

[96] Kruger, J. and Westermann, R. (2003). Acceleration techniques for gpu-based volume rendering. In *Visualization, 2003. VIS 2003. IEEE*, pages 287–292.

[97] Kulla, C. and Fajardo, M. (2012). Importance sampling techniques for path tracing in participating media. *Comp. Graph. Forum*, 31(4):1519–1528.

[98] Křivánek, J., Bouatouch, K., Pattanaik, S., and Žára, J. (2008). Making radiance and irradiance caching practical: adaptive caching and neighbor clamping. In *ACM*

SIGGRAPH 2008 classes, SIGGRAPH '08, pages 77:1–77:12, New York, NY, USA. ACM.

[99] Křivánek, J. and Gautron, P. (2009). Practical global illumination with irradiance caching. *Synthesis Lectures on Computer Graphics and Animation*, 4(1):1–148.

[100] Křivánek, J., Gautron, P., Pattanaik, S., and Bouatouch, K. (2005). Radiance caching for efficient global illumination computation. *Visualization and Computer Graphics, IEEE Transactions on*, 11(5):550–561.

[101] Lagae, A. and Drettakis, G. (2011). Filtering solid Gabor noise. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 51:1–51:6. ACM.

[102] Lagae, A., Lefebvre, S., Cook, R., DeRose, T., Drettakis, G., Ebert, D. S., Lewis, J. P., Perlin, K., and Zwicker, M. (2010). State of the Art in Procedural Noise Functions. In Hauser, H. and Reinhard, E., editors, *EG 2010 - State of the Art Reports*, pages 1–19. Eurographics, Eurographics Association.

[103] Lagae, A., Lefebvre, S., Drettakis, G., and Dutré, P. (2009). Procedural noise using sparse Gabor convolution. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2009)*, 28(3):54:1–54:10.

[104] Lagae, A., Lefebvre, S., and Dutré, P. (2011). Improving Gabor noise. *IEEE Transactions on Visualization and Computer Graphics*, 17(8):1096 –1107.

[105] Levoy, M. (1988). Display of surfaces from volume data. *Computer Graphics and Applications, IEEE*, 8(3):29–37.

[106] Levoy, M. (1990). Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9(3):245–261.

[107] Li, W., Mueller, K., and Kaufman, A. (2003). Empty space skipping and occlusion clipping for texture-based volume rendering. In *Visualization, 2003. VIS 2003. IEEE*, pages 317–324.

[108] Lindemann, F. and Ropinski, T. (2010). Advanced light material interaction for direct volume rendering. In *Proceedings of the 8th IEEE/EG international conference on Volume Graphics*, VG'10, pages 101–108, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[109] Livingston, M., Decker, J., and Ai, Z. (2012). Evaluation of multivariate visualization on a multivariate task. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2114 –2121.

[110] Lord Rayleigh, J. W. S. (1871). On the scattering of light by small particles. *Philosophical Magazine*, 61:447–454.

[111] Marin, T., Wernick, M. N., Yang, Y., and Brankov, J. G. (2010). Motion-compensated reconstruction of gated cardiac spect images using a deformable mesh model. In *IEEE Int. Conf. on biomedical imaging: from nano to macro*, pages 520–523. IEEE Press.

[112] Max, N. (1995). Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108.

[113] Max, N. L. (1986). Atmospheric illumination and shadows. In *ACM SIGGRAPH Computer Graphics*, pages 117–124. ACM.

[114] McDonald, R. J., Gray, L. A., Cloft, H. J., Thielen, K. R., and Kallmes, D. F. (2009). The Effect of Operator Variability and Experience in Vertebroplasty Outcomes. *Radiology*, 253(2):478–485.

[115] Meißner, M., Huang, J., Bartz, D., Mueller, K., and Crawfis, R. (2000). A practical evaluation of popular volume rendering algorithms. In *IEEE Symp. on Volume visualization*, pages 81–90. ACM.

[116] Mie, G. (1908). Beiträge zur optik trüber medien, speziell kolloidaler metallösungen. *Annalen der Physik*, 330(3):377–445.

[117] Miller, J. R. (2007). Attribute blocks: Visualizing multiple continuously defined attributes. *IEEE Computer Graphics and Applications*, 27:57–69.

[118] Moreland, K. (2009). Diverging color maps for scientific visualization. In *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part II*, ISVC '09, pages 92–103, Berlin, Heidelberg. Springer-Verlag.

[119] Mueller, C., Hodgson, J. M., Brutsche, M., Bestehorn, H.-P., Marsch, S., Perruchoud, A. P., Roskamm, H., and Buettner, H. J. (2003). Operator experience and long term outcome after percutaneous coronary intervention. *Can J Cardiol*, 19:1047–51.

[120] Navkar, N. V., Tsekos, N. V., Stafford, J. R., Weinberg, J. S., and Deng, Z. (2010). Visualization and planning of neurosurgical interventions with straight access. In *Proceedings of the First international conference on Information processing in computer-assisted interventions*, IPCAI'10, pages 1–11. Springer.

[121] Nicodemus, F. E. (1965). Directional reflectance and emissivity of an opaque surface. *Appl. Opt.*, 4(7):767–773.

[122] Nirenstein, S., Blake, E., and Gain, J. (2002). Exact from-region visibility culling. In *Eurographics Workshop on Rendering*, EGRW, pages 191–202. Eurographics Association. ACM ID: 581921.

[123] Nishita, T., Miyawaki, Y., and Nakamae, E. (1987). A shading model for atmospheric scattering considering luminous intensity distribution of light sources. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 303–310. ACM.

[124] NVIDIA (2011). *NVIDIA CUDA Programming Guide 4.0*. NVIDIA Corporation.

[125] Ofek, E. and Rappoport, A. (1998). Interactive reflections on curved objects. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 333–342, New York, NY, USA. ACM.

[126] Pfaffelmoser, T., Reitinger, M., and Westermann, R. (2011). Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields. *Computer Graphics Forum*, 30(3):951–960.

[127] Phong, B. T. (1975). Illumination for computer generated pictures. *Commun. ACM*, 18:311–317.

[128] Pöthkow, K. and Hege, H.-C. (2011). Positional uncertainty of isocontours: Condition analysis and probabilistic measures. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1393–1406.

[129] Pöthkow, K., Weber, B., and Hege, H.-C. (2011). Probabilistic marching cubes. *Comput. Graph. Forum*, 30(3):931–940.

[130] Prast, S. and Frühstück, A. (2013). Caustics, light shafts, god rays.

[131] Rezk-Salama, C., Engel, K., Bauer, M., Greiner, G., and Ertl, T. (2000). Interactive volume on standard pc graphics hardware using multi-textures and multi-stage

rasterization. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, HWWS '00, pages 109–118, New York, NY, USA. ACM.

[132] Ribardière, M., Carré, S., and Bouatouch, K. (2011a). Adaptive records for irradiance caching. *Comput. Graph. Forum*, 30(6):1603–1616.

[133] Ribardière, M., Carré, S., and Bouatouch, K. (2011b). Adaptive records for volume irradiance caching. *The Visual Computer*, 27(6-8):655–664.

[134] Rieder, C., Ritter, F., Raspe, M., and Peitgen, H.-O. (2008). Interactive visualization of multimodal volume data for neurosurgical tumor treatment. *Comput. Graph. Forum*, 27(3):1055–1062.

[135] Robertson, D., Campbell, K., Lau, S., and Ligocki, T. (1999). Parallelization of radiance for real time interactive lighting visualization walkthroughs. In *Supercomputing, ACM/IEEE 1999 Conference*, pages 61–61.

[136] Ropinski, T., Kasten, J., and Hinrichs, K. (2008a). Efficient shadows for gpu-based volume raycasting. In *Proceedings of the 16th International Conference in Central Europe on Computer Graphics, Visualization (WSCG08)*, page 17?24. Citeseer.

[137] Ropinski, T., Meyer-Spradow, J., Diepenbrock, S., Mensmann, J., and Hinrichs, K. H. (2008b). Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum (Eurographics 2008)*, 27(2):567–576.

[138] Rößler, F., Tejada, E., Fangmeier, T., Ertl, T., and Knauff, M. (2006). GPU-based multi-volume rendering for the visualization of functional brain images. In *SimVis*, pages 305–318.

[139] Salama, C. R. (2007). Gpu-based monte-carlo volume raycasting. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 411–414, Washington, DC, USA. IEEE Computer Society.

[140] Scherzer, D., Wimmer, M., and Purgathofer, W. (2011). A survey of real-time hard shadow mapping methods. *Computer Graphics Forum*, 30(1):169–186.

[141] Schlegel, P., Makhinya, M., and Pajarola, R. (2011). Extinction-based shading and illumination in gpu volume ray-casting. *IEEE TVCG*, 17(12):1795–1802.

[142] Schott, M., Pegoraro, V., Hansen, C., Boulanger, K., and Bouatouch, K. (2009). A directional occlusion shading model for interactive direct volume rendering. In *Proceedings of the 11th Eurographics / IEEE - VGTC conference on Visualization*, EuroVis'09, pages 855–862, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[143] Schroeder, W., Martin, K., and Lorensen, B. (2006). *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition*. Kitware, 4th edition.

[144] Schroeder, W., Martin, K., and Lorensen, B. (2007). *The Visualization Toolkit, Third Edition*. Kitware Inc.

[145] Schumann, C., Bieberstein, J., Trumm, C., Schmidt, D., Bruners, P., Niethammer, M., Hoffmann, R. T., Mahnken, A. H., Pereira, P. L., and Peitgen, H.-O. (2010). Fast automatic path proposal computation for hepatic needle placement. volume 7625 of *Proceedings of the SPIE*, pages 76251J–1 – 76251J–10.

[146] Shamir, R. R., Tamir, I., Dabool, E., Joskowicz, L., and Shoshan, Y. (2010). A method for planning safe trajectories in image-guided keyhole neurosurgery. In *Proceedings of the 13th international conference on Medical image computing and computer-assisted intervention: Part III*, MICCAI'10, pages 457–464. Springer.

[147] Shanmugam, P. and Arikan, O. (2007). Hardware accelerated ambient occlusion techniques on gpus. In *Proceedings of the I3D, ACM*.

[148] Shenas, H. H. and Interrante, V. (2005). Compositing color with texture for multivariate visualization. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '05, pages 443–446, New York, NY, USA. ACM.

[149] Sloan, P.-P., Kautz, J., and Snyder, J. (2002). Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21(3):527–536.

[150] Šoltészová, V., Patel, D., Bruckner, S., and Viola, I. (2010). A multidirectional occlusion shading model for direct volume rendering. *Computer Graphics Forum*, 29(3):883–891.

[151] Stam, J. (1995). Multiple scattering as a diffusion process. In Hanrahan, P. M. and Purgathofer, W., editors, *Rendering Techniques 1995*, Eurographics, pages 41–50. Springer Vienna.

[152] Steinberger, M., Kainz, B., Kerbl, B., Hauswiesner, S., Kenzel, M., and Schmalstieg, D. (2012). Softshell: dynamic scheduling on gpus. *ACM Trans. Graph.*, 31(6):161:1–161:11.

[153] Tang, Y., Qu, H., Wu, Y., and Zhou, H. (2006). Natural textures for weather data visualization. In *Proceedings of the conference on Information Visualization*, pages 741–750. IEEE Computer Society.

[154] Taylor, R. (2002). Visualizing multiple fields on the same surface. *IEEE Computer Graphics and Applications*, 22:6–10.

[155] Thompson, W., Fleming, R., Creem-Regehr, S., and Stefanucci, J. K. (2011). *Visual Perception from a Computer Graphics Perspective*. A K Peters/CRC Press.

[156] Urness, T., Interrante, V., Marusic, I., Longmire, E., and Ganapathisubramani, B. (2003). Effectively visualizing multi-valued flow data using color and texture. In *In Proceedings of the IEEE Symposium on Visualization*, VIS '03, pages 115–121. IEEE Computer Society.

[157] Urschler, M., Werlberger, M., Scheurer, E., and Bischof, H. (2010). Robust optical flow based deformable registration of thoracic ct images. In *MICCAI Workshop Medical Image Analysis in the Clinic: A Grand Challenge*, LNCS. Springer.

[158] Vaillant, M., Davatzikos, C., Taylor, R., and Bryan, R. (1997). A path-planning algorithm for image-guided neurosurgery. In *Joint Conf. Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery*, volume 1205 of *LNCS*, pages 467–476. Springer.

[159] van Wijk, J. J. (1991). Spot noise texture synthesis for data visualization. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '91, pages 309–318. ACM.

[160] Vancamberg, L., Sahbani, A., Muller, S., and Morel, G. (2010). Needle path planning for digital breast tomosynthesis biopsy. In *Robotics and Automation (ICRA)*, pages 2062–2067.

[161] Veach, E. (1998). *Robust monte carlo methods for light transport simulation*. PhD thesis, Stanford University, Stanford, CA, USA. AAI9837162.

[162] Veach, E. and Guibas, L. J. (1995). Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 419–428, New York, NY, USA. ACM.

[163] Veas, E. E., Mendez, E., Feiner, S. K., and Schmalstieg, D. (2011). Directing attention and influencing memory with visual saliency modulation. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 1471–1480. ACM.

[164] Viard, R., Betrouni, N., Rousseau, J., Mordon, S., Ernst, O., and Maouche, S. (2007). Needle positioning in interventional mri procedure: real time optical localisation and accordance with the roadmap. In *Engineering in Medicine and Biology Society (EMBS)*, pages 2748–2751.

[165] Villard, C., Soler, L., and Gangi, A. (2005). Radiofrequency ablation of hepatic tumors: simulation, planning, and contribution of virtual reality and haptics. *Comput. Methods Biomech Biomed. Engin.*, 8(4):215–227.

[166] Ward, G. J. and Heckbert, P. S. (1992). Irradiance Gradients. *1992 Eurographics Workshop on Rendering*, pages 85–98.

[167] Ward, G. J., Rubinstein, F. M., and Clear, R. D. (1988). A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph.*, 22(4):85–92.

[168] Ware, C. (2004). *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc.

[169] Weber, C., Kaplanyan, A. S., Stamminger, M., and Dachsbacher, C. (2013). Interactive direct volume rendering with many-light methods and transmittance caching. *Proceedings of the Vision, Modeling, and Visualization Workshop*.

[170] Weiskopf, D. (2006). *GPU-Based Interactive Visualization Techniques*. Mathematics and visualization. Springer.

[171] Williams, L. (1978). Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 12(3):270–274.

[172] Wolf, I., Vetter, M., Wegner, I., Böttger, T., Nolden, M., Schöbinger, M., Hastenteufel, M., Kunert, T., and Meinzer, H.-P. (2005). The medical imaging interaction

toolkit. *Medical Image Analysis*, 9(6):594 – 604. ITK - Open science - combining open data and open source software: Medical image analysis with the Insight Toolkit.

[173] Wonka, P., Wimmer, M., Zhou, K., Maierhofer, S., Hesina, G., and Reshetov, A. (2006). Guided visibility sampling. In *ACM Transactions on Graphics*, volume 25, pages 494–502. ACM.

[174] Wyman, C. and Nichols, G. (2009). Adaptive caustic maps using deferred shading. *Comput. Graph. Forum*, 28(2):309–318.

[175] Yu, H., Wang, C., Grout, R., Chen, J., and Ma, K.-L. (2010). In situ visualization for large-scale combustion simulations. *Computer Graphics and Applications, IEEE*, 30(3):45–57.

[176] Yu, X., Li, F., and Yu, J. (2007). Image-space caustics and curvatures. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, PG '07, pages 181–188, Washington, DC, USA. IEEE Computer Society.

[177] Zhang, Y., Dong, Z., and Ma, K.-L. (2013). Real-time volume rendering in dynamic lighting environments using precomputed photon mapping. *IEEE TVCG*, 19(8):1317–1330.

[178] Zhang, Y. and Ma, K.-L. (2013). Fast global illumination for interactive volume visualization. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '13, pages 55–62, New York, NY, USA. ACM.