FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Semantic Screen-Space Occlusion for Multiscale Molecular Visualization

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Thomas Bernhard Koch
Matrikelnummer 01526232

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dr.techn. Ivan Viola
Mitwirkung: David Kouřil, MSc

Wien, 1. Jänner 2018

Thomas Bernhard Koch	Ivan Viola

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Semantic Screen-Space Occlusion for Multiscale Molecular Visualization

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Thomas Bernhard Koch
Registration Number 01526232

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dr.techn. Ivan Viola
Assistance: David Kouřil, MSc

Vienna, 1st January, 2018

Thomas Bernhard Koch	Ivan Viola

Erklärung zur Verfassung der Arbeit

Thomas Bernhard Koch
Forchtenauerstraße 6, 7203 Wiesen

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Jänner 2018

Thomas Bernhard Koch

Acknowledgements

I would like to thank everyone that has supported me and provided feedback throughout my work on this thesis. Special thanks goes to my advisor Ivan Viola for always providing constructive criticism and for encouraging me to present my work at the Central European Seminar on Computer Graphics. I would also like to thank David Kouřil for his support during the implementation phase.

Kurzfassung

Bei der Visualisierung von großen biologischen Datensätzen ergibt sich das Problem der visuellen Übersättigung. Dem Problem wird oftmals mittel Level of Detail Schemata und durch Färbung der visualisierten Strukturen entgegengewirkt. Umgebungsverdeckung und Schatten werden oftmals verwendet, um die Raumwahrnehmung zu verbessern, trägt jedoch der visuellen Übersättigung bei zu ausgiebiger Verwendung. In dieser Arbeit wird ein Ansatz erforscht, um Schattierungstechniken im Bildraum dazu zu verwenden, um der visuellen Übersättigung entgegen zu wirken. Dazu werden Informationen über die hierarchische Struktur der molekularen Modelle berücksichtigt wird. Abhängig von dieser Information werden Schattierungstechniken nur selektiv angewandt. Dieser Ansatz wird demonstriert, indem eine veränderte Variante von Screen-Space Directional Occlusion und Screen-Space Ray Traced Shadows in Marion, einem interaktiven System zur Visualisierung von großen biologischen Datensätzen, implementiert wird.

Abstract

When visualizing large biological data sets the problem of visual clutter arises. This problem is often counteracted with level of detail schemes and appropriate coloring of the visualized structures. Ambient occlusion and shadows are often used in order to improve the depth perception which however promote visual clutter when used excessively. In this work an approach is explored to utilize screen-space shading techniques to reduce the visual clutter by using hierarchy information that is provided with molecular models. This is achieved by selectively applying screen-space shading techniques based on the semantics of the visualized hierarchically structured data. This technique is demonstrated by implementing a hierarchy-aware version of screen-space directional occlusion and screen-space ray traced shadows into Marion, an interactive rendering system to visualize biological data.

Contents

CHAPTER 1

Introduction

1.1 Problem Statement

When visualizing large biological datasets visual clutter can become a problem. Scientific illustrators therefore often simplify illustrations by omitting detail and by using shadows to improve the depth perception. In current systems like Marion [MKS+17], similar techniques are used in order to reduce the visual clutter. Inspired by traditional scientific illustrations, level of detail schemes are used to abstract the shapes of individual molecules while colors are used to group similar molecules. Depth is conveyed via ambient occlusion which also improves shape perception. However, it can be observed that excessive ambient occlusion promotes visual clutter, especially when viewing the dataset in its entirety and the objects occluding each other become smaller and smaller on the screen as can be seen in Figure 1.1.

1.2 Goal of the Thesis

The goal of this thesis is to reduce the visual clutter introduced by excessive ambient occlusion (see Figure 1.1) by restricting ambient occlusion to certain regions depending on the semantics of the objects in the scene and the zoom level. Thereby the hierarchical structure of the objects in the scene gradually becomes more emphasized when zooming in. This is similar to a level of detail scheme which is used to abstract the geometry of objects in the scene. Furthermore, to allow for more control over the illumination of the scene, a three-point lighting system and light attenuation is implemented. This also allows to draw the attention of the user to specific regions which can be used for storytelling scenarios.

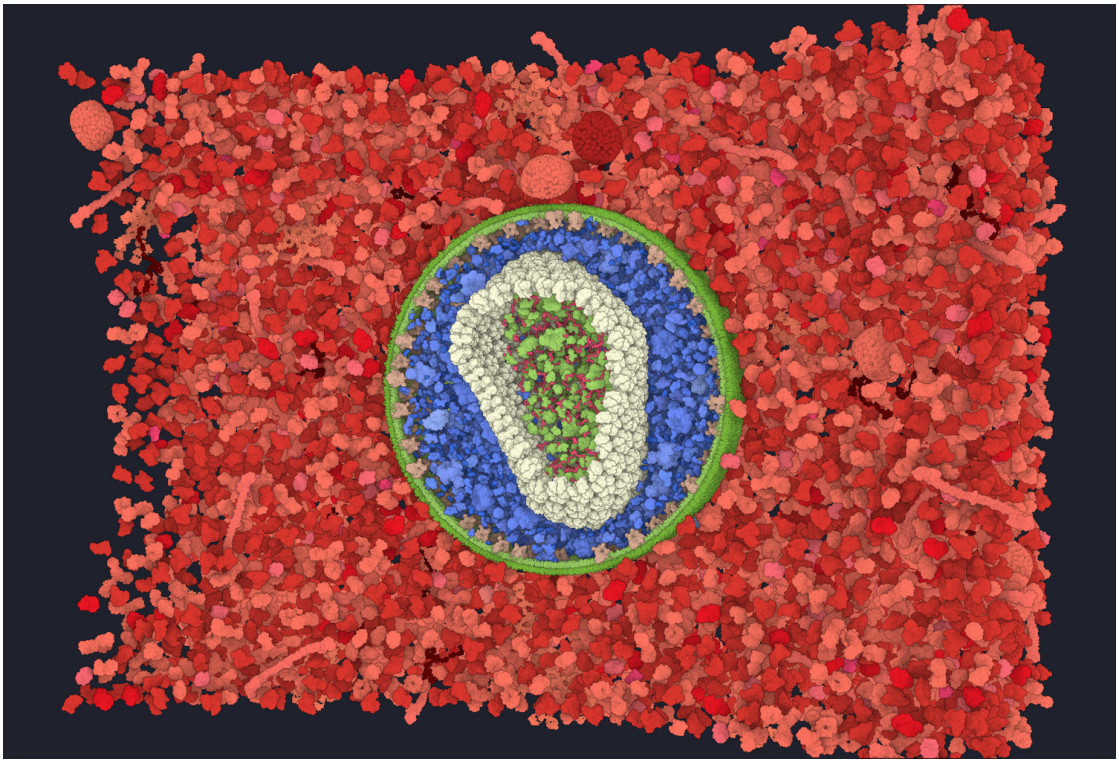


Figure 1.1: A visualization of HIV surrounded by blood plasma rendered by Marion. Ambient occlusion is used to convey depth. However, when zooming further out so that the dataset is viewed in its entirety, unnecessary detail is revealed by ambient occlusion between individual molecules promoting visual clutter.

1.3 Structure of the Thesis

This thesis is structured as follows: In Chapter 2 basic theoretical background and related work is discussed. The techniques and methods that have been used are explained in Chapter 3. In Chapter 4 the implementation is discussed and pseudocode of the implementation is given. Results are presented in Chapter 5 before concluding the work with Chapter 6 where also future work is discussed.

CHAPTER 2

Background and Related Work

In this chapter basic theoretical background and related work is discussed. At first, background about ambient occlusion and Monte Carlo integration is given (Section 2.1). In the second part of this chapter related work is discussed (Section 2.2).

2.1 Ambient Occlusion

Ambient occlusion (AO) is an important technique when rendering realistic scenes. It improves the perception of geometry and depth by producing various effects like soft shadows and darkening of corners and wrinkles. The more occluded a point is, the darker it should appear. Such effects are achieved by taking the occlusion of a point by other geometry into account. In general, the occlusion of a point can be determined by the following integral:

$$A(x, \vec{n}) = \frac{1}{\pi} \int_{\Omega} V(x, \vec{\omega}) \, |\vec{\omega} \cdot \vec{n}| \, \mathrm{d}\vec{\omega} \tag{2.1}$$

where A is the ambient occlusion at point x with surface normal $\vec{n}$, Ω is the unit hemisphere above x, $\vec{\omega}$ represents the directions originating at x along the hemisphere and V is the visibility function along a direction, returning 1 if occluded and 0 otherwise. The ambient occlusion factor of a point x is commonly either computed in object- or in screen-space. In object-space multiple rays have to be traced in directions $\vec{\omega}$ which makes calculating AO in object-space a computationally expensive operation. Also, ray tracing gets more expensive the more complex the geometry is and additional acceleration structures are often necessary. As a result screen-space techniques to approximate AO in real-time have been developed which are applied as a post-process and are less costly than ray tracing in object-space.

Monte Carlo Integration

Integrals like in Equation 2.1 generally do not have analytic solutions. To approximate the value of such integrals Monte Carlo methods are commonly used. Monte Carlo methods estimate the value of integrals by random sampling. The quality of the approximation therefore depends on the distribution and the number of samples that are used. In the case of AO, the value of the integral (Equation 2.1) can be estimated by randomly sampling the hemisphere. In general, the value of an one dimensional integral can be estimated by the following Monte Carlo estimator:

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \tag{2.2}$$

where N is the number of samples, $X_i \in [a, b]$ is a uniform random variable and $p(x)$ is the probability density function (PDF) corresponding to X_i.

The efficiency and accuracy of the approximation can be improved by *importance sampling*. The idea is to draw the samples X_i from a PDF $p(x)$ that is similar to the integrand $f(x)$ of the approximated integral. This makes the estimator converge more quickly which results in less visible noise in the final image without increasing the number of samples. In Equation 2.1 the visibility is weighted by the cosine of the angle between the sampling direction and the surface normal. This means that for samples closer to the top of the hemisphere the visibility is given a higher weight than for samples that are close to the bottom. Therefore, more samples closer to the top of the hemisphere should be used. This is accomplished by using the PDF

$$p(\theta, \phi) = \frac{\cos\theta}{\pi} \tag{2.3}$$

where θ is the polar angle and ϕ is the azimuthal angle.

2.2 Related Work

One of the first techniques to produce an effect similar to AO in screen-space called *depth darkening* was described by Luft et al. [LCD06]. The method relies on simple image manipulation techniques and mainly darkens areas along edges. The result looks significantly different compared to ray traced AO. Bavoil et al. [BSD08] introduce a technique called *horizon based ambient occlusion* (HBAO). Like many other AO techniques, HBAO relies on the depth buffer. The idea is to sample the depth buffer along a user-defined number of directions starting from the pixel for which AO should be approximated. Occluders are then detected by searching the highest depth value along each direction. McGuire et al. [MOBH11] describes a method similar to HBAO to calculate an occlusion factor. The main idea here is that a specifically shaped attenuation function is used to cancel expensive terms in the visibility integral. Also, additional parameters have been introduced which allow to control the appearance of the effect.

Many AO techniques have in common that first an ambient occlusion factor is determined which is then multiplied by the calculated illumination. This separation of the occlusion and illumination calculation leads to the problem that the lighting is ignored when computing AO, thus resulting in equally darkened areas. Therefore, neither the direction nor the color of the lights in the scene are considered when calculating AO. *Screen-space directional occlusion* (SSDO) by Ritschel et al. [RGS09] removes the decoupling of the occlusion and the illumination calculation. SSDO therefore produces colored soft shadows and takes the direction and the color of the light into account. Also, a method is described to produce one indirect bounce of light in screen-space. SSDO has been implemented as part of this thesis.

There is a range of different techniques to generate samples on a hemisphere that can be used for SSDO. A very simple technique is to randomly distribute samples. This can lead to clumping and leaves gaps which results in artifacts in the final image. Poisson disk sampling prevents clumping and gives a more uniform distribution of samples. Sampling techniques that generate samples which are uniformly distributed over a unit square are generally favorable. Compared to other techniques, Halton sequences give a very uniformly distributed sampling pattern [WLH97]. An important property of the Halton sequence is the low discrepancy. The discrepancy is used as a measure for the uniformity of a distribution of samples. The more uniformly a sample set is distributed, the lower is the discrepancy [S$^+$91]. Another important property is that the Halton point set is hierarchical, which means that if N samples are generated, the first k samples are the same for all Halton point sets where $N \geq k$. This allows to generate new samples without having to regenerate the previous samples.

The generation of the Halton sequence is based on the following principles [Nie92]: Any integer $n \geq 0$ with terms a_j in a base $b \geq 2$ can be expressed as:

$$n = \cdots a_2 a_1 a_0 = \sum_{j=0}^{\infty} a_j(n)\, b^j \tag{2.4}$$

The radical-inverse function $\phi_b(n)$ which reflect the terms a_j of the integer n around the decimal point is defined by:

$$\phi_b(n) = 0.a_0 a_1 a_2 \cdots = \sum_{j=0}^{\infty} a_j(n)\, b^{-j-1} \tag{2.5}$$

The n-th Halton point is then given by:

$$x_n = (\phi_{b_1}(n), \ldots, \phi_{b_s}(n)) \tag{2.6}$$

There are various tools and systems to visualize multiscale molecular data. VMD [HDS96] is a tool to display and analyze large biomolecular systems, especially the results of molecular dynamics simulation. Megamol [GKM$^+$15], a prototyping framework, is a visualization tool that focuses on fast rendering of large particle systems. cellVIEW [MAPV15]

uses the game engine Unity for rendering large biomolecular datasets. The presented approach is demonstrated by implementing hierarchy-aware SSDO and hierarchy-aware screen-space ray traced shadows into Marion. Marion [MKS+17] is an interactive system to support biologists in creating multiscale visualizations of biological data for efficient public dissemination. The system derives a visual representation of the data, which is inspired by traditional scientific illustrations, and renders it in real-time. Compared to traditional illustrations, the creation of such is therefore less time consuming. In order to maintain clarity of the scene and to reduce visual clutter a level of detail scheme and dynamic color mapping [WMW+16] is used.

CHAPTER 3

Method

In this chapter the core techniques and methods that are used are explained. SSDO [RGS09] (Section 3.1) is used for the illumination and AO calculation. It correctly produces colored soft shadows and one indirect light bounce. The samples used for SSDO are generated using Halton sequences in order to get uniformly distributed sample points [WLH97] (Section 3.1). The semantics of the visualized data i.e. the hierarchical structure of the visualized cells is incorporated into the AO calculation to reduce the visual clutter of the scene (Section 3.2). To reduce the noise produced by SSDO a bilateral filter is applied (Section 3.3). To improve the realism of the scene and to allow for more control over the illumination three-point lighting and light attenuation is implemented (Section 3.4).

3.1 Screen-Space Directional Occlusion

Compared to other AO techniques, SSDO combines the occlusion and the illumination calculation. This leads to more realistic looking shadows which take the color and the direction of the lights in the scene into account as explained in the following. SSDO also allows to render one indirect bounce of light.

Direct Lighting

Instead of calculating an occlusion factor and the illumination separately, the occlusion and the illumination calculation is combined by calculating the illumination just from unoccluded directions. The advantage is that the color and the direction of the lights are taken into account resulting in colored soft shadows. The occlusion and illumination calculation is combined as follows:

$$L_{\text{dir}}(\mathbf{P}) = \sum_{i=1}^{N} \frac{\rho}{\pi} L_{\text{in}}(\vec{\omega}_i)\, V(\vec{\omega}_i)\, (\vec{\omega}_i \cdot \vec{n})\, \Delta\vec{\omega}_i \tag{3.1}$$

where L_{dir} is the reflected radiance at $\mathbf{P}$ with surface normal $\vec{n}$, $\frac{\rho}{\pi}$ is the diffuse BRDF, L_{in} is the incoming radiance at $\mathbf{P}$ from direction $\vec{\omega}_i$ and V is the visibility function calculating the occlusion. The N sampling directions $\vec{\omega}_i$ are uniformly distributed over the hemisphere that is centered at $\mathbf{P}$ and oriented around $\vec{n}$. The visibility function V is evaluated in screen-space. For this a 3D position and a normal of each pixel is required. At each position $\mathbf{P}$ the hemisphere is sampled by taking a random step $\lambda_i \in [0, r_{\text{max}}]$ in direction $\vec{\omega}_i$, where r_{max} is the radius of the hemisphere. To determine the visibility in direction $\vec{\omega}_i$, sample points are projected back to screen-space. By reading from the position buffer, the 3D position can be computed and the point can be projected onto the surface.

If the projection onto the surface increases the distance of the point to the camera, the sample point is *above* the surface. The corresponding sampling direction $\vec{\omega}_i$ is then classified as *unoccluded* and the illumination is calculated from this direction. Otherwise the sample points are *below* the surface and are treated as *occluders*. Figure 3.1 illustrates this algorithm by an example with $N = 3$ sample points.

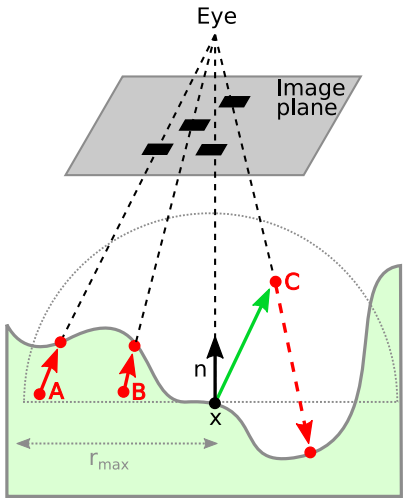


Figure 3.1: Adapted from Ritschel et al. [RGS09]. x is the 3D position of the pixel for which AO should be approximated. A, B and C are the randomly chosen sample points (red points) within the hemisphere which is centered at x with a radius of r_{max}. The red arrows point to the position after projecting the points onto the surface. x is just illuminated from the direction of sample point C (green arrow) because C is *above* and A and B are *below* the surface.

Indirect Bounce

SSDO makes it also possible to calculate one indirect bounce of light. The idea is that an occluder just blocks the direct light coming from a light source but not the indirect light. To incorporate indirect light, only the sample points which are treated as an occluder

are considered. A small normal-oriented patch is placed at each of these sample points and the corresponding pixel color L_{occ} is used as the sender radiance. A form factor is used to calculate how much light from the sender is received:

$$F = \frac{A_s \, (\vec{n} \cdot \vec{\omega}_i) \, (\vec{n}_{\text{occ}} \cdot -\vec{\omega}_i)}{|\mathbf{P} - \mathbf{P}_{\text{occ}}|} \tag{3.2}$$

where $A_s = \pi \, r_{\text{max}}^2 \, / \, N$, $\vec{n}_{\text{occ}}$ is the occluder normal and $\mathbf{P}_{\text{occ}}$ is the occluder position. The indirect bounce is then approximated by:

$$L_{\text{ind}}(\mathbf{P}) = \sum_{i=1}^{N} \frac{\rho}{\pi} \, L_{\text{occ}}(\mathbf{P}_{\text{occ}}) \, (1 - V(\vec{\omega}_i)) \, F \tag{3.3}$$

where $L_{\text{occ}}(\mathbf{P}_{\text{occ}})$ is the pixel color at the occluder position $\mathbf{P}_{\text{occ}}$.

Sampling

As discussed in Section 2.1, the value of the AO integral can be approximated by Monte Carlo integration, in particular, by sampling the hemisphere. There is a range of different techniques to generate samples. In the scope of this thesis Halton sequences are used to generate samples on a hemisphere used for SSDO. Figure 3.2 shows the distribution of 250 samples.

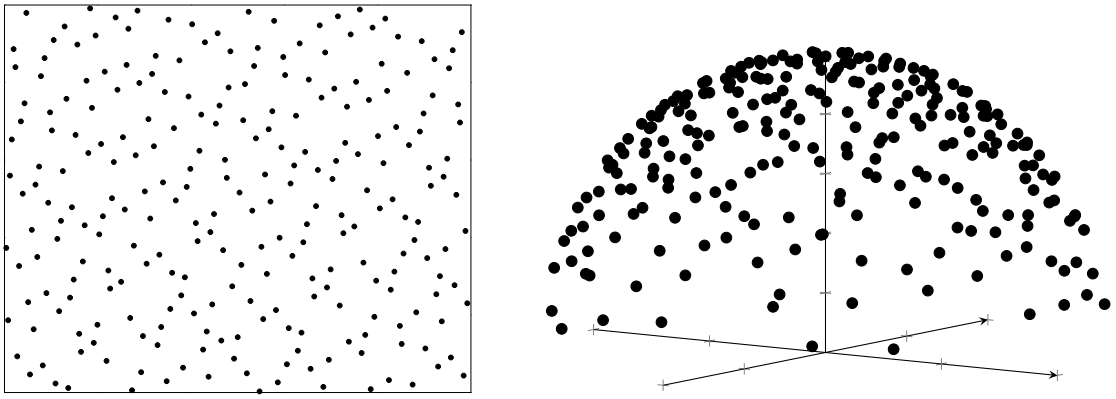


Figure 3.2: Left: Samples generated using Halton sequences ($N = 250$). Right: The samples generated by Halton sequences mapped to a hemisphere. The samples are cosine-weighted, therefore there are more samples closer to the top of the hemisphere.

As discussed in Section 2.1, more samples closer to the top of the hemisphere should be used to approximate the value of the integral (Equation 2.1) more efficiently. In order to generate cosine-weighted points on a hemisphere, *Malley's method* [PJH16] is used. The idea is to first generate uniformly distributed points on a unit disk beneath the hemisphere and then project these points up to the hemisphere. This results in cosine-weighted sample points on a hemisphere (Figure 3.2). Uniformly distributed samples

on a unit disk can be obtained by $(r, \theta) = (\sqrt{u_1}, 2\pi u_2)$ where u_1 and u_2 are uniformly distributed random variables. The cosine-weighted sample points on the hemisphere are then calculated by [PJH16]:

$$\begin{aligned} x &= \cos(2\pi u_2)\sqrt{1-u_1} \\ y &= \sin(2\pi u_2)\sqrt{1-u_1} \\ z &= \sqrt{u_1} \end{aligned} \tag{3.4}$$

3.2 Selective Occlusion Shading

Biological models are often divided into multiple hierarchically structured compartments that are typically separated by lipid membranes. For instance, the HIV virion possesses as spherical form (see Figure 3.3) and consists of three compartment. The innermost compartment, consisting of RNA genomes, is surrounded by capsid. The compartment between the outermost lipid membrane and the capsid consist of several different proteins. The blood plasma that surrounds the HIV virion can also be considered as a compartment.

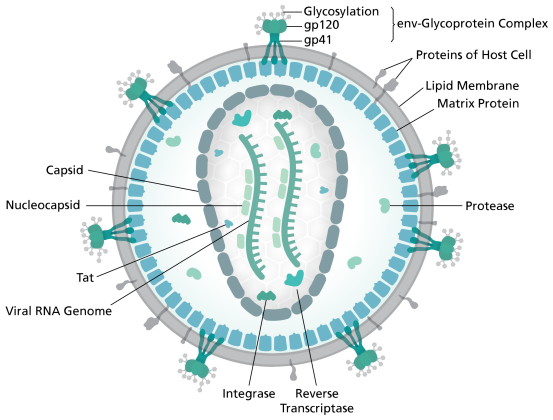


Figure 3.3: An illustration of HIV. Taken from [Spl14].

The information about the hierarchical structure of such compartments can be used to make existing screen-space shading methods hierarchy-aware. This significantly improves the flexibility of such techniques and allows to convey and emphasize desired shapes.

Based on the Labels on Levels approach by Kouřil et. al. [KvK+18], which is used in Marion, the scene is subdivided into different semantic regions depending on the scene objects' hierarchy. The subdivision is controlled using user-defined depth thresholds that divide the depth of the camera into several segments. The resulting regions buffer is shown in Figure 3.4. Differently to Labels on Levels, the subdivision does not depend on the distance of individual objects to the camera but solely on the distance of the camera to the center of the scene. This way artifacts are avoided as explained later. This behavior is shown in Figure 3.5.

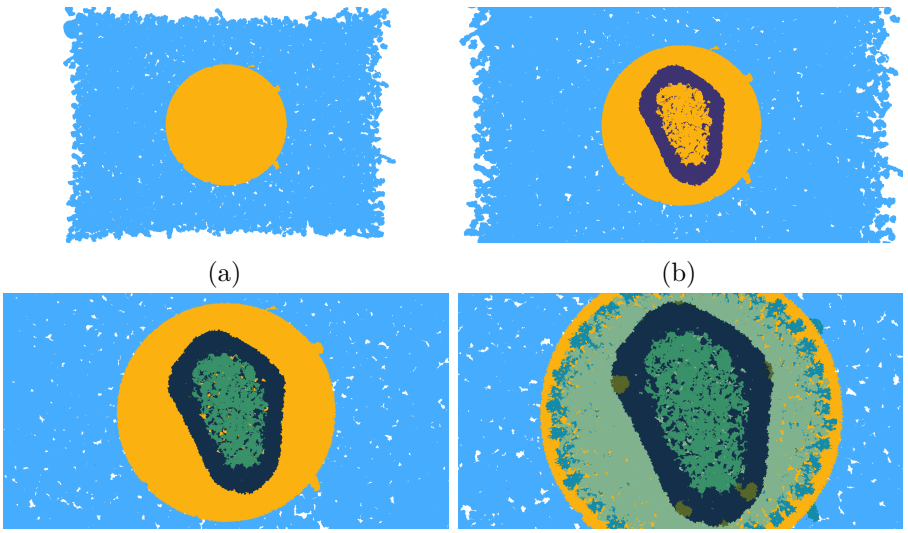
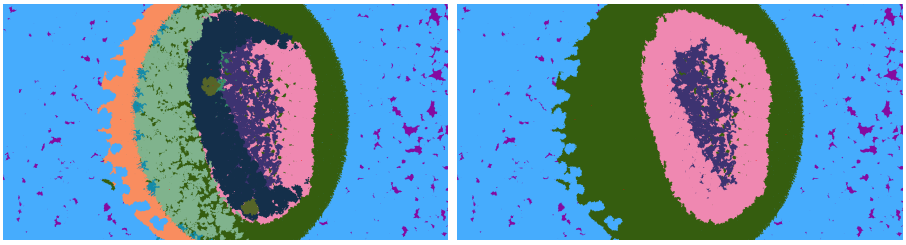


Figure 3.4: Regions buffer at increasing scale levels i.e. zooming in. Each color represents a semantic region. With large distance from the camera, the scene is subdivided into two regions (a), representing the blood plasma compartment in blue and the whole HIV virion in orange. When zooming in, the virion is further subdivided (b) resulting in a separate region for the capsid, depicted in dark blue. This continues in the same fashion (c, d) until also the innermost compartment is represented with a region.

The regions buffer is then used during the occlusion calculation so that there is just ambient occlusion between different semantic regions. Occluders are approximated in screen-space, as explained by Ritschel et. al. [RGS09]. First, sample points for each pixel at 3D position x within a normal oriented hemisphere are chosen. The sample points are back-projected into screen-space. Now screen-space coordinates are known. The sample points are then projected onto the surface by reading the world-space position from a position buffer using the obtained screen-space coordinates. Also, the semantic region at the position of the sample point is fetched from the regions buffer. A sample point is then classified as an occluder, if it moved closer to the viewer by the projection onto the surface and if it belongs to a different semantic region than the current pixel. This way, there is just ambient occlusion between objects of different regions (see Figure 3.6).

3.3 Bilateral Filtering

Screen-space ambient occlusion techniques typically produce noise artifacts due to an insufficient number of sample points. In order to reduce noise, smoothing filters are usually used. Instead of a simple Gaussian low-pass filter, which would also blur edges, bilateral filtering [TM98] [BCCS12] is used. A bilateral filter is a smoothing filter which preserves edges by combining nearby pixel values based on e.g. a Gaussian distribution.



(a) (b)

Figure 3.5: The difference in semantic regions subdivision: (a) The subdivision into semantic regions in Labels on Levels depends on the distance of objects to the camera. (b) Here, the subdivision depends on the distance of the camera to the center of the scene. The subdivision in (a) causes shadowing artifacts as semantic regions are gradually subdivided when the distance to the camera changes i.e. when zooming and, for example, shadows are falsely cast from the blue onto the pink region.

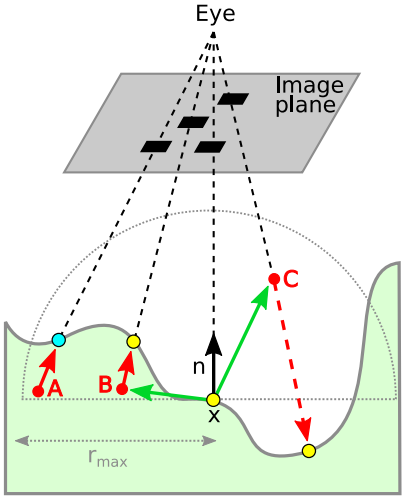


Figure 3.6: Based on Figure 3.1. The condition when a sample point is classified as an occluder is extended, compared to Ritschel et. al. [RGS09], taking also the semantic regions the sample points belong to into account. The illumination of x is calculated from the directions of sample point B and C (green arrows) because they belong to the same region (represented by the color of the dot on the surface) as x. The light is occluded from the direction of sample point A because it belongs to a different region.

The edge preserving property is ensured by also weighting nearby pixel values by their similarity e.g. the differences of their depth values or normal vectors. The bilateral filter is defined by:

$$O(x) = \frac{\sum_{x_i \in \Omega} I(x_i)\, f(x_i, x)}{\sum_{x_i \in \Omega} f(x_i, x)} \tag{3.5}$$

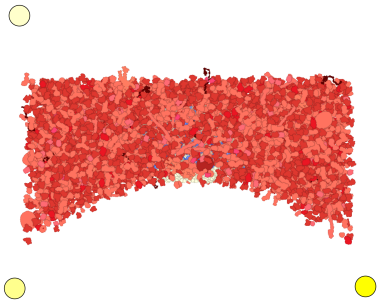


Figure 3.7: The HIV scene from the top and the three lights. The key light is in the bottom right corner, the fill light left to it and the back light is behind the object.

where O is the filtered image, x are the coordinates of the pixel that's filtered, Ω is the neighborhood around x, I is the original image and $f(x_i, x)$ weights the pixels by their closeness:

$$f(x_i, x) = g_r(|I(x_i) - I(x)|)\, g_s(|x_i - x|) \tag{3.6}$$

where g_r is the range attenuation function measuring the photometric similarity of x_i and x and g_s is the spatial attenuation function measuring the geometric closeness between x_i and x. g_r and g_s can be Gaussian functions.

If g_r and g_s are Gaussian functions, the computational complexity for an image with S pixels and a kernel size of σ is $\mathcal{O}(S\,\sigma^2)$. Some filters can be separated into two one-dimensional kernels so that the image is first filtered in the x-dimension and then in the y-dimension. This reduces the computational complexity. The bilateral filter is not separable but gives very similar results when it is separated [PvV05]. This is used in the scope of this thesis and reduces the computational complexity of the bilateral filter to $\mathcal{O}(2S\sigma)$.

3.4 Three-Point Lighting

Inspired by film and photography, a three-point light setup is used. Originally there was just a single light which followed the camera. Three lights are now used: A key light, a fill light and a back light. The key light, placed to the side of the object, creates the main illumination and is the brightest. The fill light, placed on the opposite side of the key light, is responsible for softening the scene and is darker than the key light. The back light is placed on the same side as the fill light but behind the object, and is the darkest of the three lights. The back light should help to separate the object from the background. The placement of the lights in the HIV scene can be seen in Figure 3.7.

To reduce the brightness of a light source over the distance and to improve the realism, the intensity of lights in a scene is attenuated by their distance. The attenuation factor is calculated as follows:

$$A = \frac{1}{A_c + A_l * d + A_q * d^2} \tag{3.7}$$

where A_c is a constant, A_l a linear and A_q a quadratic term. d is the distance to the light source. The value of A_c is usually kept at 1. The terms $A_l * d$ and $A_q * d^2$ allow to simulate the effect that light sources are bright nearby and quickly lose intensity further away.

CHAPTER 4

Implementation

In this chapter an excerpt of the visualization pipeline of Marion is presented and the modifications and implementation details are discussed (Section 4.1). The chapter is concluded with performance considerations (Section 4.2).

4.1 Pipeline

A hierarchy-aware version of SSDO and screen-space ray traced shadows have been implemented into Marion. Figure 4.1 gives an overview of the visualization pipeline. The pipeline consists of multiple passes that each render their output into a texture which is similar to G-buffer [ST90]. In the following, the notation ⊗ is used to refer to the output images in Figure 4.1.

The scene containing HIV in blood plasma is generated using cellPACK [JAAA+15]. In the first step, the scene is rendered using the approach of Le Muzic et. al. [MAPV15]. This step outputs a G-buffer containing world space coordinates ① and normals ②. In the next pass the current frame is subdivided into regions ④ using the Labels on Levels approach by Kouřil et. al. [KvK+18] depending on the depth segment the camera currently is in. Also, the subdivision into regions of the subsequent depth segment is stored in a buffer ⑤. The resulting regions buffers ④ and ⑤ later act like a mask to determine between which regions ambient occlusion and shadows should be applied. In the second pass diffuse shading is applied using the Lambert lighting model. The result ⑥ is used as the sender radiance of the occluders when calculating the indirect light bounce in the SSDO pass. In the next pass the occlusion factor ⑦ and the indirect light bounce ⑧ is calculated. Both buffers ⑦ and ⑧ are then blurred using a bilateral filter to reduce the noise. Before combining the results of the previous passes, shadows are rendered into a texture ⑪ which is also blurred to get softer shadows ⑫. In the following sections the passes are explained in more detail.

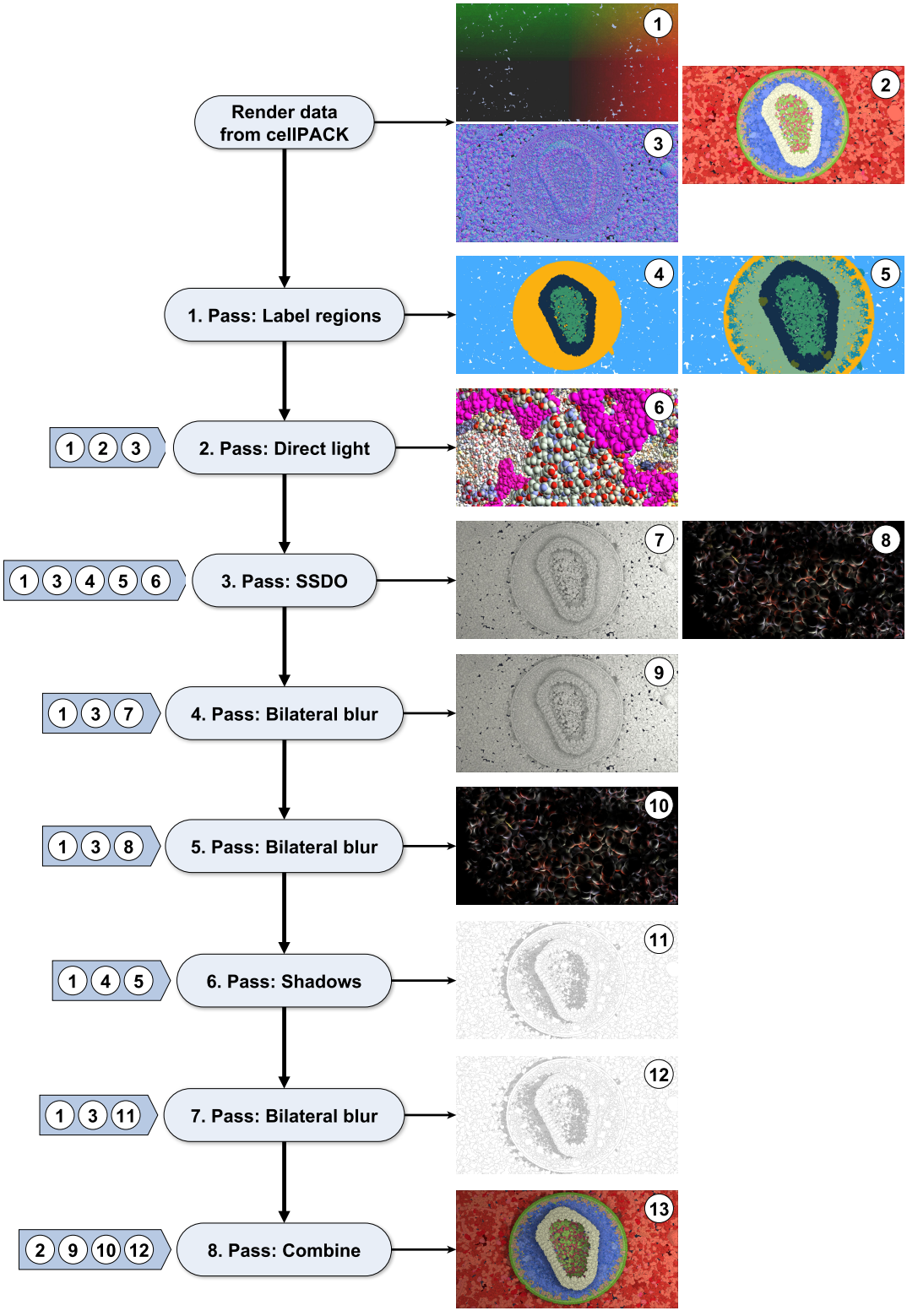


Figure 4.1: Excerpt of the visualization pipeline in Marion. The input of the passes is given on the left side of the ellipses, the output is given on the right side.

1. Pass: Label Regions

Based on the data coming from cellPACK, which contains information about the hierarchical structure of the cells in the scene, and user-defined depth thresholds, the current frame is labeled with compartment IDs. Pseudocode of the implementation is shown in Algorithm 1.

Algorithm 1 Label Regions

Input: User-defined depth thresholds.
Output: The compartment IDs that are associated with the closest and the second closest depth threshold.

1: $t \leftarrow$ determine closest depth threshold to the camera
2: **return** $\Big($compartment id associated with depth threshold $t \hookleftarrow$
3: compartment id associated with depth threshold $t+1\Big)$

The positioning of the depth thresholds is task-dependent and therefore has to be chosen by the user. Using equally sized depth segments would also give sufficient results for this scene.

Two buffers are the result, containing for each pixel: The compartment ID associated with the depth threshold closest (line 2) and second closest (line 3) to the camera. For example, if (b) in Figure 3.4 shows the compartments associated with the depth threshold t closest to the camera, then (c) would be the compartments associated with the threshold $t+1$.

2. Pass: Direct Light

The second pass takes the position texture ①, the texture containing the scene without shading ②, and the normal texture ③ as input, and outputs a diffuse shaded version of the current frame in a texture. This texture is then used for the indirect light bounce calculation in pass 3. The Lambert lighting model is used for the diffuse shading:

$$I = (\vec{l} \cdot \vec{n})\, C_S\, C_L\, I_L \tag{4.1}$$

where $\vec{l}$ is a vector from the surface to the light, $\vec{n}$ is the surface normal, C_S is the color of the surface, C_L is the color of the light and I_L is the intensity of the light. The light sources are attenuated according to Equation 3.7. This is calculated for each light in the scene.

3. Pass: SSDO

SSDO is implemented as the third pass. Pseudocode of implementation is shown in Algorithm 2.

Algorithm 2 SSDO

Input:

- The textures *posTex* and *normTex* containing world space coordinates ① and normals ③ for each pixel from pass 1.
- The texture *regionsTex* ④ and *nextRegionsTex* ⑤ containing the labeled regions from pass 1.
- The texture *diffTex* containing the diffuse shaded version of the current frame ⑥ from pass 2.
- A texture *noiseTex* containing noise.
- An array *depthThresholds* of user-defined depth thresholds.
- Arrays *lightPos*, *lightColor* and *lightStrength* containing the position, color and intensity of each light.
- An array *halton* containing sample points that are uniformly distributed in an unit square.
- User-defined radii *radiusNear* and *radiusFar*, user-defined values *bounceStrength* and *kernelSize* and user-defined light attenuation parameters A_c, A_l and A_q.

Output: The direct lighting and the light bounce.

procedure projectToHemisphere ▷ See Equation 3.4
    $r \leftarrow$ sample *noiseTex*
    $s \leftarrow$ get sample point from *halton*
    $u = s + r - \lfloor s + r \rfloor$
    return $\left(\sqrt{u_1} * \cos(2\,\pi\,u_2) \quad \sqrt{u_1} * \sin(2\,\pi\,u_2) \quad \sqrt{1-u_1}\right)$
end procedure

procedure calculateLightBounce
    $f \leftarrow$ calculate form factor according to Equation 3.2
    occluderColor $\leftarrow$ sample *diffTex* at sample point
    return *f* * *occluderColor* * *bounceStrength*
end procedure

procedure calculateOcclusionFactor(r)
    posSample $\leftarrow$ get sample point within hemisphere with radius r using projectToHemisphere()
    offset $\leftarrow$ back-project sample position *posSample* to screen-space
    sampleDepth $\leftarrow$ sample depth from *posTex* using *offset*
    rangeCheck $\leftarrow$ smoothstep(0, 1, r / $|position_3 - sampleDepth|$)
    return $(sampleDepth \geq posSample_3) * rangeCheck$
end procedure

position $\leftarrow$ sample *posTex*
normal $\leftarrow$ sample *normTex*

```
24:
25: for i = 0 to kernelSize do
26:     occlusionNear ← calculateOcclusionFactor(radiusNear)
27:     occlusionNear ← occlusionNear ∗ lerp(1, 0, cameraDepth)
28:     if occlusionNear > 0 then
29:         lightBounce ← lightBounce + calculateLightBounce()
30:     end if
31:
32:     occlusionFar ← calculateOcclusionFactor(radiusFar)
33:     compartment ← sample regionsTex at the current position
34:     nextCompartment ← sample nextRegionsTex at the current position
35:     sampleCompartment ← sample regionsTex at the position of the sample point
36:     if compartment = sampleCompartment then
37:         occlusionFar ← 0
38:     else
39:         occlusionFar ← linearly interpolate between compartment and nextCompartment
    based on the camera position between the two depth thresholds the camera currently
    is in
40:     end if
41:
42:     occlusionSum ← calculate lighting weighted by occlusionNear and occlusionFar
    from each light. Attenuate light sources according to Equation 3.7 using A_c, A_l and
    A_q
43: end for
44: lightBounce ← interpolate lightBounce depending on the depth
45: return occlusionSum, lightBounce
```

The user-defined variable *kernelSize* determines the number of iterations of the main for-loop (line 34) and thus the number of samples. In order to have AO between individual atoms but also between larger, more distant structures like individual molecules, the occlusion of a point is determined using a small (lines 26-30) and a large (lines 32-39) radius of the hemisphere. The Halton samples that are projected onto the hemisphere (line 15) have been generated via the algorithm given by [WLH97]. The radii are scene-dependent and user-defined. The occlusion calculation of near and far occluders have been moved to a single shader. Originally, this took two passes. This way maintainability is improved because code duplication is kept to a minimum.

The occlusion between small structures like atoms becomes less relevant the further the user zooms out, therefore the occlusion factor is linearly interpolated weighted by the position of the camera (line 27). Similarly, the indirect light bounce which is just calculated on atom level (line 29) is weighted by the depth (line 43). This is necessary mostly because the light bounce rendered in the distance "flickered" during camera movement becoming more apparent when fewer samples are used. For the occlusion factor of larger and more distant occluders the compartments are taken into account

(lines 34-39). There is no occlusion if the sample point and the current point belong to the same compartments. In order to smoothly fade the occlusion in and out as the user zooms, the occlusion factor is linearly interpolated depending on the camera position (line 38). As mentioned in [BSD08], the factor A_s from Equation 3.2 is used as a user-defined parameter, here denoted as *bounceStrength*, to manually control the strength of the light bounce (light 11).

4., 5. and 7. Pass: Bilateral Blur

The bilateral blur [TM98] is separated to reduce the computational complexity (Section 3.3). Therefore, two passes are necessary: The first one blurs the image in x-direction and the second one in y-direction. Pseudocode of the implementation is show in Algorithm 3.

Algorithm 3 Bilateral Blur

Input: The image which should be blurred, the normals and positions from pass 1, the user-defined kernel size *amount* and user-defined depth and normal thresholds t_d and t_n.
Output: The blurred image.

1: **for** $i = -amount$ **to** $amount$ **do**
2: $s \leftarrow$ sample position
3: **if** ($|$current depth $-$ depth at $s| < t_d$) $\wedge$ (current normal $\cdot$ normal at $s > t_n$) **then**
4: $w \leftarrow e^{-(i/(amount * 1.5))^2}$
5: $color \leftarrow$ sample color weighted by w
6: $sum \leftarrow sum + w$
7: **end if**
8: **end for**
9: **return** $color/sum$

The results of several passes in the pipeline are blurred: The output of the SSDO pass—the texture containing the occlusion factor ⑦ and the light bounce texture ⑧—is blurred to reduce the noise and the result of the shadow pass ⑪ is blurred to soften the shadows. Currently, the kernel size is user-defined, allowing the user to change the strength of the blur. Usually, in order to keep the number of calculations to a minimum, the filter kernel is pre-computed and hard-coded.

6. Pass: Shadows

Shadows are produced using screen-space ray tracing which is often used for screen-space reflections [Val14] [Sai16]. Screen-space ray tracing works by ray tracing against a depth buffer. The shadows pass is implemented as the sixth pass but could have been implemented as an earlier pass since it only relies on the world space coordinates ① and the regions buffers ④ and ⑤ from the first pass. Pseudocode of the implementation can be seen in Algorithm 4. The algorithm is similar to Algorithm 2. Also, similarly to

Algorithm 2 the shadow factor is linearly interpolated (line 9) so that it fades in and out smoothly when zooming.

Algorithm 4 Shadows

Input: The texture *posTex* containing world space coordinates ①, *regionsTex* ④ and *nextRegionsTex* ⑤ containing the labeled regions from pass 1 and a user-defined distance *step*.

Output: A shadow factor.

1: *samplePoint* ← current world space position from *posTex*
2: **while** *samplePoint* ≠ key light position **do**
3: *samplePoint* ← go distance *step* in direction of the key light source
4: *offset* ← back-project sample position *samplePoint* to screen-space
5: *sampleDepth* ← get sample depth from *posTex* using *offset*
6: **if** *sampleDepth* > $samplePoint_3$ ∧ compartment ID at sample point ≠ current compartment ID **then**
7: *shadowFactor* ← *shadowStrength*
8: **break**
9: **end if**
10: **end while**
11: *shadowFactor* ← linearly interpolate between *compartment* and *nextCompartment* based on the camera position between the two depth thresholds the camera currently is in
12: **return** *shadowFactor*

8. Pass: Combine

The last pass combines the output of the previous passes by multiplying the initially rendered scene ②, the occlusion factor ⑨ determined using the hierarchy-aware SSDO approach and the shadow factor ⑫. The light bounce ⑩ is simply added on resulting in the final image.

4.2 Performance Measurements

The performance impact of various aspects have been measured and are presented in this section. The test system includes an Intel Core i5-6600K CPU and a NVIDIA GeForce GTX 1070 GPU. The performance values have been measured at a resolution of 1920x1080.

Measurement	State	ms/frame	% increase
Light bounce	off	80	
	on	88	10
Shadows	off	95	
	on	97	2.1
Blur kernel size	1	96	
	10	98	2.1
SSDO samples	16	116	
	64	206	77.6

Table 4.1: Timing values for the HIV scene. The values for the light bounce have been measured at the atom level similar to Figure 5.3. All the other values have been measured further zoomed out similar to Figure 5.2 (b). The values for the light bounce, shadows and blur kernel size have been measured with 16 samples used for SSDO.

As can be seen in Table 4.1, the indirect light bounce increases the frame time by 10%. Shadows are cheap with an increase of just 2.1%. Also, increasing the kernel size from 1 to 10 of one blur pass increases the frame time by 2.1%. Throughout this thesis a blur radius of 3 is used to blur the occlusion texture, 6 for the indirect bounce and 5 for the shadows. It was possible to reduce the number of samples needed for the AO pass from 64 that were originally necessary when random sampling was used to 16 and still get satisfactory results.

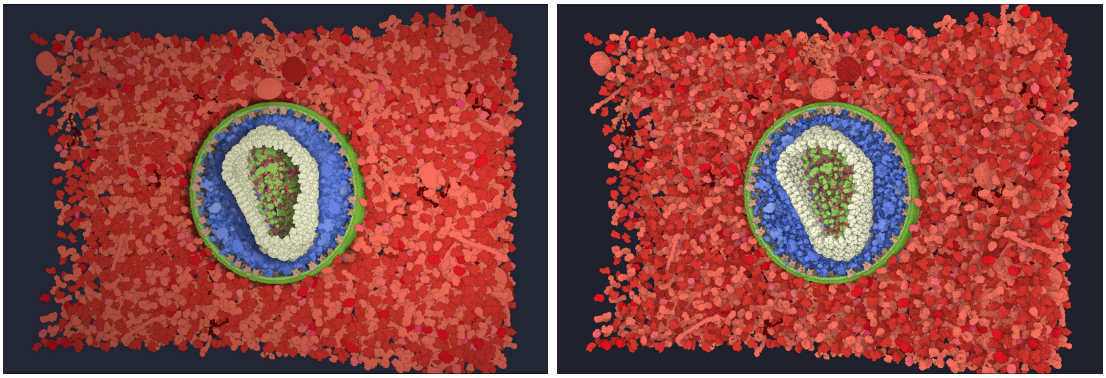
CHAPTER 5

Results

In this chapter results are presented and discussed, showing the HIV scene used throughout the document. The goal was to reduce the visual clutter caused by excessive ambient occlusion which became apparent when viewing the dataset in its entirety and the objects occluding each other became smaller and smaller. This has been improved by incorporating information about the structure of the molecular model into the occlusion calculation thus restricting ambient occlusion just to certain regions i.e. compartments depending on the zoom level. In Figure 5.1 a comparison of the HIV scene with and without hierarchy-aware SSDO can be seen. With semantic AO the image looks much calmer because there is just ambient occlusion around larger structures. When zooming into the scene, the structure of the virus is gradually emphasized by slowly appearing soft shadows caused by ambient occlusion.

Figure 5.2 shows this behavior. It can also be observed that the occlusion between individual atoms and individual proteins, respectively, is gradually appearing when zooming in, thus supporting the level of detail system hiding unnecessary detail. Zooming further in reveals individual atoms as can be seen in Figure 5.2 (d) and Figure 5.3. It can be observed that the light bounce improves the realism of the scene (see Figure 5.3).

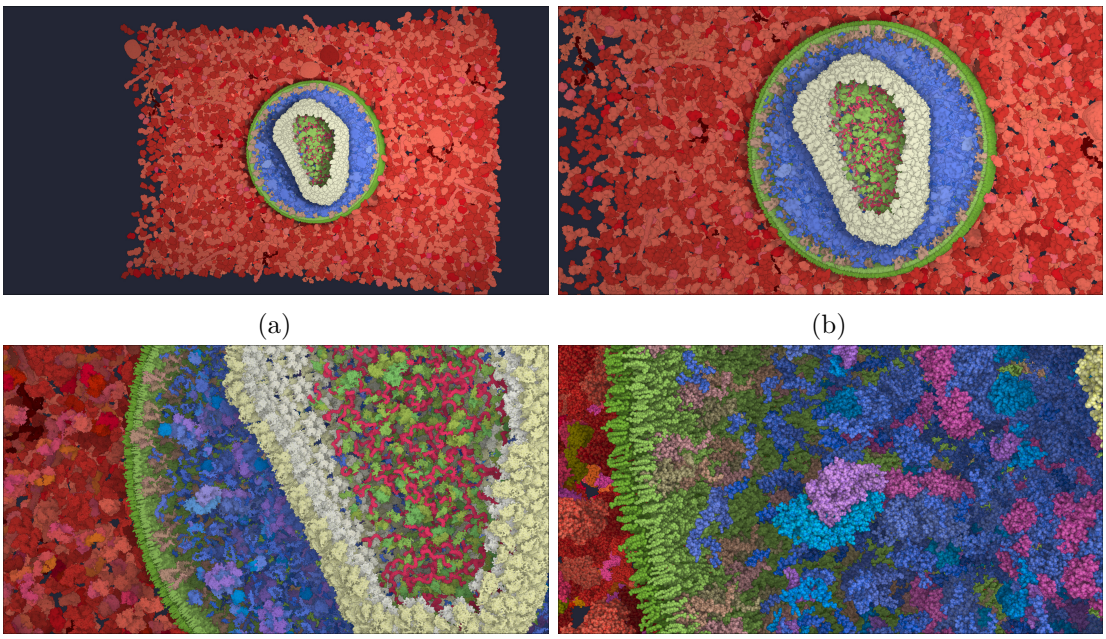
Adding shadows started as an experiment but has been left implemented due to the surprisingly good results and to demonstrate how much the scene benefits from shadows. Shadows provide additional depth cues and convey information about the size, shape and location of the objects in the scene. The impact of shadows on the perception of the size of objects can be seen in Figure 5.2. It can be observed that the perception of the size of the capsid and how much it protrudes is much more obvious with shadows. The shadows are slowly faded out when zooming in preventing the scene to get too dark. Due to the limitations of shadows produced by ray tracing the depth buffer—the main limitation being having no information about the geometry outside of the camera's view–shadows are just shown between compartments and not between individual atoms. This way the limitations are not as apparent.



(a) (b)

Figure 5.1: Comparison of the HIV scene with (a) and without (b) semantic AO. Without semantic AO the scene looks noisy and cluttered due to the small size of the molecules on the screen.

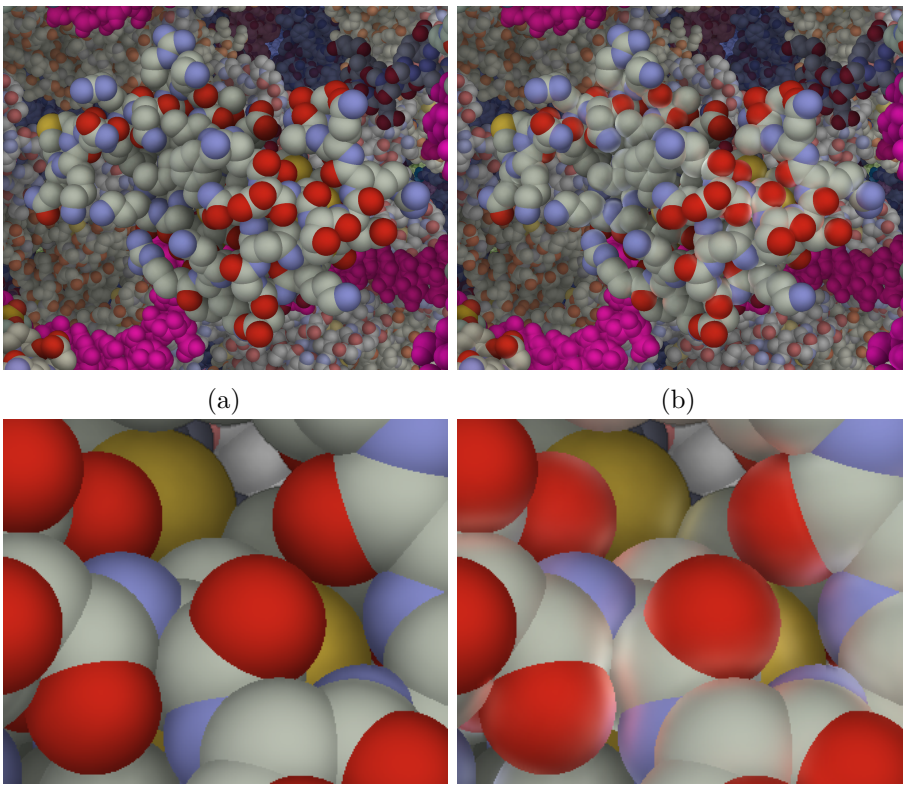
The three-point light setup that has been implemented into Marion allows for more realistically illuminated scenes and greater artistic freedom. Figure 5.4 shows different examples on how the light setup can be used and which type of effects can be achieved.



(a) (b)

(c) (d)

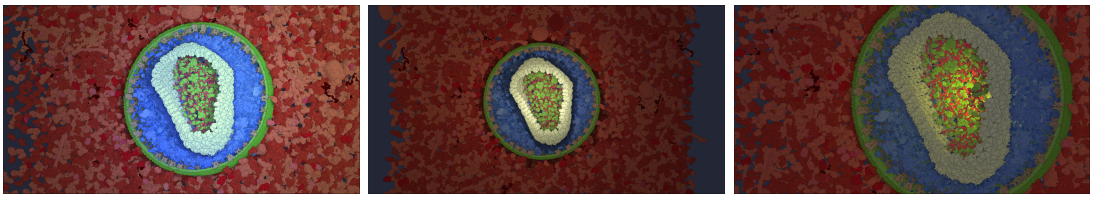
Figure 5.2: Zooming into the HIV scene. Shadows and soft shadows caused by ambient occlusion are slowly appearing gradually emphasizing the structure of the cell. First, shadows are just visible around and on the inside of the capsid and around the outermost lipid membrane (a). When zooming further into the scene soft shadows on the inside of the outermost membrane become visible revealing further details (b). The shadows (a) cast by the capsid and the lipid membrane are in turn faded out again preventing the scene getting too dark at atom level. Zooming in further reveals soft shadows between molecules of the same compartment (c, d) before individual atoms are revealed (d).



(a) (b)

(c) (d)

Figure 5.3: The top row shows the difference of the HIV scene at atom level with (a) and without (b) light bounce. The light bounce is especially visible between pink and white atoms (c, d).



(a) (b) (c)

Figure 5.4: Examples that showcase the flexibility of a three-point lighting setup. In (a) the color of the key and the fill light have been changed to a more green and blue color tone which helps to further set the virion apart from the surrounding blood cells. In (b) the key light is moved closer to the capsid and the quadratic attenuation term has been set so that the light intensity falls off steeply highlighting the capsid. (c) is an example for using lights to mark points of interest that should catch the attention of the user.

CHAPTER 6

Conclusion and Future Work

The presented approach which allows to make existing screen-space shading methods, such as screen-space ambient occlusion, aware of the hierarchy of the visualized multi-scale models. Especially interactive rendering systems, where large biological datasets are visualized resulting in scenes with thousands of molecules, can benefit from our method as they typically utilize screen-space shading methods.

The presented method has been demonstrated by implementing hierarchy-aware SSDO and screen-space ray traced shadows into Marion, a state-of-the-art interactive molecular visualization system. The technique operates in screen-space and could thus be implemented in other rendering frameworks with probably only small modifications necessary.

The method could also be used to actively guide users through the scene by employing the illumination for highlighting the essential structures. By emphasizing just certain structures, the attention of the user can be drawn to a desired direction.

In future work, we would like to investigate and expand other screen-space shading methods to take advantage of information about the hierarchical structure which is often provided with molecular models. Additionally, the performance of the presented technique could be improved by reusing samples of previous frames.

Bibliography

[BCCS12] Francesco Banterle, Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. A low-memory, straightforward and fast bilateral filter through subsampling in spatial domain, February 2012.

[BSD08] Louis Bavoil, Miguel Sainz, and Rouslan Dimitrov. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 Talks*, SIGGRAPH '08, pages 22:1–22:1, New York, NY, USA, 2008. ACM.

[GKM+15] S. Grottel, M. Krone, C. Müller, G. Reina, and T. Ertl. Megamol – a prototyping framework for particle-based visualization. *IEEE Transactions on Visualization and Computer Graphics*, 21(2):201–214, Feb 2015.

[HDS96] William Humphrey, Andrew Dalke, and Klaus Schulten. Vmd: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33 – 38, 1996.

[JAAA+15] Graham T Johnson, Ludovic Autin, Mostafa Al-Alusi, David S Goodsell, Michel F Sanner, and Arthur J Olson. cellpack: a virtual mesoscope to model and visualize structural systems biology. *Nature methods*, 12(1):85, 2015.

[KvK+18] D. Kouřil, L. Čmolík, B. Kozlíková, H. Wu, G. Johnson, D. S. Goodsell, A. Olson, M. E. Gröller, and I. Viola. Labels on levels: Labeling of multi-scale multi-instance and crowded 3d biological environments. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2018.

[LCD06] Thomas Luft, Carsten Colditz, and Oliver Deussen. Image enhancement by unsharp masking the depth buffer. *ACM Trans. Graph.*, 25(3):1206–1213, July 2006.

[MAPV15] Mathieu Le Muzic, Ludovic Autin, Julius Parulek, and Ivan Viola. cellview: a tool for illustrative and multi-scale rendering of large biomolecular datasets. In Katja B editor, *Eurographics Workshop on Visual Computing for Biology and Medicine*, September 2015.

[MKS+17] Peter Mindek, David Kouřil, Johannes Sorger, Daniel Toloudis, Blair Lyons, Graham Johnson, Meister Eduard Gröller, and Ivan Viola. Visualization

multi-pipeline for communicating biology. *IEEE Transactions on Visualization and Computer Graphics*, 24(1), 2017.

[MOBH11] Morgan McGuire, Brian Osman, Michael Bukowski, and Padraic Hennessy. The alchemy screen-space ambient obscurance algorithm. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, HPG '11, pages 25–32, New York, NY, USA, 2011. ACM.

[Nie92] Harald Niederreiter. *Random Number Generation and quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[PJH16] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[PvV05] T. Q. Pham and L. J. van Vliet. Separable bilateral filtering for fast video preprocessing. In *2005 IEEE International Conference on Multimedia and Expo*, pages 4 pp.–, July 2005.

[RGS09] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D '09, pages 75–82, New York, NY, USA, 2009. ACM.

[S+91] Peter Shirley et al. Discrepancy as a quality measure for sample distributions. In *Proc. Eurographics*, volume 91, pages 183–194, 1991.

[Sai16] Sakib Saikia. *Screen Space Reflections in Killing Floor 2*, 2016. `https://sakibsaikia.github.io/graphics/2016/12/25/Screen-Space-Reflection-in-Killing-Floor-2.html` (accessed May 27, 2018).

[Spl14] Thomas Splettstößer. *Diagram of the HIV virion*, 2014. `www.scistyle.com` (accessed May 27, 2018).

[ST90] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-d shapes. *SIGGRAPH Comput. Graph.*, 24(4):197–206, September 1990.

[TM98] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846, Jan 1998.

[Val14] Michal Valient. *Reflections and volumetrics of Killzone Shadow Fall. Presentation at SIGGRAPH Advances in Real-Time Rendering in Games course*, 2014. `http://advances.realtimerendering.com/s2014/valient/Valient_Siggraph14_Killzone.pptx` (accessed May 27, 2018).

[WLH97] Tien-Tsin Wong, Wai-Shing Luk, and Pheng-Ann Heng. Sampling with hammersley and halton points. *J. Graph. Tools*, 2(2):9–24, November 1997.

[WMW+16] Nicholas Waldin, Mathieu Le Muzic, Manuela Waldner, Meister Eduard Gröller, David Goodsell, Ludovic Autin, and Ivan Viola. Chameleon dynamic color mapping for multi-scale structural biology models. In *Eurographics Workshop on Visual Computing for Biology and Medicine*, 2016.