SOFT SHADOW MIP-MAPS

by

Yang Shen

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computing

School of Computing

The University of Utah

August 2016

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Yang Shen**

has been approved by the following supervisory committee members:

Cem Yuksel	, Chair	**03/16/2016** Date Approved
Charles D. Hansen	, Member	**03/16/2016** Date Approved
Erik L. Brunvand	, Member	**03/16/2016** Date Approved

and by **Ross T. Whitaker** , Chair/Dean of

the Department/College/School of **Computing**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

This document introduces the Soft Shadow Mip-Maps technique, which consists of three methods for overcoming the fundamental limitations of filtering-oriented soft shadows. Filtering-oriented soft shadowing techniques filter shadow maps with varying filter sizes determined by desired penumbra widths. Different varieties of this approach have been commonly applied in interactive and real-time applications. Nonetheless, they share some fundamental limitations. First, soft shadow filter size is not always guaranteed to be the correct size for producing the right penumbra width based on the light source size. Second, filtering with large kernels for soft shadows requires a large number of samples, thereby increasing the cost of filtering. Stochastic approximations for filtering introduce noise and prefiltering leads to inaccuracies. Finally, calculating shadows based on a single blocker estimation can produce significantly inaccurate penumbra widths when the shadow penumbras of different blockers overlap.

We discuss three methods to overcome these limitations. First, we introduce a method for computing the soft shadow filter size for a receiver with a blocker distance. Then, we present a filtering scheme based on shadow mip-maps. Mipmap-based filtering uses shadow mip-maps to efficiently generate soft shadows using a constant size filter kernel for each layer, and linear interpolation between layers. Finally, we introduce an improved blocker estimation approach. With the improved blocker estimaiton, we explore the shadow contribution of every blocker by calculating the light occluded by potential blockers. Hence, the calculated penumbra areas correspond to the blockers correctly. Finally, we discuss how to select filter kernels for filtering.

These approaches successively solve issues regarding shadow penumbra width calculation apparent in prior techniques. Our result shows that we can produce correct penumbra widths, as evident in our comparisons to ray-traced soft shadows. Nonetheless, the Soft Shadow Mip-Maps technique suffers from light bleeding issues. This is because our method only calculates shadows using the geometry that is available in the shadow depth map. Therefore, the occluded geometry is not taken into consideration, which leads to light bleeding. Another limitation of our method is that using lower resolution shadow mip-map

layers limits the resolution of the shadow placement. As a result, when a blocker moves slowly, its shadow follows it with discrete steps, the size of which is determined by the corresponding mip-map layer resolution.

CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor, Dr. Cem Yuksel, for the continuous support of my masters study and this research project. His guidance helped me in all the time of developing the algorithm and writing the thesis.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Charles Hansen and Dr. Erik Brunvand, for their insightful comments and the hard questions which incentivized me to widen my research from various perspectives.

I thank my colleagues Ian Mallett and Konstantin Shkurko for help developing the algorithm and writing the thesis. Also I thank my friend Yuntao Ou for sharing his high-end machine for generating some of the results.

Last but not least, I would like to thank my family for supporting me spiritually throughout writing this thesis and my life in general.

CHAPTER 1

INTRODUCTION

Shadows are important for visualizing 3D geometry. Over the past decades, computer graphics researchers have made great progress on shadow techniques to enhance the realism for interactive and real-time applications. In particular, soft shadow has received increasingly more attention due to its potential to better approximate the shadow in the real world. Using current generation graphics hardware, rendering soft shadow in real time is finally possible. However, efficiently computing accurate soft shadow still remains challenging [1, 4, 5].

Soft shadow is cast by an area light, and featured with penumbras. The amount of the shadow in penumbra area is proportional to the visible fraction of the light source. Imagine a scene with a light source and a piece of geometry. A fully lit geometry point receives light from the entire light source. On the other hand, a fully obscured geometry point cannot receive any light. If the geometry point receives part of the light, it falls in the penumbra region.

We favor soft shadow for its realism. While hard shadow is easy to calculate, for it is produced by point lights, hard shadow typically does not provide a realistic appearance. In the real world, any light source has some size, rather than being just a point. The realism of soft shadows is reflected in the following aspects [5]:

- Shadows help us understand relative object position and size in a scene. For example, without shadow we cannot perceive the position of an object in space from a single image.
- Shadows can also help us understand the geometry of a complex receiver.
- Shadows provide useful visual cues that help in understanding the geometry of a complex occluder.
- The softness of the shadow boundaries provide visual cues about the distance between the blocker and the receiver.

The challenge of efficiently computing soft shadow is to achieve high-quality shadow with real-time performance. Since the shadow value of every geometry point is determined by the visible fraction of the light source, a common approach to closely approximate the visible fraction is sampling the light source and dividing the total number of samples by the number of visible samples. However, this approach usually cannot perform fast enough for real-time rendering because sampling the light source is expensive. Another approach for computing plausible soft shadow is using a shadow map rendered from the light center to approximate soft shadows for all geometry points. Theoretically, this approach cannot generate physically accurate soft shadow because it assumes any point of the light source sees the geometry from the same position as the light center, while every point in the light sees the geometry differently. For a small light source, this assumption works well. For a large light source, however, this assumption does not hold. Nonetheless, this approach yields good performance because accessing shadow maps on the GPU is fast, and there are possibilities to achieve acceptable soft shadow quality.

Our work is based on soft shadow computation using shadow maps. We choose using a shadow map because of its high performance on the GPU and its potential of achieving high-quality soft shadows. We introduce the Soft Shadow Mip-Maps technique, which consists of three methods for overcoming the fundamental limitations of filtering-oriented soft shadows. First, we introduce a method for computing the soft shadow filter size for a receiver with a blocker distance. Then, we present a novel filtering scheme based on shadow mip-maps and trilinear filtering. Mipmap-based filtering uses shadow mip-maps to efficiently generate soft shadows using a constant size filter kernel for each layer, and linear interpolation between layers. Next, we discuss an improved blocker estimation approach. With the improved blocker estimaiton, we explore the shadow contribution of every blocker by calculating the light occluded by potential blockers. Hence, the calculated penumbra areas correspond to the blockers correctly. Finally, we discuss how to select filter kernels for filtering.

CHAPTER 2

BACKGROUND

Shadows created by any type of light source can be easily computed by ray tracing. In ray tracing, shadows are calculated by tracing shadow rays from the surface to the light sources, and penumbras can be calculated by distributing these secondary rays. The secondary ray can be traced to any point on the light source, not just a single light source location. The distribution of the shadow rays must be weighted according to the projected area and brightness of different parts of the light source. The number of rays traced to each region should be proportional to the amount of the light's energy that would come from that region if the light was completely unobscured. The proportion of lighted sample points in a region of the surface is then equal to the proportion of that light's intensity that is visible in that region [6].

Shadow mapping [7] was introduced to the computer graphics world by Lance Williams in 1978. The principle of the basic shadow mapping algorithm is fairly simple: for a scene featured with a point light, a few pieces of geometry, and a camera, creating shadows follows a binary test—whether a geometry point is visible from the light source. If it is visible from the light source, it is fully lit; otherwise, it does not receive any light, thus it falls in the shadow. Based on this principle, generating shadows is typically done in two passes:

- First, render the scene from the light source's view and store the depths to a z-buffer, or depth map.
- Second, render the scene from the camera's view. Shadows are created by testing whether a geometry point is visible from the light source, by comparing the value stored in the z-buffer and the geometry point's distance to the light source.

2.1 Filtering Shadows

A shadow map keeps depth values, therefore, they cannot be filtered using standard texture filtering method. When filtering shadows, we look up depth values from the shadow

map, then do shadow tests (depth comparisons) and filter the results of the tests.

Percentage-Closer Filtering (PCF) [8] is devised to filter shadow signals in order to reduce aliasings on hard shadow boundaries. The filtering is applied on the depth test of every sample in the filter kernel. Although PCF is mostly useful for reducing artifacts, the blurred shadows generated by small filter kernel sizes can be taken as visually plausible soft shadows for small light sources. For larger light sources, the corresponding filter kernel sizes become larger. Therefore, generating smooth soft shadows requires many accesses on the depth texture, which greatly decreases the performance. More importantly, using a constant filter size makes the penumbra the same size everywhere, which is not representative of realistic soft shadows. Therefore, it is important to adjust the filter kernel size according to the relative positions of the shadow caster and receiver. Figure 2.1 shows the soft shadow effects by PCF.

Prefiltering is an efficient approach for enhancing texture filtering performance. However, we cannot directly apply prefiltering for shadow-map filtering, because shadow-map filtering does not filter the depth values, but rather the depth comparison results. As depth comparision outputs binary results, it is not linear. We cannot change the computation orders if a filtering formula contains nonlinear functions. Thus, directly applying prefiltering in shadow-map filtering is not easy. There are two main ways of getting around this limitation [1]:

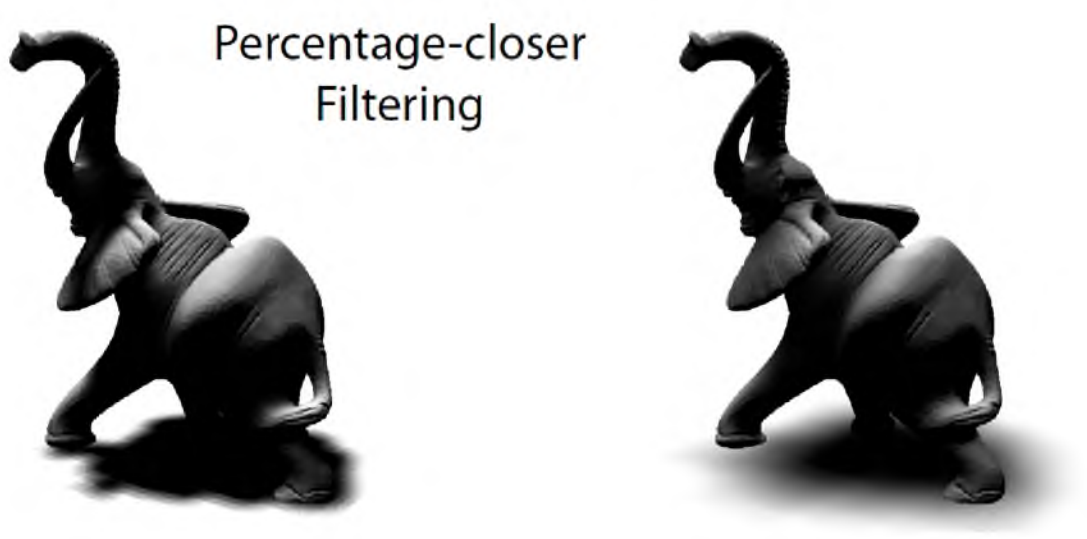

Figure 2.1: PCF resembles soft shadows from a small area light source (Left). For larger filter kernels, umbra is underestimated (right, shadow under left foot). [1]

- Interpret the depth samples as a distribution and then model the depth comparison statistically.
- Approximate the depth comparison function with a linear combination of functions that are linear in the depth component.

Variance Shadow Maps (VSM) [9] stores two components, the mean and mean square of a distribution of depths, instead of storing a single depth value, that can be easily precomputed in a manner similar to mip-mapping. With the two components, the variance over any filter region can be efficiently computed. With the variance, the blocker is approximated by the upper bound of one-tailed version of Chebyshev's inequality. VSM achieves a noticeable approximation under low depth complexity. However, light bleeding happens under high depth complexity, which is caused by small variance. Light bleeding can be avoided using Layered Variance Shadow Maps (LVSM) [10], but with high cost. Arbitrary rectangular filter kernels can be evaluated at runtime using summed area tables [11].

Convolution Shadow Maps (CSM) [12] approximates depth comparison by linearizing it with a weighted summation of basis terms in Fourier space. Thus, the coefficients of the Fourier basis terms can be precomputed. CSM also suffers from light bleedings, as appeared in VSM.

Exponential Shadow Maps (ESM) [13] replaces the Fourier basis terms in CSM with exponential basis terms. ESM achieves better quality and performance with less storage, compared with CSM and VSM. But the exponential approximation is not valid where the depth is in front of the reference depth, and such cases have to be handled by resorting to PCF.

Lauritzen and McCool [10] propose using the variance shadow map approach and applying it to depth maps that were warped by an exponential function, leading to a fast and robust solution that mostly avoids light bleedings.

2.2 Filtering-Based Soft Shadows

One of the commonly applied soft shadow techniques in interactive and real-time applications are filtering-based solutions [1]. The general idea is to use adaptive penumbra size to blur the shadow map to generate soft edges around the umbra. The penumbra size is computed based on a blocker distance estimation. Varying prefiltering methods are applied in different soft shadow techinques.

Percentage-Closer Soft Shadows (PCSS) [14] approximates visually plausible soft shadows with a small light source and performs at real-time rates. The idea is based on the observation that soft shadow becomes sharper as objects contact each other, and more blurry or softer the farther they are apart. PCSS uses a parallel planes approximation to estimate the penumbra size and utilizes the standard PCF to filter a single shadow map generated from the light center. The rendering pipeline of PCSS goes as follows:

- **Blocker search.** For a geometry point to be shaded (the "receiver"), a neighbor area surrounding the sample projected by the receiver in the depth map is searched to find the average blocker distance. The average blocker distance is the result of averaging neighbor samples closer to the light source than the receiver. The neighbor area is defined according to the light size, the distance from the receiver to the light, and the predefined near-plane distance.

- **Penumbrae estimation.** Using a parallel planes approximation, penumbrae width is estimated based on the light size and the blocker/receiver's distances to the light using similar triangle:

$$w_{Penumbra} = (d_{Receiver} - d_{Blocker}) \cdot w_{Light}/d_{Blocker} \tag{2.1}$$

- **Filering.** Now a standard PCF is applied to get blurred shadows. The kernel size of the filtering is proportional to the estimated penumbra width.

The underlying assumption of PCSS is that all points on an area light share the same distance to every geometry point. Such an assumption allows us to use one single shadow map to look up depth values viewed from any point of the area light, which is easy to program and results in good speed. Nonetheless, failure cases occur for larger light sources. When the light size becomes larger, this assumption is more likely to be violated and umbras tends to be underestimated. One major limitation of PCSS is that it involves many shadow map accesses in blocker search and filtering steps to get smooth result, which slows down the computation. To enhance the performance, many prefiltering soft shadow techniques have been devised.

Convolution Soft Shadow Map (CSSM) [15] is based on Convolution Shadow Map (CSM) [12], which approximates the traditional shadow test function with the convolution in Fourier space. The convolution prefiltering theory can be applied in both the average blocker depth step and the soft shadow computation step of PCSS framework. However, achieving high-quality soft shadows increases the number of Fourier basis terms to at least

four, so that large amounts of texture memory are required to store Fourier basis terms, making it less practical.

To overcome the memory problem of CSSM, Baoguang *et al.* [16] introduce Variance soft shadow mapping (VSSM), which takes the advantage of Variance shadow map (VSM) [9] that is based on a one-tailed version of Chebyshev's inequality, and requires a much lower amount of texture memory. Summed area variance shadow maps [11] evaluates VSM at runtime, which allows arbitrary rectangular filter kernels, hence it can also be plugged into the PCSS scheme to generate soft shadows.

Exponential soft shadow mapping (ESSM) [17] extends Exponential shadow maps (ESM) [13] to the PCSS scheme [14] to accelerate the computation by expressing the average depth of the shadow-map samples closer to the light as a convolution. ESSM yields high-quality soft shadows with high performance.

Selgrad *et al.* [2] introduce a prefiltering technique for multilayer shadow maps [18] to approximate accurate soft shadows in real time. The core idea is to reproduce the penumbra of an area light by blurring the occluder and taking a single sample, as shown in Figure 2.2. The blurred occluder is obtained by progressively merging the opacities of multilayer shadow maps. The performance of this method is not as fast as previous work, such as Exponential Soft Shadow Mapping (ESSM) [17], but this method captures shadows with multiple blockers and hence is well suited for complicated scenes. The quality is partially based on a set of parameters. Failure cases happen if the parameter setting is not proper. With too few base layers, the soft shadow quality of complicated scenes cannot be sufficed. Limiting the maximum filter size leads to very hard shadows. Additively merging fragments whose depth values are not close enough results in jagged shadow boundaries.

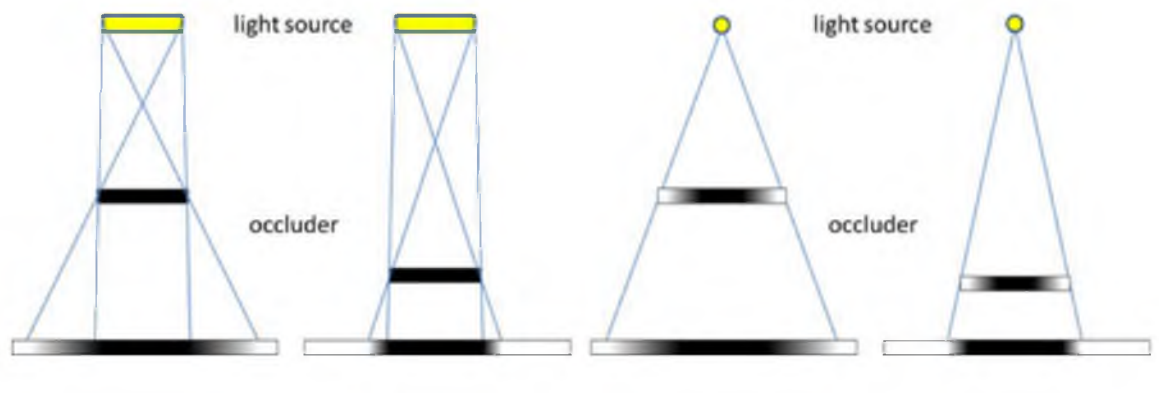

Figure 2.2: Umbra and penumbra of an area light source (left). The same shadow can be obtained using an unsharp occluder and a point light source (right). [2]

Targeting higher quality, Shen *et al.* [19] introduce an advanced filtering method and employ adaptive shadow map partitioning guided by a perceptual resolution prediction metric that exploits the typically low-frequency nature of penumbrae.

Schwarzler *et al.* [20] uses temporal coherence to avoid shadow recomputation in areas that were not changed hence it achieves high performance for fully dynamic scenes with PCSS.

To summarize, most filtering-oriented techniques are about improving the filtering performance. However, with the parallel planes approximation of PCSS, they share the same failure case with PCSS: larger light sources lead to overestimated umbrae.

2.3 Other Soft Shadowing Methods

Occlusion textures [21] approximate the occlusion in the scene with prefiltered occlusion textures. The visibility of the light source at a geometry point is estimated by accumulating the occlusion caused by each texture, using a novel formula based on probabilities. This method yields plausible soft shadows at high frame rates, and the performance is independent of light size. However, the blocking contribution of every fine object can be overestimated or missed.

Atty *et al.* [3] propose soft shadow mapping, which computes soft shadows by accumulating the discretized occluder areas. This method uses shadow map to approximate the blockers and unproject the shadow-map texels into world space. The resulting micro-occluders are then backprojected onto the light source, and by aggregating the occluded parts, the light's visibility is determined (see Figure 2.3). Since Soft Shadow Mapping combines scalar occlusion values for individual occluders, it suffers from the occluder overlapping artifact (see Figure 2.4).

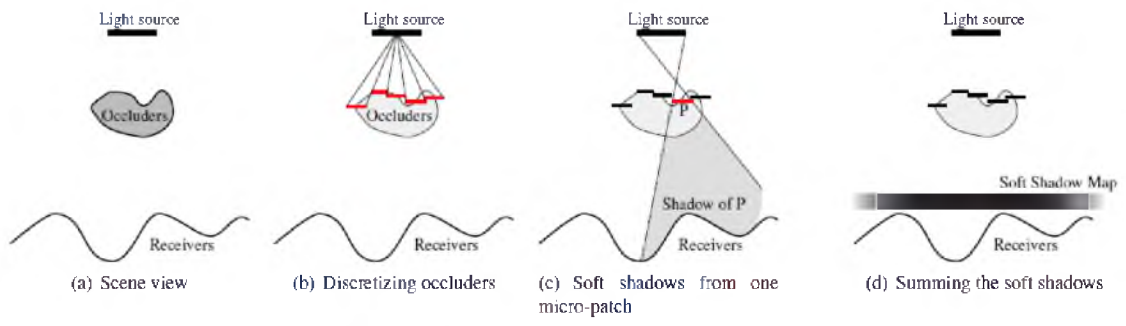

Figure 2.3: Algorithm overview of Soft Shadow Mapping. [3]

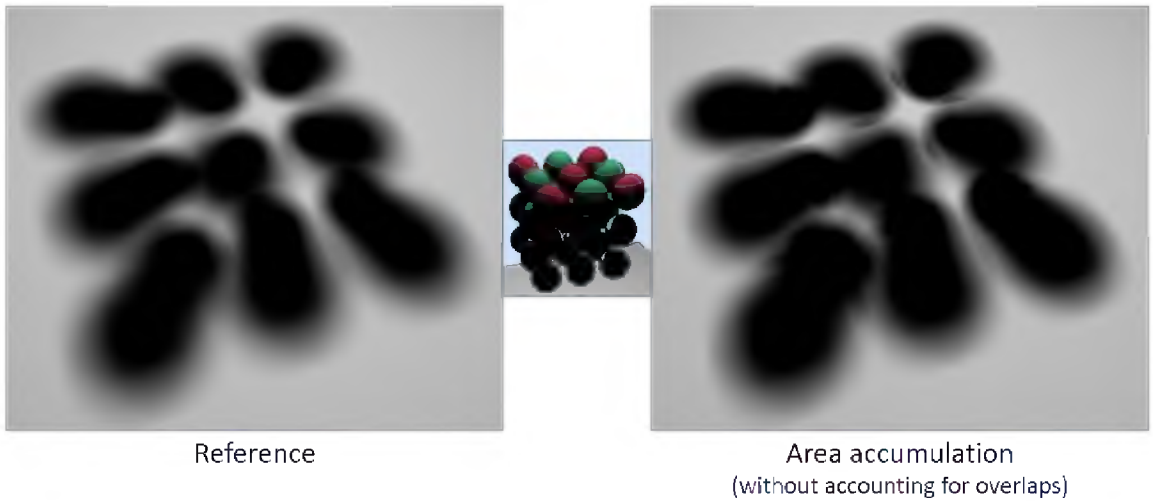

Figure 2.4: Overlapping artifact in Soft Shadow Mapping. [1]

Occlusion bitmasks [22] handles the occluder overlapping artifact by sampling the visibility rather than accumulating occluder areas. Soft shadow map is physically plausible and renders high-quality soft shadows at real-time rates. It is not physically accurate because the subset of occluders are actually those visible from the light center, not from points on the light source.

Wyman [23] introduced penumbra maps to approximate soft shadow in real time. Using object silhouette edges, as seen from the center of an area light, a map is generated containing approximate penumbral regions. Rendering requires two lookups in penumbra and shadow maps. Penumbra maps allow arbitrary dynamic models to easily shadow themselves and other nearby complex objects with plausible penumbrae.

Soft Shadow Volumes [24] build on shadow volumes for hard shadows and additionally employ penumbra wedges [25] to account for penumbra regions. The penumbra wedges are constructed for all silhouette edges and encompass the resulting penumbrae. For each covered pixel, the edge is backprojected onto the light, and finally, the covered light area of the corresponding occluders is computed using Green's theorem. However, the method suffers from the occluder fusion problem, leading to wrong results if occluders overlap. This is reduced by depth complexity sampling [26] where the light area is represented by several light sample points, and a counter for each sample point is maintained, keeping track of the number of occluders overlapping the sample point. This was originally developed for offline ray tracing, and a GPU variant exists as well [27].

Avoiding the high fill rate of shadow-volume-based methods, view-sample mapping [28]

inserts the view samples into an alias-free shadow map and then rasterizes the occluders' triangles into this map. For each shadow-map entry, an occlusion bitmask is maintained, and the light sample points overlapped by a triangle are set. Ultimately, the number of occluded points yields the amount of occlusion. This method not only produces accurate results, but is also reasonably fast for interactive applications. A related method is soft irregular shadow mapping [29], which makes some compromises concerning accuracy in favor of visual smoothness, abandoning point sampling of the light visibility and resorting to silhouettes instead of triangles.

CHAPTER 3

SOFT SHADOW MIP-MAPS

In this chapter, we introduce the Soft Shadow Mip-Maps technique by progressively replacing the PCSS stages with our methods. Our soft shadow rendering scheme is based on the filtering-based soft shadowing framework. Filtering-oriented techniques filter shadow maps with varying filter sizes determined by desired penumbra widths. Different varieties of this approach have been commonly applied in interactive and real-time applications. Nonetheless, they share some of the fundamental limitations of PCSS:

- Soft shadow filter size used by PCSS is not guaranteed to be the correct size for producing the right penumbra width based on the light source size,
- Filtering with large kernels for soft shadows requires a large number of samples, and stochastic approximations introduce noise, and
- Blocker estimation can produce highly inaccurate penumbra widths when the shadow penumbras of different blockers overlap.

We discuss three methods to overcome these limitations. First, we introduce a method for computing the soft shadow filter size for a receiver with a blocker distance. Then, we present a novel filtering scheme based on shadow mip-maps and trilinear filtering. Mipmap-based filtering uses shadow mip-maps to efficiently generate soft shadows using a constant-size filter kernel for each layer, and linear interpolation between layers. Next, we discuss an improved blocker estimation approach. With the improved blocker estimaiton, we explore the shadow contribution of every blocker by calculating the light occluded by potential blockers. Hence, the calculated penumbra areas correspond to the blockers correctly. Finally, we discuss how to select filter kernels for filtering.

3.1 Soft Shadow Filter Size

In this section, we introduce how to compute the soft shadow filter size for a receiver with a given blocker distance. Soft shadow filter size is important because it determines

whether the right region on the depth map is used for calculating the visibility of the light. The filter size estimation in PCSS does not provide the correct filter size that corresponds to an area light size because it is determined by scaling the estimated penumbra width with a user-defined factor. We replace the filter size estimation in PCSS with our approach such that the filter size of the average blocker distance can be accurately calculated. The approach we use is similar to the filter size computation used in [2].

Let us consider a simple scene geometry that consists of a horizontal planar light, a horizontal planar blocker and a horizontal planar receiver, where the light has a circular area. This simple scene perfectly fits the assumptions of PCSS with a single blocker that is a constant distance away from the light source. Consider the computation of the soft shadow filter size on the geometry point directly below the light center, as shown in Figure 3.1. We use this specific case for approximating the soft shadow filter size of points in the scene. In PCSS, the soft shadow filter size r_{pcss} is proportional to the penumbra size $r_{penumbra}$, estimated using

$$\begin{aligned} r_{penumbra} &= r_L \cdot (d_R - d_B)/d_B \\ r_{pcss} &= k \cdot r_{penumbra}, \end{aligned} \tag{3.1}$$

where k is a user-defined factor to get proper soft shadow filter size by scaling the penumbra size as shown in Figure 3.2. If we find a blocker for a receiver, the blocker could fully or partially occlude the receiver. We need to consider the potential blockers within the correct filter region, which is defined by filter size and the center position of the filter, for determining the visibility of the light. PCSS assumes the blocker creates a penumbra region. Then, the penumbra size is calculated using Equation 3.1 and scaled to get the filter size to determine whether the receiver is in the penumbra region. In this process, while the penumbra size is computed using the light size, picking the correct filter size for the given blocker distance relies on setting the correct scaling parameter.

In our method, we first compute the region where potential blockers exist at the blocker distance. When the geometry point is placed directly under the light center, the center of the filter region is also directly under the light center at the blocker distance. We use the blocker size r_B to represent the size of the region, as shown in Figure 3.3. r_B is determined by the blocker distance d_B, light size r_L, and receiver distance d_R, using similar triangles, such that

$$r_B = r_L \cdot (d_R - d_B)/d_R. \tag{3.2}$$

Now that we have the size of the blocker region, we can convert it to the filter size in texture space. Suppose the image plane for the depth map is placed at the blocker distance,

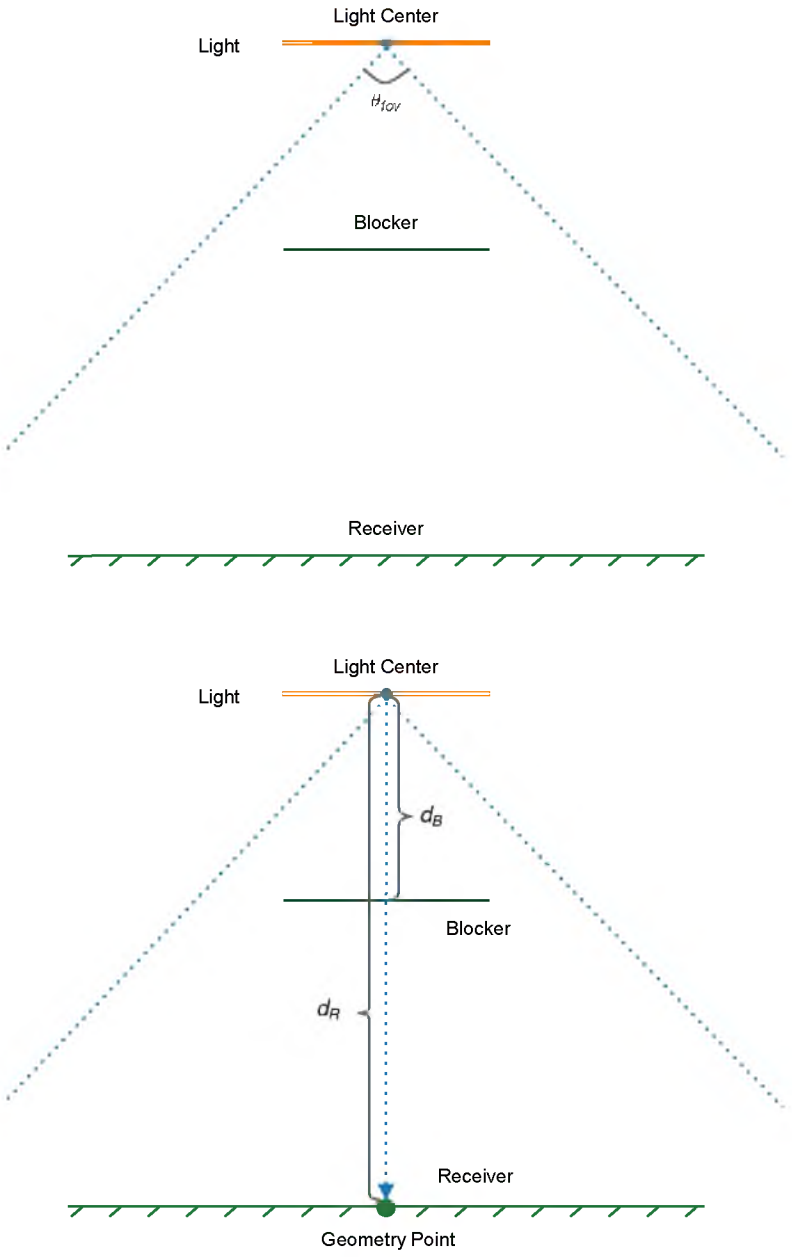

Figure 3.1: 2D visualization of the geometry with a horizontal planar light, a horizontal planar blocker, and a horizontal planar receiver.

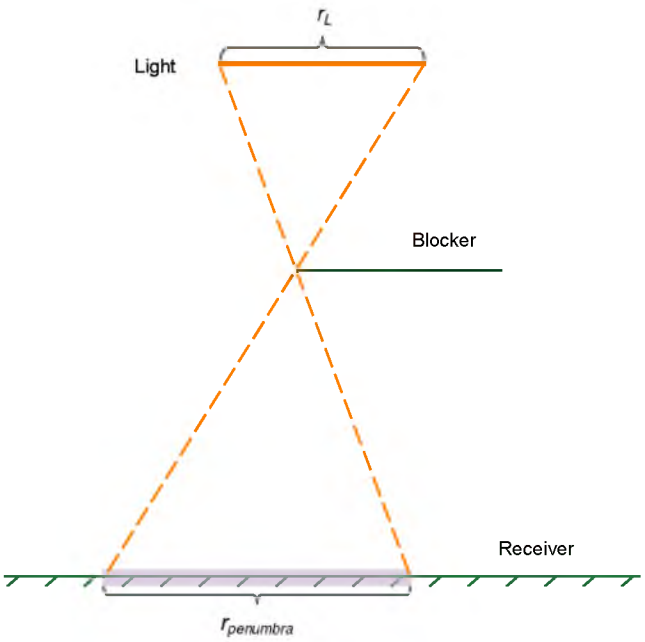

Figure 3.2: Soft shadow kernel size calculation in PCSS.

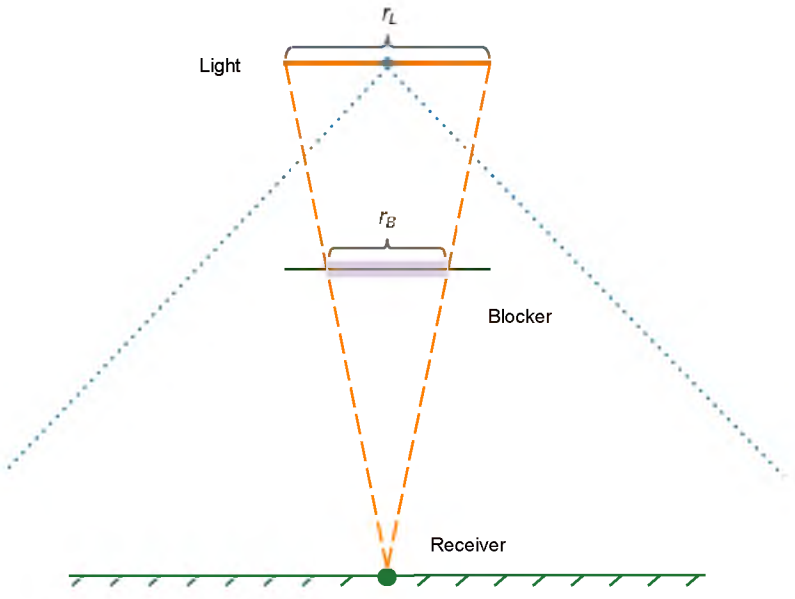

Figure 3.3: Size of the blocker region at the blocker distance.

as shown in Figure 3.4, the geometry size r_G at the blocker distance d_B is determined by d_B and the field of view of the light center θ_{fov}. The filter size r is then the ratio of r_B and r_G, such that

$$r = \frac{r_B}{r_G} = \frac{r_L \cdot (d_R - d_B)/d_R}{2 \cdot d_B \cdot tan(\theta_{fov}/2)}, \tag{3.3}$$

where θ_{fov} is the field of view angle used for rendering the depth map.

As mentioned in the beginning of the section, we use this specific case to calculate the soft shadow filter size of other geometry points. For receivers that are not placed right in front of the light center, the filter region on the depth map becomes an ellipse, as shown in Figure 3.5. Nonetheless, we use a circle for approximating the filter regions, which provides a good enough approximation in practice, especially for relatively small θ_{fov}.

3.2 Mipmap-Based Filtering

In this section, we replace the filtering of PCSS with a mipmap-based filtering scheme. Instead of using a variable filter size as in PCSS, we apply a filter with a constant filter kernel size on each mip-map layer to get shadows that have the desired penumbra width. We compute the visibility of the light on a receiver by linearly blending the filtered shadows of the two layers with filter sizes that are closest to the soft shadow filter size, which is computed using Equation 3.3. Mipmap-based filtering gets smooth shadows at stable frame rates because the constant filter kernel size yields approximately constant access and filtering time for the depth maps. On the other hand, in PCSS, filtering is performed on a single depth map. Therefore, depending on the filter size, it may need many samples for calculating smooth shadows. When stochastic filtering is used for limiting the number of samples, it leads to noise in the final shadow results. Our mipmap-based approach avoids these problems.

For computing the shadow on a receiver, we start with the blocker estimation of PCSS, which produces an average blocker distance estimation for a receiver. Then, we calculate the filter size with the average blocker distance using Equation 3.3. Finally, we use mipmap-based filtering to compute the shadow value.

In PCSS, a given receiver point is assigned an average blocker distance d_B using the blocker estimation. As shown in Figure 3.6, the blocker search region is defined by the light size r_L, the receiver distance, and the depth of the shadow map (user defined). A number of pseudo random samples are generated within the blocker search region. The depth values of all samples that are less than the receiver distance are averaged as the average blocker distance d_B.

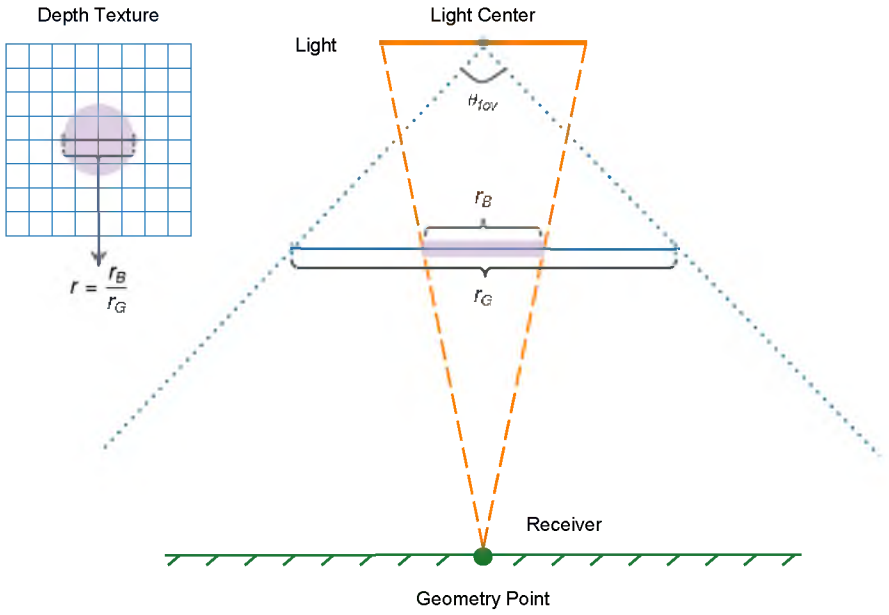

Figure 3.4: Soft shadow filter size using our method. The filter size is the ratio of the blocker size and geometry size at the blocker distance. The depth texture in the side shows the corresponding filter region.

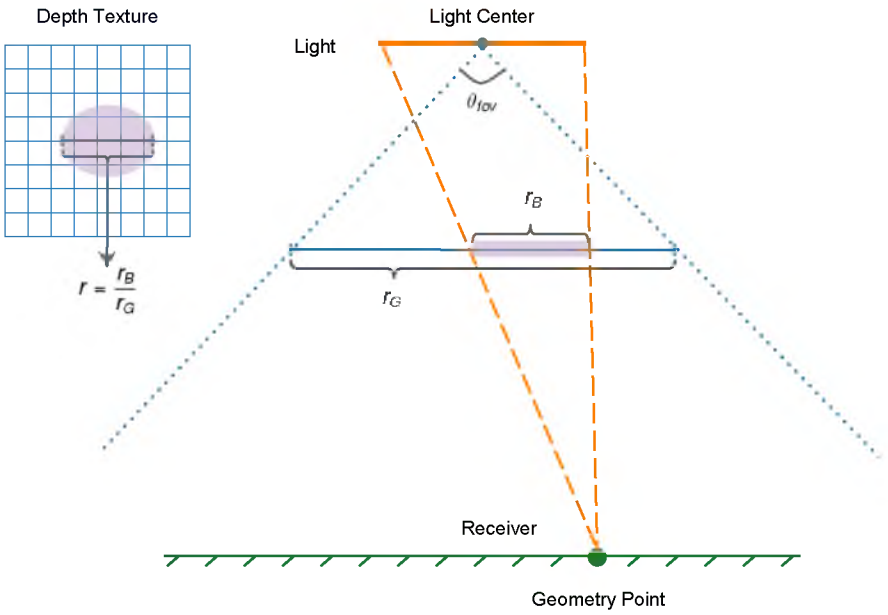

Figure 3.5: Soft shadow kernel size of receiver point not in front of the light center.

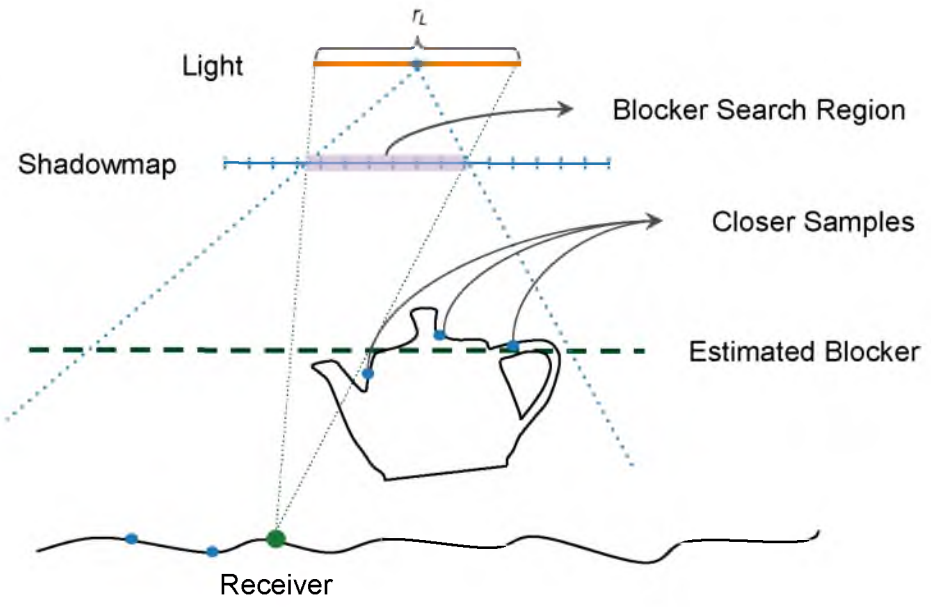

Figure 3.6: Blocker Estimation in PCSS.

With the average blocker distance d_B, we can calculate the filter size r using Equation 3.3. Then, we use r for computing the shadow using the shadow mip-maps.

Shadow mip-maps are a sequence of depth maps that have progressively lower resolutions. A single shadow mip-map layer is rendered in the same way as the depth map with the same resolution. The depth value of each texel is the distance between the closest geometry and the light center. Figure 3.7 shows the process of generating two layers of shadow mip-maps. In Figure 3.7(a), the shadow map is shown with a blue grid that represents texels. The light is treated as a camera with perspective projection placed at the light center. In Figure 3.7(b), the same geometry is rendered to a lower-resolution shadow map featured with orange grids. Figure 3.8 is a visualization of the two shadow mip-map layers together with the geometry, such that in light space, the dotted lines show how the texture centers correspond to the geometry.

3.2.1 Filtering with a Constant Size Kernel

The constant global filter kernel size is a user-defined value n, which defines an $n \times n$ texel region that is accessed during filtering. The value of n determines the filter size of each mip-map layer, such that

$$r_i = n \cdot h_i, \tag{3.4}$$

where r_i is the filter size of layer i, and h_i is the texel size (i.e., the reciprocal of layer i's resolution).

Now that we have each layer's filter size r_i, we can use r_i to determine whether layer i is one of the two relevant mip-map layers that should be used for shadow computation. As explained in Section 3.1, any blocker distance corresponds to a soft shadow filter size that can be computed using Equation 3.3. In the case of shadow mip-maps, each layer i corresponds to a particular depth d_i that is determined by its filter size r_i. Using Equation 3.3, the depth position d_i of a layer can be written as

$$d_i = \frac{r_L \cdot d_R}{2 \cdot d_R \cdot r_i \cdot \tan(\theta_{fov}/2) + r_L}. \tag{3.5}$$

Let us consider an example that the constant global filter kernel size $n = 2$ with 5 mip-map layers. In this case, the shadow mip-maps correspond to a layout as in Figure 3.9.

A given blocker with a particular blocker depth d_B falls somewhere in between two mip-map layers, as shown in Figure 3.10. We select the layers closest to the blocker for linearly blending their filtered shadows. The closest layers are highlighted with blue rectangles in Figure 3.11. Let a and b be the two corresponding layers right before and right after d_B, such that $r_a < r \leq r_b$, where $a < b$ and $r_b = 2r_a$, as shown in Figure 3.12.

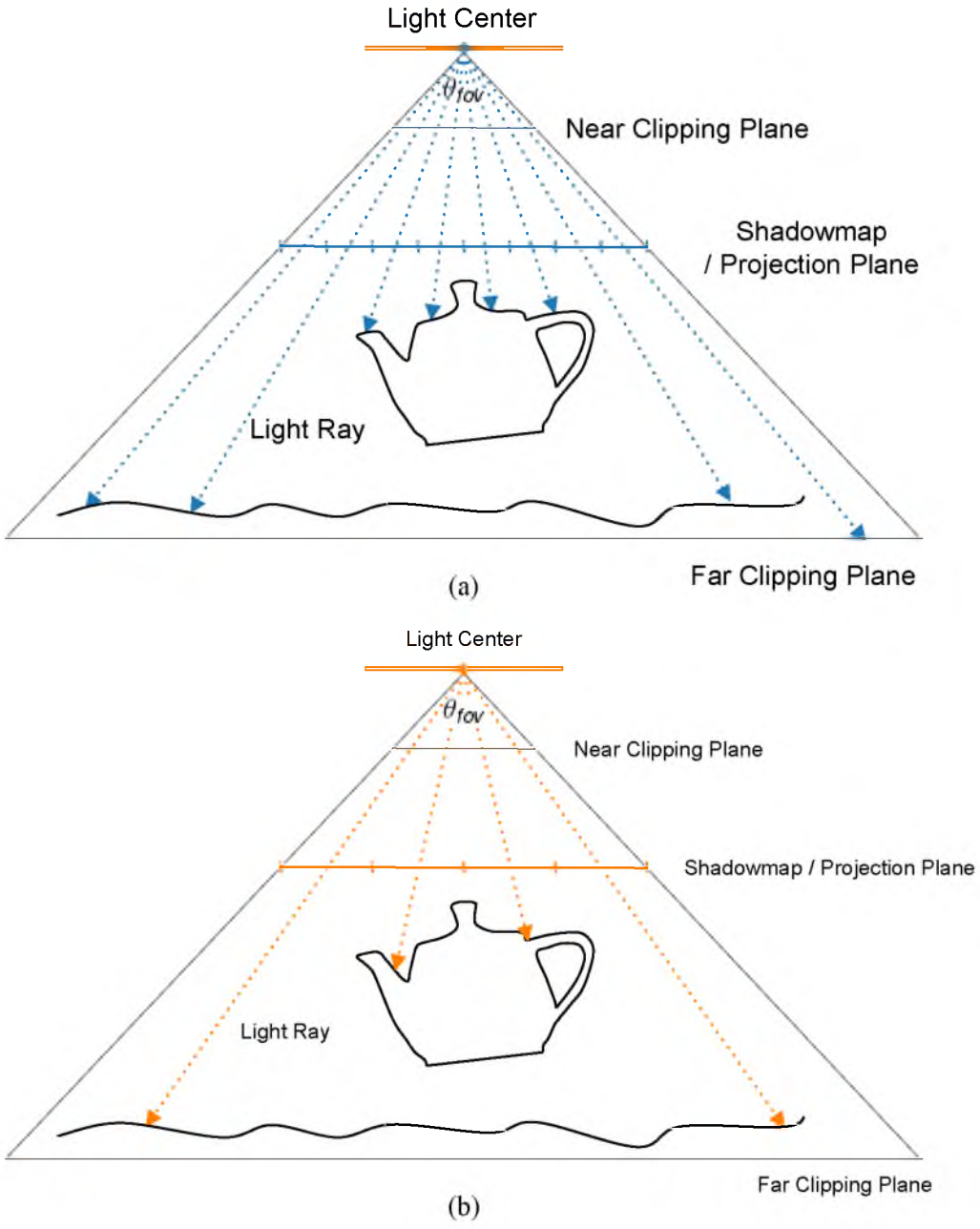

Figure 3.7: Generate shadow mip-maps from light center. The blue and orange grids are 2D shadow texture units. The dotted lines through the grids are light rays toward texel centers. The depth value of a texel is the distance of the first-hit geometry by the light ray to the light center. Black shapes are geometries. (a) High Resolution Layer. (b) Low Resolution Exmaple.

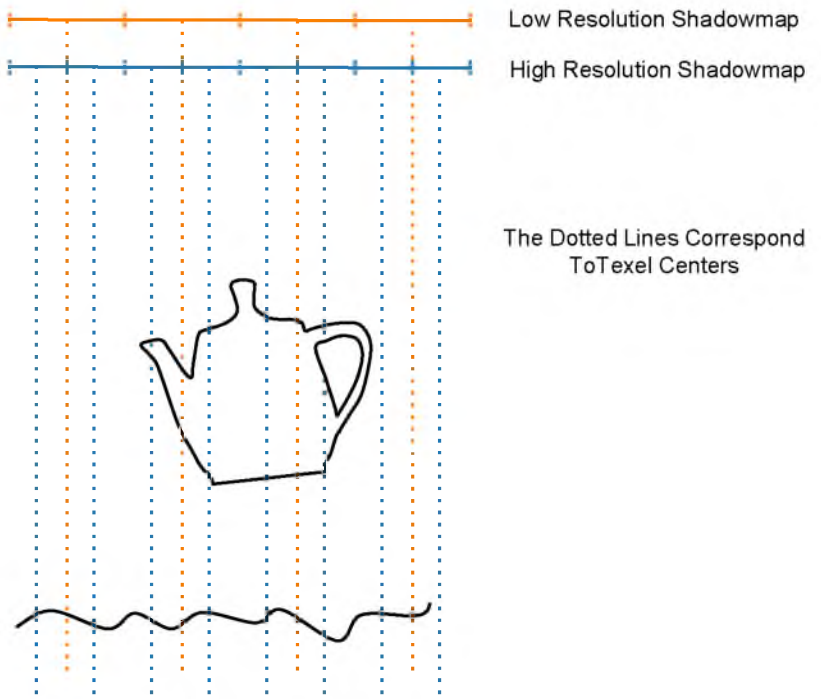

Figure 3.8: Shadow Mip-Maps.

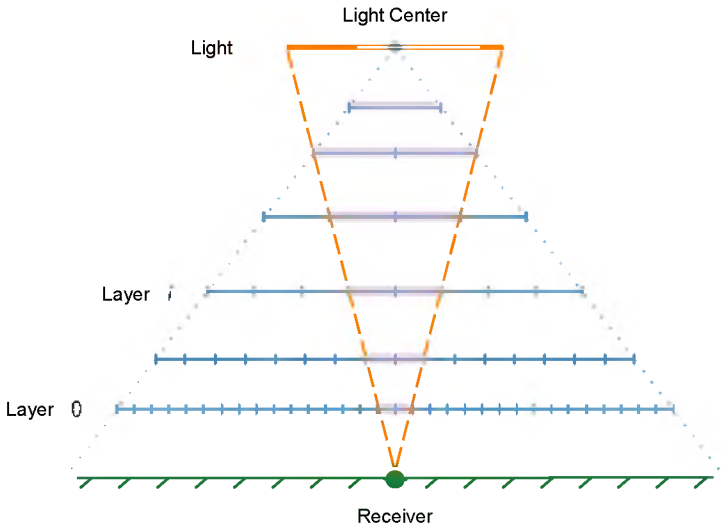

Figure 3.9: Shadow mip-maps layers are placed in their corresponding depth determined by their filter sizes. The filter region of each layer is highlighted with purple rectangle.

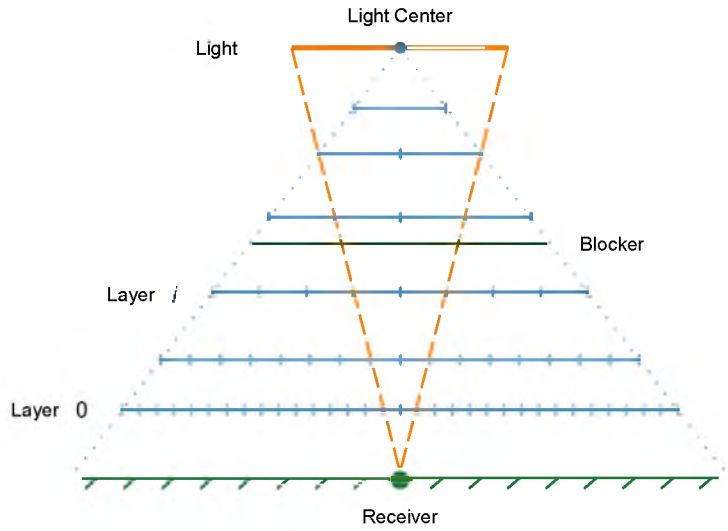

Figure 3.10: Shadow mip-maps layers and the single blocker are placed together.

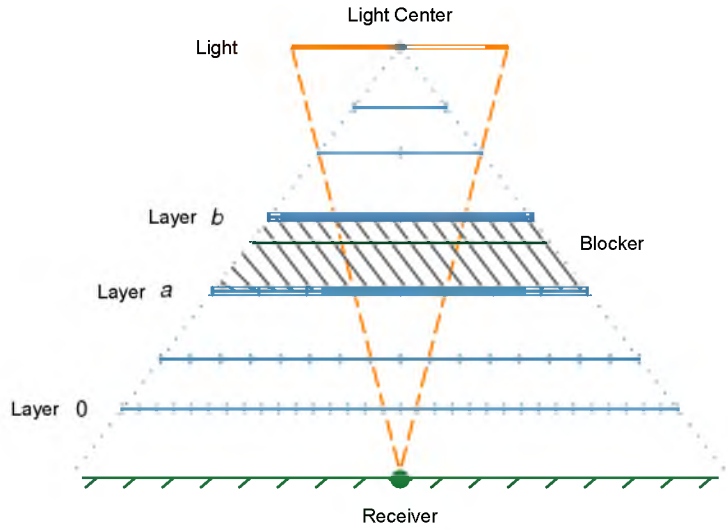

Figure 3.11: Choosing the highlighted layers to blend for the single blocker.

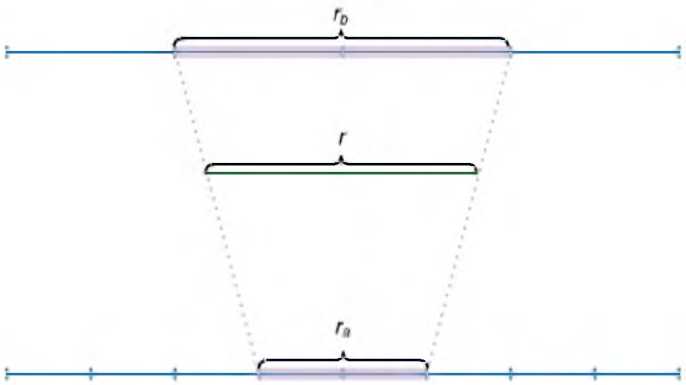

Figure 3.12: The closest shadow mip-map layers associated with filter size r.

Once the two relevant mip-map layers a and b are determined, we can filter each layer with its filter size r_i and a filter kernel. The filter kernel will be discussed in detail in Section 3.4. The filtering on each layer will result in blurred shadows with a penumbra width that corresponds to r_i. We use s_i as the filtering result, which is calculated using

$$s_i = \sum_{j \in r_i}^{texels} f_j s_{ij}, \tag{3.6}$$

where f_j is the filter value at texel j and s_{ij} is a binary shadow value that is computed using depth comparison of the receiver distance d_R and the texel's depth value d_{ij}, such that

$$s_{ij} = \begin{cases} 0, & \text{if } d_{ij} < d_R + d_{bias}; \\ 1, & \text{otherwise.} \end{cases} \tag{3.7}$$

3.2.2 Blending Shadow Mip-Maps

The shadow value s that corresponds to the filter size r can be approximated as a linear combination of their shadows s_a and s_b computed using layers a and b, such that

$$s = w_a \cdot s_a + w_b \cdot s_b, \tag{3.8}$$

where w_a and w_b are the weights determined by differences in the filter sizes using

$$w_b = \frac{r - r_a}{r_b - r_a} = \frac{2r - r_b}{r_b} \tag{3.9}$$

$$w_a = 1 - w_b = \frac{2r_a - r}{r_a}. \tag{3.10}$$

This way, mipmap-based filtering can be used for calculating the shadow on a receiver, due to a single blocker with a depth d_R that is estimated using the blocker estimation of PCSS. The depth d_R conresponds to a soft shadow filter size r, which is determined using Equation 3.3. The filter size r is then used for determining which two layers should be blended as well as the weights used for blending. Hence, the filtering result with a filter size r is approximated using a linear interpolation of the two layers with the closest filter sizes.

3.3 Improved Blocker Estimation

So far the only stage of PCSS that is not replaced is the average blocker estimation. In this section, we introduce an improved blocker estimation approach to replace the average blocker estimation in PCSS.

PCSS assumes there is a single blocker for every receiver, and searches a blocker region to compute the average blocker distance for the receiver. The average blocker distance

is then used for estimating the penumbra width for filtering. However, the single blocker assumption produces incorrect penumbra widths when the shadow penumbras of multiple blockers overlap. In this case, penumbra width near the overlapped shadow areas should be estimated by the depth of the blocker associated with the penumbra.

In our method, we explore the shadow contribution of every blocker via estimating the light obscured by possible blockers. Hence, the calculated penumbra areas correspond to the blockers correctly. Texels used for blocker estimation are the same as the mipmap-based filtering. In PCSS, accurate average depth needs a large number of samples, which is expensive.

Let us take an example to illustrate the improved blocker estimation. As shown in Figure 3.13, while estimating the blockers, we treat every texel in the filter region of each layer as a blocker. The texel's blocker distance is the depth value stored in itself: d_{ij}. Assume that we are looking at texel j of layer i (highlighted by the red rectangle in Figure 3.13), its blocker distance d_{ij} may be different from the depth range of layer i. Because each layer's depth is determined based on its kernel size r_i, d_{ij} is taken to determine how much light is obscured by the texel.

We use s_{ij} to represent the obscured light by texel j. If d_{ij} is greater than the receiver distance d_R plus a small user-defined shadow map bias value d_{bias}, the texel does not occlude

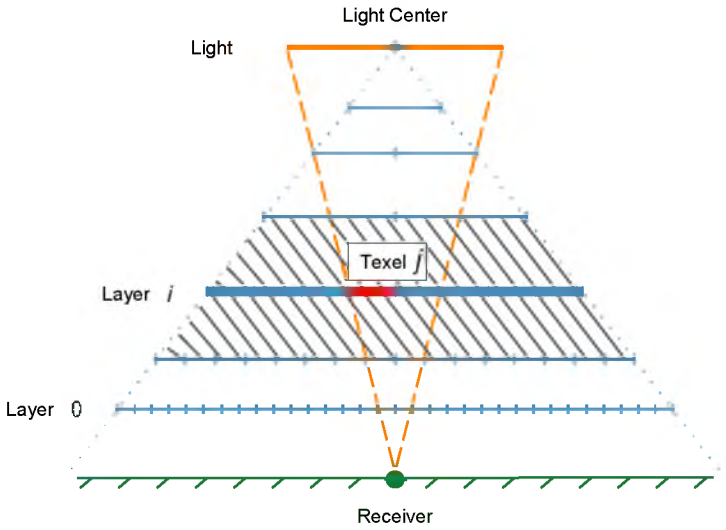

Figure 3.13: The Improved Blocker Estimation.

any light coming to the receiver and s_{ij} is 0. Otherwise, we measure the texel's influence to the receiver by calculating a texel weight w_{ij} that tells where the real geometry depth stored in the texel falls in the depth range of layer i. For every layer, we define a *depth region* within which the layer has influences. The depth region of layer i is between the filter depth of the previous layer $(i-1)$ and the filter depth of the next layer $(i+1)$. The filter depth of a layer i is defined in Equation 3.5.

If d_{ij} exceeds the depth range, the texel does not have any influence to the receiver, w_{ij} is 0 and the shadow cast by the corresponding blocker is handled by another mip-map layer. Otherwise, w_{ij} is calculated by its soft shadow filter size and the filter sizes of the layers that created the depth range where d_{ij} falls, such that

$$w_{ij} = \mathbf{w}(r \mid r_0 < r \leq r_{m-1}) = \begin{cases} \frac{2r_i - r}{r_i}, & \text{if } r > r_i; \\ \frac{2r - r_i}{r_i}, & \text{if } r \leq r_i; \\ 0, & \text{otherwise.} \end{cases} \tag{3.11}$$

where m is the number of shadow mip-map layers. The first and last mip-map layers need be handled differently. For very small filter size $r < r_0$ (smaller than the filter size of layer 0 with the highest resolution), we determine the texel weight using

$$\mathbf{w}(r \mid r \leq r_0) = \begin{cases} 0, & d_R < d_{ij} + d_{bias}; \\ 1, & \text{otherwise.} \end{cases} \tag{3.12}$$

On the other hand, when the filter size r is so large that it exceeds the filter size of the bottom mip-map layer, r_{m-1}, we consider the receiver blocked by the texel and we use

$$\mathbf{w}(r \mid r > r_{m-1}) = 1 \tag{3.13}$$

In summary, the texel weight w_{ij} can be calculated differently in three cases:

$$w_{ij} = \begin{cases} \mathbf{w}(r \mid r_0 < r \leq r_{m-1}), & \text{if } r_0 < r \leq r_{m-1}\ ; \\ \mathbf{w}(r \mid r \leq r_0), & \text{if } r \leq r_0\ ; \\ \mathbf{w}(r \mid r > r_{m-1}), & \text{if } r > r_{m-1}\ . \end{cases} \tag{3.14}$$

This way, the shadow contribution of texel j of layer i s_{ij} is formulated as

$$s_{ij} = w_{ij} = \mathbf{w}(r) \tag{3.15}$$

Now for a receiver, we have the amount of lighting obscured by every texel in the filter region of the layers, we can combine them as the total blocked light to the receiver. Let s

be the visibility of the receiver, representing the amount of lighting coming to the receiver, such that

$$s = 1 - s_B, \tag{3.16}$$

where s_B is the obscured light from possible blockers. s_B is calculated using all mip-map layers i and all texels j within the filter radius r_i using

$$s_B = \sum_{i}^{layers} \sum_{j \in r_i}^{texels} f_j s_{ij}. \tag{3.17}$$

In this equation, texels in the filter region of each layer are taken as blockers to determine the obscured light, r_i is the filter size, s_{ij} is the amount of blocked light from texel j of layer i, and f_j is the filter kernel that is used to weight s_{ij} accordingly using the real coordinate of the texel and receiver on layer i.

3.4 Filter Kernels

In this section, we talk about varying kernels for filtering. Different filter kernels may result in shadows that have intensively different qualities.

We do filtering to smooth jagged shadow edges. When selecting filter kernels, we need to carefully consider the following properties:

- The volume under the the filter kernel function should add up to 1 in the filter region: $\sum_{j \in r_i}^{texels} f_j = 1$;
- The filter kernel should give us smooth results.

While filtering shadows, we multiply a texel's shadow value with the volume under the kernel function in the domain (texel area).

Box filtering kernel supplies a linear filtering using an average operation on the texels covered by the filter region, as shown in Figure 3.14. The volume of the box is 1. When filtering each texel, the texel's filter value is the volume under the box kernel in the covered texel area, as shown in Figure 3.15. Box filtering kernel is fast and easy to compute, but it is not effective in hiding pixelation artifacts unless a large enough filter size is used.

Similar to box filtering kernel, pyramid filtering kernel also supplies a linear filtering on the filter region. Pyramid filtering kernel produces smoother filtering results than the box kernel because it uses a pyramidal volume to filter the shadow map, as shown in Figure 3.16.

Other filter kernels that use the kernel value at the texel center to filter the shadow map, instead of using the volume under the filter kernel on the filter region, may also provide

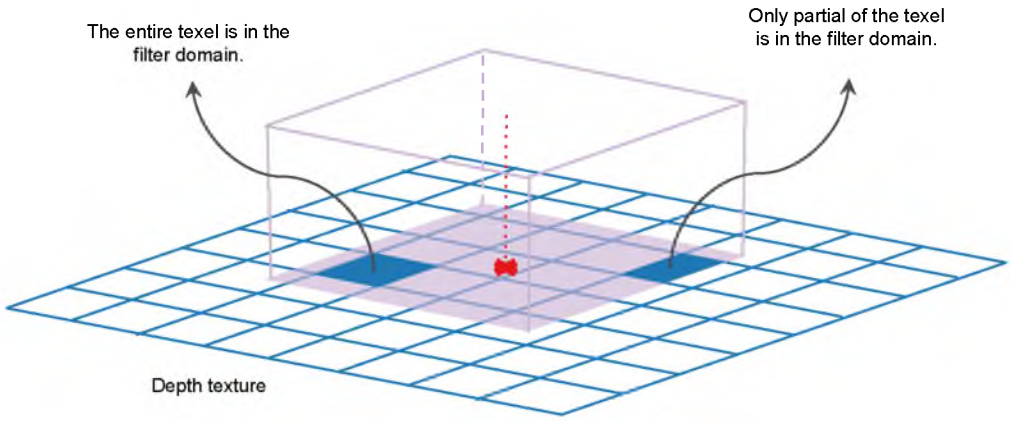

Figure 3.14: Box filtering on depth texture.

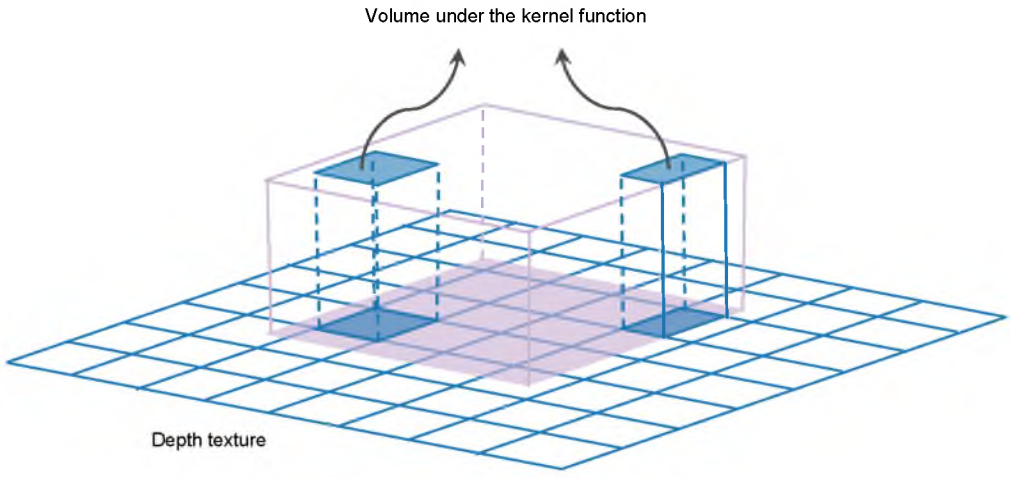

Figure 3.15: Volume under the filter kernel.

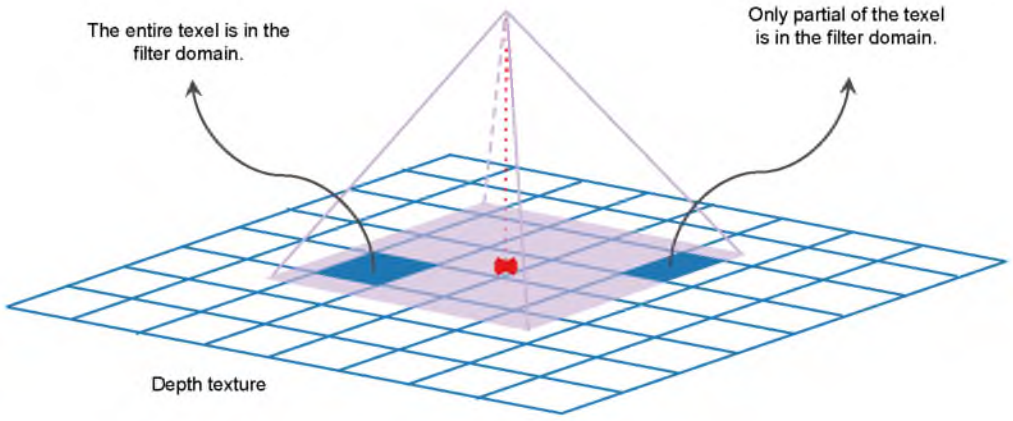

Figure 3.16: Pyramidal filtering on depth texture.

acceptable filtering results (smooth shadow), but they might require large filter sizes. One difficulty with this approach is to make sure that the filter values used for each texel add up to 1.

3.5 Implementation

The implementation of the Soft Shadow Mip-Maps technique is straightforward. The algorithm runs in two passes: first, we render the scene from the light view and generate the shadow mip-maps; second, we render the scene from the camera view. When generating the shadow mip-maps, we render the scene to the first mip-map layer to store the depths, then downsampling the first mip-map layer. During downsampling, each texel of the lower resolution layer is assigned by taking the minimum texel value of the four corresponding texels of the higher resolution layer. When rendering the scene to the camera view, we evaluate the visibility of a fragment in the fragment shader using our algorithm, then shade the fragment with its visibility. Algorithm 1 shows the overview of the implementation.

We use world space to store the depths and calculate the fragment visibility to avoid transformations of light size. When storing the world space depths to the first shadow mip-map layer, we normalize the depths using a normalization factor while rendering the scene from the camera view, and we retrieve the world space depths by applying the reciprocal of the normalization factor to the depths in the shadow mip-map layers.

The depth bias for each shadow mip-map layer needs to be carefully adjusted to avoid shadow acne and artifacts, as the shadow mip-map layers have different resolutions. We

Render scene from light source:
  Create the first layer of the soft shadow mip-map;
  Create the shadow mip-maps through downsampling;
Render scene from the camera, in the fragment shader:
  Initialize the total blocked light: $s_B = 0$;
  for *shadow mip-map layer i:* **do**
    for *texel j in the filter region:* **do**
      Compute the weight w_{ij} using Equation 3.14;
      Filter the weight and accumulate it to the total blocked shadow s_B using Equation 3.17;
    end
  end
  Compute the visibility s using Equation 3.16;
  Shade the fragment using the estimated visibility s;

Algorithm 1: Implementation of Soft Shdow Mip-Maps

suggest to first apply a constant depth bias to get rid of shadow acne. While shadow acne can be easily removed by a constant depth bias, shadow artifacts can happen in lower resolution shadow mip-map layers because the shadow-map texels cover large ranges of depths. This kind of artifacts often become obvious when polygons are almost parallel to the light direction. Second, to reduce the shadow artifact, we apply a slope-scaled depth bias for the fragment's depth for each shadow mip-map layer. The slope of a fragment is the tangent of the angle between the normal vector and the light direction at the fragment. It is used to avoid large depth bias on contact shadows. We accordingly adjust the slope by scaling the slope based on the texel size of the shadow mip-map layer. Specifically, we scale the texel size by a constant value, with which we scale the slope to get the depth bias of a shadow mip-map layer. Algorithm 2 shows the second pass with depth bias steps.

Render scene from the camera, in the fragment shader:
Initialize the total blocked light: $s_B = 0$;
Compute the fragment's distance to the light source: d_R;
Apply constant depth bias $d_{constbias}$ to avoid shadow acne: $d_R = d_R - d_{constbias}$;
for *shadow mip-map layer i:* **do**
    Apply slope scaled depth bias $d_{adaptivebias}$ to avoid shadow artifacts: $d_{R_i} = d_R - d_{adaptivebias}$;
    for *texel j in the filter region:* **do**
        Compute the weight $w_{ij} = \mathbf{w}(r(d_{R_i}))$ using Equation 3.14;
        Filter the weight and accumulate it to the total blocked shadow s_B using Equation 3.17;
    end
end
Compute the visibility s using Equation 3.16;
Shade the fragment using the estimated visibility s;

Algorithm 2: Second Pass with Depth Bias Steps

CHAPTER 4

RESULTS AND LIMITATIONS

In this chapter, we demonstrate the shadow quality produced by Soft Shadow Mip-Maps and compare the performance of Soft Shadow Mip-Maps with Variance Soft Shadow Mapping. All the experiences were run on a PC with a quad-core 2.60GHz Intel i7-6700HQ CPU, an NVIDIA Geforce GTX 970M, and 16GB memory. For all the examples, we use a small quad-shaped light source as the basic configuration for the scenes. Box filtering is used to filter the shadow mip-map layers as it brings small computation overhead yet produces competitive results.

4.1 Quality

In this section, we first show the occluded light by each layer. Then, we discuss how the shadows in the penumbra region differ from ideal penumbra shadows using a 2D example. We also examine how filter size influences the shadow quality of Soft Shadow Mip-Maps. Finally, we compare our results with Percentage Closer Soft Shadows and ray traced soft shadows.

4.1.1 Occluded Light

In this section, we illustrate how mip-map layers occlude the receiver by showing the occluded lights from the layers. In Figure 4.1, the image on the top shows a simple scene with increasing penumbra widths. The scene is configured with a planar ground on which stands a tall box. The ground and box are respectively the receiver and occluder in our context. On the contact surfaces of the receiver, the penumbra widths are relatively sharper than other penumbra areas of the receiver. As the blocker gets farther from the receiver, the penumbra width increases progressively. The three images on the bottom show occluded lights from related mip-map layers. From left to right, the mip-map layer is respectively the first, second, and third. By accumulating the occluded lights from the mip-map layers and reverting them to visible lights, we get the result in the image on the top.

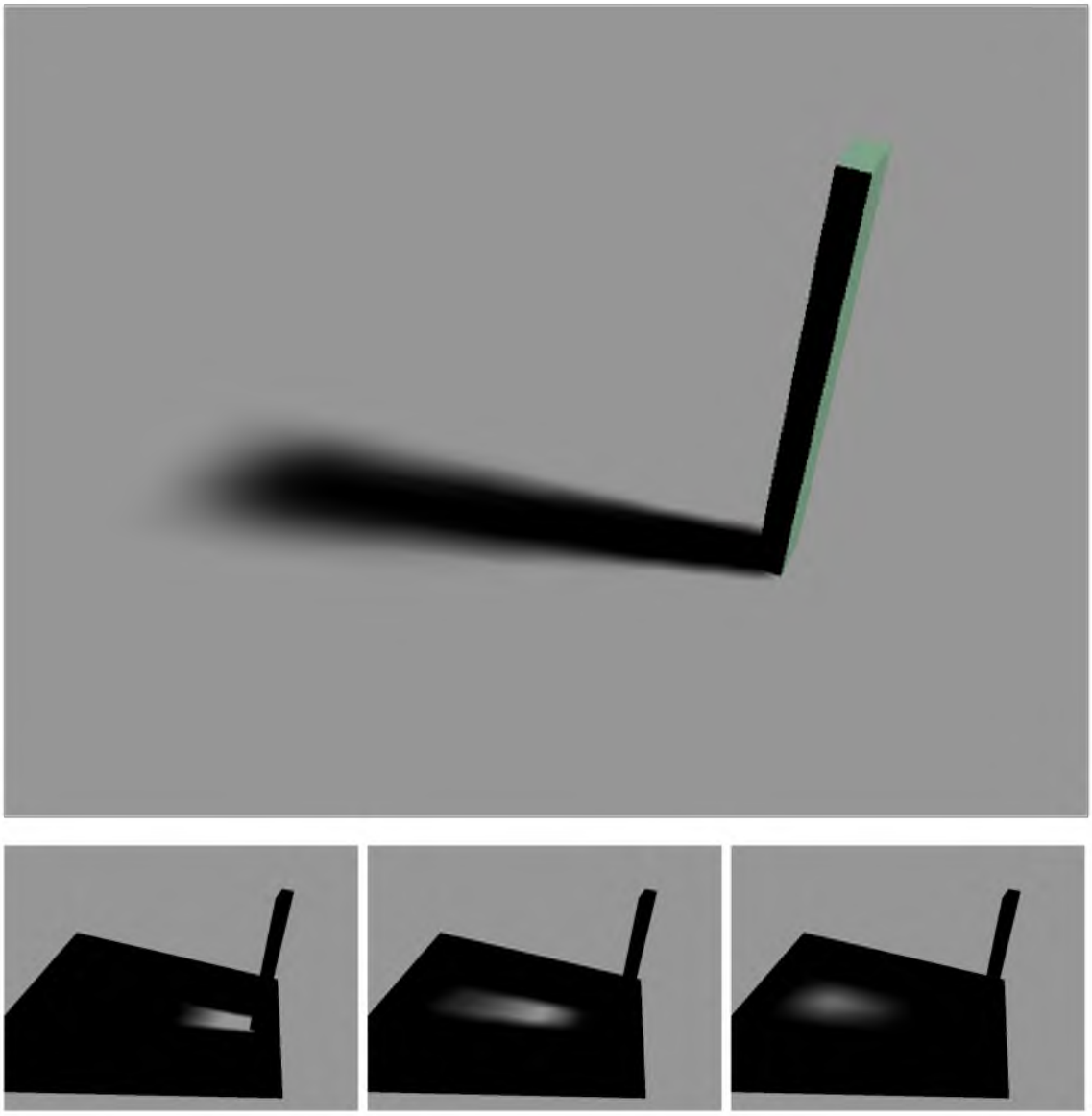

Figure 4.1: Occluded lights from soft shadow mip-map layers. Top: the scene with progressively increasing penumbra width. Bottom: occluded lights from the first, second, and third mip-map layers.

A mip-map layer is taken to compute shadows for a specific geometry point when the blocker depth of the geometry point falls in the layer's depth region. In this example, the first mip-map layer (Figure 4.1 bottom-left) is taken to compute shadows for the surface around the contact area because the blocker depths around contact area are among the largest depth region of the scene. So the corresponding depth region on the mip-map layers is reflected on the first layer's depth region. As the blocker depth decreases and enters the second mip-map layer's depth region, the second mip-map layer is taken to compute shadows (Figure 4.1 bottom-center). Similarly, the third mip-map layer is taken to compute shadows when the blocker depth enters the third mip-map layer's depth region (Figure 4.1 bottom-right). Since the minimum blocker depth stays in the third mip-map layer's depth region, no consecutive mip-map layers are needed for computing shadows.

The amount of occluded light by a mip-map layer is determined by the distance between the blocker depth and the mip-map layer's depth. In Figure 4.1 the left image on the bottom, the first mip-map layer's impact of occluded light stays stable in the beginning, because the blocker depths are larger than the first layer's depth, in which condition the first layer fully occludes the light (Equation 3.12). When the blocker depth gets smaller than the first layer's depth and then exits the first mip-map layer's depth region, the first mip-map layer's impact of occluded light gradually decreases to zero. As for the second mip-map layer (Figure 4.1 center image on the bottom), its impact of occluded light increases first as the blocker depth gets closer to the second mip-map layer's depth, and then decreases to zero as the blocker depth gets smaller than its depth and finally exceeds the second mip-map layer's depth region. The third mip-map layer's impact of occluded light is similar to the second mip-map layer. As shown in Figure 4.1 the right image on the bottom, the amount of occluded lights increases first and disappears where the geometry points are not occluded.

4.1.2 Penumbra Difference

Our results cannot achieve zero penumbra difference when compared against ray traced shadows, because the penumbra regions of the layers occluding the receiver do not fully overlap. When the occluded light in the penumbra regions is added up together to get the total occluded light, the amount of shadow in the final penumbra region does not increase uniformly from the umbra boundary to the fully lit (no shadow) boundary. However, ideally the shadow amount should increase linearly. Figure 4.2 shows a 2D version of the penumbra changes of our algorithm. From top to bottom, the blue and green lines with little grids are two consecutive mip-map layers L_i and L_{i+1}. Below these layers, the black lines represent

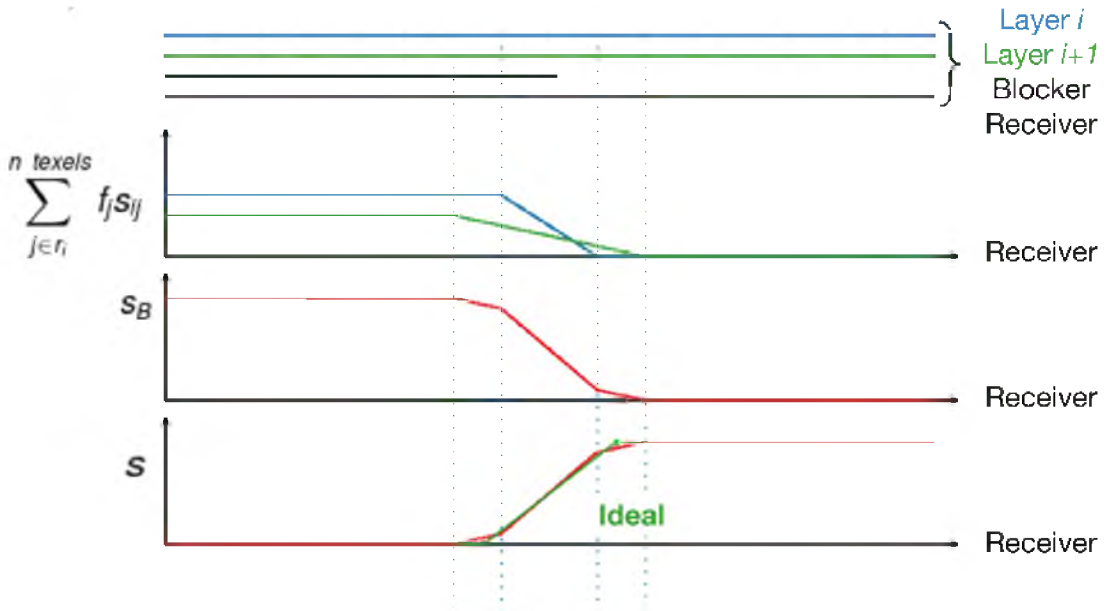

Figure 4.2: Shadow curve in 2D with linear filtering.

the blocker and receiver geometries. Below them, the three graphs show blocked lights by each layer, total blocked light, and final shadow on the receiver. The ideal shadow amount is shown as the green lines on the bottom graph.

Note that our shadow approximation deviates from the ideal shadow because our penumbra width is wider than the ideal penumbra width, and the shadow amount change rate is not constant in the penumbra region. The penumbra width of our algorithm is wider than ideal penumbra width because the penumbra width is determined by larger filter region of the mip-map layers. In this case, it is L_{i+1}'s filter region. Since we weight the shadow amount in L_{i+1}'s filter region with a nonzero coefficient, shadows are produced in the entire filter region. However, the ideal penumbra width is smaller than L_{i+1}'s filter size, so that shadows should not cover L_{i+1}'s entire filter region. The shadow amount change rate of our algorithm does not keep the same in the penumbra area because the shadow amount is calculated by linearly combining shadows from the mip-map layers, which are linear functions with different change rates in subdomains of the penumbra region. While the shadow amounts of L_i and L_{i+1} are both linear functions in their own filter regions, their linear combination causes changes of the rate because their filter regions do not fully overlap. Specifically, L_i's filter region is half of the size of L_{i+1}'s filter region, where the shadow amount change rate in the overlapped filter region is the sum of L_i and L_{i+1}'s

shadow amount change rates. For the other half filter region of L_{i+1}, the shadow amount change rate is L_{i+1}'s change rate, because L_i's shadow amount does not change in these regions.

In Figure 4.3, the left image shows the shadow rendered by our method, and the right image is the ray traced result. The bottom image shows the penumbra difference of our result and the ray traced result, which corresponds to the third graph of Figure 4.2. In the penumbra difference image, the red and green area in the floor mark the area where the shadows have differences. Specifically, the green area means the ray traced shadow is darker than our result, and the red area means our result is darker than the ray traced result. The intensity is 8x of the shadow difference.

4.1.3 Filter Size

We filter the shadow-map layers to smooth the jagged shadow edges and enhance the shadow quality. Filter size determines which shadow-map layers occlude the receiver. With a small filter size, the layers occluding the receiver come from low-resolution mip-map layers, so the occluded light from each layer is relatively coarse. In Figure 4.4 top, the left image shows our result of shadows cast by spheres, and the right image shows the ray traced reference. Because of the small filter size, The penumbra widths do not perfectly reflect the sphere's boundaries.

Using a larger filter size, the layers occluding the receiver have higher resolutions, so the occluded light from each layer is smoother and approaches the ray traced reference. In Figure 4.4 middle, the penumbra width of our result is closer to ray traced penumbra width compared with the penumbra width in Figure 4.4 top.

However, in practice we cannot use a very large filter size. This is because when the filter size is too large, the layer occluding the receiver would be layer 0 of the shadow mip-map which has the highest resolution. In this case, the umbra area would be underestimated and penumbra area would be overestimated, since very large filter size causes small depths of the mip-map layers, so that most geometry depths would be larger than the largest layer depth, i.e., layer 0's depth. When a geometry depth is larger than layer 0's depth, the texels of layer 0's filter region have full impact of the occluded light on the geometry point, according to Equation 3.12. This would bring unnecessary extra lights to the geometry point. Generally we should avoid making most geometry depths larger than the largest layer depth, which causes overestimated penumbra and expensive computation. The ideal filter size should generate shadow mip-map layer depths evenly distributed in the geometry depths, so we could compute shadows for varying penumbra widths without taking too

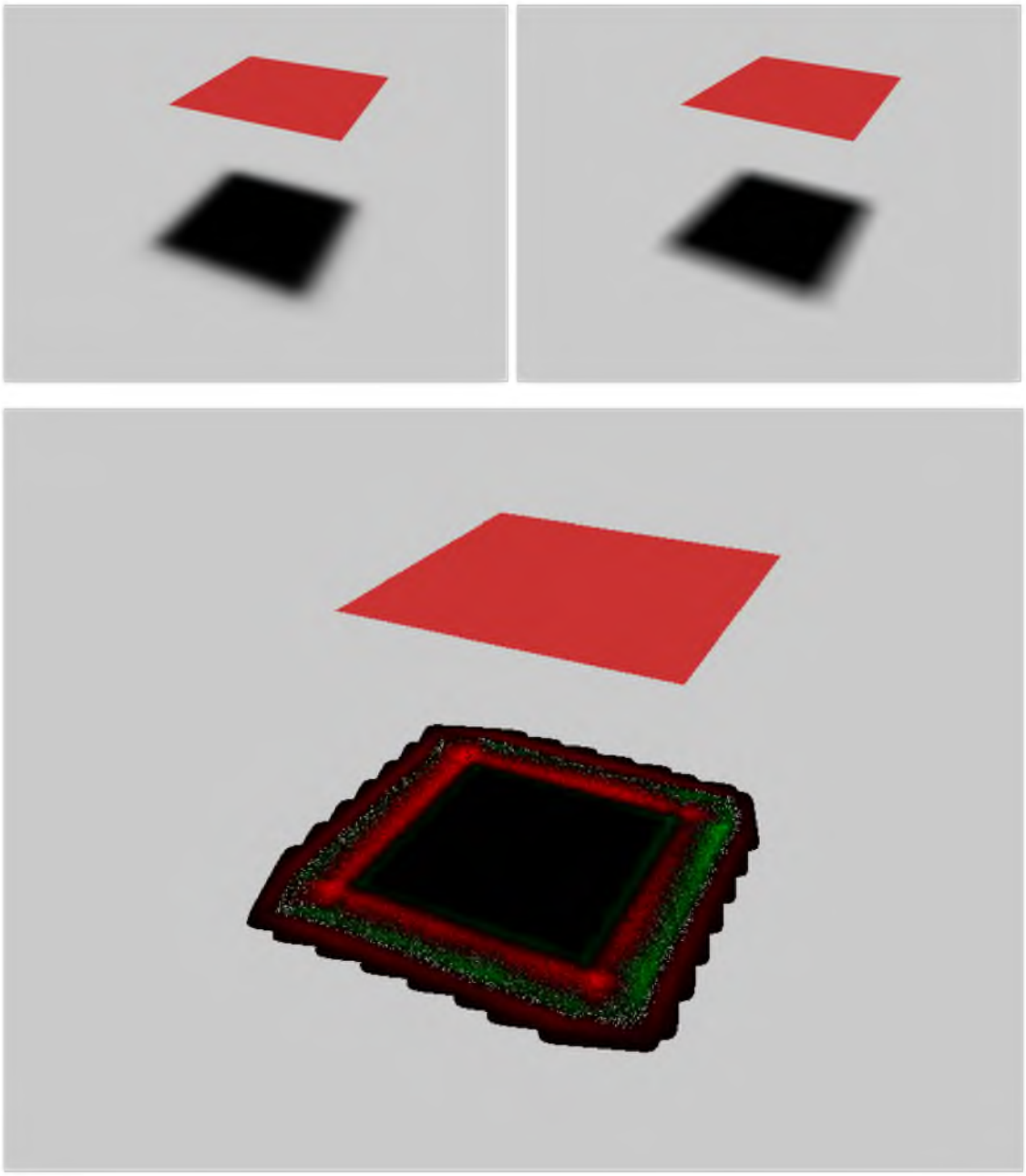

Figure 4.3: (Left) Soft Shadow Mip-Maps. (Right) Ray Traced Soft Shadow. (Bottom) Penumbra difference. The penumbra difference is magnified 8x.

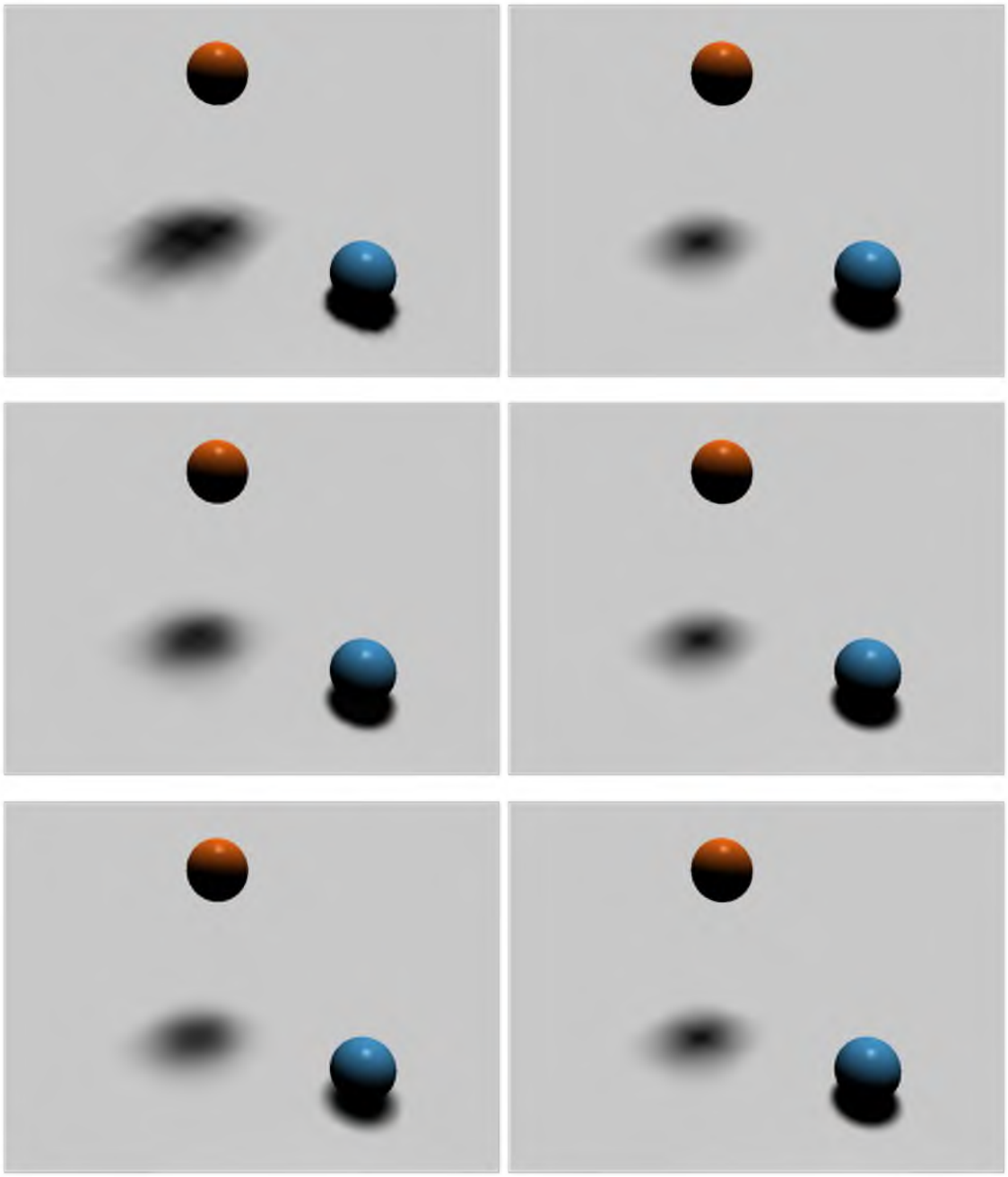

Figure 4.4: (Top) Soft Shadow with a Small Filter Size. Left: Soft Shadow Mip-Maps with a filter size of 2. Right: Ray Traced Shadow. (Middle) Soft Shadow with a Larger Filter Size. Left: Soft Shadow Mip-Maps with a filter size of 5. Right: Ray Traced Shadow. (Bottom) Soft Shadow with a Very Large Filter Size. Left: Soft Shadow Mip-Maps with a filter size of 9. Right: Ray Traced Shadow.

many texels on each shadow mip-map layer. As shown in Figure 4.4 bottom, the shadow is smoother, but the penumbra width is too wide when compared to the ray traced result.

4.1.4 Compare with PCSS and Ray Traced Soft Shadows

In our method, shadows are calculated by the right geometry that occludes the receiver. Whereas in Percentage Closer Soft Shadows, the receiver may be affected by geometries that do not occlude it. As shown in Figure 4.5, when the shadow boundaries of the planes get closer, the penumbra widths generated using PCSS deviate from the ray traced reference as compared with our method.

For complex models, our method produces closer results to the ray traced reference as compared to Percentage Closer Soft Shadows in terms of the smoothness of shadows in the penumbra area (Figure 4.6).

Our algorithm can be applied to a large span of light sizes for complex scenes, whereas Percentage Closer Soft Shadows cannot properly handle very large light sources. Figures 4.7, 4.8, and 4.9 show the same scene with gradually increased light sizes. In these figures, the left images are rendered by our algorithm, the middle images are ray traced references, and the right images are the PCSS results. Compared with PCSS, our algorithm generates smoother shadows with less noise and closer results to the ray traced references. As the light size increase, the parameters of PCSS need to be adjusted accordingly (the scaling factor of estimated penumbra width for the filter size, the depth of the shadow-map plane, the radius of the scene, etc.) to achieve similar results to the ray traced references. On the other hand, our algorithm only needs adjusting the filter size and the depth bias for approximating the results of the ray traced references with varying light sizes.

4.2 Performance

In this section, we compare the performance of our algorithm with Variance Soft Shadow Mapping, since Variance Soft Shadow Mapping provides a highly efficient soft shadow computation, which substantially outperforms other methods. Figure 4.10 shows a scene with 3968 faces, where our algorithm achieves a good balance of performance and quality. Figure 4.11 shows the performance number in millionseconds per frame for our algorithm and Variance Soft Shadow Mapping. As the filter size of our method increases, the number of texels being accessed per frame grows quadratically, which brings more time consumption during the rendering. Therefore, Variance Soft Shadow Mapping provides significantly faster shadow computation than our Soft Shadow Mip-Maps for larger filter sizes. However, as we discussed in Section 4.1.3, Soft Shadow Mip-Maps can produce soft shadows with

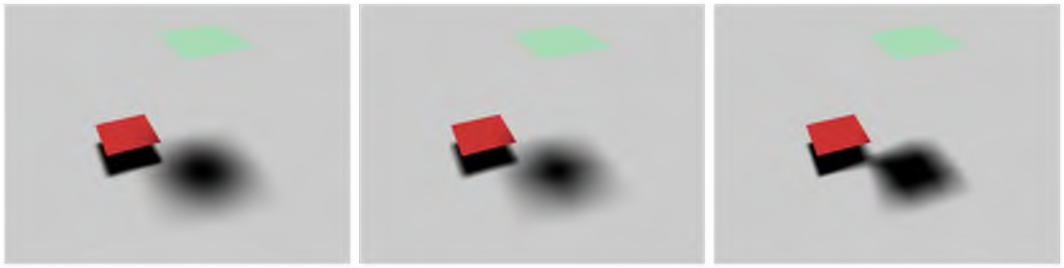

Figure 4.5: In Soft Shadow Mip-Maps, penumbra width is closer to the ground truth than Percentage Closer Soft Shadows. Left: Soft Shadow Mip-Maps. Middle: Ray Traced Reference. Right: Percentage Closer Soft Shadow.

Figure 4.6: In Soft Shadow Mip-Maps, the shadow in penumbra area is smoother than Percentage Closer Soft Shadow. Left: Soft Shadow Mip-Maps. Middle: Ray Traced Reference. Right: Percentage Closer Soft Shadow.

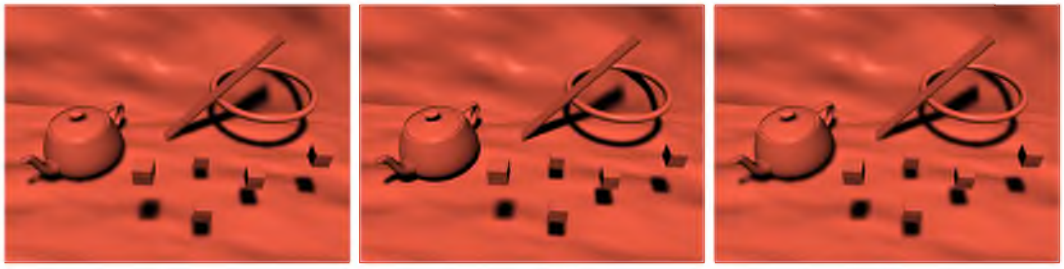

Figure 4.7: Complex scene with a small light source. Left: Soft Shadow Mip-Maps. Middle: Ray Traced Reference. Right: Percentage Closer Soft Shadow.

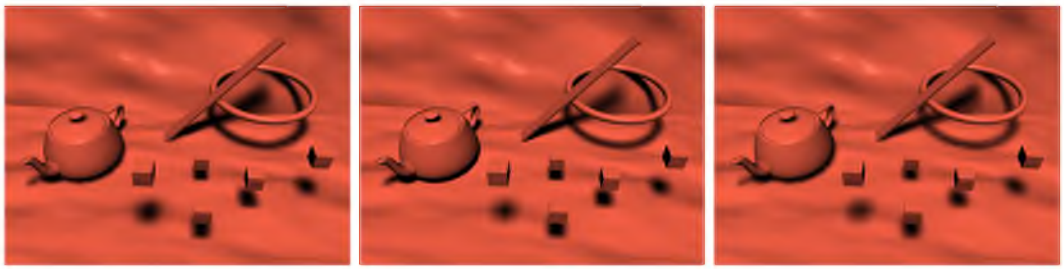

Figure 4.8: Complex scene with a larger light source than Figure 4.7. Left: Soft Shadow Mip-Maps. Middle: Ray Traced Reference. Right: Percentage Closer Soft Shadow.

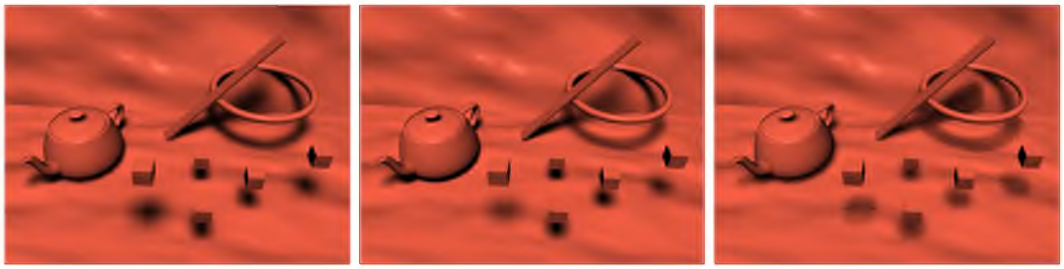

Figure 4.9: Complex scene with a larger light source than Figure 4.8. Left: Soft Shadow Mip-Maps. Middle: Ray Traced Reference. Right: Percentage Closer Soft Shadow.

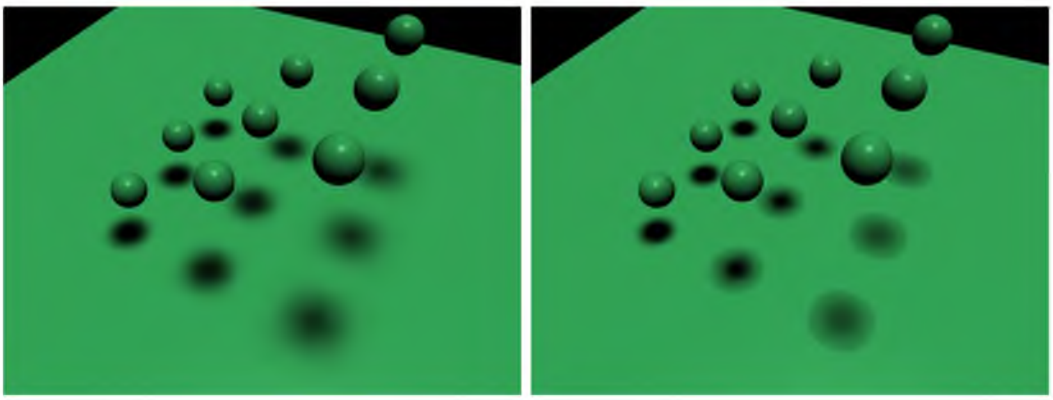

Figure 4.10: A scene with 3968 faces rendered by Soft Shadow Mip-Maps and Variance Soft Shadow Mapping. Left: Soft Shadow Mip-Maps with a depth map of size 256×256 and CPU-generated mip-maps. Right: Variance Soft Shadow Mapping with a Variance Shadow Map of size 256×256, and a GPU generated Summed Area Table.

Figure 4.11: Performance Comparison between Soft Shadow Mip-Maps and Variance Soft Shadow Mapping.

acceptable quality using small filter sizes, where the performance is closer to Variance Soft Shadow Mapping. From our experience, the best practice of filter size of our algorithm is 4 or 5, in which case it takes about 7 millionseconds to render 3968 faces. Variance Soft Shadow Mapping achieves very high performance with a GPU generated Summed Area Table, yet the quality it supplies is at a similar level as Percentage Closer Soft Shadows.

4.3 Limitations

Our Soft Shadow Mip-Maps have two important limitations: light bleeding and discrete shadow placement. In this section, we discuss these limitations in details.

4.3.1 Light Bleeding

Our method suffers from light bleeding when a geometry point is occluded by overlapped blockers at different depths to the light source. In Figure 4.12, the blue and red occluders partially overlap along the view direction of the light source. Ideally, the area on the ground occluded by the blockers should be completely in the umbra area, as shown in the ray traced reference (Figure 4.12 right image). However, in our result (Figure 4.12 left image), there is light bleeding around the boundaries of the shadows cast by the overlapping blockers, as highlighted by the yelllow parallelogram in Figure 4.13.

The reason for light bleeding is that our method only calculates shadows using the geometry available in the shadow mip-maps. Therefore, the occluded geometry is not taken into consideration, which leads to light bleeding. Specifically, light bleeding happens in *critical filter regions* of associated mip-map layers. Critical filter region is the filter region of a mip-map layer on the depth transition between the blockers, as bounded by the vertical dotted line pairs in the same color in Figure 4.14. In Figure 4.14, we approximate light bleeding in 2D with linear filtering. The top four lines are mip-map layers involved in the shadow computation. Below the mip-map layers is the geometry, which includes a higher blocker, a lower blocker, and a receiver, consecutively. Below them, the three graphs show the blocked light of each mip-map layer, the total blocked light, and the final shadow of the receiver, respectively.

The first graph in Figure 4.14 shows the blocked light of each mip-map layer. The dotted vertical lines highlight the boundaries of the critical filter regions of the mip-map layers. We define the left boundary as the starting point of the critical filter region, the right boundary as the ending point. The colors of the curve of the blocked light and the critical filter region boundaries of a mip-map layer are the same as that of the mip-map layer. Assume the blue and green mip-map layers i and $i+1$ are taken to compute the

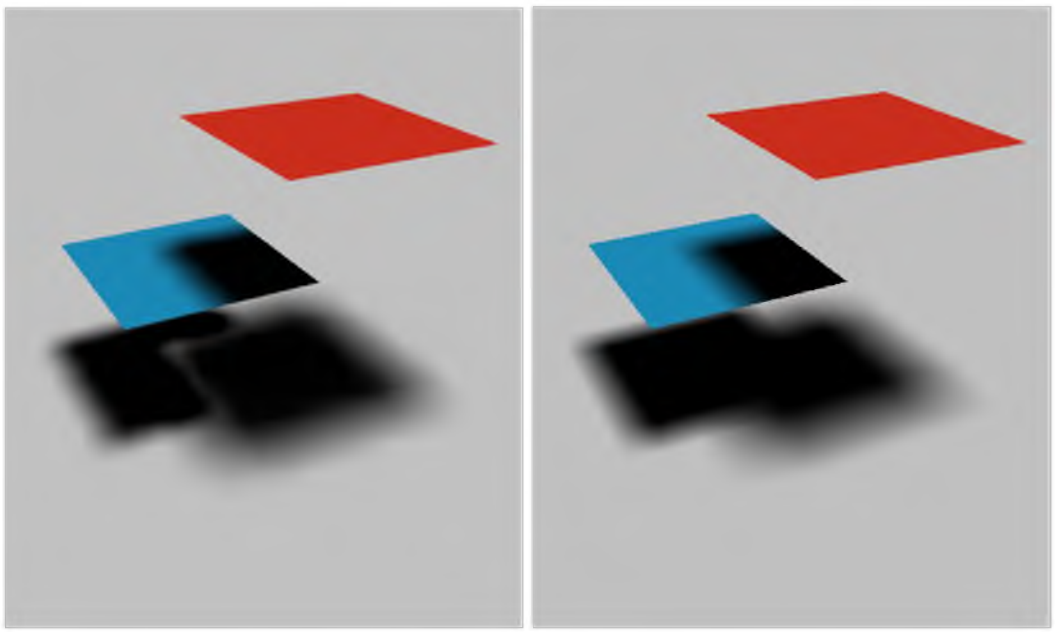

Figure 4.12: Test Scene for Light Bleeding. Left: Results of Soft Shadow Mip-Maps. Right: Ray Traced Reference.

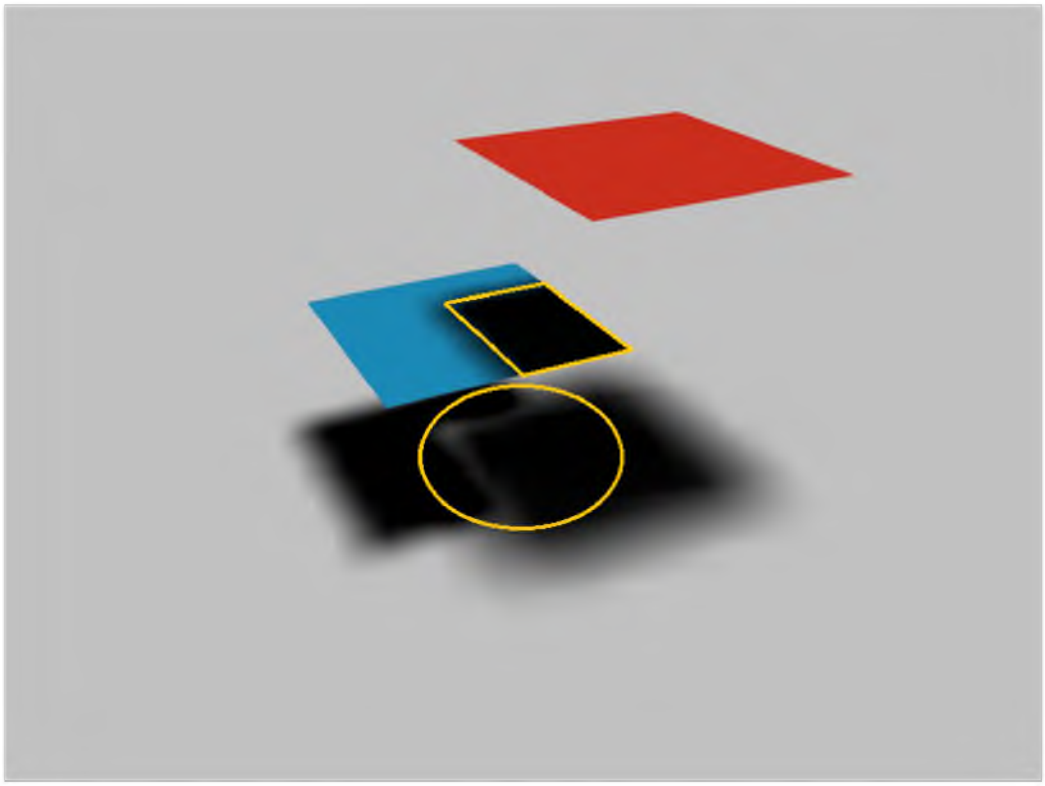

Figure 4.13: Light Bleeding Highlights. On the blue blocker, the yellow parallelogram highlights the blocked area. On the ground, the yellow ellipse hightlights light bleedings.

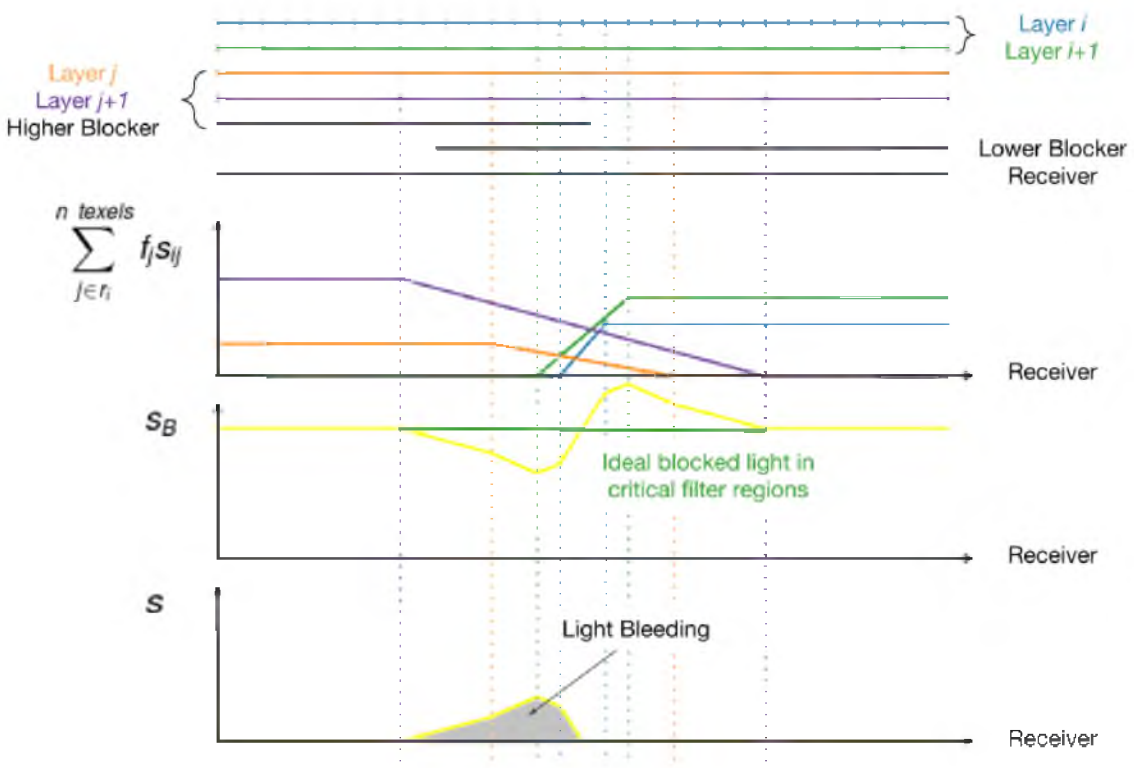

Figure 4.14: Light Bleeding Curves in 2D with Linear Filtering.

blocked light of the lower blocker, then the blocked light of each layer is 0 before the critical filter region, and linearly increases to the maximum value by the end of the critical filter region. The maximum blocked lights of mip-map layers i and $i+1$ add up to 1. On the other hand, when the orange and purple mip-map layers j and $j+1$ are taken to compute the blocked light of the higher blocker, the blocked light of each layer adds up to 1 before the critical filter region, and linearly decreases to 0 by the end of the critical filter region. Since the higher blocker produces wider penumbra width than the lower blocker, the orange mip-map layer is higher than or equal to the blue mip-map layer, i.e., $j \geq i$. When the orange mip-map layer is equal to the blue one, the blocked light of the orange mip-map layer is smaller than the blue one, so that the purple mip-map layer blocks more light than the green one and results in a larger penumbra width.

The second graph in Figure 4.14 shows the total blocked light by the mip-map layers. The total blocked light in areas out of the critical filter regions is 1. Within the critical filter regions, the total blocked light is not 1 because the inequality of the critical filter regions causes inequality of the absolute values of the slopes of the linear transitions of the blocked lights of the mip-map layers, which leads to a nonflat curve for the sum of the blocked light of each mip-map layer. However, the ideal curve for total blocked light in the critical filter regions should be flat, as depicted by the green solid line.

The third graph in Figure 4.14 shows the final shadow on the receiver. As a result of nonflat total blocked light, the final shadow is also not flat as the positive value of the opposite of the total blocked light, which results in light bleeding.

Therefore, our Soft Shadow Mip-Maps technique produce correct shadows only for the blocker geometry that is visible to the light source. If a part of a blocker geometry is not in any of the mip-map levels, it cannot be included in the shadow computation, so it does not produce any shadow. Fixing this light bleeding issue requires the ability to store multiple blocker depths for each shadow-map texel.

4.3.2 Discrete Shadow Placement

Another limitation of our method is that using lower resolution shadow mip-map layers limits the resolution of the shadow placement. As a result, when a blocker moves slowly, its shadow follows it with discrete steps, the size of which is determined by the corresponding mip-map layer resolution. Therefore, while our method produces reasonable soft shadows for a single image, soft shadows generated using our method can move substantially from frame to frame. Since the closest geometry to the light source is perceived as discrete patches in the mip-map layers that correspond to all texels of the layer, the blocker can

project to the same patch in lower resolution mip-map layers when moving. Hence, as long as the blocker is projected to the same texel on the corresponding mip-map layer, the shadow position remains unchanged.

CHAPTER 5

CONCLUSION

We introduced the Soft Shadow Mip-Maps technique for rendering soft shadows in real-time with the geometry information supplied in shadow mip-maps. Our work is based on the filtering-oriented soft shadow computation framework. In order to produce penumbra widths from the associated blockers, three methods are presented: how to precisely compute the soft shadow filter size for a receiver with a given blocker distance, how to efficiently filter shadows using shadow mip-maps, and how to estimate occluded light from potential blockers. We solved issues regarding shadow penumbra width calculation apparent in prior techniques by avoiding the average blocker estimation through estimating the occluded light from every potential blocker, which produces penumbra widths from the associated blockers.

REFERENCES

[1] E. Eisemann, U. Assarsson, M. Schwarz, M. Valient, and M. Wimmer, "Efficient real-time shadows," in *ACM SIGGRAPH 2013 Courses*, ser. SIGGRAPH '13. New York, NY, USA: ACM, 2013, pp. 18:1–18:54. [Online]. Available: http://doi.acm.org/10.1145/2504435.2504453

[2] K. Selgrad, C. Dachsbacher, Q. Meyer, and M. Stamminger, "Filtering multi-layer shadow maps for accurate soft shadows," *Computer Graphics Forum*, vol. 34, no. 1, pp. 205–215, 2015. [Online]. Available: http://dx.doi.org/10.1111/cgf.12506

[3] L. Atty, N. Holzschuch, M. Lapierre, J.-M. Hasenfratz, C. Hansen, and F. Sillion, "Soft shadow maps: Efficient sampling of light source visibility," *Computer Graphics Forum*, vol. 25, no. 4, Dec 2006. [Online]. Available: http://maverick.inria.fr/Publications/2006/AHLHHS06

[4] A. Woo, P. Poulin, and A. Fournier, "A survey of shadow algorithms," *IEEE Comput. Graph. Appl.*, vol. 10, no. 6, pp. 13–32, Nov. 1990. [Online]. Available: http://dx.doi.org/10.1109/38.62693

[5] J.-M. Hasenfratz, M. Lapierre, N. Holzschuch, and F. Sillion, "A survey of real-time soft shadows algorithms," pp. 753–774, Dec 2003. [Online]. Available: http://maverick.inria.fr/Publications/2003/HLHS03a

[6] R. L. Cook, T. Porter, and L. Carpenter, "Distributed ray tracing," in *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '84. New York, NY, USA: ACM, 1984, pp. 137–145. [Online]. Available: http://doi.acm.org/10.1145/800031.808590

[7] L. Williams, "Casting curved shadows on curved surfaces," in *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '78. New York, NY, USA: ACM, 1978, pp. 270–274. [Online]. Available: http://doi.acm.org/10.1145/800248.807402

[8] W. T. Reeves, D. H. Salesin, and R. L. Cook, "Rendering antialiased shadows with depth maps," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87. New York, NY, USA: ACM, 1987, pp. 283–291. [Online]. Available: http://doi.acm.org/10.1145/37401.37435

[9] W. Donnelly and A. Lauritzen, "Variance shadow maps," in *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, ser. I3D '06. New York, NY, USA: ACM, 2006, pp. 161–165. [Online]. Available: http://doi.acm.org/10.1145/1111411.1111440

[10] A. Lauritzen and M. McCool, "Layered variance shadow maps," in *Proceedings of Graphics Interface 2008*, ser. GI '08. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2008, pp. 139–146. [Online]. Available: http://dl.acm.org/citation.cfm?id=1375714.1375739

[11] A. Lauritzen, "Summed-area variance shadow maps," in *GPU Gems 3*, H. Nguyen, Ed. Addison-Wesley, 2008, pp. 157–182.

[12] T. Annen, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz, "Convolution shadow maps," in *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, ser. EGSR'07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 51–60. [Online]. Available: http://dx.doi.org/10.2312/EGWR/EGSR07/051-060

[13] T. Annen, T. Mertens, H.-P. Seidel, E. Flerackers, and J. Kautz, "Exponential shadow maps," in *Proceedings of Graphics Interface 2008*, ser. GI '08. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2008, pp. 155–161. [Online]. Available: http://dl.acm.org/citation.cfm?id=1375714.1375741

[14] R. Fernando, "Percentage-closer soft shadows," in *ACM SIGGRAPH 2005 Sketches*, ser. SIGGRAPH '05. New York, NY, USA: ACM, 2005. [Online]. Available: http://doi.acm.org/10.1145/1187112.1187153

[15] T. Annen, Z. Dong, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz, "Real-time, all-frequency shadows in dynamic scenes," in *ACM SIGGRAPH 2008 Papers*, ser. SIGGRAPH '08. New York, NY, USA: ACM, 2008, pp. 34:1–34:8. [Online]. Available: http://doi.acm.org/10.1145/1399504.1360633

[16] Z. Dong and B. Yang, "Variance soft shadow mapping," in *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ser. I3D '10. New York, NY, USA: ACM, 2010, pp. 18:1–18:1. [Online]. Available: http://doi.acm.org/10.1145/1730804.1730990

[17] L. Shen, J. Feng, and B. Yang, "Exponential soft shadow mapping," in *Proceedings of the Eurographics Symposium on Rendering*, ser. EGSR '13. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2013, pp. 107–116. [Online]. Available: http://dx.doi.org/10.1111/cgf.12156

[18] F. Xie, E. Tabellion, and A. Pearce, "Soft shadows by ray tracing multilayer transparent shadow maps," in *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, ser. EGSR'07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 265–276. [Online]. Available: http://dx.doi.org/10.2312/EGWR/EGSR07/265-276

[19] L. Shen, G. Guennebaud, B. Yang, and J. Feng, "Predicted virtual soft shadow maps with high quality filtering," *Computer Graphics Forum*, vol. 30, no. 2, pp. 493–502, 2011. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2011.01875.x

[20] M. Schwärzler, C. Luksch, D. Scherzer, and M. Wimmer, "Fast percentage closer soft shadows using temporal coherence," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ser. I3D '13. New York, NY, USA: ACM, 2013, pp. 79–86. [Online]. Available: http://doi.acm.org/10.1145/2448196.2448209

[21] E. Eisemann and X. Decoret, "Plausible image based soft shadows using occlusion textures," in *2006 19th Brazilian Symposium on Computer Graphics and Image Processing*, Oct 2006, pp. 155–162.

[22] M. Schwarz and M. Stamminger, "Bitmask soft shadows," *Computer Graphics Forum*, vol. 26, no. 3, pp. 515–524, 2007. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2007.01074.x

[23] C. Wyman and C. Hansen, "Penumbra maps: Approximate soft shadows in real-time," in *Proceedings of the 14th Eurographics Workshop on Rendering*, ser. EGRW '03. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 202–207. [Online]. Available: http://dl.acm.org/citation.cfm?id=882404.882434

[24] U. Assarsson and T. Akenine-Möller, "A geometry-based soft shadow volume algorithm using graphics hardware," in *ACM SIGGRAPH 2003 Papers*, ser. SIGGRAPH '03. New York, NY, USA: ACM, 2003, pp. 511–520. [Online]. Available: http://doi.acm.org/10.1145/1201775.882300

[25] T. Akenine-Möller and U. Assarsson, "Approximate soft shadows on arbitrary surfaces using penumbra wedges," in *Proceedings of the 13th Eurographics Workshop on Rendering*, ser. EGRW '02. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2002, pp. 297–306. [Online]. Available: http://dl.acm.org/citation.cfm?id=581896.581935

[26] S. Laine, T. Aila, U. Assarsson, J. Lehtinen, and T. Akenine-Möller, "Soft shadow volumes for ray tracing," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1156–1165, Jul. 2005. [Online]. Available: http://doi.acm.org/10.1145/1073204.1073327

[27] V. Forest, L. Barthe, and M. Paulin, "Accurate shadows by depth complexity sampling," *Computer Graphics Forum*, vol. 27, no. 2, pp. 663–674, 2008. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2008.01164.x

[28] E. Sintorn, E. Eisemann, and U. Assarsson, "Sample based visibility for soft shadows using alias-free shadow maps," *Computer Graphics Forum*, vol. 27, no. 4, pp. 1285–1292, 2008. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2008.01267.x

[29] G. S. Johnson, W. A. Hunt, A. Hux, W. R. Mark, C. A. Burns, and S. Junkins, "Soft irregular shadow mapping: Fast, high-quality, and robust soft shadows," in *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, ser. I3D '09. New York, NY, USA: ACM, 2009, pp. 57–66. [Online]. Available: http://doi.acm.org/10.1145/1507149.1507159