BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

HIGH-QUALITY SHADOW RENDERING FROM COMPLEX LIGHT SOURCES

DOCTORAL THESIS
DISERTAČNÍ PRÁCE

AUTHOR
AUTOR PRÁCE

Ing. JAN NAVRÁTIL

BRNO 2015

Contents

CHAPTER 1

Introduction

One of the real life tasks which benefit from computational performance of computers is the generation of images, animation and visualization. Computers can synthesize images in a nearly photorealistic quality and in real-time. Realistic rendering of global illumination has been considered the most time consuming part of the computer graphics for many years. The most difficult part of the evaluation is computing light transport and visibility correctly for every point of the entire scene.

In realistic image rendering, numerous visual phenomena have to be taken into account. They appear as a consequence of light scattering in the scene, e.g. caustics, reflections, and shadows. Each of these topics is worth discussing. In this thesis, the shadows for interactive applications will be further investigated.

Shadows constitute an important part of computer graphics rendering methods because they allow enhanced perception of the depth relations in the scene. Shadows help human viewers to correctly perceive object positions in the virtually created scene. Virtual scenes are often very dynamic, so if we want to achieve high quality shadows the algorithm has to be robust and work regardless of the scene's configuration, specifically the light and camera position. An example can be seen in applications for modeling and visualization where developers require fast and accurate shadow casting, independent from light types, camera position and scene complexity.

Shadow rendering in 3D applications has been investigated for many years. Various approaches have been published and their usage depends on the application and on the required quality of results. The main challenge is to evaluate a visibility between a rendered point and a light source. The highest quality is achieved with off-line rendering techniques, such as Ray Tracing or Radiosity. However, their rendering times are far from interactive rates. It can take hours or days to produce an high quality realistic image with radiosity or ray tracing. Frequently used algorithms in interactive applications are shadow volumes or shadow mapping. The shadow mapping algorithm is fast and easy to implement on GPUs despite its limitations in resolution and consequently in quality of rendering.

This thesis is mainly focused on resolving issues that appear in a shadow mapping algorithm. This algorithm renders the shadows in two steps. In

the first step, the discrete representation of the scene is stored into a depth texture from light point of view. Then, the values in the texture are used for shadow computation from camera point of view. The representation of the scene is discretized because of the limited resolution of the texture. The resolution provides a number of samples that can be used for shadow computation. Since the textures are rectangular, samples are evenly distributed. This may produce artifacts on shadow boundaries and thus decrease the overall visual quality of the rendered image.

The purpose of this work is improvement of shadow quality in the shadow mapping algorithm. Some methods try to reduce the aliasing artifacts by adapting the distribution of samples to the current scene configuration. This mostly depends on the mutual position of the camera, the scene, and the light source. Typically, in outdoor scenes, modelling the sun as a complex light source is not very efficient, and actually not necessary. In this case, a directional light source is used as an approximation. For this simple light source, the parameterization of sampling distribution is very straightforward and easy to implement. In order to render shadows from more complex light sources, different approach needs to be employed in comparison to methods that deal with a directional light source or a spotlight.

When dealing with complex light sources in the shadow mapping algorithm, the representation of a scene in the depth texture has to be modified, and the texture generation process as well. Therefore, shadow quality improvement techniques that are successfully used with simple light sources are no longer directly usable.

The goal of this thesis is to introduce an approach that is able to render a scene with complex light sources, reduce aliasing artifacts on the shadow boundaries and also improve the quality of shadows regardless of the type of the scene and its configuration. The main contributions are improved shadow quality through better sampling of the scene, utilization of the shadow map warping for sampling improvement and evaluation of shadow quality.

CHAPTER 2

Global Illumination and Shadows

When people observe an image of some object, such as a room or a scene, they most likely want the image to look like a real photograph. It is mainly related to the images generated by a computer. Computers have to simulate at least how the scene is illuminated by a light. Computation of global illumination is considered as the most difficult and time consuming task in 3D computer graphics. However, high quality images can be produced, such that they are hardly distinguishable from real photographs.

The time complexity arises from a fact that these approaches simulate natural behavior of the light. Today, such simulation is usually based on geometric optics model where light travels in straight lines. The most difficult part of this simulation is evaluating mutual „visibility“ of every two points on the scene surface and summing up their light contribution. During the years, most of the expensive parts of global illumination have been replaced by simple models, or approximated by less complex algorithms capable of running in real-time [14]. This allows to render scenes with dynamic content, light sources and cameras, while retaining a plausible level of realism.

Shadow rendering is one of the areas where physical-based rendering is being replaced by simple algorithms in order to run the application at interactive rates.

2.1 Realistic Image Synthesis

Methods for the computation of global illumination are able to synthesize images in photo-realistic quality, because they simulate physical rules of light distribution. This consists of light transport from a light source to an object in a scene or how the light is reflected from various kinds of materials, All these phenomena have to be taken into account and stored in the global scene description. This includes geometry data, properties of materials and specification of shape and properties of light sources. Physical based description of the properties is defined by a model.

Most of the existing algorithms can be categorized into two basic groups. Firstly, *point sampling* approaches where the scene is evaluated independently

for every pixel of the output image. Secondly, *finite elements* methods where the scene is divided into set of elements and the illumination is computed with respect to mutual relations of the elements. The following text gives a brief overview of representative techniques from for each group. Specifically, *Ray Tracing* [18] and *Radiosity* approaches are presented.

The idea of photon mapping is slightly different comparing to ray tracing or radiosity. The main difference can be seen in a different representation of the illumination information. Instead of storing the illumination tightly connected with the geometry, it is stored in a separate data structure called *photon map* [5].

2.2 Shadows in Interactive Applications

Generation of realistic images takes second or hours in the global illumination algorithms. Computation of illumination in interactive applications requires approximation of the most expensive parts of the algorithms [13]. The problem can be divided into two parts.

Firstly, an expensive computation of BRDF on a surface that is currently lit can be approximated by shading models (e.g. Blinn-Phong, Cook-Torrance). In this case, form factors are not needed and the geometric relations are neglected. The shading models evaluate illumination based on position of geometry and light source. The shading models, however, do not provide any information of whether the surface lies in shadow or not.

The second part of the global illumination that has to be approximated is computation of shadows. The global illumination algorithms compute shadows either by evaluating intersection of shadow rays with geometry (Ray Tracing), or it results from a small number of photons in a photon map (photon mapping). Neither of these approaches is applicable in interactive applications without additional simplification [10].

The most popular algorithms that are used for rendering shadows in interactive applications are planar shadows and the shadow volumes algorithm [3]. The shadow mapping algorithm is investigated in detail in Section 2.3 since the main contribution of the thesis is improvement of the algorithm.

2.3 Basics of Shadow Mapping Algorithm

This section investigates basic principles of the Shadow Mapping algorithm [19]. It describes individual steps of the algorithm, its advantages and disadvantages. Further, it provides some implementations details in order to present all necessary aspects that are needed to render shadows on GPU.

2.3.1 Shadows in Two Steps

The key concept of all shadow rendering algorithms is that a point on a surface is considered to be visible to the light source if there is no occluder between the surface point and the light source. The visibility test might be a difficult and an expensive task for complex light sources.

In the Shadow Mapping algorithm, three basic types of the light sources can be considered: *directional light source*, *spotlight* and *omnidirectional light source* . However, the algorithm can be used with some additional improvements for complex light sources as well (see Section 3.3). The directional light source is the simplest one. It is used mostly in outdoor scenes where most of the light comes from the sun which is considered to be in infinity. Because of this, all rays can be considered parallel. The spotlight is defined by its position and direction of a „cone“. The light is emitted from a point in space into the directions limited by the cone. The area of the illuminated part of the scene is defined by the field-of-view angle. The omnidirectional light source is also represented as a point in space, but it shines into all directions.

A basic approach to decide whether some object is occluded by another is to compare its distance to a camera or an observer. In the rasterization pipeline, a depth buffer is used to store information about the distance of the object to the camera. In every pixel, the depth buffer stores only the value of the closest object. In the Shadow Mapping algorithm, the scene is rendered from the virtual camera in the position of the light source. Then, the depth buffer contains information about the distance to the objects that are closest to the light source. This implies that these objects are directly lit by the light source and everything behind them is in the shadow. The subsequent rendering pass from the camera point of view can read the depth information from the depth buffer and decide whether the surface being rendered is in the shadow or not.

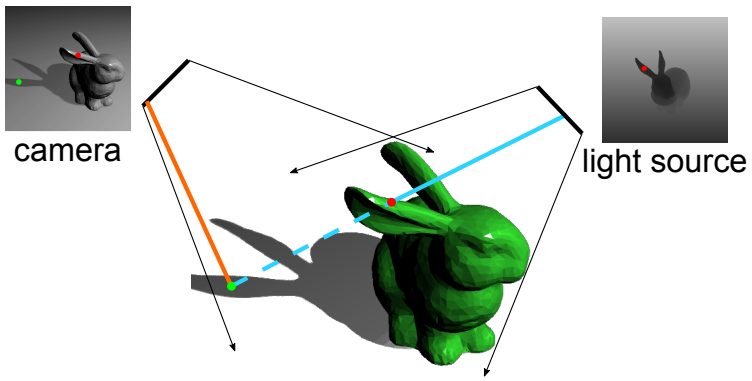

Figure 2.1: Illustration of the basic principle of the Shadow Mapping algorithm. The depth stored in the shadow map (red dot) is less than projected pixel visible from a camera (green dot).

Naturally, to implement the shadow maps on contemporary computers, one must consider exploitation of GPUs. Contemporary GPUs have already integrated support for accelerated computation of the shadows. For instance in OpenGL, there is API function [1] that allows usage of 3-component vector for sampling the texture with the depth values:

```
float texture( sampler2DShadow sampler, vec3 P);
```

The function fetches the depth value from the shadow map using texture coordinates that are stored in the first two component of vector P. Then, the depth value is compared with the third component that should hold the referencing depth value of currently rendered fragment. The function returns 0, if the P. z value is greater that the value in the texture, otherwise it returns 1. The code snippet written in GLSL can look like:

```
1 vec3 texCoords = lightModelViewProjection *
  io_ObjSpacePosition;
2 texCoords.xyz = normalize( texCoords.xyz );
3 texCoords.z = (Length - near)/(far - near);
4 vec3 P = vec3( 0.5*texCoords.xy + 0.5, texCoords.z);
5 float shadow = texture( shadowMap, P);
6 vec4 fragColor = shadow*color;
```

From the above fragment of code, it is, hopefully, obvious that the shadow map implementation in contemporary GPU is straightforward and efficient.

2.3.2 Shadow Mapping Issues

The Shadow Mapping algorithm is considered to be very efficient and flexible approach, but it suffers from some issues and visual artifacts including aliasing. As the depth information is usually stored in a texture, the size of the texture represents the total number of depth values that can be fetched in order to compute a shadow. It is common that multiple surface points with different distances to the light source are projected to a single shadow map texel. This incorrect sampling rate leads to unpleasant visual artifacts and aliasing. Section 3.1 explains the aliasing in Shadow Mapping and methods for its elimination in detail. The following text discusses the most common visual artifacts produced by Shadow Mapping that are immediately noticeable.

Artifacts caused by wrongly computed self-shadow that arises on the object surface is called a *surface (shadow) acne*. The depth value in the shadow map is quantized, but points from the surface do not all have the same depth. Consequently, some fragments on the surface lie in shadow while other fragments are considered as lit by the light (see Figure 2.2).

[1] `https://www.opengl.org/documentation/glsl/`

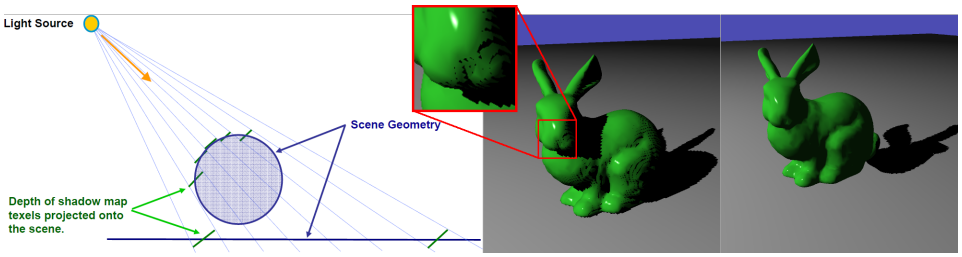

Figure 2.2: Illustration of source of the shadow acne (left). Depicted surface acne (middle) and Peter-panning effect (right).

The solution could be adding some bias to the surface in order to eliminate a difference along the pixels. A slope of the surface might be take into account to achieve a better results. For instance, the polygon offset is successfully used for this purpose. Simultaneously, when rendering objects with closed geometry, the front face culling can be enabled. It causes that the depth map stores distances to the polygons farther from the camera.

However, an excessive usage of the bias could lead to another artifact on shadows. This second common visual artifact, called disconnected shadow, or *Peter Panning*. It makes the shadow detached from the object and the object appears to be floating in the air. This usually happens when the algorithm compares two depth values that are close to each other. When the bias is applied, the shadow test may mistakenly evaluate the fragment to be lit.

To be sure that the shadow test pass for correct fragments, the bias has to be adjusted. Also, the view frustum of the light source has to fit as much as possible in order to improve precision of discrete depth quantization.

CHAPTER 3

Shadow Quality and Complex Light Sources

As the basic Shadow Mapping algorithm has been introduced in chapter 2, this chapter focuses on visual quality of rendered images. It describes how the quality of shadows is influenced by incorrect sampling and aliasing, and how the aliasing error can be measured. Further, the chapter presents some optimization techniques that eliminate disadvantages of the Shadow Mapping algorithm related to visual artifacts. It shows that the techniques are designed and implemented for simple light sources where they achieve good results.

This work, however, focuses on improvement quality of shadows for complex light sources where the techniques for simple light sources fail. This chapter provides an overview of various methods for rendering shadows cast from omnidirectional light sources. It describes the principles of each method and discusses their advantages and disadvantages.

3.1 Deriving the Error Metric

The aliasing in the Shadow Mapping algorithm is a significant visual artifacts. It appears namely on shadow edges due to low resolution of the shadow map, because a single shadow map texel cannot cover all the details for object further from light sources . The shadow map sampling rate is typically insufficient to handle all the scene details sufficiently well.

The sampling scheme is very similar for both camera and light source. The camera and the light source sample scenes through pixels representing rectangular areas. All the light rays going through the given rectangular pixel and the light source or the camera form a beam defined by the pixel. Given a beam from the camera projects through a pixel onto a scene surface with width w_i'. Similarly, a beam from the light source through a shadow map texel is projected on the surface with width w_l'. The aliasing error on such surface can be approximated by the ratio of the projected beam widths:

$$m = \frac{w_l'}{w_i'} \tag{3.1}$$

As it can be seen in Figure 3.1, the aliasing error does not depend only on beam widths and distance of the surface to the camera or light source but also on

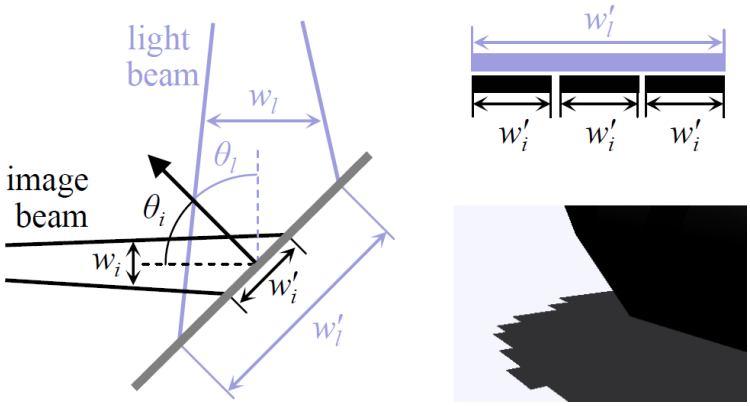

Figure 3.1: Illustration of beams projected from eye and light source and their widths on a surface [8].

surface orientation. Stamminger et al. [16] described these two types of aliasing: *perspective* and *projection*. The aliasing error according to Stamminger can be quantified as:

$$m = \frac{w_l'}{w_i'} \approx \frac{w_l}{w_i} \frac{\cos\theta_i}{\cos\theta_l} \tag{3.2}$$

where w_i and w_l are the widths of the image and light beams at the point of intersection and θ_i and θ_l are the angles between the surface normal and the beam directions.

Perspective aliasing is caused when the shadow map is undersampled because of light source distance while projection aliasing appears when the direction of light rays is parallel to the surface so that shadow stretches along the surface. The perspective aliasing is the most common one in the Shadow Mapping algorithm. It occurs when more than one point on the geometry is projected to the single texel in the shadow map. Pixels by the near plane are more dense in post-perspective space than pixels by the far plane. However, the sampling rate of the shadow map remains the same over the entire view frustum. Because of this, more pixels map to the same texel in the shadow map.

Some methods exist that attempt to reduce the perspective aliasing artifacts on shadow boundaries. The shadow map can be filtered [2] that causes the shadows to be smooth which, however, is not always desired. The „correct“ approach would be to use high shadow map resolution for objects near to the camera and low resolution for distant objects. Naturally, the high resolution is not necessary for shadows far from camera as the fine scene details are not visible due to perspective projection.

In some approaches [4], multiple shadow maps with different resolutions are used. They are stored in a hierarchy based on resolution and they adapt to the level of detail desired in individual locations of the rendered frame. Unfortunately, this technique needs multiple rendering passes and some additional data structures, so acceleration in hardware is not efficient.

The projective aliasing is not easy to eliminate and since it is less intrusive, existing approaches neglect it. The following sections describe how the perspective aliasing can be reduced which leads to elimination of jagged edges in the output image. It happens when the width of the image beam w_i equals the to the light beam width w_l.

The Shadow Mapping algorithm works with two types of samples. View samples are pixels that correspond to points on a scene surface described by their 3D position (and other properties such as color, normal vector etc.). They are generated by sampling the scene from a camera point of view. Shadow samples are generated by sampling the scene from a light source point of view. In both cases, the sampling is performed using an orthogonal grid with a predefined resolution.

However, multiple view samples can be projected onto one shadow sample and then aliasing can be observed in a final image as jagged edges of the shadows. This is caused by uniform rasterization of a texture produced by a graphics hardware. One solution is to parameterize the sampling using a warping function. The function enlarges important parts of a scene in order to increase shadow sampling rate. This technique increases a probability that shadows for different view samples are resolved by different shadow samples. There are two types of the warping function - *global* and *local*. The global warping function can be defined by a transformation matrix. This warping function mostly depends on a mutual position of a camera, a light source and geometry and ignores properties of view samples [16]. The local warping function is derived from properties of view samples and scene analysis [6, 15]. These approaches are described in detail in the following sections.

3.2 Methods for Reducing Aliasing

Some methods exist to reduce the aliasing errors caused by the sampling mechanisms used in the Shadow Mapping algorithm. As the shadow map size is typically given by the hardware limitations, these methods exploit non-uniform sampling of the shadow maps either through non-linear mapping or using discrete smaller maps with different resolutions.

3.2.1 Perspective Shadow Maps and Parallel-Split Shadow Maps

Perspective Shadow Maps [16] differ from standard shadow maps in that they are generated after perspective transformation, i.e. in normalized device coordinates. It causes reduction of the perspective aliasing (see Section 3.1) on the shadow boundaries in a rendered image. The idea of the *Parallel-Split Shadow Maps* approach [20] is to split the view frustum in a certain distance from the camera into several parts in order to minimize the oversampled areas and thus make use the shadow map efficiently.

3.2.2 Rectilinear Texture Warping

Rosen [15] introduced an adaptive shadow mapping approach that also addresses the aliasing issue. He suggested the *Rectilinear Texture Warping* (RTW) technique that is capable of rendering quality shadows. Unlike CSMs, the RTW uses only one shadow map to cover the entire scene and a set of importance functions for adaptive scene sampling. The shadow map can be generated per-frame, and it supports fully dynamic scenes. As the camera and the light source moves, the RTW adaptively changes the sampling rate, whereas the standard shadow map remains unchanged as it can be seen in Figure 3.2.

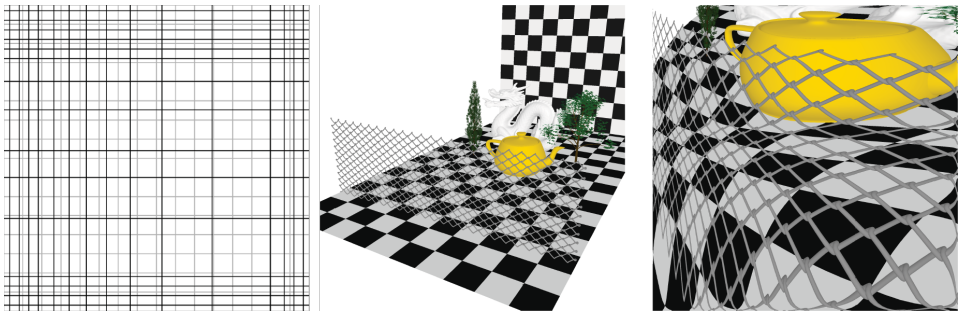

Figure 3.2: Illustration of rectilinear warping scheme [15].

The crucial step in the algorithm is creating the importance map. The importance can be analyzed in three different ways:

- **Forward** - Firstly, the depth map is rendered from the light point of view. Then, the depth map is analyzed and the importance map is built.
- **Backward** - The depth map is rendered from camera point of view, projected to the light space and analyzed.
- **Hybrid** - Combination of most valuable results from both approach in cost of higher computation time.

The importance analysis in all options is performed using an arbitrary number of analytical and heuristic-based functions. The output of the analysis is the *importance map* which serves as an input to next steps of the algorithm:

```
1 Build the importance map
2 Convert 2-D importance map into 1-D warping maps
  begin
3 |   Collapse rows/columns to 1-D importance maps
4 |   Blur importance maps
5 |   Build warping maps from importance maps
  end
6 Render the RTW shadow map
7 Render the output image from the desired view
```

Algorithm 1: RTW algorithm

In the Step 2, the importance map is further processed in order to get the warping map. Firstly (in Step 3), the maximal importance value from every row and every column is stored into two 1D importance maps. These maps are blurred in Step 4 in order to smooth the differences between adjacent samples and ensure coherency. Finally (in Step 5), the positions are shifted according to value in the 1D importance map. The computed offsets are then used to build the warping maps.

The Steps 6 and 7 are well known steps from the standard Shadow Mapping algorithm. Step 6 renders the shadow map and Step 7 computes shadows. However, in both of these steps, the newly built warping map is used for rendering of the shadow map as well as computation of shadow map coordinates when the shadow is computed.

The biggest disadvantage of the RTW algorithm is the rectilinear grid. In order to maintain high quality shadows, the algorithm selects the maximal importance value from the importance map for a given row and column, respectively. This may introduce unneeded resolution for the remaining part of the row or column. In the worst case, it may lead to decrease in quality of the output. Another disadvantage that is common for all warping approaches is that the scene has to be finely tessellated. The warping of the shadow map curves the long edges of triangles (see Section 2.3.2). This artifact is not visible when the triangles are reasonably small. Contemporary GPUs supports hardware tessellation. It slightly increases processing time, but also improves quality of the rendered image.

Jia et al. [6] introduced *Distorted Shadow Mapping* (DSM) algorithm that detects shadow silhouettes from depth discontinuities in the standard shadow map. It does not employ any regular grid, and increases the sampling rate locally. However, the DSM algorithm does not consider any other view information, and it relies only on information from the shadow map. Hence, some important details might be missing.

3.3 Omnidirectional Shadow Mapping

3.3.1 Cube Shadow Maps

In order to create shadow maps for an omnidirectional light source, the Cube Shadow Maps algorithm proposes to point the virtual camera into six directions. The view direction of the virtual camera should be oriented along directions defined by the axes of the local coordinate system of the cube: positive X, negative X, positive Y, negative Y, positive Z and negative Z. This is almost identical to the way how a cube map for environment mapping is generated except that in this case depth values are stored instead of color.

Basics of the Cube Shadow Maps

The faces of the cube represent shadow maps and directions of the faces shows the particular direction for the virtual camera . In order to cover the whole environment, the traditional Shadow Mapping algorithm exploits cube maps to visualize shadows cast from point lights. To fill the data in the cube shadow map, six render passes have to be performed. The GPUs generally support the cube shadow maps which are thus easy to implement.

The biggest disadvantage of the Cube Shadow Maps is that six render passes are often too expensive. This fact can cause rapid decrease of performance for complex scenes with high number of polygons. Even if per-object frustum culling is applied, rendering of shadows is still very expensive in comparison to rendering of the rest of the scene.

3.3.2 Dual-Paraboloid Shadow Maps

The following text discusses an alternative approach for rendering shadows cast from omnidirectional light sources. The *Dual–Paraboloid Shadow Mapping* algorithm (DPSM) [1] maps 3D positions of a geometry into 2D map. The Dual-Paraboloid mapping can be used for creating maps of an environment and among other environment mapping approaches, such as cubical or spherical, the algorithm introduces better performance in comparison to the cubical mapping, and better quality in comparison to the spherical mapping. The algorithm is based on two paraboloids attached back-to-back, each capturing one hemisphere. This section introduces principles of the Dual–Paraboloid Shadow Mapping algorithm, and how it can be used for rendering shadows.

Mathematical Background

In principle, the idea is based on a mirror. Imagine a totally reflective mirror in a shape of a paraboloid that reflects incident rays from a single hemisphere into the direction of the paraboloid (see Figure 3.3). The rays may carry some information about the environment (such as color or distance) and the information

can be stored into a rectangular map. The 2D coordinates are computed from the point on the paraboloid surface where the ray intersects the paraboloid.

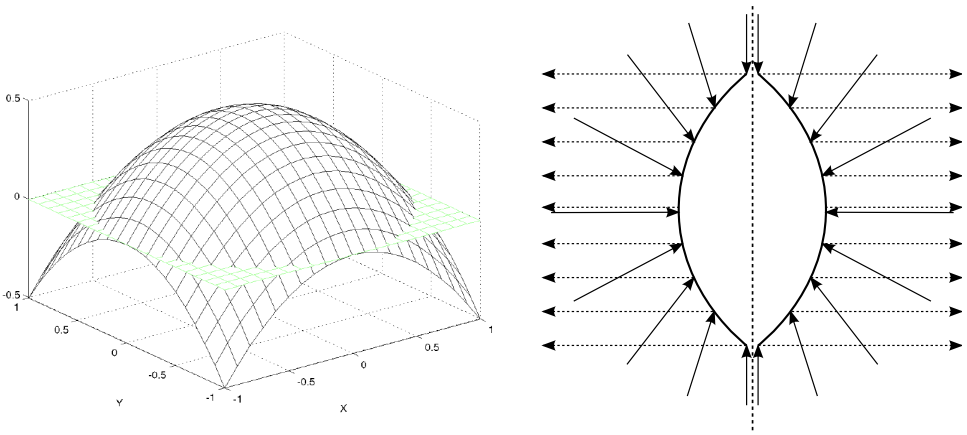

Figure 3.3: (Left) The paraboloid itself. (Right) Two paraboloids attached back-to-back can capture the environment from all directions [1].

To implement the Dual–Paraboloid Shadow Mapping algorithm on GPU, it is necessary to understand how the mapping actually works. This knowledge will be then used for writing shaders for GPUs. The paraboloid itself is given by:

$$f(x,y) = \frac{1}{2} - \frac{1}{2}(x^2 + y^2), \quad x^2 + y^2 \leq 1 \tag{3.3}$$

The key concept of the paraboloid mapping is that all incident rays are reflected in the same direction. The first task is to find the point on a paraboloid surface where the ray is reflected. This can be computed using the surface normal vector.

A paraboloid surface point P is given by:

$$P = (x, y, f(x,y)) \tag{3.4}$$

To compute the normal vector in P, the tangent vectors have to be computed by taking the partial derivatives of the function with respect to x and y. The resulted cross product gives the normal vector:

$$T_x = \frac{\delta P}{\delta x} = \left(1, 0, \frac{\delta f(x,y)}{\delta x}\right) = (1, 0, -x) \tag{3.5}$$

$$T_y = \frac{\delta P}{\delta y} = \left(0, 1, \frac{\delta f(x,y)}{\delta y}\right) = (0, 1, -y) \tag{3.6}$$

$$N_P = T_x \times T_y = (x, y, 1) \tag{3.7}$$

The derived normal vector for the point P is now known for every point on the paraboloid surface and it expresses the x and y coordinate of the map.

Based on the information mentioned above, the mapping can now be defined. The normal vector for the entire paraboloid surface can be computed as a sum of the incident ray and the reflected ray. As mentioned above, the reflected ray is always going to be $(0, 0, 1)$ for the front paraboloid and $(0, 0, -1)$ for the back paraboloid, respectively. This is a crucial concept that it is the same for a given hemisphere (see Figure 3.3). The normal vector can be computed as:

$$N_P \Leftrightarrow V_{incident} + V_{reflected} \tag{3.8}$$

Based on the Eq. 3.7, the previous equation can be expressed as:

$$N_P = (x, y, 1) \Leftrightarrow V_{incident} + V_{reflected} = V_{sum} \tag{3.9}$$

The final step for getting the x and y coordinates is to divide all components of V_sum by its z part:

$$N_P = \frac{1}{z_{sum}} (x_{sum}, y_{sum}, z_{sum}) = \left(\frac{x_{sum}}{z_{sum}}, \frac{y_{sum}}{z_{sum}}, 1 \right) \tag{3.10}$$

As both x and y coordinates of the paraboloid surface is now computed, they express the point on the surface from which the incident ray is reflected. Also, they express the coordinates to the map where the information from the environment is going to be stored. The following text presents how this concept can be applied to rendering shadows using the Shadow Mapping algorithm.

Depth Texture Generation and Shadow Rendering

When rendering shadows cast from an omnidirectional light source, the Shadow Mapping algorithm requires to render the shadow map for the entire scene. Based on the concept of the Dual-Paraboloid mapping, it needs only two render passes to capture the whole environment.

The omnidirectional shadow rendering works in the same way as the traditional Shadow Mapping algorithm (see Section 2.3). The virtual camera is placed in the position of a light source. According to the position and the orientation of the light source, the appropriate model-view-projection matrix has to be found. In this case, the projection matrix can be identity, because the projection is performed by the paraboloid mapping. The model-view matrix provides information on where is the scene divided into two hemispheres and it also expresses the direction of the paraboloid.

The vertex shader on GPU parametrizes only the geometry vertices. The remaining part of the rendering process is unchanged. It means, that the paraboloid mapping is applied only in the vertex shader and rasterization of polygons are performed in the traditional way. The vertex shader can be written as:

```
1 vec4 vertexEyeSpace = in_ModelViewMatrix * vec4(in_Vertex,1.0);
2 vertexEyeSpace.xyz = normalize( vertexEyeSpace.xyz );
3 vertexEyeSpace.z += 1.0;
4 vertexEyeSpace.xy /= vertexEyeSpace.z;
```

The input geometry is transformed to the light space using model-view matrix. The resulting vector is normalized and it will serves as the incident ray for the paraboloid mapping. The next step is to sum the incident ray with the reflection vector which is $(0, 0, 1)$ (Line 3). Finally, the result is divided by the z part in order to derive the x and y coordinates. To process the vertex further in the pipeline, the z and w coordinates have to be set as well:

```
1 vertexEyeSpace.z = (Length - near)/(far - near); vertexEyeSpace.w =
  1.0;
```

The depth value from the z coordinate is stored in the shadow map in a fragment shader. The values from the shadow map will be used in the next step where the shadow is computed.

In the final render pass of the Shadow Mapping algorithm, the depth values are read from the shadow map and compared with the depth of the current fragment. The shadow computation is now performed in the fragment shader. However, the steps for computing coordinates to the shadow map are very similar with the first rendering pass. The same concept of the paraboloid mapping is used as well:

1. Find a vector from the light source to the desired object.
2. Use this vector to calculate s and t coordinates (one pair for each hemisphere).
3. Sample both paraboloid maps with the coordinates.
4. Process the sampled values.

The implementation of all steps in the fragment shader is:

```
1 texCoords.xyz = normalize( texCoords.xyz );
2 texCoords.z += 1.0; texCoords.x /= texCoords.z; texCoords.y /=
  texCoords.z;
3 texCoords.z = (Length - near)/(far - near); texCoords.w = 1.0;
4 return vec3( 0.5*texCoords.xy + 0.5, texCoords.z);
```

The resulting x and y coordinates are normalized in order to sample the texture in range [0..1]. The z coordinates holds the depth of the rendered fragment.

After this step, all the necessary pieces of information are derived for computing shadows using the Shadow Mapping algorithm. The texture coordinates are derived using the paraboloid mapping, and the depth value that is going to be compared with the value stored in the shadow map is also computed.

The Dual-Paraboloid Shadow Mapping minimizes the amount of used memory and the number of render passes that are necessary to cover the whole

environment in comparison to . the Cube Shadow Maps technique. Other parameterization can certainly be found but the proposed parabolic parameterization maintains its simplicity and performance, e.g. in GPU implementation [12].

Nevertheless, the DPSM algorithm has also some disadvantages. While in the Cube Shadow Map approach all the transformations needed to create the shadow map are linear, they do not need any extra treatment on GPUs. This mainly concerns the interpolation process between vertex and fragment shader (see Section 2.3.2). When using the DPSM algorithm, the rendered scene needs to be finely tessellated, because the mapping is not linear and it does not work well for large polygons. Unfortunately, it may introduce new bottlenecks and artifacts on the connected parts of front and back paraboloids.

CHAPTER 4

Improved Texture Warping for Complex Light Sources

The Cube Shadow Maps and Dual-Paraboloid Shadow Mapping are methods that are capable of rendering shadows cast into all directions. Since they both are based on the Shadow Mapping algorithm, they suffer from issues caused by the limited resolution of the shadow map represented by a raster image. The main issue is related to the quality of shadows which is usually decreased by aliasing. The Parallel-Split Shadow Maps is a technique that can reduce aliasing in outdoor scenes and large environments. It is optimized for directional light sources and spotlights. These types of light sources are the most common in outdoor environment.

This chapter introduces a novel technique for improving quality of shadows cast by omnidirectional light sources. It shows how to improve process of shadow map rendering in order to get a better sampling distribution of the scene. It utilizes non-orthogonal warping scheme and it is applicable also for complex light sources.

The core of the thesis can be expressed by the following statement: *Parameterization of shadow map coordinates based on simple scene analysis can reduce aliasing error of the shadows cast by complex light sources.*

Section 3.1 shows that the highest aliasing error can be observed close to the near plane of the camera view frustum. For some scenarios, for instance outdoor scenes lit by the sunlight, the aliasing error can be successfully reduced with the PSSM algorithm . However, PSSMs do not address the shadow quality for omnidirectional light sources. The shadow quality for this type of light sources is discussed in the thesis. They present one of the three types of light sources that can be usually seen in indoor scenes.

Section 3.3 explains how difficult is to compute shadows for omnidirectional light source. The thesis shows how to improve the quality of shadows regardless of the mutual position of the light source and the camera. The improvements are implemented in both Cube Shadow Maps and Dual-Paraboloid Shadow Mapping algorithm. Moreover, omnidirectional light sources are successfully employed not only for direct illumination, but also as virtual point lights for computing of indirect illumination .

An example of a critical scenario is when the light source is inside the camera view frustum. The scenario introduces two main challenges. Firstly, shad-

ows have to be cast into all directions. Secondly, the aliasing error is not distributed uniformly but it depends on mutual position of the light source and the camera, and the current scene configuration. The uniform distribution of the aliasing error is observed from the light source point of view when the light source is outside the frustum. This applies to all types of light sources. When the light source is inside the frustum, the alias error changes unevenly. The approach presented in the thesis handles both of the challenges.

4.1 Improved Non-orthogonal Texture Warping

The approaches described in previous sections still do not address the problem of reducing aliasing in general case. They can improve the quality of shadows for cases where the aliasing error is distributed evenly in the shadow map [17, 11].

4.1.1 Importance-driven Error Reduction

Section 3.1 defines the aliasing error and shows how it can be measured. It was also shown that the aliasing error can be evaluated for any point in the camera view frustum. The idea of the proposed algorithm is to modify the projection to the shadow map according to value of the aliasing error.

The value of the aliasing error expresses whether the projection of a surface to the shadow map is undersampled (the value greater than 1), or oversampled (the value less than 1). In case of undersampled areas, jagged shadow edges appear. When the aliasing error is projected to the light space, it helps to identify the undersampled and oversampled regions in the shadow map. In these regions, the sampling rate has to be increased or decreased, respectively. the sampling rate can be modified using an improved parameterization of the mapping function. The result of the improved parameterization is that all of the points in the undersampled regions are mapped on a larger area so that the sampling improves while the sampling density of the previously oversampled regions is reduced.

Let us suppose that possible method for projection control is a grid that is placed over the shadow map. By default, the grid cells are rectangular and the projection corresponds to the standard Shadow Mapping algorithm. By changing positions of vertices on the grid, the projection can enlarge the undersampled parts locally and reduce the oversampled parts. Movements of the vertices should be managed so that the reprojected shadow map reaches the equilibrium state. The best result is achieved when the alias error is completely removed so that it equals to 1 in every pixel. However, due to geometric limitations of the shadow map this situation is not achievable. Therefore, the feasible solution is provided when the aliasing error is as constant as possible over the entire shadow map. When the grid reaches a steady state, the shadow

map is regenerated with the derived warping function. The same function has to be used in the shadow rendering step. The warping grid projects the surface points on different positions in the shadow map and hence the texture coordinates have to be parametrized using the same warping function.

This section presents the key concept of the NoTW approach. The idea of warping grid illustrates how the projection can be modified. The grid no longer appears in the following text and the improved mapping is derived using a set of warping functions.

The idea of the parameterization of the texture coordinates using warping functions is crucial for the remaining text. It presents the efficient way of improving the shadow quality based on the values of the aliasing error projected to the light space.

4.1.2 Introduction to Improved Texture Warping

Rosen [15] introduced the first method that addressed problem of important regions distributed in the depth texture. He introduced the rectilinear warping maps that could easily control the sampling in particular parts of the depth texture. This could be controlled by importance function and the approach could be used for point light sources without complex modification. Nevertheless, the rectilinear warping schema is not completely local and some parts of a scene may receive resolution higher or lower than required and that situation is not optimal.

Similar approach was published by Jia et al. [6]. They do not limit the approach to rectilinear grid; therefore, they can control the results more precisely. However, this approach needs multiple render passes of the scene to analyze the scene and decides the dividing schema. This can introduce certain issues for complex scenes.

The improved warping parameterization described in this thesis reduces the aliasing artifacts, and it allows to render high quality shadows regardless of a light source or a camera position in the scene.

The approach computes an improved parameterization based on importance driven depth texture warping. It identifies regions in the depth texture where the sampling is not optimal and enlarge this regions in order to get higher sampling rate. Before the traditional Shadow Mapping algorithm, an additional step of generating the non-orthogonal warping functions have to be applied. These functions are used later during the shadow rendering.

The main contributions are:

- Introduction of a novel importance function for determining sampling rate of depth texture. This function extends the set of functions introduced by Rosen et al. [15].

- The Non-orthogonal Texture Warping (NoTW) scheme which leads to better control of importance-based warping without affecting the nearest regions in the texture (in the same row and/or column).

The Non-orthogonal Texture Warping (NoTW) algorithm is partially based on Rectilinear Texture Warping (RTW) approach [15] (see Section 3.2.2 for details). The RTW approach utilizes various properties of view samples, e.g. distance to a camera, normal vector or edge detection. The warping function can be constructed using forward, backward or hybrid analysis.

The first step in the forward analysis is rendering of the scene from the light source point of view. Then, the importance map is computed. In the backward analysis, the G-buffer with the scene's depth and color is rendered from a camera point of view. Then, the importance analysis is performed using samples projected into the light space. The hybrid analysis combines both approaches.

The backward analysis is the fastest method because it requires a scene to be rendered only two times. The first rendering pass is used to create a depth buffer from the camera. The second rendering pass creates a warped shadow map. Its complexity is linear with relation to the number of light sources.

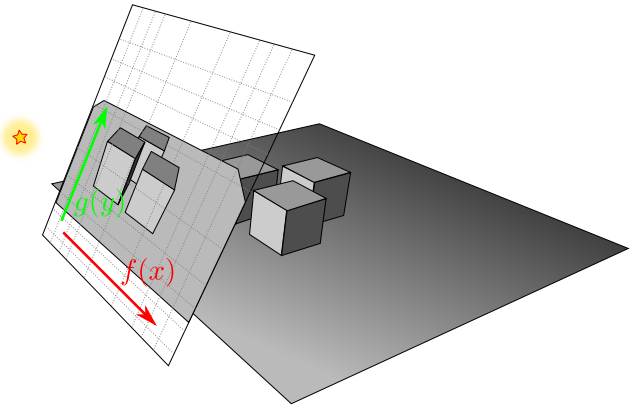

Figure 4.1: Two 1D warping functions enlarge parts of the scene that are important according to the importance map. It is not always optimal with the rectangular grid.

The warping function in RTW is composed of two 1D warping functions that operate in projection plane of a light source (see Figure 4.1). These functions are derived from an *importance map*. The importance map is constructed by projection of view samples onto the projection plane of a light source. Multiple view samples can be projected into one pixel of the importance map. In every pixel, the importance value is computed based on the view sample properties. The 1D warping functions are derived separately for column and rows according to a maximal importance value. Since the functions parameterize vertical and horizontal component of the shadow map separately they produce an orthogonal warping grid.

4.1.3 Shadow Rendering Using Warping Functions

The basic idea of the NoTW algorithm presented in the thesis is to achieve better distribution of view samples in the shadow map. Every shadow sample

resolves shadow for all view samples that were projected on it (the detail explanation of view and shadow samples and their relation to the aliasing error are presented in Section 3.1). The ideal situation occurs when one texel from the shadow map samples a surface that is projected onto one pixel in the image space. However, this is hardly achievable in most of the scenes because of the scene complexity, geometry and mutual position of the camera and the light source. Assume that the best result is observed when the number of view samples for all shadow samples is the same.

In NoTW algorithm, the importance map has the same resolution as the shadow map. Every pixel in the importance map stores the number of view samples that were projected onto the given shadow map texel. The importance map can be created by projection of view samples into to the light space and increase a counter by one. This step can be easily accelerated by contemporary GPUs.

The complete algorithm for computing shadow consists of the following steps:

1 Render a scene from a camera point of view to G-buffer
2 Project every view sample into the importance map
3 Compute prefix-sum for every row in the importance map
4 Construct the set of warping functions for rows according to Equation 4.4. Use the prefix-sum from the Step 3
5 Smoothen the set of warping functions, e.g. using weighted average
6 Project every view sample onto the importance map (and increment by 1) leveraging the set of warping functions created in the previous step
7 Repeat the Steps 2-5 for all columns
8 Create shadow map using both sets of warping functions
9 Evaluate shadows in the scene using G-buffer, the set of warping functions and the warped shadow map

Algorithm 2: Non-orthogonal Texture Warping.

The first step is generation of the G-buffer. Apart from other properties, it contains positions of view samples. The importance of the samples is then analyzed. The steps 2-7 are the most important ones and they are used to construct the set of 1D warping functions. The warping functions are derived in different manner than Rosen [15]. For every row and every column, 1D warping function is constructed separately and thus it does not allocate unneeded resolution in other parts of the shadow map. The degree of freedom for warping functions is increased using this approach . The steps are described in detail in the following section.

4.1.4 Construction of 1D Warping Functions

For one row of the importance map, let us assume a function $f(x)$ that returns the number of view samples on a normalized position x and its corresponding prefix-sum function $g(x)$:

$$n = f(x) \quad x \in \langle 0, 1 \rangle \tag{4.1}$$

$$s = g(x) = \int_0^x f(x)dx \tag{4.2}$$

For evenly distributed view samples in the row, the ratio of the number of view samples on all positions before x, i.e. $g(x)$, and the total number of view samples $g(1) = N$ is equal to ratio of the position x and the row length:

$$\frac{g(x)}{g(1)} = \frac{x}{1} \tag{4.3}$$

Expression $g(x)/g(1) > x/1$ implies that there are more view samples than the number of samples x and thus the area needs to be enlarged to achieve uniform sampling rate. On the other hand, expression $g(x)/g(1) < x/1$ implies that there are less view samples and the area can be smaller.

Now, the warping function can be derived so that it is defined as an offset $o(x)$ that has to be added to the actual view sample position. The offset function is given by:

$$o(x) = \frac{g(x)}{N} - x \tag{4.4}$$

Let us assume that the view sample is projected onto a particular row in the shadow map. Then, a new sample position x' in the row is given by:

$$x' = x + o(x) \tag{4.5}$$

Before the algorithm proceeds with construction of warping functions for columns, the importance map has to be recomputed again. But now, the newly derived set of 1D warping functions for rows are applied. After this step, the number of view samples that have to be redistributed in a given column is nearly constant (see Figure 4.2). When the 1D warping functions for columns are derived, all the view samples are distributed more uniformly.

Section 3.2.2 mentioned that the RTW algorithm constructs two warping functions - for vertical and horizontal direction, respectively. This approach is improved in this work by constructing set of warping functions for all rows and all columns at the same time. Nevertheless, these functions have to be smoothed in order to limit the warping amplitude. Otherwise, the large polygons that are linearly rasterized would not be processed by the warping functions correctly. The quality can be controlled by adjusting the size of smoothing window when averaging the warping functions. The wider the window is the smoother are the warping functions. The smoothing step is included in

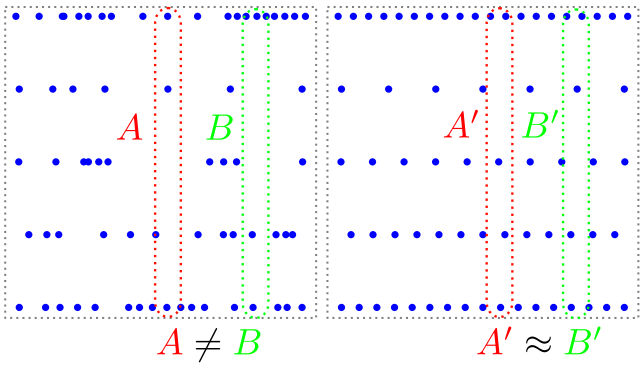

Figure 4.2: (Left) Five rows of the importance map. Blue dots indicate view samples. (Right) the importance map constructed using the set of row warping functions. Columns in the left do not contain the same number of view samples. Columns in the right contains approximately the same number of view samples.

the RTW algorithm as well. It can be implemented, for instance, as a weighted average of the results based on the number of view samples on a row or a column, respectively (see Figure 4.3).

The complete warping function can be expressed as:

$$\begin{aligned} warp(x,y) &= (x + o_x^{(i)}(x), y + o_y^{(j)}(y)) \\ i &= \lfloor y \cdot w \rfloor \\ j &= \lfloor (x + o_x^{(i)}(x)) \cdot w \rfloor \end{aligned} \tag{4.6}$$

where w is the shadow map resolution (number of pixels in one row), $o_x^{(i)}(x)$ is a warping function for i^{th} row, $o_y^{(j)}(y)$ is a warping function for j^{th} column.

When both sets of warping functions are applied, the view samples projected onto the projection plane of a light source are better spread as it can be seen in Figure 4.4.

Once both sets of the warping functions are constructed, the shadow map can be rendered (see Step 8 of the proposed Algorithm 2). a surface point with world space coordinate $v = (v_0, v_1, v_2, 1)$ is projected onto the shadow map .

4.1.5 Minimal Shadow Frustum Extension

The Non-orthogonal Texture Warping algorithm is extended with an additional improvement. The technique for finding a *Minimal Shadow Frustum* (MSF) [16] was implemented, and it was extended using rotating caliper . Using this technique, the NoTW algorithm projects only parts of the scene that are visible in the camera view frustum and occluders outside the frustum that cast shadows on objects inside the frustum. However, since the MSF algorithm is complex, it runs on CPU and thus it may influence rendering speed. Moreover, issues caused by precision of floating point operations have to be considered during implementation.

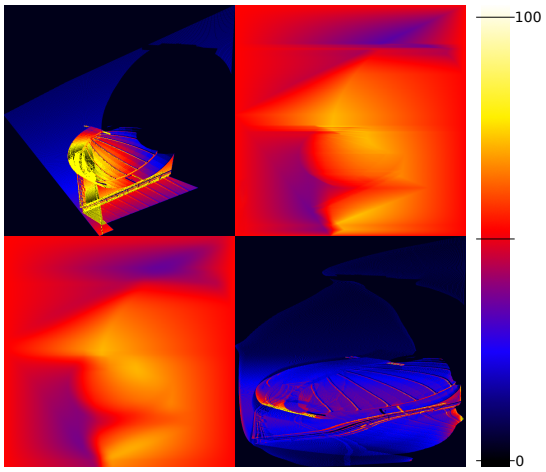

Figure 4.3: (Top, left) Importance map. (Top, right) A set of warping functions for every row of the importance map. (Bottom, left) Smoothed warping functions. (Bottom, right) the importance map after application of row warping functions - importance map for columns. Yellow color in warping functions means positive offset for a particular position in the row.

Rosen presented Desired View (DV) function that works similarly to the MSF. However, he did not clearly show how it influences the overall quality. The NoTW algorithm supports the DV as well, but it is only used as pre-process step before computing the importance map. The DV simply finds minimum and maximum view samples coordinates in the importance map. In addition, the MSF rotates the bounding box to an optimal position and adjusts near and far planes. Rosen computes the DV in the RTW approach from the importance map by finding first/last row and column that contains an importance value greater than zero. In the NoTW approach, the DV is computed by parallel reduction over the set of view samples projected into the shadow map space. It does not contribute to warping process, but it only crop the relevant part of shadow map.The DV function can be applied before construction of the warping functions .

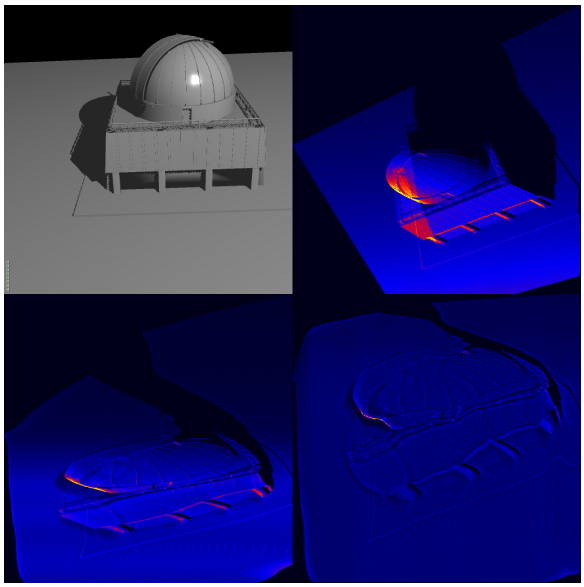

Figure 4.4: (Top, left) Scene rendered from a camera point of view. (Top, right) the importance map created from view samples. (Bottom, left) Reprojected view samples using only row warping functions. (Bottom, right) Reprojected view samples using both sets for warping functions.
It can be seen that importance is more spread across the importance map in the final stage. Black parts of second image are pixels with no view samples. These pixels correspond to those shadow map pixels that are useless - they resolve shadowing equation for invisible parts of the scene. In final image, these black parts almost disappear.

CHAPTER 5

Experimental Results and Discussion

The Non-orthogonal Texture Warping scheme has been evaluated on various scenes. The experiments performed and described in this chapter show that the approach is fast and capable of rendering high-quality shadows for complex light sources. Also, various improvements and extensions that can be used together with the NoTW algorithm are discussed.

5.1 High-quality Shadows

The NoTW algorithm improves the Shadow Mapping algorithm. The most important contribution of the NoTW algorithm is the reduction of the aliasing error in a scene and increasing the quality of rendered shadows. In the Shadow Mapping algorithm, poor quality shadows can be rendered which produces "jagged" shadow edges. In order to evaluate precision of rendered shadows, the Shadow Volumes algorithm was chosen as the ground truth, because it provides sample-precise shadows (see Figure 5.1).

This Section presents various scenes on which the evaluation has been performed. The output images show incorrectly computed shadow pixels in red color.

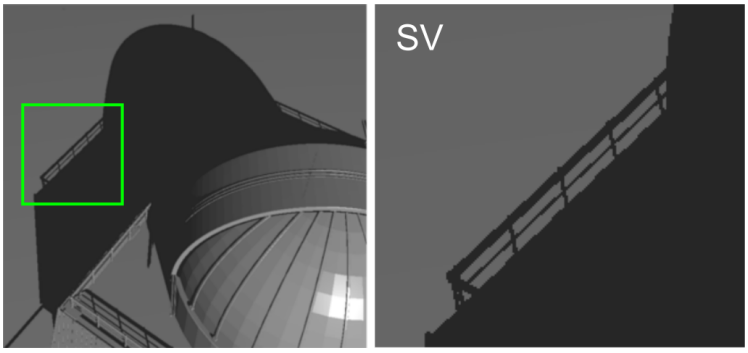

Figure 5.1: The reference image illustrates the Observatory scene (left) and zoomed detail of the image (right) that is used for evaluation of the quality.

5.1.1 Comparison with Standard Shadow Mapping

This section shows differences in quality between the standard Shadow Mapping algorithm as introduced in Section 2.3 and the NoTW algorithm. The results were measured for Observatory scene on 1024×1024 resolution of the output image and with 512×512 resolution for the shadow map. In Figure 5.2, differences from the reference solution are presented.

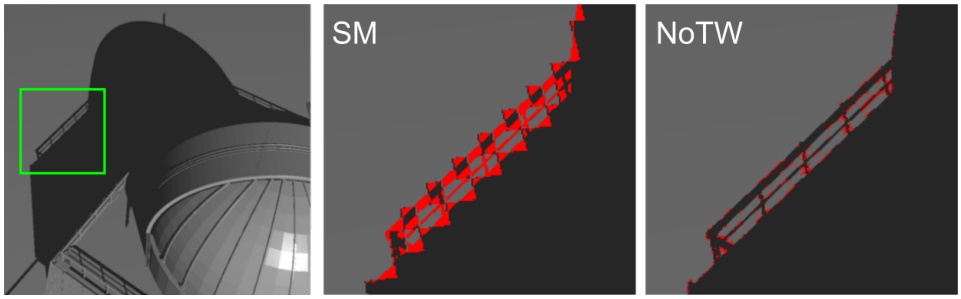

Figure 5.2: (Left) the reference image of Observatory. (Middle) the detail rendered with the Shadow Mapping algorithm. (Right) the same detail rendered with the Non-orthogonal Texture Warping algorithm.

The basic Shadow Mapping algorithm has no ability to focus on the current camera view. It covers the whole scene with the shadow map and the aliasing error in this case is really high. The red pixels in Figure 5.2 (left) illustrate that many view samples were projected onto a single shadow map texel. The NoTW algorithm, on the other hand, project the view samples more uniformly to the shadow texels.

5.1.2 Comparison with RTW

The Rectilinear Texture Warping (RTW) algorithm is the most similar approach to the NoTW approach and since some improvements of the RTW algorithm are suggested in Section 4.1, the visual quality has been explicitly compared to the RTW algorithm as well. Implementation of RTW algorithm with backward analysis has been used for creation of the importance map. Both the Distance to Eye and the Desired View importance functions were enabled in all reference images (see Section 3.2.2 for more details about the importance functions).

Figure 5.3 (left) shows that the sampling distribution is more uniform in the RTW algorithm in comparison to the standard Shadow Mapping algorithm presented in the previous section.

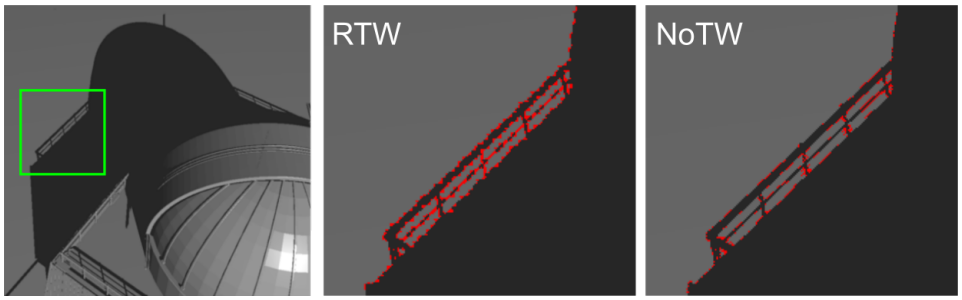

Figure 5.3: (Left) the reference image of Observatory. (Middle) the detail rendered with the Rectilinear Texture Warping algorithm. (Right) the same detail rendered with the Non-orthogonal Texture Warping algorithm.

5.2 Performance

This section presents experiments related to the speed of the Non-orthogonal Texture Warping algorithm (NoTW). Every improvement in quality can bring additional computation cost, however, it still has to maintain interactive rates. Moreover, the Shadow Volumes algorithm defines a lower boundary for speed. In the following text, all approaches have been compared to fully optimized and accelerated Silhouette-based Shadow Volumes approach introduced by Milet et al. [9].

5.2.1 Basic Shadow Algorithms

Table 5.1 shows frame times for all experimental scenes. This is a basic performance comparison of the NoTW algorithm with different approaches.

Scene	Conf. room	Sponza	Observatory
triangles	126665	261978	52583
gbuffer	2.16	2.229	1.84
SV	9.64	18.41	14.96
SM	0.21	0.40	0.16
RTW	3.14	3.47	3.02
NoTW	**3.63**	**3.84**	**3.23**

Table 5.1: Performance comparison of implemented methods for different scenes. Times are in milliseconds.

The accelerated Shadow Volumes algorithm (SV) introduced by Milet et al. is the slowest. It can be seen that the frame times depends on the scene complexity. This is a common property of all shadow rendering algorithms and namely the Shadow Volumes. On the other hand, the standard Shadow Mapping algorithm (SM) is the fastest approach, but it has the worst quality of

the output as described in Section 5.1. The RTW as well as NoTW approaches performs better than SV and the timings are almost equal.

5.3 Complex Light Sources

From the Shadow Mapping algorithm point of view, omnidirectional light sources are considered to be complex light sources. They require additional computation steps to be capable of rendering shadows into all direction.

Omnidirectional light sources introduce an advanced use case and it brings additional complexity to the algorithm. The Non-orthogonal Texture Warping algorithm supports also this type of light sources and this section presents visual as well as performance comparison of the Cube Shadow Mapping (CubeSM) and Dual-Paraboloid Shadow Mapping (DPSM) algorithms (presented in Section 3.3) extended with the NoTW scheme. It shows that the NoTW algorithm is applicable to arbitrary use case when it is integrated into existing algorithms for omnidirectional shadow rendering and extended with a zooming feature (e.g. Desired View function or Minimal Shadow Frustum extension).

5.3.1 Omnidirectional Light Sources

To validate robustness of the NoTW algorithm, a simple but still general use case was chosen. The light source is considered as a dynamic object that can be easily visible in the camera view frustum as it travels through a scene. The reason is that in this case, the CubeSM as well as the DPSM have to fully use their resources. In Figure 5.4, one such a use case is depicted.

The next step in evaluation of the visual quality is comparison of the rendered shadow maps (see Figure 5.5). It illustrates how the improved parameterization modifies the shadow map and how the vertices are moved from their initial positions. It is expected that when warping functions are applied, the scene rendered into the shadow map is highly deformed and objects are not be clearly recognizable. Also, it is necessary to apply the same functions in the process of computation shadows when the samples are projected into the light space. The warping scheme in the NoTW as well as RTW algorithm has to ensure that all samples are projected on the correct place in the shadow map.

Finally, Figure 5.6 illustrates the count maps that were analyzed in order to derive the warping functions. Closer look shows that DPSM algorithm is more efficient in using the space available in the map. This is the reason why the DPSM algorithm extended with the NoTW scheme produces better results. Since one side of the paraboloid covers a bigger part of the scene than one cube face frustum, there is more oversampled regions in the shadow map rendered with the DPSM approach. In other words, there is more space where the view samples can be distributed.

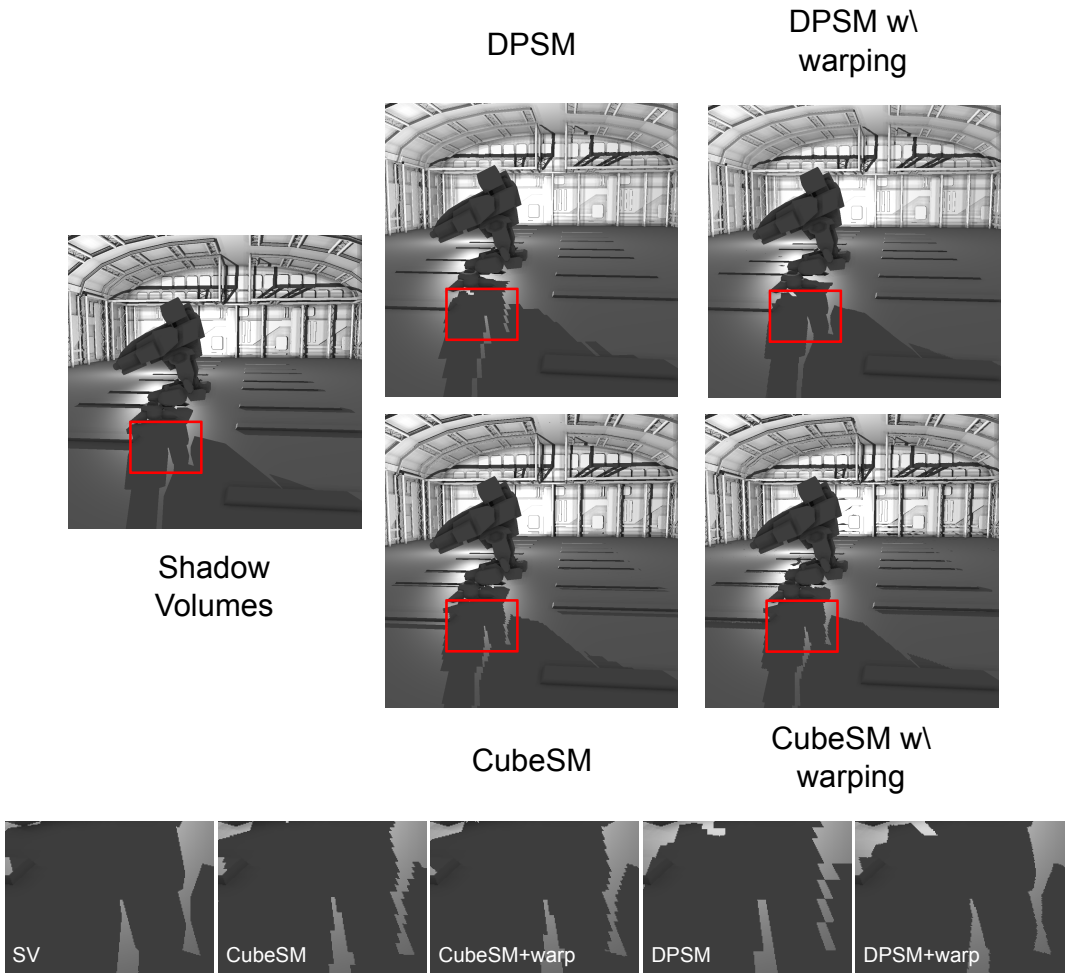

Figure 5.4: Shadow quality can be compared e.g. according to shadow boundaries. „Jagged“ edges shows that the ratio between view samples and shadow samples is high. Shadow Volumes algorithm rendered the reference image.

However, the warping functions had to be smoothed as described in Section 4.1.4. Therefore, the warping functions do not distribute the view samples over the entire shadow map. The smoothing factor is controlled by the user and it was set manually for each of the testing scenes.

Performance

Since the Shadow Mapping algorithm consists of various steps, execution times of the steps were also measured for all tested approaches.

The Table 5.2 shows that the biggest impact in NoTW approach is observed in rendering of the shadow map, because of the importance map creation, analysis, and deriving of the warping functions. It has to be noted that even though the Shadow Volumes algorithm is fully optimized and capable of running in real-time, the frame times are not stable between frames. It depends on complexity of the scene and in the worst case, rendering of single frame took 16ms.

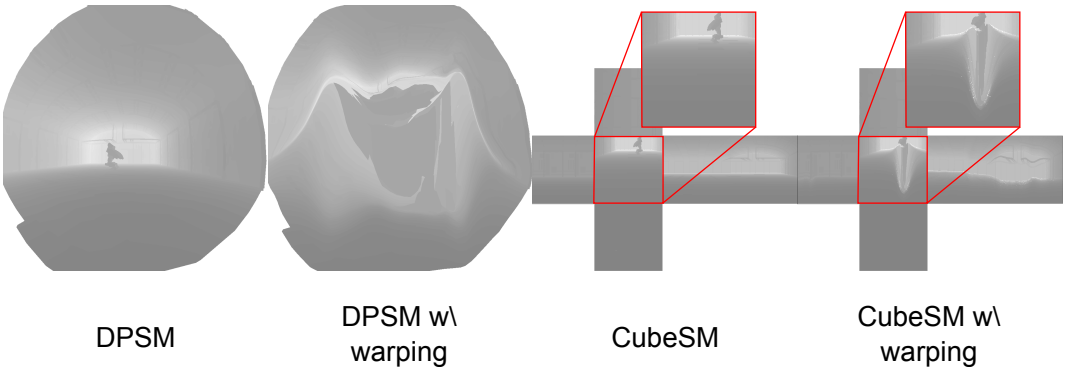

Figure 5.5: Comparison of shadow maps. The warping functions cause the scene is hardly recognizable.

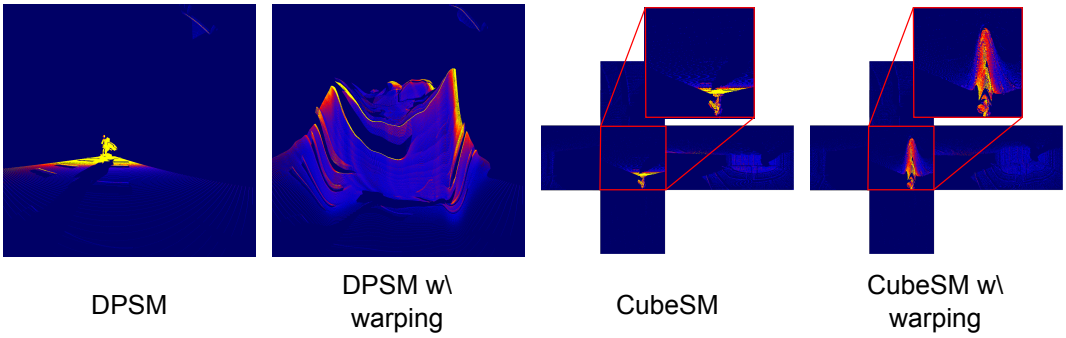

Figure 5.6: Comparison of count maps. For the Cube Shadow Mapping, only one face contains most of the shadow samples.

For shadow mapping-based approaches, the times were stable.

Method	Frame time	SM rendering	Shadow computation
SV	4.8*	N/A	N/A
CubeSM	2.5	0.38	0.10
DPSM	2.3	0.16	0.09
CubeSM w/ warping	**5.8**	**3.6**	**0.11**
DPSM w/ warping	**3.4**	**1.2**	**0.10**

Table 5.2: Performance comparison of implemented methods for omnidirectional shadow rendering. Times are in milliseconds.

5.3.2 Effect of Desired View

Experimental results also showed that extension of the NoTW algorithm with the Desired View (DV) function (or Minimal Shadow Frustum extension, MSF) is major part of decreasing alias error, but in some situation it is not sufficient. The main reason for focusing on these methods is that it should confirm whether the DV or MSF is not sufficient enough to render images of the similar

quality. Since Rosen [15] described this importance function, however, he did not show any results.

The MSF or DV perform better than the texture warping techniques when a small part of a scene is rendered. However, in real world scenes the camera renders a bigger part of a scene and in this case the warping techniques perform better (see Figure 5.7). The MSF or DV do not generate the view frustum small enough and thus artifacts on shadow edges are more apparent. The performance of DV and MSF depends on current hardware setup. MSF performs better than DV when running on fast CPU and slow GPU.

The effect of the DV function is nicely visible in Figure 5.7. First image shows the Observatory scene rendered using the traditional Shadow Mapping algorithm with a directional light source. Second image, shows how the scene is zoomed on the part visible in the camera view frustum when only the DV is applied as a pre-process step in the NoTW algorithm. The RTW includes the DV function in the set of importance functions by default and it is used to analyze the importance map and derive the warping functions. The last image depicts the NoTW algorithm. It is similar to the result from the RTW algorithm but the parameterization is a bit different which leads to lower alias error (see the shadow maps in Figure 5.7).

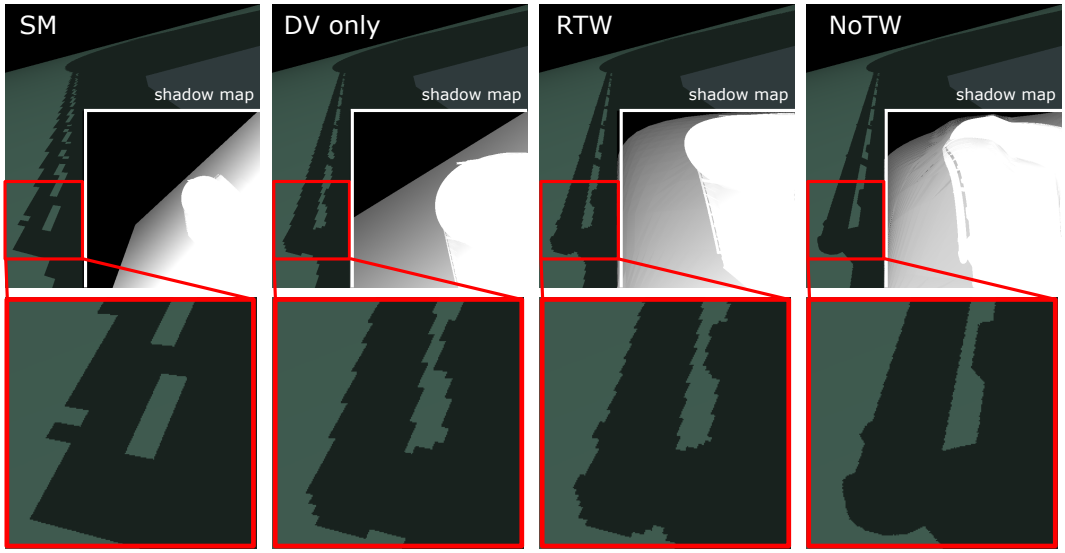

Figure 5.7: Images show shadow maps for Observatory scene. (From left to right) Shadow Mapping (SM), NoTW with only DV function (no warping applied), Rectilinear Texture Warping (RTW), complete NoTW including warping.

The Desired View (DV) function and Minimal Shadow Frustum extension, MSF) help to generate the high-quality shadows with a small additional cost. However, when the warping techniques employ all their features, the results are even better and the impact on performance is not crucial (see Table 5.3)

The similar effect as the DV function has the Improved Paraboloid Shadow Mapping (IPSM) [17] approach . The IPSM has been designed for optimization

Method	time per frame
SM	1.596
MIN-SM	1.7
SV	8.750
RTW	3.296
NoTW	**4.708**
NoTW-DV	**2.521**

Table 5.3: Performance comparison of implemented methods. Times are in milliseconds.

of the Dual-Paraboloid Shadow Mapping algorithm in cases when the light source is outside the camera view frustum.

5.4 Discussion

The motivation for this work has been an idea that most of the researchers have been focused on improving shadow quality in the Shadow Mapping algorithm, but only for directional light sources, spotlights and large environments, e.g. Parallel-Split Shadow Maps [20] . The Non-orthogonal Texture Warping (NoTW) algorithm has been initially designed for methods involved in omnidirectional shadow casting where improving of the shadow quality has not been explicitly investigated.

Experiments presented in this Chapter showed that the NoTW algorithm is successfully usable in various environments without any modification. Algorithms that depend on view and scene context could be replaced with one solution. Applications can save some time when they do not have to deal with an expensive switching between multiple methods with different data structures and demands on resources.

The initial proof-of-concept has been implemented on contemporary GPUs and the algorithm runs in interactive rates. At this point, there is a space for further optimizations and improvements. The deriving of the warping functions can be further optimized with parallel processing units, e.g. CUDA.

5.4.1 Limitations

However, the solution has also some disadvantages. The NoTW algorithm as well as the RTW algorithm have to deal with the linear rasterization unit. Figure 4.4 (bottom right) shows how the warping functions distorted the space. Nowadays, the rasterization pipeline can handle only the polygonal mesh. If the warping function changes rapidly between two vertices, some errors can be seen . Lloyd introduced the nonlinear rasterizer [7] that could replace the traditional rasterization pipeline and allow for processing non-linear data.

In the experiments, a few techniques have been used in NoTW to deal with

these errors. Firstly, it utilized the adaptive tessellation provided by OpenGL. The similar improvement was suggested by Rosen et al [15]. Further, the quality can be controlled by adjusting the size of smoothing window. Another solution is to use weights during smoothing step. It can influence sizes of offset values. In the experiments, these parameters were manually set to fit the current view. When they were set inappropriately, the warping functions do not work correctly so that it totally deforms the shadow map and also produces artifacts and incorrectly computed shadows in the output image.

In the future work, some constraints have to be defined that should be involved in deriving of the warping functions. It should allow for adjusting the parameters automatically and render the output image with the highest quality.

The limitation of the NoTW algorithm is also missing support for Minimal Shadow Frustum extension (see Section 4.1.5). It should perform better that the Desired View function, but due to precision issues in floating point arithmetic it ended only as a prototype with a very poor performance. However, the basic version of the technique has been used in the standard Shadow Mapping algorithm.

5.4.2 Implementation Details

The algorithm has been implemented in OpenGL 4.4 using compute shaders. For creation of the importance map, image atomic operation *imageAtomicAdd* that occurs in OpenGL has been used to save time spent on GPU. The results were measured on a PC running Intel Core i7 4790 with 16GB of memory. The scenes were rendered on a high-end GPU: NVidia GTX 980 and Titan X. Operation system was Linux Ubuntu 14.04.2.

The solution requires additional memory in comparison to the basic Shadow Mapping algorithm. Deferred shading has been used for creation of the G-buffer that requires set of 2D textures. Two one-channel floating point 2D textures have been used for storage of the warping functions with the same resolution as the shadow map. Furthermore, the algorithm requires few textures for storing temporary results - the importance map, prefix sum map and storage for warping functions. The additional memory requirements are thus dependent on the shadow map resolution.For instance, when using the shadow map with resolution $w = 1024$, additional 20 MBytes of the memory needs to be allocated.

The memory requirements can be decreased by using e.g. another format of textures. For instance, 16bit textures for the importance map or prefix-sum map. Also, with increasing number of lights, the memory requirements increase only for storing the warping functions: $8w^2$[bytes] for one light source.

CHAPTER 6

Conclusion

The main goal of this work has been improvement of the shadow rendering based on the Shadow Mapping algorithm using Non-orthogonal Texture Warping of the shadow maps. The goal of the work has been achieved and its main contribution is experimental evaluation of the hypothesis that parameterization of shadow map coordinates based on simple scene analysis can reduce aliasing error of the shadows cast by complex light sources.

The experimental results demonstrated that the reducing of aliasing error in shadows could be achieved by modification of projection mapping. This has been evaluated on various use cases. Further details can be found in Chapter 5. Evaluation of the hypothesis included rendering shadows in indoor as well as outdoor scenes with various configurations. The experiments showed that the non-orthogonal warping scheme is applicable to standard Shadow Mapping algorithm and it improved the sampling rate for complex light sources as well.

Results of the work can be applied in various computer graphics applications that rely on quality of shadows in real time. The range of possible applications is from CAD systems e.g. in architecture where shadow rendering is critical for the realistic perception of the buildings to computer games where shadow rendering is nowadays required even in complex scenes and appreciated by the game players as a part of gaming virtual reality.

Future work will be focused on further improvements of robustness and balancing the warping parameters and on better estimation of the error distribution on the shadow maps. Possible other direction of focus would be extensive evaluation of the method on large and complex scenes and measurement of improvement and combination of the warping with other GPU functions. Very interesting direction would be to connect all approaches that employ nonlinear functions with nonlinear rasterization pipeline.

Bibliography

[1] Stefan Brabec, Thomas Annen, and Hans-Peter Seidel. Shadow mapping for hemispherical and omnidirectional light sources. In *In Proc. of Computer Graphics International*, pages 397–408, 2002.

[2] Michael Bunnell and Fabio Pellacini. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, chapter Shadow Map Antialiasing. Pearson Higher Education, 2004.

[3] Franklin C. Crow. Shadow algorithms for computer graphics. *SIGGRAPH Comput. Graph.*, 11(2):242–248, 1977.

[4] Randima Fernando, Sebastian Fernandez, Kavita Bala, and Donald P. Greenberg. Adaptive shadow maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 387–390, New York, NY, USA, 2001. ACM.

[5] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[6] Nixiang Jia, Dening Luo, and Yanci Zhang. Distorted shadow mapping. In *Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology*, VRST '13, pages 209–214, New York, NY, USA, 2013. ACM.

[7] D. Brandon Lloyd, Naga K. Govindaraju, Steven E. Molnar, and Dinesh Manocha. Practical logarithmic rasterization for low-error shadow maps. In *Proceedings of the 22Nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH '07, pages 17–24, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[8] D. Brandon Lloyd, David Tuft, Sung-eui Yoon, and Dinesh Manocha. Warping and partitioning for low error shadow maps. In *Proceedings of*

the 17th Eurographics Conference on Rendering Techniques, EGSR '06, pages 215–226, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[9] Tomáš Milet, Jozef Kobrtek, Pavel Zemčík, and Jan Pečiva. Fast and robust tessellation-based silhouette shadows. In *WSCG 2014 - Poster papers proceedings*, pages 33–38. University of West Bohemia in Pilsen, 2014.

[10] Tomáš Milet, Jan Navrátil, Adam Herout, and Pavel Zemčík. Improved computation of attenuated light with application in scenes with multiple light sources. In *Proceedings of SCCG 2013*, pages 155–160. Comenius University in Bratislava, 2013.

[11] Jan Navrátil, Pavel Zemčík, Roman Juránek, and Jan Pečiva. A skewed paraboloid cut for better shadow rendering. In *Proceedings of Computer Graphics International 2012*, page 4. Springer Verlag, 2012.

[12] Brian Osman, Mike Bukowski, and Chris McEvoy. Practical implementation of dual paraboloid shadow maps. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames*, Sandbox '06, pages 103–106, New York, NY, USA, 2006. ACM.

[13] Jan Pečiva, Jaroslav Přibyl, and Jan Navrátil. Close-to-photorealistic lighting for simulations and cad. In *2011 International Simulation Multiconference - SCS (SCSC, SPECTS, GCMS) - SCSC Proceedings (HardCopy) 11*, pages 1–2. SCS Publication House, 2011.

[14] Jan Pečiva, Pavel Zemčík, and Jan Navrátil. Mimicking pov-ray photorealistic rendering with accelerated opengl pipeline. In *WSCG'2011 Communication Papers Proceedings*, 19-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pages 149–156. University of West Bohemia in Pilsen, 2011.

[15] Paul Rosen. Rectilinear texture warping for fast adaptive shadow mapping. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, pages 151–158, New York, NY, USA, 2012. ACM.

[16] Marc Stamminger and George Drettakis. Perspective shadow maps. *ACM Trans. Graph.*, 21(3):557–562, July 2002.

[17] Juraj Vanek, Jan Navrátil, Adam Herout, and Pavel Zemčík. High-quality shadows with improved paraboloid mapping. In *Advances in Visual Computing*, Lecture Notes in Computer Science 6938, pages 421–430. Faculty of Information Technology BUT, 2011.

[18] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, June 1980.

[19] Lance Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 12(3):270–274, August 1978.

[20] Fan Zhang, Hanqiu Sun, Leilei Xu, and Lee Kit Lun. Parallel-split shadow maps for large-scale virtual environments. In *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications*, VRCIA '06, pages 311–318, New York, NY, USA, 2006. ACM.