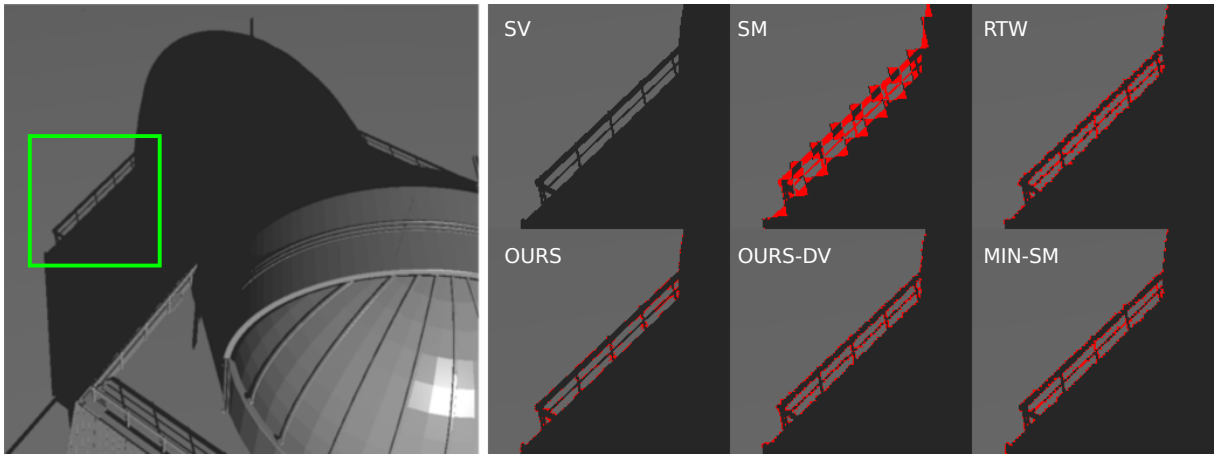
An Improved Non-Orthogonal Texture Warping for Better Shadow Rendering

Tomáš Milet
Brno University of Technology Czech Republic
imilet@fit.vutbr.cz

Jan Navrátil
Brno University of Technology Czech Republic
inavrati@fit.vutbr.cz

Pavel Zemčík
Brno University of Technology Czech Republic
zemcik@fit.vutbr.cz



The figure shows the difference in quality. Images are zoomed on shadows cast from Observatory scene for different methods. Red pixels are wrongly evaluated. From left to right: Shadow Volumes (SV), Shadow Mapping (SM), Rectilinear Texture Warping (RTW), our solution, our solution using only desired view (DV), SM + minimal shadow frustum.

ABSTRACT

In interactive applications, shadows are traditionally rendered using the shadow mapping algorithm. The disadvantage of the algorithm is limited resolution of depth texture which may lead to aliasing artifacts on shadow edges. This paper introduces an improved depth texture warping with non-orthogonal grid that can be employed for all kinds of light sources. For instance, already known approaches for reducing aliasing artifacts are widely used in outdoor scenes with directional light sources but they are not directly applicable for point light sources. We show that the improved warping parameterization reduces the aliasing artifacts and we are able to present high quality shadows regardless of a light source or a camera position in the scene.

Keywords

shadow-mapping, alias, warping, local warping, minimal frustum, shadows

1 INTRODUCTION

Images rendered on computers are still being improved with various visual effects. Nowadays, computers can synthesize images in nearly photorealistic quality and in real-time. One of the most important visual cues, still worth improving, are shadows. The two key algorithms for shadow rendering [Wil78, Cro77] have been accelerated on GPUs.The shadow volumes approach [Cro77] suffers from the need to render large amount of data to gather necessary information for rendering shadows. On the other hand, shadow mapping approach [Wil78] is limited by the size of depth texture. In this paper, we addressed this problem with the improved depth texture parameterization that makes use of the available resources more efficient.

The resolution of the shadow map determines the number of samples that can be utilized. Recently, some

approaches were introduced that improve sampling of the important parts of the scene. These approaches work well for outdoor scenes where we can expect that the major part of lighting comes from the directional light source [ZSXL06]. This type of light sources can be processed efficiently with the shadow mapping algorithm. On the other hand, the basic shadow mapping algorithm cannot easily address point light sources and it needs additional improvements, e.g. using cube shadow maps, or an alternative parameterization [BApS02, Ros12, JLZ13]. In this case, the improved sampling is difficult to achieve using approaches mentioned above because the algorithms would need non-trivial modification. We omit the most complex types of light sources such as area lights or volume-based emitters in this paper.

Since the point light sources can cast shadows into all directions, the regions where the need to improve scene sampling exists can be distributed throughout the depth textures. It all depends on the scene complexity and also on the mutual position of the light source and the camera. While outdoor scenes are lit by directional light source, the important parts are located in front of the camera and the importance decreases with a distance from the camera.

Various approaches have been introduced that can parameterize the shadow map and thus improve sampling on selected parts of the scene based on the scene analysis [VNHZ11, NZJP12, Ros12, JLZ13]. However, neither of these approaches is fully automatic or robust enough and they work only in few cases when the scene is lit with directional light sources or the light source is outside a camera view frustum.

Our solution computes an improved parameterization based on importance driven depth texture warping. We can identify regions in the depth texture where the sampling is not optimal and enlarge this regions in order to get higher sampling rate. We employed modern GPUs in the warping process thus these additional computing steps have no crucial impact on an overall performance.

We introduce an additional step that is performed before the traditional shadow mapping algorithm is applied. In this step, a non-orthogonal warping grid is computed and this grid is used during the shadow rendering step.

Our 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. [Ros12],
- the non-orthogonal warping grid which leads to better control of importance-based warping without affecting the nearest regions in the texture (in the same row and/or column).

2 PREVIOUS WORK

The shadow mapping algorithm was first published in 1978 [Wil78]. Since then, many approaches addressing its aliasing issues have been published.

Stamminger and Drettakis [SD02] introduced an idea of creating depth texture after performing of perspective projection. This step emphasizes regions in front of the camera where the aliasing error are mostly observable. However, results in this approach are dependent on the mutual position of a camera and a light source. In some case, creating depth texture in post-perspective space may lead to very unpleasant results because of the perspective transformation function. Also, the overall results are influenced by object outside the camera view frustum because they introduce additional complexity to the computation.

Fernando et al. [FFBG01] introduced a hierarchical structure by subdividing shadow map into smaller shadow map pages having different resolutions due to different level of aliasing in different parts of the current camera view frustum. With camera being dynamic, this hierarchical structure needs to be updated per frame and due to limitations of graphics hardware of that time, most of the algorithm runs on CPU.

This method was further optimized by Lefohn et al. [LSK$^+$05]. Evolution of GPU hardware allowed more of the algorithm to move on the graphic chip itself by programmable vertex and pixel shader pipeline stages.

Parallel-split shadow maps approach was introduced by Zhang et al. [ZSXL06]. The idea is to split view frustum into multiple parts according to depth, split light frustum into multiple ones and then independent shadow maps are rendered for each layer. Splitting view frustum is based on a practical splitting algorithm which averages logarithmic and uniform splitting scheme. However, this method is targeted and optimized for outdoor scenes and it would need some amount of work to adapt it for indoor scenes and namely point light sources. Also, the approach does not deal with the perspective aliasing error correctly.

Based on Zhang, Lauritzen et al. [LSL11] introduced Sample distribution shadow maps which further improves partitioning. The camera frustum is partitioned automatically based on receiver sample distribution given by depth buffer, eliminating areas with no shadow samples. This sample distribution is also used to compute tightly-bound light-space partition frusta.

The first method that addressed problem of important regions distributed in the depth texture was introduced by Rosen [Ros12]. 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.Other parts of a scene may receive unneeded resolution. This can lead to reduction in overall quality.

Similar approach was published by Jia et al. [JLZ13]. They do not limit the approach to perpendicular splitting planes; 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.

Finally, some approaches that were focused only on point light sources have been published recently [VNHZ11, NZJP12]. Nevertheless, they discussed possibilities for improving shadow quality using Dual-Paraboloid shadow maps [BApS02]. But this technique is not sufficient due to its nonlinear transformation during generation of depth textures. This introduces additional limitation regarding model quality and especially size of polygons.

2.1 Scene Sampling and Parameterization

The shadow mapping algorithm works with two types of samples. A *view sample* is a point on a scene surface that is described by its 3D position (and other properties such as color, normal vector etc.). The view samples are generated by sampling the scene from a camera point of view. Secondly, *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.

Another solution is to parameterize the sampling using a *warping function* $y = f(x)$. 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 [SD02]. The local warping function is derived from properties of view samples and scene analysis [Ros12, JLZ13].

2.2 Rectilinear Texture Warping

Our algorithm is partially based on Rectilinear Texture Warping (RTW) approach [Ros12]. Let us make a short overview of RTW algorithm using backward analysis.

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 scene from a light source point of view. Then, the importance map is computed.One additional rendering step is necessary to compute shadows. 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 a camera. The second rendering pass creates a warped shadow map. Its complexity is linear with relation to the number of light sources.

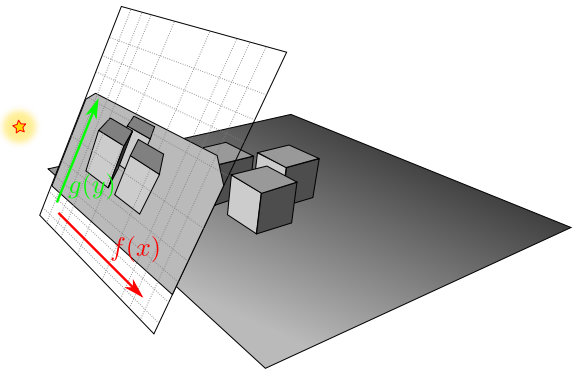


Figure 1: Two 1D warping functions enlarge parts of the scene that are important according to the importance map.

The warping function in RTW is composed of two 1D warping functions that operate in projection plane of a light source (see Figure 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.

3 SHADOW RENDERING USING NON-ORTHOGONAL WARPING GRID

The basic idea of our algorithm 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 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. Because of this fact, we can assume that the best result is observed when the number of view samples for all shadow samples is the same.

In our approach, the importance map has the same resolution as the shadow map. Every pixel in the importance map stores the number of view samples that are sampled by the given shadow sample. 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. 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

The first step is generation of the G-buffer. Apart from other properties, it contains positions of view samples that we need to analyze the importance for.

The most important are the steps 2-7 where we construct the set of 1D warping functions. We derive the warping functions in different manner than Rosen [Ros12]. For every row and every column, we construct one 1D warping function separately and thus we do not allocate unneeded resolution in other parts of the shadow map. The degree of freedom for warping functions is increased using this approach and we should not allow the situation illustrated on the Figure 2. The steps are described in detail in the following section.

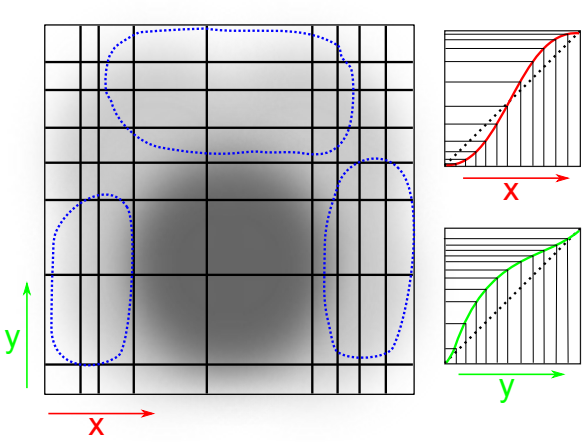


Figure 2: Importance map for RTW, Left: Combination of two 1D warping function, Right: two 1D warping function. It can be seen that blue parts are oversampled. The larger cells cover more important areas of the shadow map.

3.1 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 $g(x)$:

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

$$s = g(x) = \int_0^x f(x)dx \tag{2}$$

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:

$$\frac{g(x)}{g(1)} = \frac{x}{1} \tag{3}$$

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

$$o(x) = \frac{g(x)}{N} - x \tag{4}$$

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:

$$x' = x + o(x) \tag{5}$$

Before we proceed with construction of warping functions for columns, we need to recompute the importance map. But now, we apply the newly derived set of 1D warping functions for rows. After this step, the number of view samples that have to be redistributed in a given column is nearly constant (see Figure 3). When the 1D warping functions for columns are derived, all the view samples are distributed uniformly.

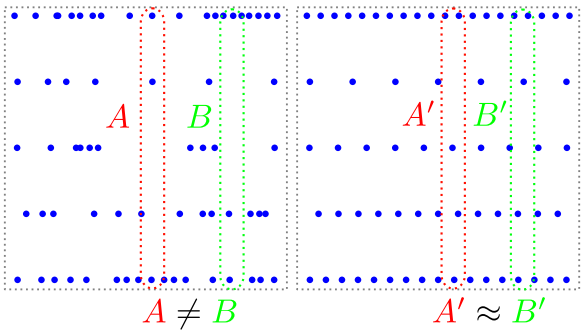


Figure 3: Left: Five rows of the importance map. Blue dots indicate view samples. Right: the importance map constructed using the set of row warping functions. Columns in the left do not contain the same number of view samples. Columns in the right contains approximately the same number of view samples.

As we mentioned in the Section 2.2, the RTW algorithm constructs two warping functions - for rows and columns respectively. We improve this approach and construct the set of warping functions for all rows and all columns. Nevertheless, we need to smoothen these functions in order to prevent them from providing too different offsets. Otherwise, the large polygons that are linearly rasterized would not be processed by the warping functions correctly. The smoothing step is included in the RTW algorithm as well. Rosen performs this step on the warping functions. However in our approach, we smoothen among all warping functions. 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).

The complete warping function can be expressed as:

$$\begin{aligned} warp(x,y) &= (x + o_x^{(i)}(x), y + o_y^{(j)}(y)) \\ i &= \lfloor y \cdot w \rfloor \\ j &= \lfloor (x + o_x^{(i)}(x)) \cdot w \rfloor \end{aligned} \tag{6}$$

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.

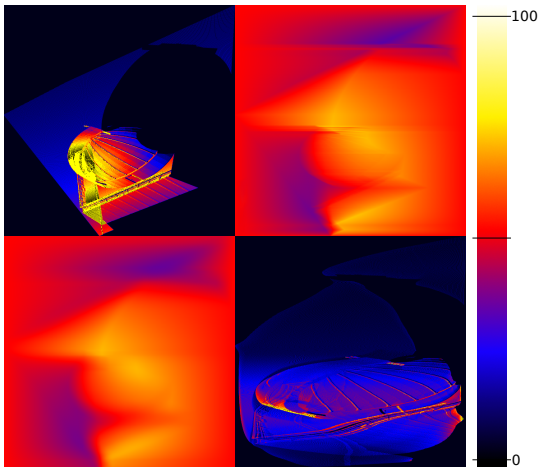


Figure 4: Top Left: Importance map, Top Right: a set of warping functions for every row of the importance map. Bottom left: smoothed warping functions, Bottom right: The importance map after application of row warping functions - importance map for columns. Yellow color in warping functions means positive offset for certain position in the row.

When we apply both sets of warping functions, the view samples projected onto the projection plane of a light source are better spread as can be seen on the Figure 5.

Once we constructed both sets of the warping functions, the shadow map can be generated (see Step 8 of the proposed algorithm). A surface point with a world-space coordinate $\mathbf{v} = (v0, v1, v2, 1)$ is projected onto the shadow map in Algorithm 1. Final shadow map can be seen in Figure 6.

Input: **v** - vertex in world space, M - light projection view matrix
Output: **p** - vertex in the shadow map clip space

1 $\mathbf{a} = M \cdot \mathbf{v}$;
2 $\mathbf{b} = ((a_1, a_2)/a_4 + 1)/2$;
3 $\mathbf{c} =$ `warp(`**b**`)`;
4 $\mathbf{d} = (\mathbf{c} \cdot 2 - 1) \cdot a_4$;
5 $\mathbf{p} = (d_1, d_2, a_3, a_4)$;

Algorithm 1: Warping function that can be used in vertex / evaluation shader. Steps 1, 2 project vertex into normalized coordinates of shadow map. Step 3 moves vertex according to warping funcions. Steps 4, 5 project vertex back into shadow map clip space.

3.2 Minimal Shadow Frustum Extension

We extended our solution with an additional improvement. We implemented an algorithm for finding a minimal shadow frustum (MSF) [SD02] and we extended it using rotating caliper. Using this technique, we project

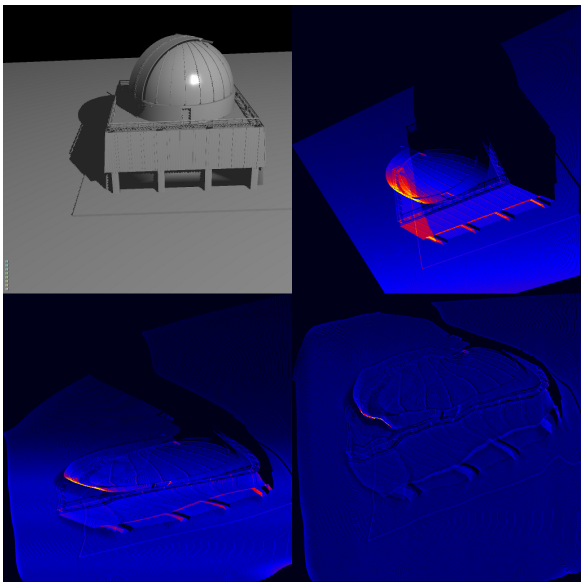


Figure 5: Top Left: Scene rendered from a camera point of view, Top Right: the importance map created from view samples. Bottom Left: reprojected view samples using only row warping functions. Bottom right: reprojected view samples using both sets for warping functions. It can be seen than importance 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.

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 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.

The goal of this additional improvement is to verify whether the MSF does not provide better results with a less cost.

Rosen et al. presented a desired view (DV) function that works similarly to the MSF. However, they did not clearly show how it influences the overall quality. We support the DV in our solution as well, but it is only used as pre-process step before computing the importance map.

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 et al. computes DV in RTW approach from the importance map by finding first/last row and column that contains an importance value greater than zero. In our solution, DV is computed by parallel reduction over the set of view samples projected into the shadow map space. DV does not contribute to warping process, it only focuses the relevant part of shadow map. We can apply the DV function before construction of the warping functions (before the Step 2 of the Algorithm 1):

3.3 Implementations Details

We implemented the algorithm in OpenGL 4.4 using compute shaders. For creation of the importance map, we used image atomic operation *imageAtomicAdd* delivered with OpenGL.

Our solution requires additional memory as compared to the basic shadow mapping algorithm. We used deferred shading for creating the G-buffer that requires set of 2D textures. For storing the warping functions, we used two one-channel floating point 2D textures. These have the same resolution as the shadow map. Further, the algorithm requires few textures for storing temporary results - the importance map, prefix sum map and storage for not smoothed warping functions. The additional memory requirements are thus dependent on the shadow map resolution.For instance, when we use the shadow map with resolution $w = 1024$, we need to allocate additional 20 MBytes of the memory.

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.

4 RESULTS AND DISCUSSION

The results were measured on a PC running Intel Core i7 4790 with 16GB of memory. We used a high-end GPU: NVidia GTX 980. Operation system was Linux Ubuntu 14.04.2.

We compared our solution with the Rectilinear Texture Warping algorithm (RTW) [Ros12], the basic shadow mapping algorithm (SM), accelerated silhouette-based shadow volumes algorithm (SV) [MKZP14] and the shadow mapping algorithm extended with the minimal shadow frustum (MIN-SM). We measured quality and speed of all approaches (see Table 1 and teaser image).

Regarding quality comparison, we selected the shadow volumes algorithm as the ground truth. It provides sample-precise shadows and moreover, it also defines a lower boundary for speed. No solution based on the shadow mapping algorithm can be slower than the silhouette-based shadow volume approach [MKZP14].

The RTW algorithm is the most similar approach to our solution. And since we suggested some improvements, the comparison to RTW is very important. We implemented RTW algorithm with backward analysis used

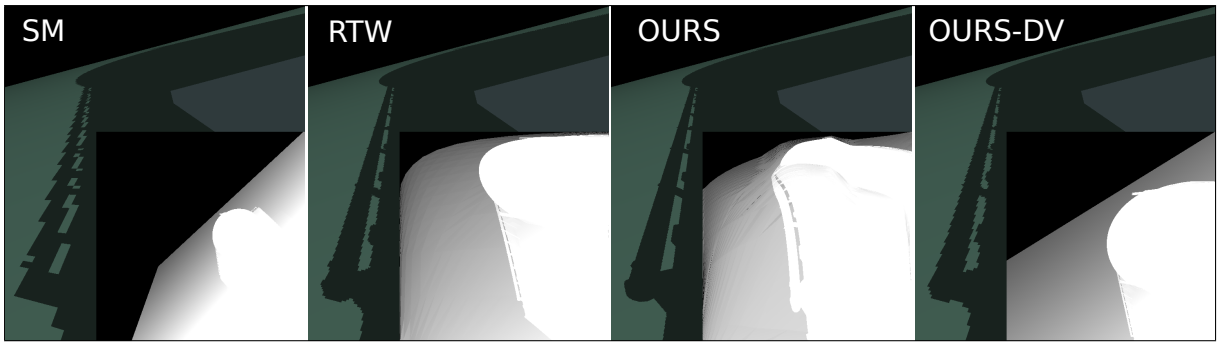


Figure 6: Images show shadow maps (grey squared images) for Observatory scene. From left to right: shadow mapping (SM), Rectilinear Texture Warping (RTW), our solution, our solution using only DV.

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

Table 1: Performance comparison of implemented methods. Times are in miliseconds. Measured for Observatory scene on 1024×1024 resolution with 512×512 resolution for the shadow map.

for creation of the importance map. The timings for the crucial steps of both approaches can be seen in Table 2. We used both the distance to eye importance function and the desired view function in all reference images (see teaser image).

Also, comparison with the shadow mapping algorithm extended with the minimal shadow frustum (MSF) shows some interesting results. The main reason for including this method is that we wanted to know whether the MSF is not sufficient enough to render images of the similar quality. Rosen did not described this extension and did not show any results.

We measured our algorithm on three scenes (see Figure 7). We selected various types of scenes (outdoor as well as indoor) in order to show that our solution can be adapted to different environment and types of light sources. Times for all scenes are shown in Table 3.

In Figure 7, you can see differences from a reference solution (shadow volumes algorithm). Red pixels are incorrectly computed.

As it can be seen in Table 3, our method is slightly slower than RTW but it produces better visual results (see teaser image and Figure 7). Measurements show that computing MSF is not expensive and it may be suitable in some situations.Also, it did not provide the best quality. Computing desired view (DV) function in our method is the third fastest method but visual results are worse than using MSF. Results also show that DV function is major part of decreasing alias error, but in some situation it is not sufficient.

scene	Conf. room		Sponza		Observatory	
	ours	rtw	**ours**	rtw	**ours**	rtw
desired view	**13.9**	89.1	**15.4**	82.5	**18.4**	86.8
imp. map	**68.4**		**61.0**		**69.0**	
shadow map	**14.4**	7.3	**19.9**	14.1	**8.2**	9.3
final pass	**3.3**	3.5	**3.7**	3.2	**4.4**	3.8

Table 2: Overhead of steps in our algorithm for different scenes. Values are in percent.

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
Ours	**3.63**	**3.84**	**3.23**

Table 3: Performance comparison of implemented methods for different scenes. Times are in miliseconds.

Minimal Shadow Frustum

The experimental results show that performance of DV and MSF depends on current hardware setup. MSF performs better than DV when running on fast CPU and slow GPU. When we compared the impact of both approaches on quality comparing the texture warping techniques, the results are following. The MSF or DV perform better 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 these cases the warping techniques perform better (see Figure 7 and 6). The MSF or DV do not generate the view frustum small enough and thus artifacts on shadow edges are more apparent.

4.1 Limitations

Our implementation as well as the RTW algorithm have to deal with linear rasterization unit. In Figure 5 (bottom right), we can see the result of our warping process. It can be seen that the warping functions distorted the

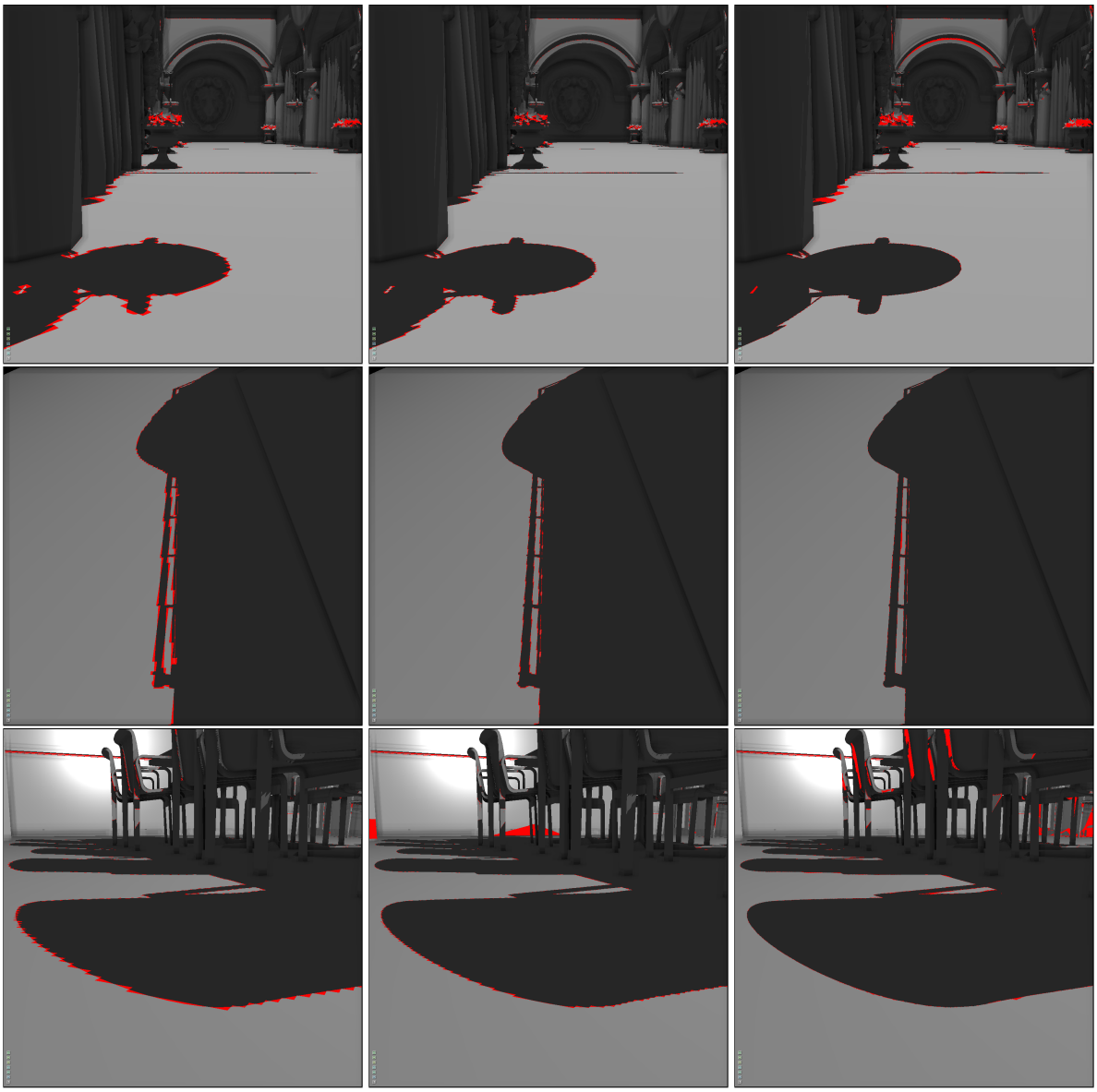


Figure 7: Images show difference between shadow mapping techniques and shadow volumes. Images in the first column show basic shadow mapping. The second column shows RTW method and third column shows our new warping method. First scene is Sponza, second scene is Observatory and last scene is Conference room. Times are shown in Table 3.

space. Nowadays, the rasterization pipeline can handle only the triangle vertices. If the warping function changes rapidly between two vertices, we can see some errors (see Figure 7 top, right for missing shadows under curtains). We used a few techniques in our solution to deal with these errors.

Firstly, we utilized the adaptive tessellation provided by OpenGL. The similar improvement was suggested by Rosen et al. Further, we modified the size of smoothing window when averaging the warping functions. The wider the window is the less different are the warping functions. In extreme case, our solution converts to the RTW algorithm. Another solution is to use weights during smoothing step. It can influence sizes of offset values.

5 CONCLUSION AND FUTURE WORK

This paper presents an extension of the Rectilinear Texture Warping algorithm achieved through the improved non-orthogonal warping grid constructed using the set of 1D warping functions. The novel importance warping function results in less artifacts at the shadow edges.

The improvement has been evaluated on various testing scenes. We showed that the method is fast and provides better results than the RTW algorithm. Also, we dis-

cussed various improvements and extensions that can be used together with our solution.

Standard methods for alias reduction globally change sampling rate using partitioning of a scene where directional light sources are commonly used. Our method change sampling rate locally and thus it can be used with other kinds of light sources using DPSM or cube maps.

The future work includes adaptation the algorithm for other visual effects, e.g. mirrors, refraction etc. We will focus on more experiments with the shadow mapping algorithm for point light source, i.e. the Cube Shadow Maps.

6 ACKNOWLEDGMENTS

This work was supported by the Ministry of Education, Youth and Sport of the Czech Republic under the research program TE01020415 (V3C - Visual Computing Competence Center). Additional data and algorithms were provided by Cadwork.

7 REFERENCES

[BApS02] 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.

[Cro77] Franklin C. Crow. Shadow algorithms for computer graphics. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '77, pages 242–248, New York, NY, USA, 1977. ACM.

[FFBG01] 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.

[JLZ13] 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.

[LSK+05] Aaron Lefohn, Shubhabrata Sengupta, Joe M. Kniss, Robert Strzodka, and John D. Owens. Dynamic adaptive shadow maps on graphics hardware. In *ACM SIGGRAPH 2005 Conference Abstracts and Applications*, August 2005.

[LSL11] Andrew Lauritzen, Marco Salvi, and Aaron Lefohn. Sample distribution shadow maps. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 97–102, New York, NY, USA, 2011. ACM.

[MKZP14] 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.

[NZJP12] 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.

[Ros12] 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.

[SD02] Marc Stamminger and George Drettakis. Perspective shadow maps. *ACM Trans. Graph.*, 21(3):557–562, July 2002.

[VNHZ11] 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.

[Wil78] Lance Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 12(3):270–274, August 1978.

[ZSXL06] 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.