EG UK Theory and Practice of Computer Graphics (2013)
Silvester Czanner and Wen Tang (Editors)

Resolution Estimation for Shadow Mapping

Michal Ferko[†]

Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia

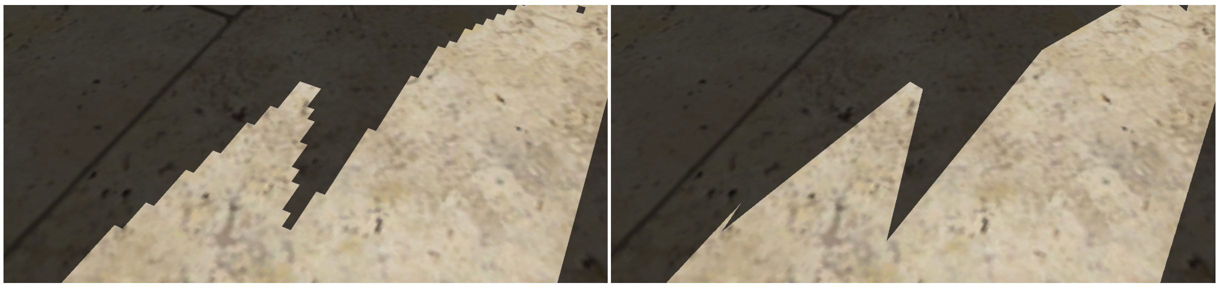

Figure 1: *Our new method outperforms standard Shadow Mapping by dynamically reducing shadow map resolution. We improve on quality as well as performance. Standard Shadow Mapping at 4096x4096 (left, 2.8 ms) and our method with resolution reduced to 2700x2300 (right, 2.4 ms).*

Abstract

We present an approach to efficiently reduce shadow map resolution while retaining high quality hard shadows. In the first step, we generate a list of sample points that are seen from the camera and then project these into light space, much like Alias-free Shadow Maps. In the next step, we analyze the list of sample points on the GPU to construct a tight light frustum for shadow rendering. After the light frustum is computed, we calculate for each sample the actual coverage in the final shadow map to estimate how large a shadow map pixel should be. From this number, we derive the lowest possible resolution to use in the shadow map while retaining nearly alias-free shadows. Our algorithm is built for a deferred renderer.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Bitmap and framebuffer operations I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Shadows are crucial for human observers to better understand scene geometry. However, the accurate computation of shadows is hard to achieve in real-time applications. To produce pixel-perfect hard shadows, Shadow Volumes [Cro77] can be used. But it requires a lot of additional geometry and the rasterization of it. The generated shadow volumes need to be updated every time an object or light moves, making it a very CPU-intensive method.

Shadow mapping [Wil78], on the other hand, is an efficient real-time technique for generating shadows that scales better with dynamic and complex scenes. It is fully GPU-accelerated while leveraging the rasterization hardware. Unfortunately, it suffers from shadow aliasing due to the fact that we use images of a limited resolution to produce shadows. We propose an approach that attempts to minimize the aliasing error and estimate the minimum resolution to be used when rendering the shadow map. An example is shown in Figure 1.

Shadow mapping renders the scene as seen from the light source and stores the depth buffer from this render pass as

[†] michalferko1@gmail.com

DOI: 10.2312/LocalChapterEvents.TPCG.TPCG13.109-114

the shadow map. This buffer is then used when rendering the scene from the camera's point of view. For every point visible from the camera, we project this point into the light's clip space (in which we stored the shadow map). This gives us (x, y) coordinates in the shadow map and z which determines how far the current point is from the light. By reading the stored depth d at the (x, y) coordinates in the shadow map and simply testing if $z < d$, we can determine which points are lit and which are not.

2. Related Work

The standard Shadow Mapping suffers from projective, perspective and depth precision aliasing [SD02]. There are several modifications that try to avoid one or several types of aliasing at once.

There are approaches that achieve completely alias-free shadows with the help of shadow maps, but these approaches add a large overhead to achieve the alias-free results. The main problem is that we need to sample in a shadow map at arbitrary points, not in an aligned grid patterns as with standard rasterization. This is due to the fact that points seen by the camera are not regularly distributed from the light's point of view.

The Alias-Free Shadow Maps method [AL04] actually does not use rasterization into a shadow map, but instead uses triangles in a kd-tree to compute the accurate shadowing term where it needs to be computed. The authors provide a CPU implementation, which can hardly be real-time for complex scenes consisting of thousands of triangles. Another method [JLBM05] uses the Irregular Z-buffer rasterization to allow storing of depths in arbitrary points in the light space image plane. When these positions are matched with points seen by the camera, we achieve alias-free shadows.

Adaptive Shadow Maps [FFBG01] perform progressive refinement of parts of the shadow map that do not have sufficient resolution. The method is not guaranteed to converge but provides high quality shadows. Resolution-Matched Shadow Maps [LSO07] improve Adaptive Shadow Maps and do not use an iterative refinement process. Depth-precision aliasing is addressed using the standard bias techniques.

A more extensive overview of shadow aliasing and alias-free methods can be found in [ESAW11].

Restricting shadow computation only to portions of the scene that are visible was suggested in [BAS00], but the authors perform it only on a lower resolution camera image. We access the G-Buffer depth, which is used in other parts of the rendering pipeline as well, and process the data on the GPU.

3. Our approach

Our approach is similar to Resolution-Matched Shadow Maps [LSO07]. However, we avoid the complex generation of shadow pages to adaptively change resolution in different portions of the shadow map. Even though their algorithm produces almost alias-free shadows, the additional overhead makes it not viable for multiple shadow casting lights. We aim at very simple and computationally inexpensive extensions that help save performance and estimate resolution while not adding a larger overhead.

Only adaptive sampling (either through adaptive resolution or some other method) solves projective aliasing well, since rendering shadow maps at very high resolutions is impractical and mostly not necessary.

The steps of our method can be easily summarized as an extended deferred shading pipeline:

1. Render the G-Buffer from the camera.
2. Analyse the G-Buffer depth with respect to the light.
3. Perform a series of downscaling steps to get tight light frustum and resolution estimate.
4. Generate shadow map by rendering the scene from the light, use recommended resolution and tight light frustum.
5. Perform the G-Buffer post-process to calculate shading, access the shadow map to evaluate shadowing term.

3.1. Tight frustum computation

We take all the points seen from the camera and project them into the light's clip space. During this step, we presume that the light casting shadows is a spot light and that it is defined by a perspective projection. Therefore, points outside the viewing frustum corresponding to this perspective projection are considered to be outside of the light volume and thus do not need to be taken into account when evaluating shadows.

For the list of all visible points, we take their light space coordinates that would correspond to the current perspective projection used for looking from the light. Points outside the light viewing frustum are not taken into account.

From all of these coordinates, we calculate minima and maxima for x and y coordinates. To get these, we perform a series of downscaling passes, much like we would construct mip-maps from a high resolution image. Instead of storing the averaged value in the next level, we store the minima and maxima of the small neighbourhood (4x4 has proven to outperform other neighbourhood sizes). For this, we use two 32-bit floating-point RGBA textures which are exchanged after every pass. We render into the first texture, then we use its data to calculate new values for the second texture, etc. The process ends after a few iterations depending on image resolution (6 iterations for resolution 1920x1080) with the

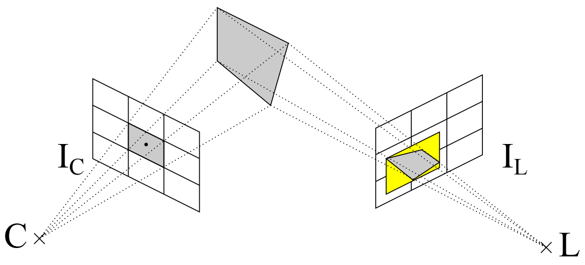

Figure 2: *The process of resolution estimation. From the camera C, we project the pixel into a polygon in world space. This is then projected into the light's clip space I_L for light L and it's bounding rectangle (yellow) is computed. The width and height of this rectangle are used to estimate new resolution.*

result being just one texel. This texel contains a 2D axis-aligned bounding box (AABB) encapsulating all points seen from the camera in light space. We use a simple crop matrix to render only into the portion of the light space by mapping $[x_{min}, x_{max}] \times [y_{min}, y_{max}] \to [-1,1] \times [-1,1]$ [BAS00].

3.2. Resolution estimation

Every pixel we see from the camera represents a small patch in the scene. Since we have limited resolution, one pixel can actually represent thousands of real scene patches, while each of these patches covers a very small portion of the pixel. Therefore, computing precisely how one pixel corresponds to several small surface patches in the scene is not efficient to compute. We make an assumption that one pixel corresponds only to one patch.

To compute the actual resolution needed for alias-free hard shadows, we need to determine the smallest required size for a shadow map pixel. We take every pixel seen from the camera, retrieve the world-space patch represented by this pixel, and then project the patch into light space. Afterwards, we compute a 2D AABB for the patch in the light's clip space, which gives the estimated area that would require one shadow map pixel.

To reconstruct the world-space patch from a pixel, we use the G-Buffer depth. Instead of sampling the depth at the pixel center, we sample points in pixel corners while using bilinear interpolation. Therefore, the orientation and size of the patch is affected by neighbouring depth values. We then project the four pixel corners into the light's clip space. As a result, we get four sample positions, from which we can easily compute the bounding rectangle and it's width and height. This whole process is shown in Figure 2.

Finally, the smallest possible light space coverage (w_{pixel}, h_{pixel}) for each of our patches needs to be determined. The shadow map resolution is then derived from this value. We perform the same downscaling as we did with the tight frustum computation, always taking 4*x*4 neighbourhoods and taking the smallest *x* and *y* extents of the bounding rectangles for all of these patches.

We use both the tight frustum and the minimum bounding rectangle to compute the actual resolution. We use the bounds $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$ to determine the width and height of the selected portion of light space, and then calculate the actual resolution $(width_{sm}, height_{sm})$ as:

$$\begin{aligned}
&-1 \le x_{min} \le x_{max} \le 1 \\
&-1 \le y_{min} \le y_{max} \le 1 \\
&width = x_{max} - x_{min} \in [0,2] \\
&height = y_{max} - y_{min} \in [0,2] \\
&(width_{sm}, height_{sm}) = \left(\frac{width}{w_{pixel}}, \frac{height}{h_{pixel}} \right)
\end{aligned}$$

If the camera viewing frustum and the light viewing frustum are identical, this returns the requested shadow map resolution to be the same as the resolution of the image generated by the camera (only if identical settings are used for the projection matrix). We do not let the shadow map exceed certain resolution. We use this limit as a user-defined parameter that allows to limit memory consumption as well as reduce computational overhead of shadow map rendering when shadows are not as important.

4. Results

The main goal of this work is to provide extremely fast tests on how to reduce shadow map resolution and effective reduction of the light viewing frustum. We tested our approach on a Windows 8 machine with an Intel Core i7-3770K processor, a GeForce GTX 680 graphics card and 16GB memory. Our application was written in C++, OpenGL and GLSL and is compatible with OpenGL3+ hardware.

We tested mainly the performance of our modification when compared to standard Shadow Mapping. The quality is guaranteed to be better thanks to the nature of our extensions. The test consisted of a pre-defined camera fly-through in the famous Crytek Sponza model. At the moment, the fast scene analysis supports only one shadow casting spot or directional light, but it is easily extensible for point lights and the capability of doing tests for multiple lights at once.

For the test, we recorded performance for standard Shadow Mapping with a 4096x4096 shadow map. With our approach, we recorded the estimated resolution (clamped to 4096x 4096), as well as the percentage of the screen our tight light frustum covers compared to the unmodified light frustum. The graph showing varying resolution is shown in Figure 3. Our tests were part of a deferred renderer, the resolution of final images was 1680x1050.

We also recorded the computation time of our algorithm compared to the standard Shadow Mapping. There is the

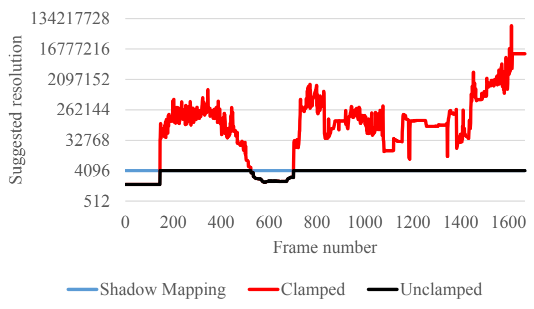

Figure 3: *The graph showing varying resolution for a short scene walkthrough. Maximum of width and height is shown.*

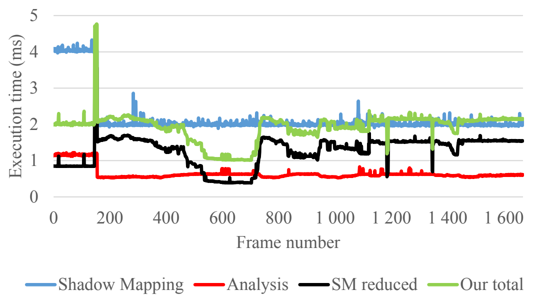

Figure 4: *Showing time of execution for our approach of up to 4096x4096 resolution (green) and standard 4096x4096 Shadow Mapping (blue) on the same animation as Figure 3.*

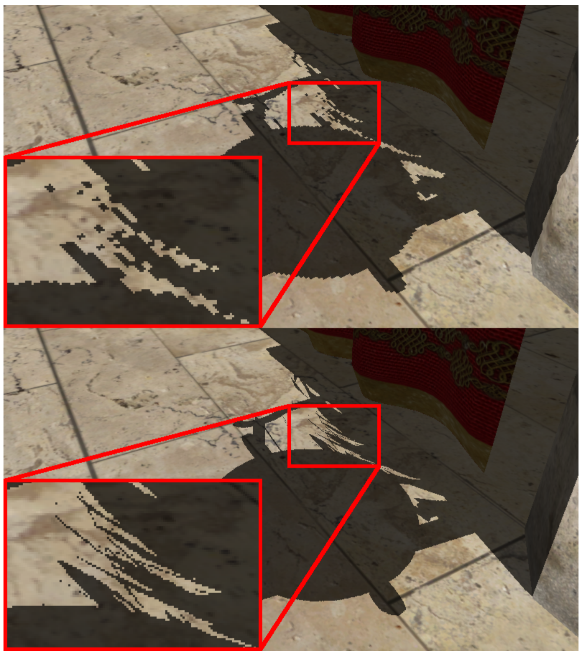

Figure 5: *Increased shadow map precision thanks to a smaller light frustum. Without frustum reduction (top) and with a tight camera-dependent frustum (bottom). The differences are most noticeable on thin geometry such as leaves.*

analysis step (which consists of two parts, but these parts are executed in one pass) and then finally the shadow map rendering step. This can be seen in Figure 4.

As we can see from the results, the analysis step takes under 1 ms when there are at least some points outside the light's viewing frustum that the algorithm can discard fast. This does not happen until frame 200 in our animation. When the shadow map resolution is reduced, we actually outperform standard Shadow Mapping thanks to faster rasterization. The performance of the scene analysis step is dependent on the resolution of the camera image. Also note that the recommended resolution also strongly depends on the final resolution.

4.1. Shadow quality

Our solution is guaranteed to have lower aliasing than classic Shadow Mapping. This is thanks to the reduction of the light's viewing volume, resulting in a denser sampling of the scene. Such an occurrence is shown in Figure 5.

The only possible source of additional aliasing for our method is the reduction of shadow map resolution. If the pixel grid determined by the lower resolution causes misalignment of shadow map pixels in the camera image, our approach will falsely evaluate those pixels. The same would occur if we used standard Shadow Mapping with a lower resolution shadow map. We determine the shadow map pixel size by its actual size in the output image, therefore these problems occur only in small neighbourhoods around the shadow boundaries determined by a full resolution shadow map. In Figure 6, you can see that the actual differences are negligible when we would achieve alias-free shadows with a higher resolution shadow map as well.

The improvements provided by our algorithm can change rapidly for a dynamic camera. We can see only a small portion of the scene and the next frame come across a large open space. The light frustum will change rapidly, resulting in sudden changes of aliased shadow edges. Such an occurrence is shown in Figure 7.

Sudden changes in shadow map resolution are not as noticeable, thanks to the small neighbourhood changes for the full resolution and suggested resolution.

If the suggested resolution exceeds the maximum dimensions we allow for a shadow map, and the original light viewing frustum is tightly enclosing all camera pixels, our approach will not reduce aliasing. This is due to the fact that both parts of our algorithm already determined that there is no space for improvement (either of the light's frustum or

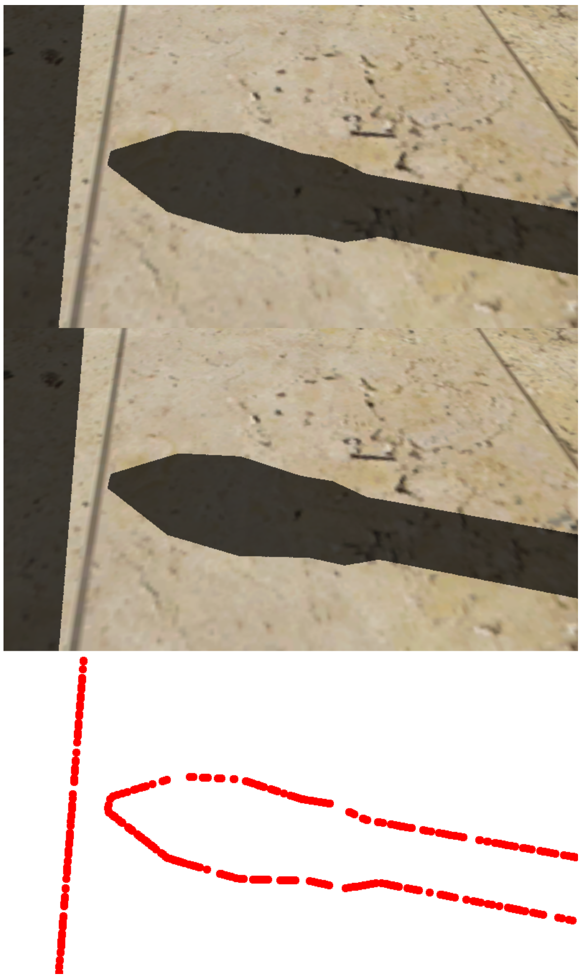

Figure 6: *Possible misalignment due to reduced resolution. 4096x4096 shadow map (top), reduced to 1920x1520 shadow map (middle) and marked pixel differences with red dots (bottom).*

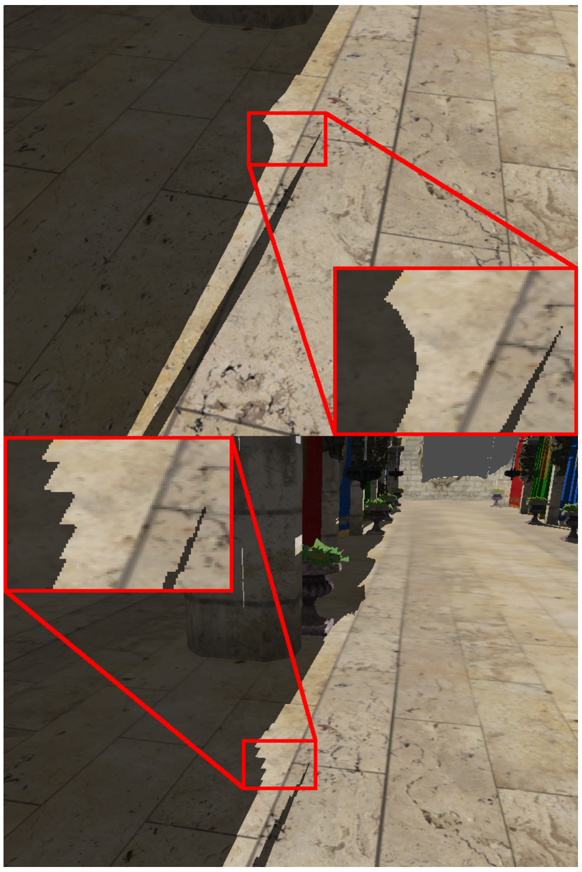

Figure 7: *Sudden change of shadow aliasing. Only a small portion of the scene is seen in the first frame (top) and when the camera moves only slightly, the frustum is suddenly much larger (bottom).*

the shadow map resolution). Such a situation is a very special case and in general will never occur.

5. Conclusions & Future work

We have presented a combined approach to generate hard shadows with aliasing kept low using simple and efficient modifications that are fast to compute on a GPU. Our approach is aimed at extending a deferred renderer. We use the G-buffer depth to minimize the light's view frustum as well as estimate the required resolution for the shadow map to retain alias-free shadows.

At the moment, our method reduces perspective aliasing greatly, but does not improve much on projection aliasing. We would need adaptive resolution for portions of the shadow map to address projection aliasing, since rendering very high resolution shadow maps (as suggested by the resolution estimate) is not practical.

We plan to extend our method to dynamically change resolution of portions of a shadow map. We intend to achieve this with a GPU-accelerated implementation through multiple viewports in OpenGL 4 and geometry shaders to duplicate geometry that should be shared between viewports. The important part will be detecting how to slice the light's viewing volume. A fast GPU cluster analysis with respect to light space pixel sizes should be performed on all samples seen from the camera. Several fixed-size parts of different resolution could be used to ensure stable performance, but this also means that we will not always reach alias-free shadows. The projection aliasing will, however, be strongly reduced.

Acknowledgements

We would like to thank Marko Dabrovic and Frank Meinl for the Crytek Sponza model. This research was supported in part by the SPINKLAR-3D project, VEGA 1/1106/11.

References

[AL04] AILA T., LAINE S.: Alias-free shadow maps. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques* (Aire-la-Ville, Switzerland, Switzerland, 2004), EGSR'04, Eurographics Association, pp. 161–166. 2

[BAS00] BRABEC S., ANNEN T., SEIDEL H. P.: Practical shadow mapping. *Journal of Graphics Tools 7* (2000), 9–18. 2, 3

[Cro77] CROW F. C.: Shadow algorithms for computer graphics. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1977), SIGGRAPH '77, ACM, pp. 242–248. 1

[ESAW11] EISEMANN E., SCHWARZ M., ASSARSSON U., WIMMER M.: *Real-Time Shadows*. A K Peters/CRC Press, 2011. 2

[FFBG01] FERNANDO R., FERNANDEZ S., BALA K., GREENBERG D. P.: Adaptive shadow maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 387–390. 2

[JLBM05] JOHNSON G. S., LEE J., BURNS C. A., MARK W. R.: The irregular z-buffer: Hardware acceleration for irregular data structures. *ACM Trans. Graph. 24*, 4 (Oct. 2005), 1462–1482. 2

[LSO07] LEFOHN A. E., SENGUPTA S., OWENS J. D.: Resolution-matched shadow maps. *ACM Trans. Graph. 26*, 4 (Oct. 2007). 2

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 557–562. 2

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1978), SIGGRAPH '78, ACM, pp. 270–274. 1