Pixel Accurate Shadows with Shadow Mapping

Christian Luksch*
MatNr. 0525392

Institute of Computer Graphics and Algorithms
Vienna University of Technology

Abstract

High quality shadows generated by shadow mapping is still an extensive problem in realtime rendering. This work summarizes some state-of-the-art techniques to achieve pixel accurate shadows and points out the various problems of generating artifact free shadows. Further a demo application has been implemented to compare the different techniques and experiment with alternative approaches.

Keywords: Pixel Accurate Shadows, Shadow Mapping, Deferred Shading

1 Introduction

The shadow mapping algorithm introduced by Williams [Williams 1978] is an efficient way to determine the shadow projected by a light in a scene. Thereby the light-view depth values are rendered to a texture which is used to classify the visibility of the scene-fragments relative to the light. In theory this algorithm has very little limitations and performs well on modern graphics hardware. On the other hand shadow mapping hugely suffers from aliasing artifacts, which all together make pixel accurate shadows for all kinds of scenes and camera positions very difficult.
Shadow mapping aliasing will occur when there is not enough information in the shadow map to do an accurate shadow test for a fragment. Because of the finite resolution of the shadow map, for a fragment in eye-space the corresponding depth value in light-space can only be approximated by sampling the nearest value or doing some sort of interpolation. The lack of accuracy causes a blocky appearance and unaesthetic incorrect shadowing results.

In the following sections we give a brief overview of typical shadow mapping problems and previous work. Then we give insight into our demo application and details of our implementation.
In Section 6 we present our research on confidence-based shadows and introduce a new technique to guarantee pixel accurate shadows. Then we review two techniques, Parallel Split Shadow Maps [Zhang et al. 2006] and Fitted Virtual Shadow Maps [Giegl and Wimmer 2007a]. We analyze their practicality for pixel accurate shadows, propose improvements and give details of our own implementations. Section 9 handles shadow map biasing, describes the ID-buffer concepts and compares results in our test scene.
Finally, we summarize the discussed techniques in a comparison, followed by the conclusion.

*e-mail: christian.luksch@aon.at

2 Shadow Mapping Problems

The whole shadow mapping process suffers from two types of aliasing artifacts, perspective and projection aliasing. Additionally there is a self shadowing problem and the fact that shadow maps can not be filtered like common textures.

Perspective aliasing: In a perspective view object near the camera are larger than distant objects. When the shadow map is rendered the scene is regularly sampled, which results in undersampling near the camera and oversampling in the distance.

Projection aliasing: This type of aliasing is independent of the camera, it only depends on the angle between the light direction and the surface normal. If the angles is almost perpendicular, the surface area has barley any shadow map resolution. It is difficult to counteract this and cannot be solved by a simple global method.

Incorrect Self-Shadowing: The shadow map can be seen as regular grid of depth samples taken from the scene, which are resampled during the shadow test. This lead to incorrect self-shadowing artifacts. Therefore some sort of distinction or biasing must be used.

Shadow Map Filtering: Filtering is very important to hide undersampling artifacts or to get anti-aliased shadow outlines in oversampled areas and increases the overall shadow quality. Common texture filtering can not be used, because interpolated depth values make no sense along object edges and will still generate sharp outlines. Special for shadow mapping developed filtering techniques has to be used.

Several techniques have been developed to improve the quality of the shadow mapping algorithm. The next section gives a brief overview and classifies these techniques.

3 Previous Work

Most of the shadow mapping techniques try to overcome the aliasing artifacts, which are the result of undersampling due to limited resolution. The ideal solution would be a depth sample in light space for each fragment in screen space. This approach has been followed by Timo Aila in ”Alias-Free Shadow maps”[Aila and Laine 2004]. Unfortunately, this requires irregular shadow map samples, which makes it hard to implement the algorithm efficiently.

Many pixel exact shadow mapping techniques use some sort of hierarchical tiling to achieve the required sampling resolution where it is needed. To this class of algorithms belongs Adaptive Shadow Maps [Fernando et al. 2001], Tiled Shadow Maps [Arvo 2004], Queried Virtual Shadow Maps [Giegl and Wimmer 2007b] and Fitted Virtual Shadow Maps [Giegl and Wimmer 2007a].

A widely used class of techniques are those that create a view-dependent reparameterization of the shadow map, so that there are more samples close to the view point. In this category

belongs Perspective Shadow Mapping (PSM) [Stamminger and Drettakis 2002], Trapezoid Shadow Mapping (TSM) [Martin and Tan 2004] and Light Space Perspective Shadow Mapping (LiSPSM) [Wimmer et al. 2004].
In comparison to standard shadow mapping the complexity of these techniques is almost the same, which makes them practical for real-time rendering. However, the quality depends on the view point and will change when the camera moves, which even can get as bad as standard shadow mapping in the so called *Duelling Frusta Case*, where the view direction is almost parallel to the light direction.

Another category of techniques are those that split the view frustum in smaller parts and create a shadow map for each of them. A possible partitioning scheme is to split by the face edges of the view frustum seen from the light, which allows to build a reparameterization for each face to optimize the sample positions. Another possibility is to slice the view frustum along the view axis (z-partitioning) [Zhang et al. 2006]. It can be combined with shadow map reparameterization as well.

In the paper Warping and Partitioning for Low Error Shadow Maps [Lloyd et al. 2006] the aliasing error of the latter two classes of techniques is extensively analyzed. It is shown that z-partitioning used with a warping technique like LiSPSM should be the best scheme to render shadows in scenes with a high depth range to reduce perspective aliasing. This will be elaborated in Section 7.

Scherzer et al. [Scherzer et al. 2007] presented a technique that reuses already rendered shadow information through temporal reprojection and use a confidence-based method to merge with the shadow rendered in the next frame. A single reparameterized shadow map is rendered each frame to achieve a high frame rate. With additional jittering exact shadows will be produced after a certain number of frames.

Beside these techniques shadow map filtering will also be required to render artifact free shadows. Percentage Closer Filtering (PCF) is a widley used technique. Another approach is Variance Shadow Mapping (VSM), which enables to use common hardware texture filters and makes large filter kernels more efficient. [Donnelly and Lauritzen 2006]

4 Implementation Overview

To compare and evaluate differences between various shadow mapping techniques, it is important to have a common test basis. We implemented our own application, which is a powerful tool to experiment with alternative approaches and it gives more control over the implemented techniques. Only a few other published shadow mapping demo applications exist and most of them only support a single technique.
Our application has been implemented in C++ and uses the DirectX 9 graphics API. The test system is a Core2Duo @ 3Ghz with 4GB Ram and a Nvidia GeForce 8800 GTS with 640MB video ram.

We used two test scenes, one random generated terrain with numerous static and dynamic objects, in which the large size and perspective aliasing plays a major role. Second a scene with the power plant mesh which has lot of fine structures and much more projection aliasing. Screenshots can be found in Figure 1.
Further a set of camera positions has been composed that show various cases of shadowing scenarios. Thereby comparable analyses can be done at different times and also allows accurate benchmarking.

5 Deferred Shadowing

Deferred Shadowing is a technique base on deferred shading to apply multiple shadow maps with preferably less overhead. Deferred shading first renders all for shading needed surface properties like surface normal, material color and specular exponent to a full screen render target. For shadowing the fragment world position is also stored in some way. The actual shading is done in a final full screen pass, whereby just only visible fragments get shaded, unlike common forward rendering. Furthermore, deferred shading enables an efficient way to render numerous lights, since the shading is shifted to screen space.
Because an arbitrary number of shadow map should be used, our implementation uses an accumulative render target to store the final overall shadow test result, which is used instead a single shadow map in the final pass when the fragments get shaded.
Such a technique requires hardware support of multiple render targets, floating point render targets and full 32-bit floating point accuracy.

5.1 Fragment World Position

As mentioned above the world position is required for shadow testing. The straightforward solution would be to directly store the World Position in an 128-bit 4 channel floating point render target. This will assure that no precision is lost, but it will also consume much memory bandwidth and unless the graphics hardware supports multiple render targets with different bit depths, storing color and surface normal is more complicated, because everything has to be packed in another 128-bit render target.
On second thought the World Position can also be recovered with the fragment screen position and its depth, so that a single 32-bit render target for the fragment depth is sufficient. Therefore either the linear eye depth or the projected depth can be used.

To recover the world position from linear eye depth either the view vector needs to be interpolated over the full screen quad or it also can be reconstructed through the fragment screen position (texture coordinate) and using the inverse view matrix to get the world position. The following equation shows the implementation using the fragment screen position s_{xy} to calculate the partial view vector v_{xy} and use it to recover the world position $wpos$:

$$v_{xy} = \left(\begin{pmatrix} 2s_x \\ -2s_y \end{pmatrix} + \begin{pmatrix} -1 - \frac{1}{rtWidth} \\ 1 + \frac{1}{rtHeight} \end{pmatrix} \right) \cdot matInvProj \quad (1)$$

$$wpos = \begin{pmatrix} depth^l \cdot \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix} \\ 1 \end{pmatrix} \cdot matInvView \quad (2)$$

It is very important to use the exact fragment center, which is achieved through the offset $\frac{1}{rtHeightWidth}$ under DirectX 9, but generally it depends on the rasterization rules of the graphics API.
To recover the world position from the projected depth a similar equation can be used.

In comparison the world position recovered from the linear depth is nearly the same as the original world position. The projected depth varies slightly in the result which probably comes from loss of accuracy, but on average it is almost the same and can be used

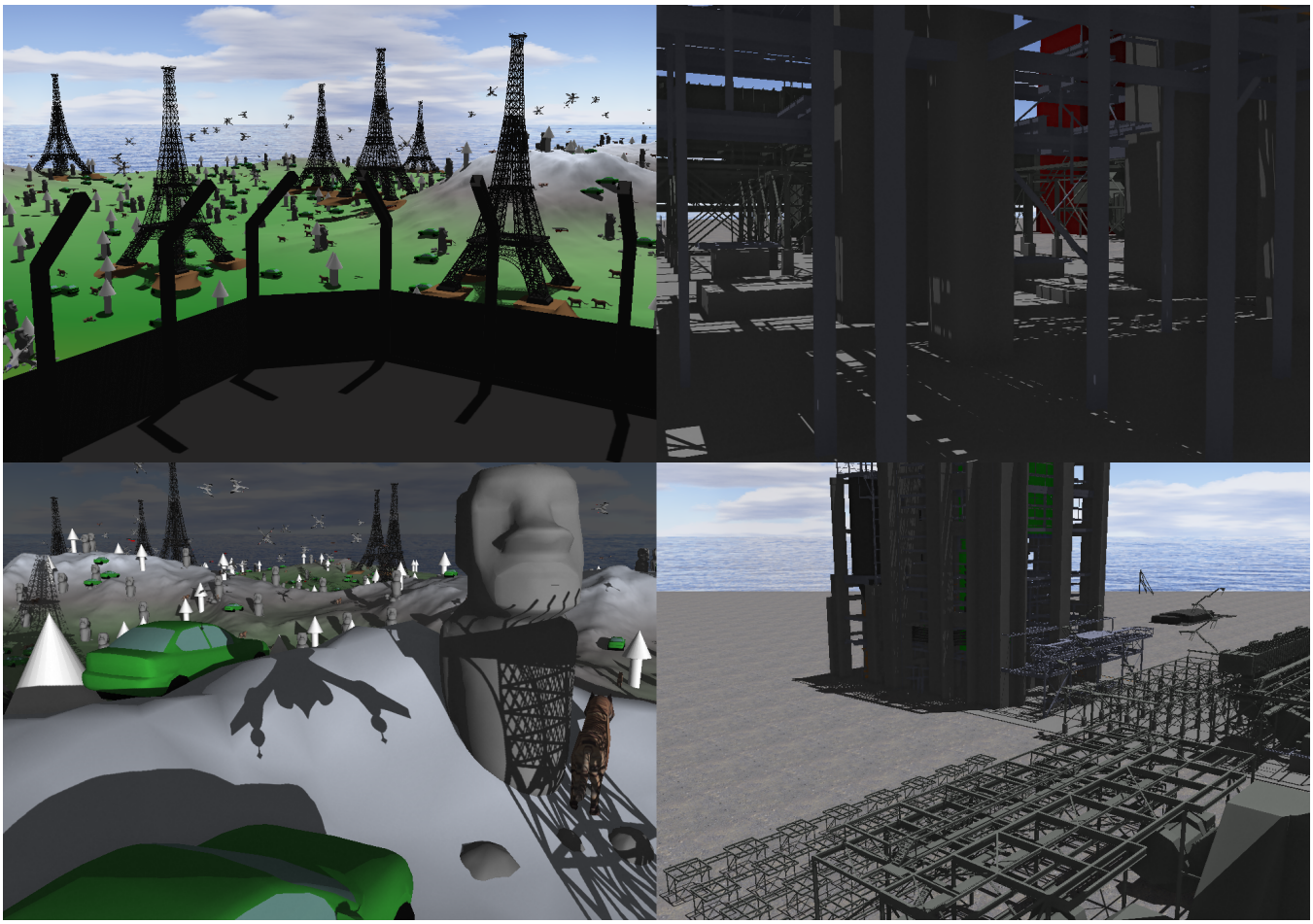

Figure 1: Screenshots of the demo application. Terrain scene (left) and power plant (right).

too without any concerns. The projected depth could also be read directly from the z-Buffer, which is possible with DirectX 10 on every graphics hardware. Under this circumstance this type of implementation should be more efficient and be preferred over separate rendered linear depth.

5.2 Implementation Details

In our implementation the following scheme of render targets is used:

- Material Properties: 8-bit per channel ARGB (32-bit)
- Compressed Surface Normal: 2 channel 16-bit floating point (32-bit)
- Depth / World Position: either 32-bit or 128-bit four channel floating point
- (32-bit ID-Buffer)

The surface diffuse color is stored in the first render target. The alpha channel could be used for the specular exponent.

Compressing the surface normal is suggested in many other deferred shading implementations. Thereby only N_x and N_y is stored and the z-component can be recovered through:

$$N_z = \sqrt{1 - {N_x}^2 - {N_y}^2} \tag{3}$$

Assuming that in view space all surface normals have a positive z-component, this equation is always true, but in practice this can not be assured. Therefore a bit to store the sign of N_z is borrowed from the material properties, so that the normal can be recovered correctly.

The purpose of the ID-Buffer will be discussed in section 9.1.

A good discussion on deferred shading can be found in Nvidia's GPU Gems 2 by a developer of the game S.T.A.L.K.E.R. [Shishkovtsov 2005].

6 Confidence-Based Shadows

To generate a high accuracy shadow a very high resolution shadow map would be required in a dimension that is far beyond graphics hardware capabilities. A series of slightly jittered shadow maps can be used to simulate higher shadow map resolutions. To combine the single shadow maps to a final result a confidence value is used to preserve the best samples [Scherzer et al. 2007].
When a fragment of the scene is shaded, it is transformed into light-space and the nearest depth value is read from the shadow

map. Unfortunately, the exact depth value is only known at the center of a texel, which usually will not be hit. Therefore, we also store a confidence value of this shadow test in the accumulation shadow buffer, based on the distance to the nearest texel center, where the depth information has been taken from. This can easily be determined through the texture coordinate tc and the resolution sm:

$$conf = 1 - \max\left(|\{tc_x \cdot sm_{xl}\} - 0.5|, |\{tc_y \cdot sm_{yl}\} - 0.5|\right) \cdot 2 \quad (4)$$

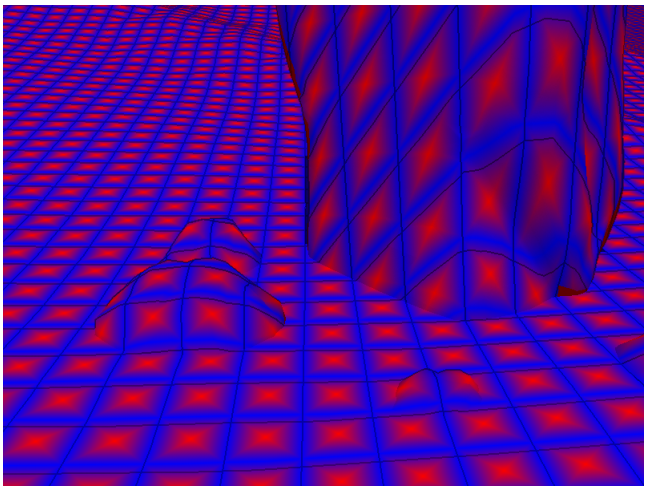

Figure 2: Illustrates the confidence of the sampling position of a projected shadow map. The confidence is high (red) at the center of the shadow map texels, because that is where the depth values has been rendered. The farther apart the lower the confidence.

Figure 2 visualizes this confidence value. Such a confidence value is bound to the light space and the shadow map resolution. This will be relevant in the discussion in section 6.5.

6.1 Jittering

The next step is to generate a series of shadow maps with different rasterization. This is achieved by rendering each shadow map with an translation offset along the light view plane in sub-pixel scale. To be able to reproduce the same result the series of random numbers has to be the same. Therefore, the Halton sequence numbers are very convenient, because they guarantee a nearly uniform distribution and appear to be random at the same time. A illustration of the generated sample positions is shown in Figure 3. It shows that doubling the sample points results in an evenly refinement, which is important when a suitable sample number hat to be chosen.

A different rasterization can also be achieved by rotating the light view, thereby the rotation angle should be taken from a Halton sequence. This method can be combined with translational jittering as well.

6.2 Accumulation

With every new pass the different rasterizations contribute new shadow information. There are several ways to achieve this, thereby

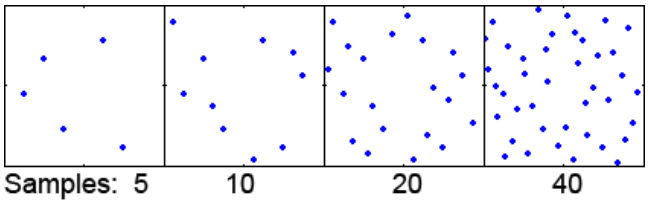

Figure 3: Sample positions of the Halton sequence with 5, 10, 20 and 40 sample points.

it has to be discovered when to stop rendering new passes and if the result is actually the exact one.

The first implemented method draws only shadow fragments with a certain confidence, we will refer to this method as *Simple Confidence*. This will draw dots at the center of the shadow map texel. After enough passes to cover the whole texel have been rendered, a continuous shadow in a higher quality is generated. Thereby a certain confidence value always requires a certain number of passes to cover the sample area, whereby the quality is also increased by a certain factor. At this point a characteristics of the Halton sequence can be seen. The even distribution of the sample points approximately refines by doubling the number of samples, whereby the simulated shadow map resolution is doubled. The sequence of useful number of sample points is shown in Figure 3. A number in between is not that optimal, because it does not evenly cover the whole texel. Also a minimum of five samples should be used which is the first number where the simulated shadow map resolution is approximately doubled.
Figure 4 shows the first five samples with a very high confidence value and a low shadow map resolution to visualize the process. For this technique the confidence value and number of passes has to be configured manually and it has to be ensured that the whole texel is covered.
A problem is that the shadow outline also increases in relation to the size of the projected shadow map texels and the configured confidence value.

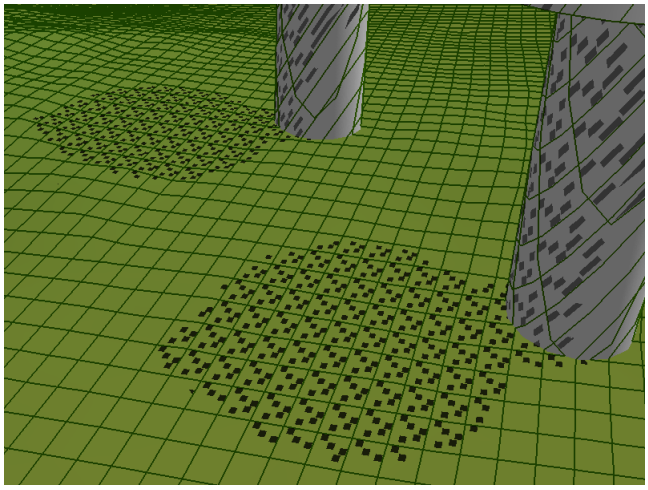

Figure 4: Translational jittering using the first five offsets given by the Halton sequence. The points represent the jittered texel centers and the grid the unjittered shadow map.

Based on the insight gained from the first method, an advanced method to accumulate confidence-based shadows has been developed, which will be referred as *Adapted Confidence*.

To no longer depend on configuring the confidence value and a fixed number of passes, new fragments are only rendered if their confidence value is higher than the current one regardless of its shadow. This principle can easily be implemented using an additional hardware depth buffer which holds the current confidence. This method also obtains that the shadow is always continuous and not composed by dots, which allows to stop at any pass without leaving falsely unshadowed holes.

However, an automatic stopping criteria can not be used, because each pass only simulates a higher shadow map resolution, which only increases the confidence value of the shadow tests, but there is no correlation to the scene properties. Unless we know how much confidence is required, such a method is not possible.

6.3 Optimal Confidence

An efficient pixel exact shadowing method should adapt on the unique scene properties of each setting. The required shadow map resolution has to be evaluated, which has to be done on fragment basis.
So far our method increases the accuracy of the shadow test globally by simulating higher shadow map resolutions with each additional pass through jittering. Actually each fragment requires a certain minimum shadow map resolution to be exactly shadowed, which is equivalent to a certain confidence value. To adapt the quality locally it is required to know which confidence is needed per fragment. This value depends on how the shadow map is projected onto a fragment. To approximate this projection we use the neighbouring fragments and project all to light-space and calculate their shadow map texture coordinates. The spanning area gives information of the required confidence. This process is illustrated in Figure 5.

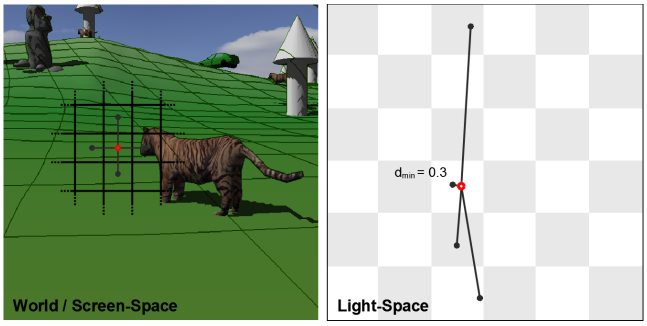

Figure 5: Illustrates how the optimal confidence is calculated. For each fragment the neighboring fragments are projected to light-space and the minimum texture coordinate distance is used to determine the required confidence. It also shows a case of discontinuity.

We calculate the distances to the neighboring fragments and use the minimum distance $tcDist_{min}$ to determine the required confidence $optConf$ in the following way:

$$optConf = 1 - tcDist_{min} \cdot smSize \tag{5}$$

In the case of discontinuities this still gives a stable result.

The optimal confidence is calculated in a separate pass after the scene is rendered and stored in full-screen texture.
This factor is now used to bias the confidence written to the depth buffer $finalConf$, in such way that the value will be 1 once the required confidence is reached and will not get overwritten anymore.

$$finalConf = \text{saturate}(conf + (1 - optConf)) \tag{6}$$

With this method it is possible to use occlusion queries to determine when to stop rendering new passes. The number of rendered fragments gives an approximation of how far the shadow is converged. We use a simple heuristic with two thresholds to configure the stopping criteria. The process is aborted when less than ε fragments are rendered for N passes. Such a rule is reasonable, because even when zero fragments have been rendered in the last pass, it is not guaranteed that all fragments reached their required confidence and a new rasterization still might contribute new fragments.

Figure 6 visualizes the required confidence calculated with this method. The high perspective aliasing in this setting shows that uniform shadow mapping requires a shadow map resolution about 50 times of the current one near the camera, on the other hand the distribution with LiSPSM is well balanced.

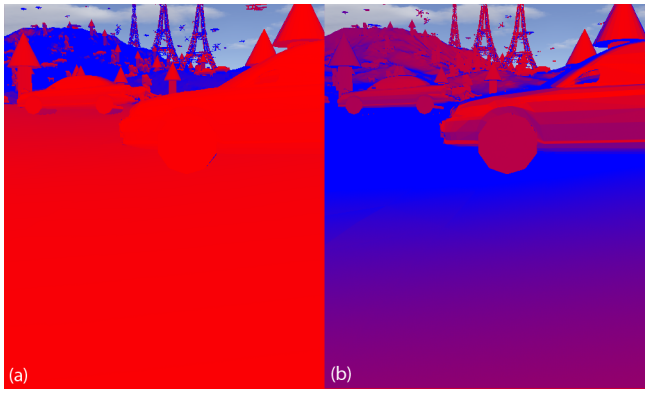

Figure 6: Minimum required confidence (red: high, blue: low) needed to generate pixel accurate shadows. (a) uniform shadow mapping, (b) LiSPSM.

6.4 Early Results

The number of passes or achievable shadow quality with the adapted confidence method depends on the initial shadow map sample distribution. On settings where LiSPSM can already eliminate most of the perspective aliasing, in a few passes projection aliasing artifacts can be eliminated very well. However, in the duelling frustum case where only uniform shadow mapping can be used, the required confidence can get to high in some areas to get a pixel accurate result in a practical number of passes.
Figure 7 shows such a duelling frustum case, but there are no objects very close to the camera so that the undersampling is moderate. It compares uniform shadow mapping with confidence-based shadow rendered in 20 passes and using a 2048^2 shadow map. (b) has been rendered with the simple confidence method using a confidence threshold of 0.5, thereby the increased shadow outline, which is about half a shadow map texel, can be noticed. (c) uses the adapted method and blending with the optimal confidence. The outlines look very fringed, because the required confidence is still not reached. This effect is only in dimension

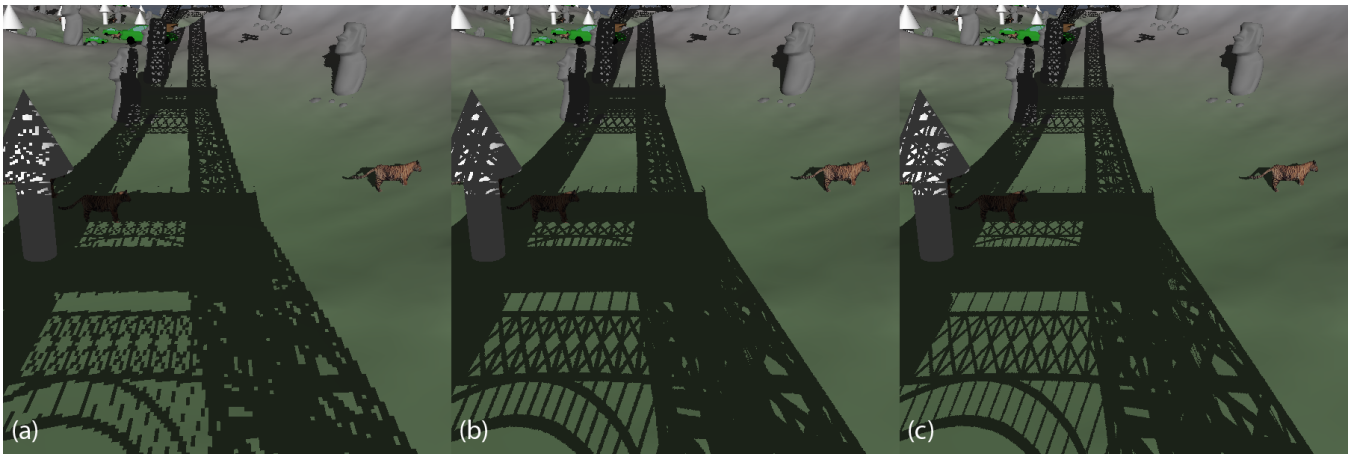

Figure 7: Comparison of uniform shadow mapping with a 20-pass confidence based shadow. (a) uniform shadow mapping (b) simple confidence based shadow: drawing dots with conf > 0.5 (c) adapted confidence based shadow accumulated with optimal confidence.

of one pixel, nevertheless, blending or blurring should hide it very well.

A problem is that shadow map filtering to generate smooth shadow outlines does not work with confidence-based shadow accumulation. Pixel exact anti-aliased shadow outlines would require a much more expensive accumulation.

6.5 Further optimizations

Looking at Figure 6 brings ideas for further optimizations.

6.5.1 LiSPSM n-Parameter Jittering

LiSPSM uses a parameter that controls the balance of the quality between front and back. The required confidence in Figure 6 (b) indicates a very well balanced sample distribution. It is the result of the automatic calculation of the LiSPSM n-parameter according to the paper. A detailed explanation how the n-parameter works can be found in [Wimmer et al. 2004].
The idea is to vary this n-parameter to focus more samples in the front or the back. This will faster converge to an exact shadow result.
The implementation showed that it definitely brings some improvement and reduces the number of required passes, but to find the optimal amount of jittering is not easy and does not yield the same result in all cases. Furthermore, if confidence-based accumulation should be used, a different reparameterization would mean another required confidence and therefore two confidence values from different passes have no relation and actually can not be compared. When the confidence values are biased by the optimal confidence the confidence-based accumulation is correct again, but it means that the optimal confidence has to be recalculated whenever n is changed.

Figure 8 compares translational jittering (a) with our implementation of n jittering (b) after 10 passes confidence-based shadow accumulation. A 1024^2 shadow map has been used, whereby markedly biasing artifacts occur near the far plane, but increasing the bias to an artifact free amount would cause hugely misplaced shadows. The biasing problem is also discussed in section 9.

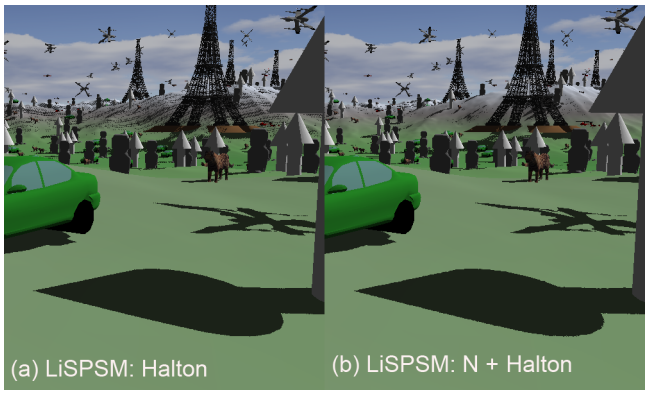

Figure 8: Comparison of translational (a) with n (b) jittering after 10 passes with a 1024^2 shadow map.

Translational jittering produces perfect shadows near the camera, but it is not capable of completely removing the biasing artifacts in the distance after 10 passes. With additional n jittering the markedly biasing artifacts nearly vanish, while the shadow quality near the camera marginally loses some accuracy, however, this can be tuned by the way the n-parameter is jittered. Overall, this is an additional jittering method that should be used if such biasing artifacts are problematic.

6.5.2 Frustum Reduction

Another approach is to always use uniform shadow mapping, thereby the shadow near the far plane converges first and no shadow further refinement is required in the backmost area, which means that the shadow frustum can be reduces. In our implementation a factor determines in which steps the frustum will be reduced. The factor can be determined with a similar method described in section 7.2 using equation 15, but the difference is that whole frustum is shadowed and therefore does not apply the exact same way and undersampling along the view direction rather occurs. A slightly higher factor may be needed, which may results in a few more required passes than with frustum splitting.

Since confidence-based shadow accumulation should be used, the

calculated required optimal confidence has to be adapted to the new projection of the reduced frustum. Due to a frustum reduction results in an uniform increment of the sampling rate in the focused area, the original calculated optimal confidence can simply be scaled by:

$$finalConf = \text{saturate}(conf + (1 - optConf) \cdot f_{reduced}/f), \quad (7)$$

where f is the view frustum far plane distance and $f_{reduced}$ the current reduced far plane distance.

The exact method is to test the backmost area that will be reduced whether the shadow is converged. This can be determined by using hardware occlusion queries in a pass that draws pixel only of fragments in this depth area that have not reached the optimal confidence. Now jittering is continued until the occlusion query returns a value below a certain threshold. It turns out that the shadow sometimes does not converge fast enough in the back, because of projection aliasing. A suitable occlusion query threshold is hard to find, because it depend on the scene. Therefore , it takes many passes until the front shadow quality is usable. Hardware occlusion query also cost quite much performance.
An other method is to reduce the frustum every pass and stop after a certain number. It does not guarantee an exact shadow result, but it comes quite fast to a useful result without remarkable artifacts.

Figure 9 shows the shadow result after 5 (a) and 10 (b) passes using a 2048 shadow map and a reduction factor of 0.55. Less passes or a lower factor will leave pixels that have wrong shadow results (c), however, by not using the confidence-based accumulation and by simply overdrawing the shadow result with the new one, these artifacts would be resolved. Nevertheless, most of the shadow maps would be wasted and frustum splitting with focusing on the depth tile would make more sense.
In comparison to frustum splitting the quality is similar, but thereby already 5 slices were sufficient to generate an acceptable shadow result (d), furthermore, shadow map filtering can be used.

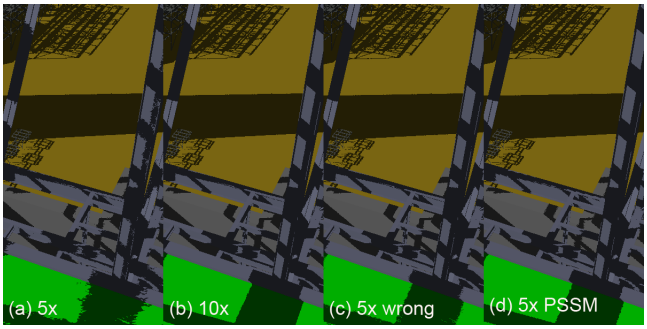

Figure 9: Frustum reduction by factor 0.55 after 5 (a) and 10 (b) passes. (c) reduction factor of 0.45 and 5 passes with wrong shadow results. (d) comparison to frustum splitting in 5 slices with optimal tiling and $\alpha = 0.8$.

Overall, the result is quite similar to frustum splitting or z-partitioning and sometimes even superior, because of additional jittering. The total rendering costs of our frustum reduction approach are mostly higher due to parts of the shadow frustum are rendered several times. However, in a case where the light direction is almost parallel to the view direction, z-partitioning also has to render lots of shadow casters several times and the shadow frusta of the slices will overlap. Considered differently the frustum reduction always proceeds like in such a case and additional uses confidence-based accumulation.

7 Frustum Splitting

This technique is based on the observation that objects in different depth layers from the camera require a different shadow map resolution, which is achieved by splitting the view frustum in depth slices (z-partitioning) and a shadow map is rendered for each. It is also known as Parallel Split Shadow Mapping (PSSM) introduced in [Zhang et al. 2006] or Cascaded Shadow Mapping, which has been implemented, but the details remains unpublished.

PSSM has originally been designed to optimize the distribution of shadow map samples for a fixed number of slices over the whole frustum only considering perspective aliasing. From analysing the aliasing error an optimal logarithmic split scheme has been evaluated. In practice this distribution is not very well when only a few slices are used, therefore, an adjustable mixture of logarithmic and uniform splitting to find the split position is:

$$C_i^{log} = n(n/f)^{i/n} \quad (8)$$

$$C_i^{uniform} = n + (n - f)(i/m) \quad (9)$$

$$C_i = \frac{C_i^{log} + C_i^{uniform}}{2} + \delta_{bias} \quad (10)$$

C_i is the distance of the i-th split plane, n and f are the frustum near and far plane distance and m the total number of slices. With δ_{bias} the split position an be adjusted for according to practical requirements of application.

For rendering m shadow maps are allocated and rendered successively, with DirectX 10 this even can be done in a single pass. Then for shadowing the depth slice of a fragment is calculated and the proper shadow map and light-space transformation is selected in the shader. They showed that already 4 slices with 512^2 shadow maps can generate a better result than a single 1024^2 shadow map and is similar than the result with a reparameterization like LiSPSM which, however, can not be used in all cases.

7.1 Improvements

A drawback that becomes noticeable, especially when shadow map filtering is used, is that the transition of the different slices can be visible, because of an abrupt change of the sampling rate, however, if undersampling is avoided it does not matter. Figure 10 illustrates such a transition. Without filtering the transition is not directly visible, but filtering hugely emphasizes it.

In many cases a combination with LiSPSM can further optimize the sample distribution and reduces the average error, which also produces smoother transitions between the slices. Lloyd et al. also suggest this combination. [Lloyd et al. 2006] They further analysed a combination with face partitioning, where the view frustum is splitted by its face edges, whereby even in a duelling frustum case a reparameterization can be used. However, the number of required shadow maps increase drastically and a similar result can be generated with only a few more depth slices too, as long as the shadow map resolution is sufficient.
Because our implementation already has a confidence based technique we also can use this to accumulate the shadow maps of the depth slices together. Thereby, the bounding box of the light frustum for each depth slice is used to clip the shadow, instead of testing the fragments only if it is in the depth slice. This always uses the entire shadow map and the highest confident samples are selected which moves the borders slightly backwards.

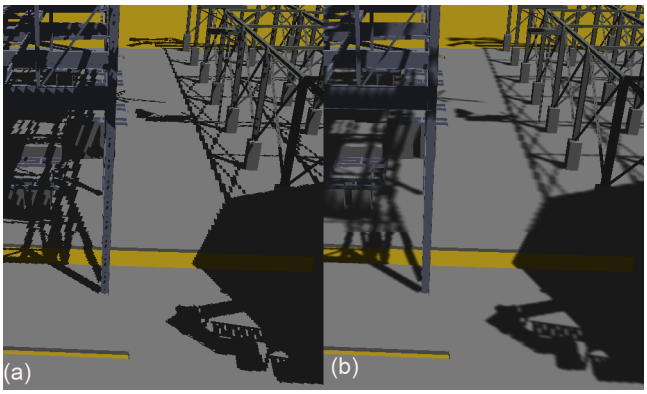

Figure 10: Illustration of transitions between the depth slices: Power plant scene shadowed with 3x512 PSSM (a) without filtering (b) 3x3 Gauss PCF.

The combination of the uniform and logarithmic split scheme, like equation 10, results indeed in an optimal average distribution of samples, however, close to the camera the error can be noticed clearly. With LiSPSM this can often be eliminated, but because we yield a high quality shadow with at least one sample per fragment in all cases, this is not satisfying. To settle more samples close to the camera we use an adjustable linear interpolation of the logarithmic and uniform split scheme:

$$C_i = \alpha C_i^{log} + (1-\alpha) C_i^{uniform} \tag{11}$$

This allows an easy way to balance the quality. A combination with $0.75<\alpha<0.85$ results in a desired behaviour. Figure 11 shows a case with perspective aliasing and shadows projected towards the camera and therefore a bad shadow quality near the camera (a). A higher α-value increases, the quality near the camera, while in this setting no quality change in the back can be seen (c). This case also allows a well reparameterization, whereby the quality can be increased similar than with a higher α-value (b), the different with LiSPSM and an α-value of 0.8 is not that big.
In the terrain scene a slight quality loss in the back is noticeable, but in cases where LiSPSM does not work are still good shadows near the camera.

Further the resolution of the shadow map needs to be increased at least to the size of the view port, when a sampling rate of at least one per fragment should be reached, otherwise the sample resolution would not even be able to cover this horizontal. For present common resolutions a 2048^2 shadow map is required. When the implementation is not that much limited by shadow map rendering the performance loss will be tolerable. We noticed a 30 percent frame rate drop between 2048^2 and 1024^2.

With the new splitting scheme nearly every case of our terrain scene is covered with enough shadow map samples and shadow map filtering can be used without any concerns. However, in the power plant scene cases where either the seams between the slices are visible or where there are not enough samples near the camera, can be found, because there are much more fine shadows projected over large distances and there is generally more projection aliasing. The only solution is to use more slices. The costs for rendering more passes can be very different, but it is mostly significant less than linear, when the shadow casters of every slice are culled properly. Because our system renders one shadow map after another and uses it to shadow the scene immediately, we can easily use an approach with a dynamic number of slices.

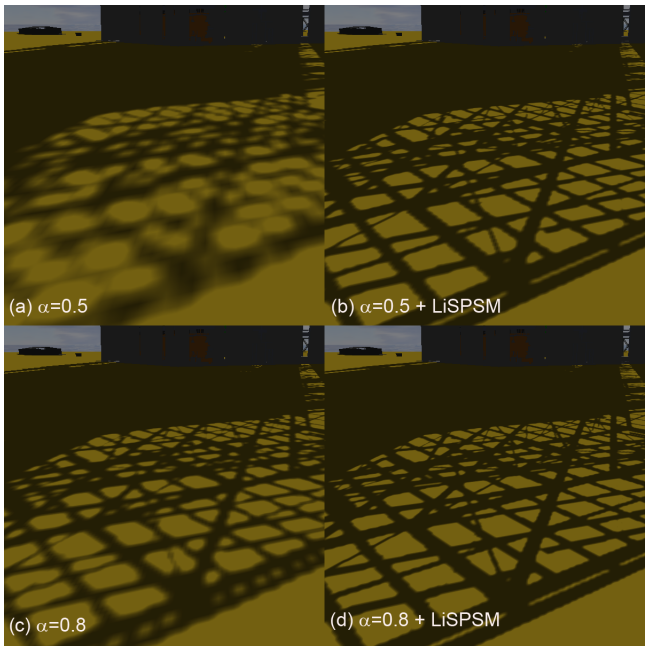

Figure 11: Comparison of different values for α and combination with LiSPSM with 4x1024 PSSM and 3x3 Gauss PCF.

7.2 Optimal Tiling

In a simplified case where the shadow receiver is a plane and the shadow map is projected on it, the projected size of a fragment can be simply determined. We assume the camera to be on plane level, then the width of a projected texel is the crucial parameter to estimate when the sampling rate becomes less then one sample per fragment, because of perspective aliasing. Only taking width into account is generally sufficient for this simplification, because the ratio between width and height depends on the angle of the camera in relation to this plane and only in the case where the camera looks orthogonal to the plane the ratio will be one. However, the shadow map projection is still considered orthogonal to the plane, which will not be given in an usually case and the projection angle to the plane could be considered too. This might only make sense for a terrain szene e.g., where the ground plane can be seen as the terrain, therefore the projected texel size could be estimated more accurately on the terrain, but on other geometry or in an arbitrary szene no such plane exists and therefore we always consider a plane through the camera view direction and orthogonal to the shadow map projection.
The following sketch illustrates a shadow map aligned for the last slice of a view frustum seen from light. From this setting an automatic estimation to set the split planes will be derived.

Known parameters are the distance of the far plane f, the shadow map resolution *smRes*, the view port resolution *res* and the projection matrix opening angle α. The width at the far plane fw therefore is determined by:

$$fw = 2f \tan\left(\frac{\alpha}{2}\right) \tag{12}$$

Because the opening angle differs from which side the shadow map is projected, the opening angle α is not clearly given. Therefore, we make an further simplification and assume $fw = \sigma f$. A practical opening angle is about 60 to 90 degree, then σ should be set to

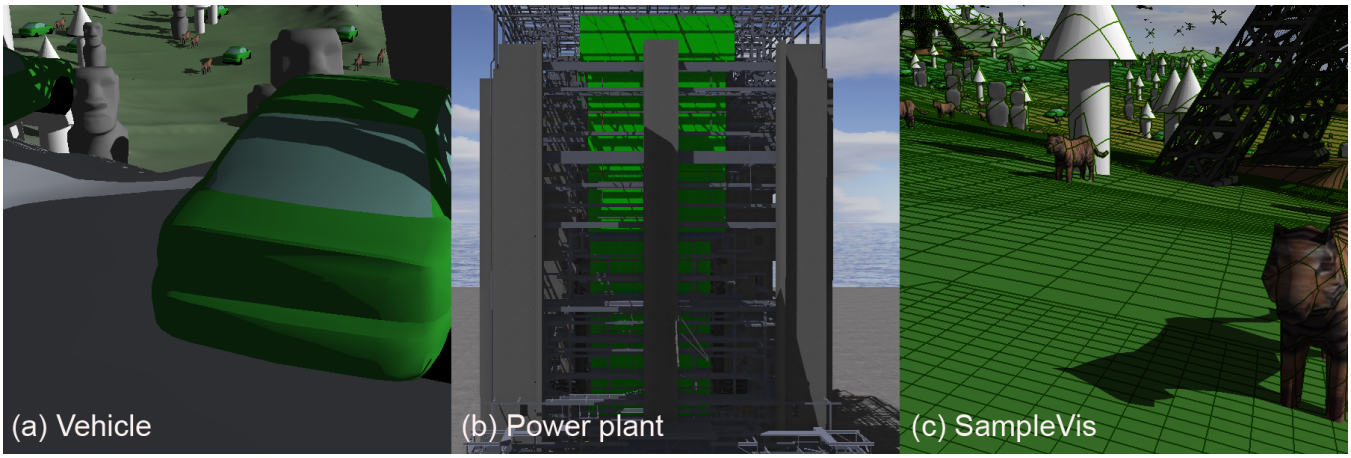

Figure 13: Results with PSSM; 6 2048^2 slices. (a) Case with usually bad projection aliasing on the vehicle. (b) Power plant scene where shadows are projected over large distances on steep surfaces. (c) Visualization of shadow map alignment. Perspective aliasing is no problem anymore, because there are always enough samples, furthermore, no transitions are visible.

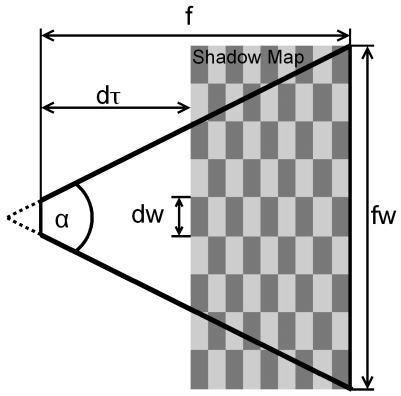

Figure 12: View frustum and shadow map projection of a slice. Distance d_τ where one shadow map sample is projected to one fragment on the screen should be found.

about 1.5. The width of a shadow map texel in world space is:

$$d_w = \frac{f_w}{smRes} \tag{13}$$

The split distance d_τ where the projected texel width approximates ρ is:

$$d_\tau = \frac{d_w \cdot res_{avg}}{\rho} \tag{14}$$

To have one shadow map sample per fragment ρ should be set to 1, but due to our simplifications and because there usually is also projection aliasing, ρ can be set to a value less or greater than 1 depending on the application.

Equation 14 is now used to determine the first split plane, because further slices have a similar geometry reapplying this scheme will lead to a reduction by the same factor. This factor can directly be calculate using:

$$f = \frac{res_{avg} \cdot \sigma}{\rho \cdot smRes} \tag{15}$$

This equation is only valid if

$$\frac{res_{avg} \cdot \sigma}{smRes} < \rho \tag{16}$$

is preserved, otherwise the factor f will be >1 and therefore the split plane would be outside the view frustum.

With the reduction factor f the view frustum is splited successively, which is continued until the new split distance d_τ is less than near plane distance n, however, this often leads to an unpractical number of slices, therefore, a value to determine maximum number of slices is additionally used and the last two slices are simply split by equation 11.

7.3 Results and Summary

PSSM gives a well distributed shadow quality with certain number of passes, whereby the rendering costs scale predictable with the scene. However, there are many cases with less shadow map samples near the camera. This can be well tuned with the adjustable combination of the uniform and logarithmic split scheme, but a pixel accurate result can still not be guaranteed.

To get pixel accurate shadows our optimal tiling estimation delivers the reduction factor for frustum splitting to overcome perspective aliasing. This factor can also be considered as tuning parameter to determine the number of required slices with a logarithmic split scheme which strongly depends on the view port and camera parameters. For a 1280x1024 view port resolution with a 60 degree camera opening angle about 6 slices each with a 2048^2 shadow map are sufficient to get a nearly pixel accurate result in most test case. However, a higher opening angle would clearly increase the number of required slices or force to use a higher resolution shadow map.
It might also be considered that if many slices are used, the required resolution along the view direction in a slice will be reduced, then a non-square texture with less resolution along the view axis could be used to save some fill rate. This, however, does not apply to a case where the light is nearly parallel to the view direction.

Figure 13 shows some results generated with PSSM. With the auto tiling estimation the optimal number of slices for perspective aliasing is known, hardly any shadow artifacts can be noticed. Cases (a) and (b) are extreme cases which usually have clearly

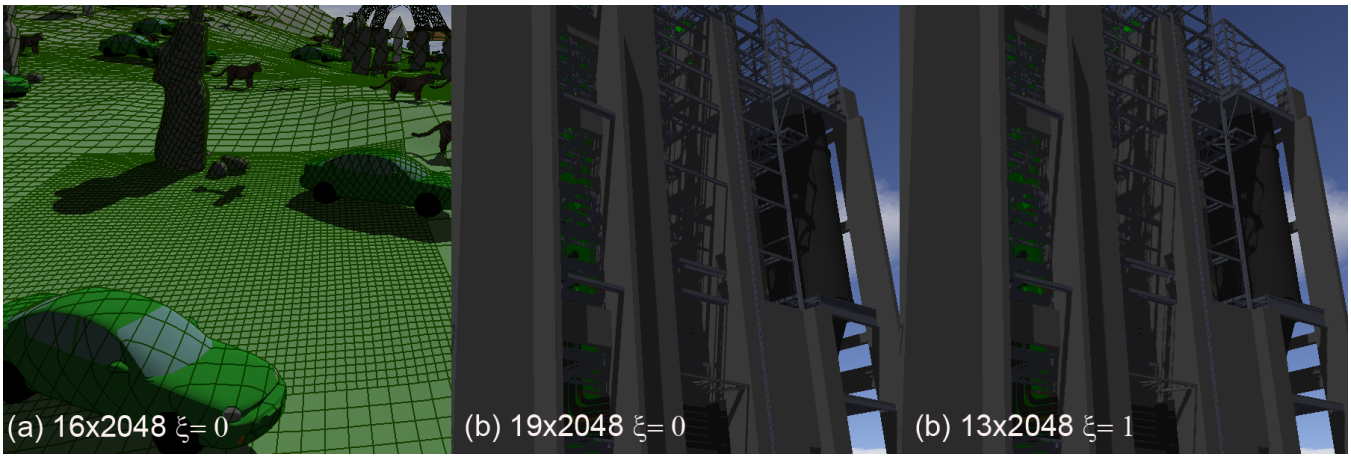

Figure 14: Results of FVSM with a 2048^2 shadow map resolution and 3x3 Gauss PCF shadow map filtering. (a) shows the quadtree like refinement of the shadow map resolution. (b) and (c) show a comparison with a different quality trade off parameter ξ.

aliasing artifacts. Image (c) also shows the alignment of the shadow map slices. When the sampling rate becomes critical horizontal, a new slice begins and there are always enough samples to have one per fragment.

Furthermore, in most cases less slices would have produced a similar result and there often is no visible shadow in some slices. A more extensive scene analysis would be required to fully adapt on the visible area and scene properties. Such an approach, but in combination with a shadow map tiling instead of frustum splitting is followed by Giegl et al. Fitted Virtual Shadow Maps [Giegl and Wimmer 2007a] discussed in the next section.

Overall, PSSM can eliminate aliasing artifacts quite well by only using a few passes. The rendering costs can also be predicted very well when a constant number of slices is used, therefore, it is well suited for realtime applications and has already been used in recent games.

8 Tiling Techniques

A further class of techniques simulate a high shadow map resolution by tiling the shadow map and refine it in a hierarchical update process. One of the first published approaches is Adaptive Shadow Maps (ASM) [Fernando et al. 2001]. It originally has not been designed to fit on the GPU, whereby most of the work is done on the CPU and therefore, does not perform very well. In 2005 an implementation of ASM on current graphics hardware has been published, but it is still limited by many expensive CPU readbacks. [Lefohn et al. 2005]

Another technique which simulates high resolution shadow maps is Queried Virtual Shadow Mapping (QVSM) [Giegl and Wimmer 2007b] and the improved version Fitted Virtual Shadow Maps (FVSM) [Giegl and Wimmer 2007a]. Thereby a virtual shadow map is splitted in equally-sized tiles. During the rendering the virtual shadow map is refined in a quadtree like fashion only where necessary. To accumulate the shadow result a similar approach like described in section 5 is used, which performs the process with little overhead.
The original QVSM uses hardware occlusion queries to measure the improvement and stop the refinement when it drops under a certain threshold. In the paper also some further optimizations to improve the performance are described, but they are negligible, because the whole process has been adapted with FVSM.

FVSM first makes an extensive scene analysis to determine which resolution is needed in each tile. Therefore the output of the first deferred rendering pass is transferred to the CPU and then the bounding box in light-space of each screen space fragment is calculated. It is similar to our approach of calculating the optimal confidence described in section 6.3, but it separately handles required u and v shadow map resolution. They render a 256x256 image of the scene whereby the required resolution and the shadow map tile of each fragment is calculated. This image is then transferred into the system RAM and each tile is statistically analysed to reject outliers by the CPU. After that a quad-tree like structure called the ”Shadow Map Tile Grid Pyramid” (SMTGP) is merged up from the tiles. In the rendering process this pyramid is traversed from top down, whereby a shadow map is rendered, if the required resolution can be covered by the supported texture resolution of the graphics hardware, otherwise the sub-tiles are processed while different required u- and v-resolution are considered too. Now it is guaranteed that only required tiles to avoid undersampling are rendered without using any hardware occlusion queries. Furthermore, an easy to use quality vs performance trade-off ξ, which simply shifts the required resolution, whereby the quality is equally reduces, can be used.

We also implemented our own version with some variations of this technique, whereat we entirely calculate the bottom level of the SMTGP on the GPU, however, whereby the statistical analysis comes rather short. The exact require resolution is calculated for 256x256 shadow map tiles and the averaged for 64x64, which turned out to be accurate enough. LiSPSM has also been disabled because it often produces very bad quality in the distance, which leads to suboptimal tiling.
Figure 14 shows results generated with our implementation. In (a) the rendered tiles in a quadtree like refinement can be seen. Image (b) and (c) compare the result with a different quality parameter ξ. Furthermore, we only used a 2048^2 shadow map, because it turned out that the number of passes only increases slightly, but is still faster in comparison to 4096^2. Probably because our culling keeps the overall rendering cost nearly the same and therefore the rendering is rather fill rate limited.

Overall FVSM performs much better than QVSM and is a well suited technique to shadow large-scale dynamic scenes without

shadow artifacts, however it requires distinct more passes than PSSM, but on the other hand it completely adapts on all shadow scenarios and thereby also projection aliasing.

9 Shadow Biasing

When the shadow map is sampled to do the shadow test, the sample position usually is not exactly at the texel center, whereby the nearest sample must be taken which will be different than the real one, especially when there is a lot of undersampling. A simple case is illustrated in Figure 15.

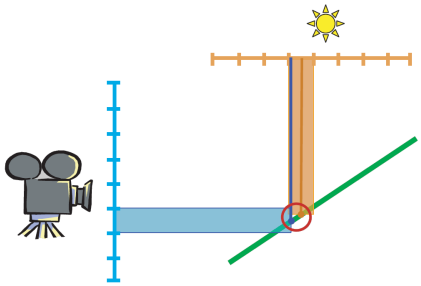

Figure 15: Shadow map resampling during the shadow test.

The different resampling lead to incorrect self-shadowing artifacts characterized by shadowed spots in the middle of a lit surface, which is called ”shadow acne”. A simple solution for this problem is to add a certain bias to the depth value before the shadow test, which has to be manually configured for each setting. Polygons with no depth slope hardly need any biasing, while for polygons that are almost parallel to the light direction a big bias is required. This can be achieved by using slope-scale biasing, where the bias is altered dependent on the depth slope of the polygon.
Because the bias has to overcome the deviation in every case a quite large value might be needed, especially when shadow map filtering is used. Although the shadow map samples are well distributed in view space when LiSPSM is used, the bias has to be set to a very large value, because the reparameterization results in shadow map samples far apart from each other near the far plane observed in world space in which the bias has to be set. This leads to noticeably misplaced shadows, also called ”peter panning”. Figure 16 compares uniform shadow mapping with the smallest suitable (a), LiSPSM and the same bias (b) and LiSPSM with a suitable bias (c).

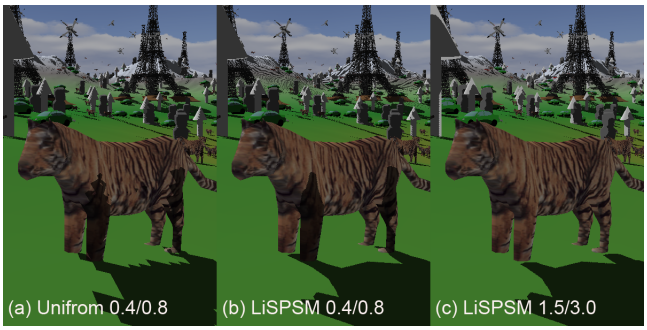

Figure 16: Comparison of different bias values with uniform shadow mapping and LiSPSM (constant / slope-scale bias). The shadow map resolution is 4096.

A possible solution would be to adapt the bias depending on the distance to the camera.

Biasing with confidence-based shadows is no real problem, because only high accurate samples are visible in the final result. The same applies for frustum splitting. Thereby, the bias can be configured for the first or last depth slice. Because the sampling rate of the other slices is in relation to the shadow map area between the two slices, this factor can be used to scale the biasing value, whereby hardly any misplaced shadows can be noticed. The same factor has been used to scale the confidence in section 6.5 when the frustum is reduced. However, we found cases where biasing is still problematic.

Results of confidence-based frustum reduction (a) and PSSM (b) at a biasing critical point is show in Figure 17.

9.1 ID Buffer

Another simple solution for the biasing problem is to use an ID buffer [Forsyth 15. May 2007] that stores the object or polygon ID instead or additionally to the depth value. A 16-bit value to store the ID would already be sufficient. The shadow test then checks if the surface ID stored in the ID buffer is equal to its own ID, hence the surface is lit, otherwise it is shadowed. However, it is not that easy, because edge acne will occur if the object behind gets hit instead. This can be solved by sampling the nearest four neighbours and only shadow the surface if all four IDs do not match. The only side effect is that the shadow shrinks, which can be clearly seen in Figure 17 (c).

Rendering object IDs can be easily achieved, by passing an additional parameter to the fragment shader when an object is rendered and write it together with the depth to the shadow map. When the objects are shaded their IDs can be used to determine inter object shadows without any bias and use the common depth comparison with a suitable bias for self shadowing, which already removes most of the peter panning syndrome. Tom Forsyth describes an approach that stores both, ID and depth in an 8-bit ID + 8-bit depth buffer. This is enough for at least 256 objects and if the IDs are assigned carefully it should be sufficient for even more, because only objects close to each other have to be differentiable. The depth also only covers every object by itself, because the depth is only needed for self shadowing.

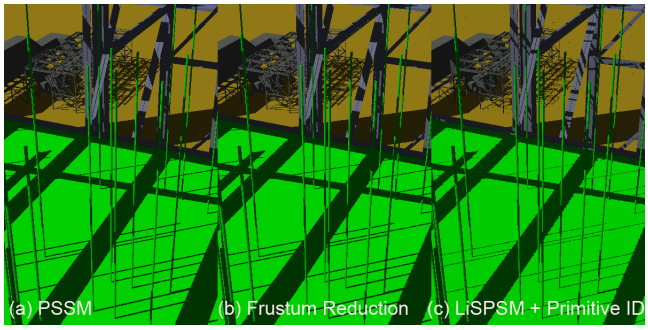

Figure 17: Shadow result in an biasing problematic case (far plane: 500 units, pipe diameter: 0.05 units). (a) PSSM with scaled bias per depth slice, (b) Frustum reduction with a similar configuration than (a), (c) LiSPSM and ID buffer instead of depth test.

The better way would be to entirely use polygon IDs. One way would be to roll out the geometry and fill it with unique IDs for each polygon which, however, makes usage of indexed geometry

impossible. A much easier and better solution is possible with the DirectX 10 API, thereby the primitive ID system value can used and added to the object ID to generate the IDs.
Because our implementation is based on DirectX 9, we used a simple hack to achieve a similar result. In a second pass which renders the deferred ID buffer, we generate a value from a combination of the position and the normal vector in the vertex shader. Then it is added to the object ID and the flat shading mode is used to draw the ID buffer. The same is done to render the shadow map. Because flat shading uses the output value of the first vertex to fill the entire polygon, it does not properly work with indexed geometry, which generates some artifacts in our scene, but a brief impression of the capabilities of using the polygon ID is possible.
Figure 17 (c) shows the result of this implementation with LiSPSM, compared to the nearly exact shadow generated in (a) and (b). Spite the artifacts because of the fine shadow structures, ID buffers work very well and especially remove peter panning artifacts of objects in the terrain scene when LiSPSM or uniform shadow mapping is used.

10 Comparison

This section compares the presented techniques in our two test scenes, a terrain and the power plant. A performance benchmark had been made, where each technique had to render ten different camera settings for three seconds. All techniques had been configured with practical settings to achieve the best possible shadow quality with the least required rendering costs. The test system was a Core2Duo @ 3Ghz with 4GB Ram and a Nvidia GeForce 8800 GTS with 640MB video ram. Figure 18 shows the achieved frames per second (FPS) in both scenes.

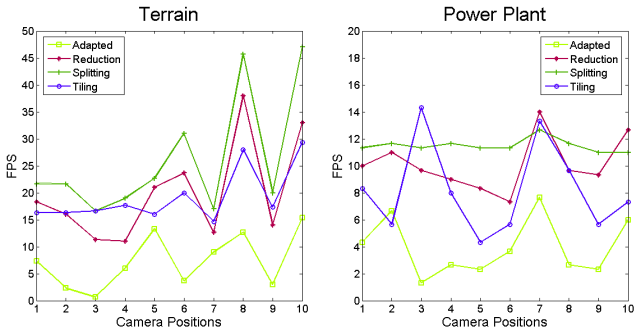

Figure 18: Performance comparison between Adapted Confidence, Frustum Reduction, Frustum Splitting and Shadow Map Tiling.

Because a previous performance analysis showed that a 2048^2 shadow map is the most efficient on the test system, this setting had been used with all techniques.

Adapted Confidence: To get a well initial shadow map alignment, LiSPSM had been used. The stopping criteria was when less than 3000 pixels had been refined in the last pass. In some unfavorable conditions the shadow only converged very slowly and the process had to be aborted after 30 passes, whereby the shadow outlines were still fringed. The graph shows that the performance strongly depended on the setting and overall was clearly below all other.

Frustum Reduction: The number of required passes depends on the visible depth range and only a few were mostly sufficient. Most camera settings could had been rendered in a useful frame rate. However, a pixel correct shadow result is not guaranteed and in a few cases some artifacts could had been seen.

Frustum Splitting: To get nearly perfect shadows in all settings, five slices had been configured. A constant high frame rate had been achieved in nearly all settings. Transitions were not visible.

Shadow Map Tiling: The scene analysis keeps the number of required passes quite low and does not cost much performance itself. Overall the performance was almost on the top, whereat artifact free pixel correct shadow had been rendered.

This final table puts all techniques together and compares some important aspects against each other:

	Pixel Accurate	Passes	Implementation	Artifacts
Adapted	yes	many	complex	fringed outlines
Reduction	optional	few	complex	marginal
Splitting	no	few	simple	(transitions)
Tiling	yes	average	complex	none

11 Conclusion

Deferred shading or shadowing is definitely the way to go when multiple shadow maps should be accumulated successively. The additional rendering cost in the first pass already compensate when the scene has a high depth complexity and when expensive shading computations should be done in the fragment shader, which is the case especially in the power plant scene.
Culling of shadow casters with techniques like PSSM or FVSM was also very important to keep the frame rate up to an interactive level.

We have shown that confidence-based techniques can produce very accurate shadows. With the optimal confidence and occlusion queries it is possible to automatically adapt on the scene properties, but the number of required passes can exceed the practical limit. Therefore, a combination with temporal reprojection [Scherzer et al. 2007], is possibly a better way to accumulate high quality shadows when the frame rate is high enough.

Furthermore, frustum splitting turned out to be a very powerful technique to overcome perspective aliasing, but the only way to completely avoid undersampling would be to make an extensive scene analysis to set optimal splits. FVSM does this and uses a simple shadow map tiling approach to refine the shadow quality, but it seem not to be the best way to tile the shadow map, because a nearly exact shadow result can be generated with only a few frustum splitted slices in most cases as well.

References

AILA, T., AND LAINE, S. 2004. Alias-free shadow maps. In *Proceedings of Eurographics Symposium on Rendering 2004*, Eurographics Association, 161–166.

ARVO, J. 2004. Tiled shadow maps. In *CGI '04: Proceedings of the Computer Graphics International*, IEEE Computer Society, 240–247.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM Press, 161–165.

FERNANDO, R., FERNANDEZ, S., BALA, K., AND GREENBERG, D. P. 2001. Adaptive shadow maps. In *SIGGRAPH 2001, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, 387–390.

FORSYTH, T., 15. May 2007. Shadowbuffers. `http://home.comcast.net/~tom_forsyth/papers/Tom_Forsyth_Shadowbuffers_GDC2007_small.ppt.zipl`.

GIEGL, M., AND WIMMER, M. 2007. Fitted virtual shadow maps. In *Proceedings of Graphics Interface 2007*, Canadian Human-Computer Communications Society, 159–168.

GIEGL, M., AND WIMMER, M. 2007. Queried virtual shadow maps. In *Proceedings of ACM SIGGRAPH 2007 Symposium on Interactive 3D Graphics and Games*, ACM Press, 65–72.

LEFOHN, A., SENGUPTA, S., KNISS, J., STRZODKA, R., AND OWENS, J. D. 2005. Dynamic adaptive shadow maps on graphics hardware. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, ACM, 13.

LLOYD, B., TUFT, D., YOON, S., AND MANOCHA, D. 2006. Warping and partitioning for low error shadow maps. In *Proceedings of the Eurographics Symposium on Rendering 2006*, Eurographics Association, 215–226.

MARTIN, T., AND TAN, T.-S. 2004. Anti-aliasing and continuity with trapezoidal shadow maps. In *Proceedings of the 2nd EG Symposium on Rendering*, Eurographics Association.

SCHERZER, D., JESCHKE, S., AND WIMMER, M. 2007. Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*, Eurographics Association, 45–50.

SHISHKOVTSOV, O. 2005. *GPU Gems 2*. ch. Deferred Shading in S.T.A.L.K.E.R., 143–166.

STAMMINGER, M., AND DRETTAKIS, G. 2002. Perspective shadow maps. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM, 557–562.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph. 12*, 3, 270–274.

WIMMER, M., SCHERZER, D., AND PURGATHOFER, W., 2004. Light space perspective shadow maps.

ZHANG, F., SUN, H., XU, L., AND LUN, L. K. 2006. Parallel-split shadow maps for large-scale virtual environments. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, ACM Press, 311–318.