Shadow Mapping Algorithms
NHTV - International Game Architecture and Design 2013

Name: Vladimir Bondarev
Student Number: 080933
Date: 4 June 2013
CONTENT

1 Introduction...........................................................................................................................................5

2 Aliasing Error........................................................................................................................................7
  2.1 Sampling Error................................................................................................................................7
  2.2 Precession Error...............................................................................................................................7
  2.3 Acne....................................................................................................................................................7
  2.4 Dueling Frusta.................................................................................................................................7

3 Previous Work......................................................................................................................................9
  3.1 Shadow Mapping Algorithm...........................................................................................................9
  3.2 Anti-Aliasing Techniques (with pros and cons).............................................................................9
    3.2.1 Warping.....................................................................................................................................9
    3.2.2 Partitioning.................................................................................................................................12
    3.2.3 Warping & Partitioning.............................................................................................................13
    3.2.4 Irregular Sampling....................................................................................................................14
    3.2.5 Filtering...................................................................................................................................17
      3.2.5.1 Percentage-Closer Filtering.................................................................................................17
      3.2.5.2 Variance Shadow Mapping.................................................................................................17
      3.2.5.3 Bilateral Filtered Shadow Maps............................................................................................18

4 Quality Measure Definition................................................................................................................19
  4.1 Aliasing...........................................................................................................................................19
  4.2 Resource.........................................................................................................................................19
  4.3 Simplicity and Flexibility................................................................................................................21
  4.4 Miscellaneous.................................................................................................................................22

5 Quality Measure Formula....................................................................................................................23
  5.1 Aliasing...........................................................................................................................................23
  5.2 Resource.........................................................................................................................................23
  5.3 Simplicity and Flexibility................................................................................................................24
  5.4 Miscellaneous.................................................................................................................................25
  5.5 Formula..........................................................................................................................................25

6 Measurement & Results.......................................................................................................................27

7 Discussion............................................................................................................................................31

8 Conclusion...........................................................................................................................................33

9 Reference............................................................................................................................................35
ABSTRACT

Original shadow mapping algorithm suffers from aliasing error. Many popular techniques aim to improve shadow mapping, unfortunately most aliasing reduction comes at the expense of memory, low latency, flexibility and complexity. Using a special formula this research attempts to answer the following theoretical question: "Which shadow mapping technique is most suitable for use in general purpose real-time graphics applications?". The formula consists of a general list of advantages and disadvantages, which are correlated into a specific score. The highest score represents a most suited algorithm for general purpose shadow mapping in real-time application. Furthermore this research briefly explains common aliasing errors, multiple shadow mapping algorithms, the compilation of the formula and measurements & results. Also, other observations can be found in the section tattled Discussion and Conclusion.

1. INTRODUCTION

Shadows contribute to a great portion of realism towards the reproduction of a visual environment. Without any shadows the geometry appears to be flat. The orientation and shape of an object is the scene is often hard to deduce. As in traditional art, painters use attached and cast shadows (see figure 1) to enrich geometric environment with visual cues.

![Figure 1](image-url)  

**Figure 1:** Area facing away from the light source is less exposed towards the direct light which results in an attached shadow. Cast shadow are also an absence of light, but is determined by the amount of occluded direct light by another surface area. Source[MKK98]/Figure 1.

The cues help our brain to understand the environment, thus the scene appears to be more realistic. This makes it equally important for computer graphics, because of that a substantial amount of research is invested towards developing efficient shadow algorithms.

Most popular approaches are based on shadow volumes [CROW77] and shadow mapping [WILL78]. Shadow volumes use stencil buffer to determinate occluded areas. In this research we are primarily focusing on the shadow mapping. To retain the scope of this research we are not going to investigate shadow volumes any further.

The original shadow mapping algorithm is introduced by Lance Williams [WILL78]. It is a simple and effective image space algorithm to generate shadows for complex scenes and curved geometry in real-time applications. As most image space algorithms, shadow mapping suffers from aliasing errors. Many already existing techniques reduce the aliasing error, but usually at the expense of resources or flexibility.

By analyzing modern techniques, a simple formula is developed to test each algorithm for its current general performance. This research paper shows that filtering techniques score the highest score. Variance Shadow Mapping has the highest score of all techniques among all categories.
2 Aliasing Error

Most benefits of shadow mapping are derived from being an image space algorithm however, it also inherits all of its drawbacks. The shadow map has a finite resolution and depth precision therefore it is crucial to match corresponding camera view fragment and shadow map pixel as close as possible together and stay inside the tolerated precision range.

2.1 Sampling Error

Undersampling in shadow mapping occurs when an area of multiple camera view fragments is matched to a single shadow map pixel. Exactly the opposite happens with oversampling, a single camera view fragment is matched to an area of multiple shadow map pixels. Both oversampling and undersampling cause aliasing error (comparable to an early stage of 3D rasterization without texture filtering). High amount of samples (oversampling) is expensive and some shadows may also experience jittering. Low amount of samples (undersampling) is beneficial towards real-time performance, but is likely to result in blocky and incorrect shadows. In real-time graphics oversampling occurs scarcely due to a limited resource constrains, focusing most research to battle the undersampling.

2.1.1 Perspective aliasing

Undersampling and oversampling are typically caused by the camera view perspective. When a uniform shadow map (homogeneously distributed samples in 3D space, see figure 3) is projected on the camera view plane, near shadow map samples are larger and thus undersampled and far pixels are oversampled. Most papers refer to this type of phenomena as perspective aliasing. Perspective aliasing is first mentioned in [SD04].

2.1.2 Projective aliasing

Another infamous sampling error phenomena occurs when the surface normal becomes more perpendicular relative to the orientation of the light. Causing transformed shadow map sample in camera space approaching the infinity thus undersampled. More on this in section 2.2.3.

2.2 Precision Error

Although precision is still an issue, it has become much less so in the last couple of years. Modern GPUs support shadow maps with 32-bit precision which provides more precision however precision errors still occur and it is important to adjust the near and far clipping planes of the light’s projection matrix to the minimum necessary distances thus keeping both planes as close as possible. If the precision required to store the range of depth values in the shadow map goes beyond the maximum possible precision that can be stored in the map, then it is possible that the shadow test will fail and result in the shadows being projected onto incorrect surfaces.

Figure 4: Aliasing Error. Source [WIM04] figure 4.
Sampling errors occur when $dp$ (shown in figure 4) is either smaller or larger than the camera view pixel. In theory shadow mapping has always some amount of error, the trick is to manage the error until it is no longer visible to the human eye.

### 2.3 Acne

Geometry with a curved surface is more susceptible to sampling errors and may demonstrate shadow map discontinuity as *acne*. Acne is a result of undersampling. The undersampling occurs when the surface becomes more parallel with the light. The more surface gets parallel with the light, the longer $dy$ (figure 4) will become. Eventually it will become infinitely long. Eventually $dp$ becomes larger than a single pixel from camera view and causes an aliasing error.

![Figure 5: (Left) Without global polygon offset. (Right) With global polygon offset. The object suffers from "acne" and the shadow on the ground has very blocky edges. Both errors are a result of undersampling. The object on the right is not completely acne-free because, the polygon offset is too small to remove all visible error.](image)

Acne can be reduced by introducing a small offset to the shadow map depth value (see figure 5). The offset can be calculated ([KOZ04] section 14.2.3) and applied to minimize visible error. This type of solution will not work on surfaces which are orientated parallel to the light (where the surface area and the error approaches infinity). Large offsets result in incorrect self-shadowing and may even penetrate through other surfaces.

### 2.4 Dueling Frusta

Dueling frusta is a well-known shadow mapping problem. It occurs when the light is facing towards the camera and also the light orientation becomes parallel with the camera face normal (see figure 6 a). While both camera and light have their own perspectives, shadows near the camera becomes heavily undersampled.

![Figure 6: Light points towards the camera. a) light orientation towards the camera that causes the most sampling error. b) Blue is undersampling and red is over sampling (source [KIL07]).](image)

This phenomenon occurs most often when the shadow map is generated using a perspective projection matrix and the field of view of the camera’s projection matrix is large. If both the light’s view matrix and the camera’s view matrix are using orthographic projection matrices, then the dueling frusta problem would not occur.
3. Previous Work

3.1 Shadow Mapping Algorithm

In 1978 Lance Williams published a paper on *Casting Curved Shadows on Curved Surfaces* [WIL78]. The paper explained how z-buffer information can be utilized to determine which surfaces are occluded and which are visible to the light. The main benefit of the algorithm is that it uses stored depth information instead of complex geometrical equations to calculate light visibility. By only needing depth values to calculate occlusion, shadow mapping becomes universal for all primitive types and greatly reduces the overall complexity.

![Figure 2: Directional light. (Left) Camera view. (Right) Shadow Map.](image)

The algorithm consists of two main steps:

- **Step I:** The scene is rasterized from the light perspective. Only depth values are required and stored into a texture which is called the shadow map.

- **Step II:** The shadow map is linearly transformed to match camera view pixels. While the scene is rasterized from the camera view, matched depth pixels from camera view and shadow map are compared. If the value from shadow map is greater than camera view value, the pixel is not visible to the light.

In the past couple of decades, shadow mapping earned its popularity in real-time rendering due to its simplicity, scalability and image space based nature which benefits greatly from modern GPU hardware. Since most calculations can be performed on the GPU, it's very fast and has low overhead. The overhead scales linearly with respect to the amount of lights used to generate shadows. Unfortunately like most image space algorithms, shadow mapping also suffers from aliasing error.

3.2 Anti-Aliasing Techniques

Improved shadow mapping techniques can be categorized in warping, partitioning, warping and partitioning, irregular-sampling, filtering and hybrid. Hybrid techniques use a combination of shadow maps and shadow volumes [CROW77] to reduce the aliasing error. Because the hybrid solution focuses on shadow-volume technique we are not going to discuss this solution in this paper.

3.2.1 Warping Techniques

Warping techniques redistribute shadow map samples with a global parameter such as a matrix which describes how the space is curved. This space-alternation allows the shadow map to focus on more important parts of the scene where the most aliasing occurs (usually near the camera). By adjusting the light view space, more samples are taken at the near plane of the camera and less at the far plane. See perspective aliasing (section 2.1.1).

**Perspective Shadow Maps** [SD02] (PSM)

Perspective Shadow Maps (PSMs) attempt to reduce perspective aliasing by adjusting the uniform sampling distribution of shadow maps. The improvement is achieved by generating shadow maps in post-perspective space of current camera. Before perspective shadow maps are generated, the scene and the light source are
transformed by the camera matrix. This results in non uniform shadow map distribution. The closer the objects are to the camera the higher the sampling.

Figure 7: (Left) Uniform Shadow Map. (Right) Perspective Shadow Map. Source [SD02].

PSM is using camera projection matrix resulting in coordinates behind the camera to be inverted, this type of phenomena is called singularity point problem. Depending on the light orientation, objects behind the camera may also cast shadows which are guaranteed to be incorrect due to the singularity point problem. Polygons behind the camera are inverted and are no longer generating correct shadows. The problem can be solved by clipping the geometry behind the camera, but doing this may exclude potential shadow-casters. This problem can also be solved by moving the camera backwards during the shadow map generation. Moving the camera backwards before shadow map generation is not ideal because doing so results in details being stored in the shadow map that may not contribute to the final render and thus sacrificing details in more relevant parts of the shadow map.

Advantages

- The shadow map resolution is distributed according to camera matrix, greatly reducing perspective aliasing on close objects to the camera.
- As fast as original shadow mapping algorithm.

Disadvantages

- Shadow casters behind the camera are inverted by the singularity point of post-perspective space. Clipping the objects behind the camera can resolve the problem, but will exclude potential shadow-casters. Moving the camera backwards will include the missing objects, but will reduce the shadow map quality.
- The shadow map resolution of near objects is greatly increased at the cost of distant object shadow error. As a result when an object is moved relative to the camera, the shadow will exert excessive flickering (discontinuation). This kind of problem is also called continuity problem.
- Limited to certain scenes, which depends on relative camera and light position [MAR04].
- Increases polygonal offset problem (projective aliasing - acne). The depth values in post-perspective space has no longer a linear distribution, which increases the complexity level to deal with projective aliasing.
- Sensitive to the dueling frustum problem.

Light Space Perceptive Shadow Maps [WIM04] (LiPSM)

This particular technique is an improvement of PSM. By using perspective transformation in light space instead of camera space the light does not change position and all light types can be treated as directional lights. This particular improvement excludes the singularity point problem which is found in regular PSMs.

Figure 8: Parameterization between uniform and perspective shadow map. Source [WIM04].
In contrast to PSM, LiSPSM can be tuned between perspective and uniform sampling distribution. The closer \( n \) (shown in figure 8) gets to infinity, the more uniform sampling distribution will be. LiSPSM uses global \( n \) parameterization (settings are identical for all fragments) to solve perspective aliasing.

**Advantages**
- Solves singularity point problem introduced by PSM.
- The shadow map is rendered from light position and this solves the nonuniform polygonal offset problem due to perspective warping.
- Tunable sampling distribution between close and distant objects (from uniform to completely PSM)
- As fast as original shadow mapping algorithm.

**Disadvantages**
- Global parameterization does not solve all aliasing. Local fragment parameterization that approximates logarithmic distribution would provide more optimal results [LLOYD08].
- Sensitive to the dueling frustum problem.

**Trapezoidal Shadow Maps** [MAR04] (TSMs)

This particular wrapping technique attempts to reduce perspective aliasing by approximating the eye's frustum-trapezoid from the light's position. TSM and LiSPSM are very similar in their approach to solve perspective aliasing. TSM computes a bounding trapezoid and LiSPSM uses camera frustum to determinate shadow map sample distribution. Trapezoidal approximation allows for higher precision on certain regions which provides greater flexibility than LiSPSM.

**Advantages**
- Reduces the discontinuity problems that plague original PSMs (non-linear depth value distribution).
- Trapezoidal-transformation allows flexible shadow sampling distribution (focusing on certain areas).

**Disadvantages**
- Finding optimal trapezoid is complex.
- Designed for large ground planes.
- Does not solve the dueling frustum problem.

**Logarithmic Perspective Shadow Maps** [LLOYD08] (LogPSMs)

Logarithmic Perspective Shadow Maps is a very high quality and low aliasing error algorithm that is heavily based on LiSPSM and explores logarithmic parameterization as suggested by [WIM04] in section 4.2. By using logarithmic distribution of parameterization LogPSM greatly minimizes the aliasing error.

![Figure 9: (Left) Uniform shadow map. (Right) LogPSM. The grid on the images indicates sampling distribution. Source [LLOYD08].](image)
Although LogPSM is capable of generating high quality shadow maps, logarithmic rasterization is not supported by current generation hardware. A suggestion is mentioned to approximate logarithmic rasterization at the vertex shader level and tessellate large projected surface area polygons. However, this leads to more complex shaders. The example in [LLOYD08] uses an unoptimized brute-force approach, rendering screen size quads at different depths. On current hardware, the LogPSM algorithm described in [LLOYD08] performs 6x slower in a best-case scenario than traditional shadow mapping techniques and performs 24x slower in a worst-case scenario. The algorithm in its current state is not very attractive to real-time graphics applications due to its heavy performance cost. As explained in [LLOYD08] section 6.3.1, having hardware support for logarithmic rasterization will allow the performance of LogPSM to approach that of LiSPSMs.

Advantages
- Has minimum perspective shadow map aliasing.
- Handles dueling frusta well.

Disadvantages
- Current hardware only supports projective rasterization.
- In order for LogPSM to work, logarithmic rasterization is approximated by vertex program. Large projected surface-area polygons must be tessellated to reduce scene error.
- The approach used in [LLOYD08] is not attractive for real-time shadows.
- Complex and not efficient to implement on current hardware.

3.2.2 Partitioning Techniques

Subdividing camera view frustum in multiple parts aims to achieve efficient sample redistribution in camera view space. Each subdivision usually has a different sample distribution density in light view space and aims to preserve constant density in camera view space.

**Cascaded Shadow Maps [MS12] (CSMs)**

Cascaded Shadow Maps is a partitioning algorithm that divides camera and light view frustum in multiple subfrusta. Each subfrustum has its own shadow map texture which is focused on a certain camera view area. With optimal parametrization this particular approach roughly approximates logarithmic perspective sample distribution with uniform shadow maps. The more the frustum is divided the better sampling approximation will be. Unfortunately, a better approximation also results in heavier GPU load.

![Sampling error: red high aliasing, green no aliasing and blue over sampling. (Left) Cascaded Shadow Mapping and (Right) Logarithmic Perspective Shadow Mapping. Source [LLOYD08] video.](image)

There are two general ways to create subfrusta: fit to scene and fit to cascade [MS12].
• **Fit to scene.** All subfrusta are created with the same near plane which results in overlapping shadow maps.

• **Fit to cascade.** All subfrusta are connected where the following subfrustum near plane is connected to previous subfrustum far plane.

Neither the fit to scene nor the fit to cascade methods are good solutions. *Fit to scene* has high wastage and *fit to cascade* has dueling frusta issues.

Unfortunately the light view volume that falls outside the view frustum is wasted (see figure 12). In this case the wastage is ~1/3 of total texture size. Without proper clipping the GPU cycles are also wasted. An additional warping technique on top of CSM will help to utilize the texture space more efficiently and improve the sampling rates as well.

**Advantages**

• Fit to scene reduces dueling frusta
• Good anti-aliasing, but still not perfect.
• Easy to implement.

**Disadvantages**

• Fit to scene has high texture wastage and requires the same objects to be rendered multiple times.
• Fit to cascade does not handle the dueling frusta problem well.
• Requires separate render pass for each subfrustum.

### 3.2.3 Warping & Partitioning Techniques

Warping & Partitioning techniques combine both partitioning and warping to improve sampling distribution.

**Light Space Cascaded Shadow Maps** [LMCY11]

Light-space Cascaded Shadow Maps (LSCSMs) combine both LiSPSM and CSM to generate high-quality shadows and resolve redundant rendering problems that occur with CSM. The redundancy occurs when view direction and light direction are not perpendicular, resulting in some geometry being rendered into neighboring...
shadow maps. The scene is split into multiple non-intersecting parts from the light view frustum. LSCSM proposes a method to generate shadow maps using multiple warped viewing frusta which are dependent on the position and size of the objects in the scene. The algorithm is also using bounding sphere tests to resolve conflicting depth information which is saved in multiple depth buffers.

Advantages

- Shadow quality is gained by approximating the sampling distribution closer to the logarithmic distribution by combining LiSPSM and CSM
- Performance gain compared to regular CSM by partially removing redundant objects which are otherwise rendered into neighboring shadow maps.

Disadvantages

- This approach is still affected by discontinuity problem (introduced by CSM) between shadow maps transitions.
- Shadows multiplied from different depth maps may exert perspective aliasing artifacts. The artifact is introduced by undersampled shadow map.

Subdivided Shadow Maps [LLOYD05]

Subdivided Shadow Maps uses two levels of frustum subdivisions for a better perspective warping approximation.

![Subdivided Shadow Maps](image)

Figure 13  a) The light is behind the camera. Shadow map on the image is projected downwards. b) Frustum is subdivided to redistribute shadow map samples per each sub-frustum. c) Another subdivision along the faces can be done to better approximate the logarithmic transformation. Source [LLOYD05].

From the light view, the camera frustum is divided into smaller subfrusta (frustum subdivision, figure 13b). Then from the camera view, another subdivision may be done along the camera frustum (face subdivision, figure 13c). Finally, each subdivision is warped. Face subdivision is done from the camera view and therefore also reduces the dueling frustum problem which is introduced by the calculations done from the light view (perspective).

Unlike CSM and PSSM this algorithm has in addition to horizontal splits, it also uses vertical splits. In the camera view space, vertical partitioning enables finer sampling distribution along vertical surfaces and results in lower aliasing error in complex scenes compared to CSM.

Advantages

- High aliasing reduction. (by a factor of 10, mentioned in the original paper [LLOYD05] section Conclusion and Future Work)
- Minimizes or eliminates the dueling frusta problem.

Disadvantages

- Runs approximately 2x slower than basic shadow mapping technique (stated by the paper).
- Relatively complex to implement.

3.2.4 Irregular Sampling Techniques

Irregular sampling techniques generate high shadow quality by using a non-uniform sampling distribution. Commonly found problem in this type of algorithms is a hardware incompatibility.
**Alias-Free Shadow Maps** [AL04]

Alias-Free Shadow Maps generates ray-tracing quality shadows by testing each individual screen pixel for light visibility. All camera view pixels are rendered and then transformed into the light view space where they are tested for light visibility. The greatest challenge is to efficiently test all transformed pixels for visibility. Transformed pixels to the light view space (figure 14 right) are no longer uniformly arranged on the image plane. This is a problem for the GPU because the conventional single rasterization pass for the visibility will become over or undersampled and will result in incorrect shadows. In [AL04] a software solution is used to divide the light view space in a hierarchy, where the correct sample is being matched.

**Figure 14:** Left, uniform sample points from the camera perspective. Right, camera perspective points are transformed into the light space for visibility test and are no longer uniformly arranged. Source [AL04].

**Advantages**

- Ray-tracing shadow quality / Pixel correct sampling.

**Disadvantages**

- Not suited for real-time application:
  - Irregularly placed samples which are not optimal for the GPU hardware.
  - Software rasterization is used to bypass GPU rasterization limitations.
  - Requires transformed pixels analysis to build a hierarchy.
  - Requires hierarchy traversal for visibility test.
- Requires an additional pass to apply shadows.

**Irregular Z-Buffer Technique** [JMB05]

Irregular Z-Buffer is a clever technique that stores irregular samples in two-dimensional grid and produces ray-tracing quality shadow maps. Instead of regular z-buffer to store depth values, this algorithm uses a two-dimensional, hierarchical data structure, to store and query visibility samples. A hashing function is used to access the correct node for each pixel. Unfortunately the data layout is not supported to run real-time on the GPU hardware.
Advantages

• Very low to no aliasing error.

Disadvantages

• The main strength is also currently the main weakness, the buildup and traversal of the data structure is not (fully) supported by the GPU. Possible counterpart that should be later researched is current GPU development. Path-tracing uses new GPU architecture which allows for complex data structures to traverse over the triangle geometry. This technique deserves to be revised according to new hardware developments.

Fast, Sub-pixel Antialiased Shadow Maps [PWCZB09]

Fast, Sub-pixel Antialiased Shadow Maps extends upon the (alias-free) accurate irregular per-pixel shadow mapping algorithm [AL04] with a fine approximate of sub-pixel shadows. Normally sub-pixel shadows can be achieved by super-sampling the shadow maps (brute approach), requiring a large number of samples per screen pixel (sometimes seen as oversampling). The negative results of oversampling is a performance drop and a memory wastage. On the other hand it increases the shadow quality and reduces the jitter by capturing finer details of the light occluders.

![Figure 16: An triangle partially occluding a pixel. The occlusion is stored with a bitmask. Source [PWXZB09].](image)

Instead of super-sampling, this algorithms records an occlusion mask (see figure 16). For each silhouette edge a line is drawn which contributes to the occlusion mask. The occluded pixel area can be represented with a half plane which is precalculated and stored in a lookup table. The results are comparable to 32x-128x super-sampling.

Advantages

• Accurate pixel correct shadows.
• Reduces jitter by representing half-plane silhouette for sub-pixel anti-aliasing.
• Greatly reduces resource cost compared to super-sampled shadow maps.

Disadvantages

• Inherits all problems introduced by alias-free algorithm [AL04].
  • GPU rasterization incompatibility which requires custom software rasterization.
• Increased algorithm complexity.
• Identification of silhouette requires CUDA and then also needs to be rasterized.

3.2.5 Filtering Techniques

Filtering techniques use image space solutions to solve visible aliasing.

**Percentage-Close Filtering (PCF)** [RSC87]
Linear hardware filtering techniques such as bilinear filtering cannot be used on shadow maps to approximate sub pixel value. An average value of neighboring shadow map pixels will provide incorrect depth data and result in shadow bleeding (see figure 19 middle).

Percentage closer filtering solves this problem by averaging shadow test results instead of interpolating neighboring shadow map pixels. An offset kernel with a scalar is used to determine where the neighboring shadow tests are taken, the results are summed and divided by the amount of samples, resulting in average shadow value (see figure 17 middle).

![Figure 17: Left, no PCF. Middle, with PCF. Right, with PCF and Linear Filtering](image)

Although bilinear filtering can result in shadow bleeding, in some cases it can provide and improvement combined with PCF (see figure 17 right).

**Advantages**
• Intuitive and easy to implement.
• Relatively fast on GPU hardware, depends on the number of samples taken.
• Very easy to combine with other sample redistribution algorithms.
• Somewhat reduced acne due to a smoother edge transition.

**Disadvantages**
• Does not solve perspective aliasing. For optimal use it should be combined with sample redistribution techniques.
• With insufficient samples taken, silhouette can still appear aliased.
• Large number of samples is required for high quality silhouette approximation.

**Variance Shadow Maps (VSMs)** [DL06]

The main disadvantage of PCF is that it requires a large number of samples to achieve descent shadow silhouette quality and neighboring depth discontinuities may also generate wrong results. VSM improves on PCF by approximating high sub-pixel shadow quality in a single sampling operation by allowing hardware filtering to be performed on the shadow map. The approximation is done by comparing a depth value to an area distribution value. This algorithm relates pixel depth to the texel area, when hardware filtering (or is prefiltered with guassian pass) is performed the sub-pixel values of depth and square depth is used to reconstruct the occlusion intensity of the projected shadow. If the consistency between depth and square depth is extreme, the sample is then simply ignored.
Advantages
• Very fast, at almost the same cost as ordinary shadow mapping.
• Supports hardware filtering such as mipmapping.
• Significantly reduced projection aliasing.
• Very good to combine with other sample redistribution algorithms.
• Reduced acne due to better depth approximation.

Disadvantages
• Requires two channels to store data.
• High variance can cause light bleeding (square depth can cause precision problems).

Bilateral Filtered Shadow Maps (BFSM) [KK09]

This algorithm uses blurring to reduce perspective aliasing. By applying a Gaussian kernel, the shadow silhouette no longer appears blocky. Unfortunately only blurring will result in shadows to appear on incorrect surfaces (this is referred to as shadow bleeding, see figure 19 middle). When applying Gaussian filtering, depth samples are also compared. When the depth difference between sampled pixels becomes greater, the contribution of the Gaussian filter is reduced. This way the depth discontinuities are detected and shadow bleeding is eliminated, resulting in smooth shadow edges.

Filtering is performed as a post-processing effect on the separate lighting buffer which only contains shadows (alpha channel in this implementation).

Advantages
• Fast, approximately 1x-2x times slower than regular Gaussian pass ([KK09] table 1).
• Reduces / hides aliasing error.
• Easy to implement

Disadvantages
• Smooth shadows are not always wanted. Reducing the kernel size may result in a visible aliasing error.
• Extra buffer to store shadow intensity and the depth values.
• No hardware depth filtering.
4. Quality Measure Definition

Measurement algorithm is based on advantages and disadvantages in section 3.2. The goal is to develop a formula which allows to perform a simple measurement of a shadow mapping techniques based on their descriptions. To find a fitting approach how to measure the algorithms mentioned in section 3.2, all advantages and disadvantage are put next to each other, sorted, categorized and refactored into a form.

4.1 Anti-Aliasing

After combining all advantages and disadvantages from section Anti-Aliasing Techniques (section 3.2), can be concluded that not all aspects of anti-aliasing are equally improved (some are even diminished). For example, PSM reduces perspective aliasing, but increases discontinuity error resulting in more severe acne and less stable shadow (jitter, comes often paired with warping techniques). This means that aliasing reduction requires more specific testing.

Anti-aliasing will be tested in following areas: projective anti-aliasing, perspective anti-aliasing, acne reduction and jitter reduction. These are the main sub-categories which will determine the overall shadow anti-aliasing quality.

Finding reasonable scaling system is very challenging, thus a simple level system is introduced. All negative effects in contrast to original shadow mapping will have the lowest score-level. Improved anti-aliasing in contrast to original shadow mapping will result in higher scores-levels.

Warping and partitioning techniques improve on original shadow mapping by better re-approximation of sample distribution, but still exert some sampling error. Therefore sampling re-approximation is the following logical level after regular shadow mapping. Pixel correct anti-aliasing can be referred to irregular shadow mapping or ray-tracing no longer struggles with sampling distribution error and is logically the following anti-aliasing level. The highest possible anti-aliasing level is sub-pixel anti-aliasing, achieved by super-sampling. "Fast, Sub-pixel Antialiased Shadow Maps" [PWCZB09] or path-tracing. Everything else falls in between.

Tested aspects of anti-aliasing:
- Projective anti-aliasing.
- Perspective anti-aliasing
- Acne reduction
- Jitter reduction

Anti-Aliasing quality level overview in chronological order:
- Increased aliasing.
- Original shadow mapping.
- Re-approximation of sample distribution.
- Pixel correct anti-aliasing.
- Sub-pixel correct anti-aliasing.

For some special techniques it is possible to fall in between two quality levels. For example: a combination of warping and partitioning techniques can provide close to pixel correct shadows.

4.2 Resource

As the hardware improves, the image quality requirement of real-time application also increases. Generating high quality anti-aliased shadows with poor use of resources renders the algorithm useless for real-time application. It is absolutely vital to determine the resource consumption of each algorithm to benchmark the real-time compatibility.

Current generation games use a large number of triangles and consume vast amounts of memory. In this section we want to evaluate how the hardware resources are used by each algorithm in contrast to original shadow mapping technique. Main performance will be tested in: performance and memory use.
Performance

The performance can be tested in many different ways. To keep measurement simple and intuitive, we want to assess all algorithms for their ability in handling large number triangles. GPU computability has one of the most significant roles in triangle rasterization. High fill-rates can insignificantly increase algorithm latency. Therefore it is also very important to test for increased fill-rates in contrast to original shadow mapping algorithm.

Handles Large Triangle Counts

Current generation of games use vast amounts of triangles (more than a million). A technique which is not fully hardware supported (rasterization on the GPU) or has a custom workarounds, will highly likely have a poor real-time performance. Regular shadow mapping is considered to be the fastest algorithm, any other complications or inefficiencies will increase the computational latency.

Each technique will end up in one of the following categories:

- Core run-time element is not supported by the GPU rasterization, requires emulation or complex data structure outside the GPU.
- Core run-time element is not supported by GPU rasterization, but can be emulated on the GPU.
- Fully supported by GPU rasterization, but has some major inefficiencies (such as requires multiple scene renders in combination with error analysis, etc).
- Performs on equal or almost equal level as original shadow mapping. May have some minor inefficiencies.

Algorithms that are already used in real-time applications such as computer-games should never fall in to first two categories (two most inefficient categories), their ability in handling large number triangles has been already proven. The categorization of each algorithm will be applied on current hardware comparability.

Not increasing fill-rates

High fill-rates causes performance stall and is preferably avoided. A high fill-rate can be associated with super-sampling or performing an extra render pass.

| No | Has an increased fill-rate, think of multiple passes or triangles appear larger on the shadow map. |
| Yes | Equal or no significant changes in fill-rates compared with original algorithm. |

Memory

This section determines if more memory is required to save any extra data or if the bandwidth is increasing. With low bandwidth the GPU can process faster and with high bandwidth slower. High memory consumption is also not a beneficial factor, which will also be taken into account. Some algorithms not always use the allocated memory in an efficient matter, wastage is some what significant and will be taken more lightly into account.

Bandwidth

Bandwidth determines how often the memory buffer needs to be accessed. Based on this criteria we can benchmark the memory flow on the GPU. The less the memory needs to be accessed, the faster the algorithm can perform.

- A very high number of samples needs to be taken to achieve descent image quality.
- A moderate number of samples needs to be taken which is inside acceptable levels, but still significantly slows down the algorithm.
• Low, none to a very low number of samples needs to be taken which results in an insignificant performance drop.

Capacity

This criteria determines relative quantity amount of memory required by the algorithm. Low memory usage is beneficial towards algorithm overall performance and gives also space and ability for other programs to make use of the memory. Large amounts of data can cause limitations to the shadow quality due to insufficient available memory to process at the same time on the GPU.

• Requires high amount of storage (super sampling).
• Requires extra memory to store data, but is in acceptable range with current hardware.
• Does not require extra memory to save extra data.

Wastage

Techniques with low memory use efficiency is a waste and requires to participate in final equation which determines algorithm quality.

• High memory wastage, 50% or allocated memory is occasionally not used.
• Moderate wastage, maximum 30% of allocated memory is occasionally not used.
• None to low wastage, maximum 10% of allocated memory is occasionally not used.

4.3 Simplicity and Flexibility

This section is responsible for grading the implementation complexity and overall use generality for various scene setups. Easy to implement and flexible algorithms are usually favored over complex algorithms which are restricted to particular scene setups. Therefore it is also important to take this measurement into the account for overall final algorithm score.

Simplicity

A simple algorithm is easier to implement correctly and perhaps also easier to combine with other algorithms if required. This section is challenging to benchmark since complexity is a relative term. We define the categories of simplicity in following terms from low to high complexity:

• Algorithm consists from multiple techniques and is not intuitive to implement. It may require a custom solution to bypass hardware limitations.
• The algorithm consists of multiple techniques, but is fairly straightforward to implement without any real challenges.
• One additional algorithm with not higher than moderate complexity.

Flexibility

Versatile algorithms can be used for various light types, scenes, positions and orientations. Unfortunately many techniques introduce various restrictions that reduce their over usefulness. For example, some warping algorithms perform poorly when the light and camera orientation approaches parallel orientation. In this section we want to know if the algorithm has following restrictions:

• Has scene restrictions.
• Has light type restriction.
• Has position restriction.

No scaling will be used here, just a simple yes/no for each applicable category.
4.4 Miscellaneous
Some major disadvantages in contrast to original shadow mapping have no major category and therefore they end up here. Minor disadvantages that end up here should be ignored.

Minor disadvantages can be considered such as:
- Technique has acne, but can be simply fixed with a constant polygon offset.
- Precision error can be caused by old hardware. (16-bit depth buffer).

Major disadvantages can be considered such as:
- Dueling frusta.
- PSM singularity problem.
- Some warping techniques posses a nonuniform polygon offset problem.
- Light/Shadow bleeding, precision error caused on current generation hardware. (32-bit depth buffer).
- Restricted to only soft silhouette.
- etc.
5. Quality Measure Formula

In this section we explain how the compilation of the formula works. Each subsection (Anti-Aliasing, Resource, Simplicity and Flexibility and Miscellaneous) from Section 3 of this research produces a normalized score from zero to one. Normalized score allows parameters to be easily tuned for final results. Each subsection-score contributes towards the final score, explained in section 5.4 score from each subsection is combined into a final score.

5.1 Anti-Aliasing "A"

As in section 3.1 indicated, there are four different criteria where the anti-aliasing quality of an algorithm will be tested (projective anti-aliasing, perspective anti-aliasing, acne reduction and jitter reduction). The scaling is defined between zero and four. 0 means the quality of specific criterion is decreased compared to original shadow mapping technique. Algorithms rewarded with 4 point have the highest shadow mapping quality.

Tested aspects of anti-aliasing are:
- a Projective anti-aliasing.
- b. Perspective anti-aliasing
- c Acne reduction
- d Jitter reduction

Quality grading options with assigned score:
- 0 - Increased aliasing.
- 1 - Original shadow mapping.
- 2 - Re-approximation of sample distribution.
- 3 - Pixel correct anti-aliasing.
- 4 - Sub-pixel correct anti-aliasing.

A represents the normalized score of anti-aliasing subsections.

Formula layout (16 is the normalization factor):

\[ A = \frac{a + b + c + d}{16} \]

Example:

\[ A = \frac{4 + 2 + 1 + 3}{16} = 0.625 \]

Score approaching 1 represents high quality anti-aliasing and score approaching 0 represents the opposite.

5.3 Resource "R"

In this section we define numerical values and weightings for Performance and Memory, both reflect the algorithm ability to run in real-time.

**Performance**

With performance we are measuring the GPU compatibility and the fill-rates of each algorithm. Each technique will end up in one of the following options with assigned score points which reflects on algorithm hardware comparability and efficiency:

- 0 points - Core run-time element is not supported by the GPU rasterization, requires
emulation or complex data structure outside the GPU.

- 1 points - Core run-time element is not supported by GPU rasterization, but can can be emulated on the GPU.
- 3 points - Fully supported by GPU rasterization, but has some major inefficiencies (such as requires multiple scene renders in combination with error analysis, etc).
- 4 points - Performs on equal or almost equal level as original shadow mapping. May have some minor inefficiencies.

The amount of points achieved with the category where the algorithm appropriately belongs to is defined by $T$. There is a gap of 1 point between assigned scores 1 and 3. The purpose of this 1 point gap is to split supported and unsupported hardware comparability more far apart.

To keep the fill-rates scoring system simple, we want to answer the statement in section 3.2 "Not increasing fill-rates" with yes or no. Yes provides one point and No zero points. We define fill-rate score with $F$.

### Memory

Bandwidth is defined with $B$ and one of the following options can be selected:

- 0 points - A very high number of samples needs to be taken to achieve descent image quality.
- 1 points - A moderate number of samples needs to be taken which is inside acceptable levels, but still significantly slows down the algorithm.
- 2 points - Low, none to a very low number of samples needs to be taken which results in an insignificant performance drop.

Capacity is defined with $C$ and one of the following options can be selected:

- 0 points - Requires high amount of storage (super sampling).
- 1 points - Requires extra memory to store data, but is in acceptable range with current hardware.
- 2 points - Does not require extra memory to save extra data.

Wastage is defined with $W$ and one of the following options can be selected:

- 0 points - High memory wastage, 50% or allocated memory is occasionally not used.
- 1 points - Moderate wastage, maximum 30% of allocated memory is occasionally not used.
- 2 points - None to low wastage, maximum 10% of allocated memory is occasionally not used.

$$R = \frac{T}{4} \times 0.8 + \frac{F + B + C + W}{7} \times 0.2$$

Ability to handle a large amount of triangles is the most important resource, therefore $\frac{T}{4}$ has the highest weighting factor. The weighting is balanced to according to all techniques.

### 5.2 Simplicity and Flexibility "S"

### Simplicity

Simplicity defined by $C$, represents an indication of algorithm complexity. Very complex to implement or understand algorithms receive 0 points, a moderate complexity 1 point and easy to implement algorithms such as PCF can be presented with 2 points:

- 0 points - Algorithm consists from multiple techniques and is not intuitive to implement. It may require a custom solution to bypass hardware limitations.
1 points - The algorithm consists of multiple techniques, but is fairly straightforward to implement without any real challenges.

2 points - One additional algorithm with not higher than moderate complexity.

**Flexibility**

This section is responsible for providing a score that is able to reflect a simple representation of algorithm flexibility. \( F \) represents a normalized sum of score points achieved per each applicable criterion. Each false statement provides a single score point:

- Has scene restrictions.
- Has light type restriction.
- Has position restriction.

\[
S = \frac{C}{2} \times 0.2 + \frac{F}{3} \times 0.8
\]

Simplicity has much lower weighting factor than flexibility. Due to low flexibility the algorithm becomes less applicable for variety of situations, which is an important factor. Simplicity is less important, it depends on the experience and abilities of the developer.

**5.4 Miscellaneous "M"**

\( N \) represents the amount of extra penalty points gained by major uncategorized inefficiencies (see section 3.4). No penalty points awarded will result in one point, the more penalty points awarded the closer the final score will get to zero. Per one major inefficiency, one penalty point is awarded.

The weighting never reaches zero, it ensures that no limit on amount of penalties exists.

**5.4 Formula**

The final score is determined by the formula below. All containing variables are explained in previous sections above.

\[
f(s) = A \times R \times S \times 0.9 + M \times 0.1
\]

Anti-Aliasing, Resource and Simplicity & Flexibility are multiplied together because they are interconnected together. Each cannot become useful without another. Miscellaneous is kept apart due to unknown possible penalty points and because we still want to reward other variables.

The final score reflects the overall score the algorithm.
6. Measurement & Results

<table>
<thead>
<tr>
<th>Anti-Aliasing</th>
<th>Project AA</th>
<th>Perspective AA</th>
<th>Acne Reduction</th>
<th>Jitter Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource</th>
<th>Sub Score A</th>
<th>Sub Score R</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Performance</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Simplicity &amp; Flexibility</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Minimum</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Miscellaneous</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Minimum</th>
</tr>
</thead>
</table>

Final Score: 0.228

Table 1: Points and scores of all techniques from section 2.3.

Graph 1: Higher scores are better; lowest score is zero and maximum score is one.
Graph 2: Higher score represents a more and better use of resources. The algorithm can better handle a large number of triangles, has low fill-rates and good memory consumption.
Graph 3: High scores reflect low complexity and high flexibility of each algorithm.

Graph 4: The highest score is one, the algorithm has no miscellaneous penalties.

Graph 5: This is the overall result of all algorithms. Higher scorers represent overall good performance, flexibility and high anti-aliasing.
Table 2: Numerical average overall score value per technique category.

<table>
<thead>
<tr>
<th>Average Overall Type Score</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warping</td>
<td>0.268</td>
</tr>
<tr>
<td>Partitioning</td>
<td>0.408</td>
</tr>
<tr>
<td>Warping &amp; Partitioning</td>
<td>0.406</td>
</tr>
<tr>
<td>Irregular Sampling</td>
<td>0.247</td>
</tr>
<tr>
<td>Filtering</td>
<td>0.419</td>
</tr>
</tbody>
</table>

Graph 6: Average overall score value per technique category.
7. Discussion

The overall measurement is a rough approximation. Any future work will require to incorporate more algorithms, better weighting and perhaps also new factors to provide more precise approximation and reliable outcomes.

Table 2 indicates the overall average results per each category. This indicates that Filtering, Partitioning and Partitioning 

& Warping have the highest scores. Overall even with penalties, filtering has the highest average score. The penalties on variance shadow mapping (table 1) are caused by occasional light bleeding and requirement to save the data in separate buffer with two channels.

On other hand an interesting observation can be found in irregular sampling algorithms (Table 1). Fast, sub-pixel antialiased shadow maps produce remarkable sub-pixel shadow quality in real-time. Other irregular algorithms underperform due to their low score in Resource section (Graph 2). The reason why the algorithm by [PWCZB09] is performing so well is to thank to latest GPU development where advanced (GP)GPU programming is supported. Main parts such as software rasterization and other analysis can all be performed or accelerated by the GPU. This indicates that some older and less popular algorithms can be again reviewed for applicability with new hardware. It also indicates that other irregular techniques may have been not fairly benchmarked, if they could be accelerated with GPGPU.

Perspective Shadow Mapping has a lower overall score than regular shadow mapping. PSM is severely limited to certain scenes, a narrow band of light orientations are supported and also may increase other types of aliasing. Warping the shadow map refocuses sampling from far to near camera distance, which due to distant undersampling also increases shadow "jaggies" in distant scene areas. PSM does not solve the undersampling issue, it only compensates undersampled areas by taking away samples from oversampled areas.

Logarithmic Perspective Shadow Mapping generates very low aliasing error shadow maps. The error is minimum and can even be ignored. Unfortunately the hardware does not support logarithmic rasterization, which greatly slows down the algorithm. With logarithmic hardware support, this would be one of most popular algorithms for shadow mapping.

Cascaded Shadow Maps perform better than regular shadow mapping. In combination with perspective warping better shadow qualities are achieved, but the complexity of the algorithms increases. CSMs do not posses greatest performance due to requirement of having multiple passes (one pass for every cascade). Also it performs in certain situations better than others. Usually CSMs are used for sun light, the reason for that is the horizontal split of the shadow maps. Having the camera differently orientated to the light, CSM might provide no benefit at all. Main struggle of CSMs are vertical surfaces. Subdivided Shadow Maps provides vertical, horizontal partitioning and space warping which, in combination has a very low aliasing error.
8. Conclusion

After reviewing many algorithms, their strengths and weaknesses it remains hard to create a formula that will judge all techniques in the same manner fairly. The results are not by any definition hard evidence of a superior algorithm, they nearly express how a certain algorithm performs in a general manner within developed formula compared to original shadow mapping technique. The formula suffers from precision and weighting biasing. It needs to be further refined and improved to achieve more accurate results. Some algorithms are purely theoretical and serve as a hardware proposal, deciding how to benchmark this kind of technique is very tricky. Logarithmic Perspective Shadow Mapping is a good example of this kind of algorithm. With hardware support LogPSM will nearly surpass every other modern technique in resource consumption and the quality of shadow, without hardware support it simply cannot perform in real-time.

According to this formula in terms of generality, filtering techniques score the highest average score. Variance Shadow Mapping has the highest score compared to all other techniques. Many modern games utilize VSM in their engine, which somewhat confirms the result. Algorithms such as Cascaded Shadow Maps are mainly focused on certain type of conditions which in some terms may be unfair for general judgment. As previously suggested, the formula needs to be improved.

Also an interesting observation is made on irregular shadow mapping techniques, revealing that new hardware allows older algorithms to become available for new application. More flexible and powerful hardware opens new possibilities for older and perhaps less popular algorithms.
Reference


