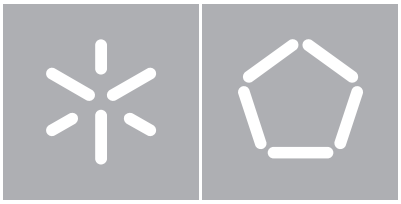
Universidade do Minho
Escola de Engenharia

Fábio da Costa Morais

Rasterização + Ray Tracing: Renderização de Sombras Sólidas

Rasterization + Ray Tracing:
Rendering of Hard Shadows

Outubro de 2015



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Fábio da Costa Morais

Rasterização + Ray Tracing: Renderização de Sombras Sólidas

Rasterization + Ray Tracing:
Rendering of Hard Shadows

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de

Professor António José Borba Ramires Fernandes

Outubro de 2015

Acknowledgements

I would like to dedicate this work to my parents for their unconditional support and faith, without it I would have never been able to complete it.

I also would like to express my gratitude to my supervisor, Professor António José Borba Ramires Fernandes, for his contribution and guidance on the research and writing of this document.

Abstract

Due to great technological advances in video cards over the last decade, several classical image-rendering algorithms have recently been adapted to run on GPUs. This made it possible for several ray tracing based global illumination techniques to perform faster and faster, achieving performance levels which are, in some cases, suitable for real-time applications. However, despite these advances, rasterization is still the most widely used technique in the 3D Computer Graphics industry for real time applications due to its efficiency generating images with reasonable visual quality. As the implementation of photorealistic techniques using ray tracing in real-time is still out of reach of today's hardware, there have been several attempts to combine rasterization and ray tracing, to obtain the best of both worlds.

This dissertation seeks to explore the benefits of an approach that combines rasterization and ray tracing and to study the feasibility of this approach using as case study the creation of hard shadows. There have been two relevant works on this particular area in the past. The first goal of these methods was to identify problematic pixels in a shadow map, i.e. pixels which according to the method could not be determined with certainty to be in shadow or light. One of such methods guarantees the correctness of the remaining pixels, whereas the other is based on a heuristic approach. The set of pixels in doubt was then sent to a ray tracer to evaluate their shadow status.

In this work we also explore an extension to one of such methods that takes into account information regarding the triangle's adjacency, thereby being able to guarantee the shadow status of more pixels based only on the data provided by the shadow map. To assert the usefulness of such approach we evaluate all three methods against a ray tracing solution, analysing the graphics pipeline, the overhead in terms of computational effort on the rasterization front, the number of pixels sent to the ray tracer, and also the visual impact of the pixels that are incorrectly assumed to be accurately classified by the shadow map.

Resumo

Devido aos grandes avanços tecnológicos em placas de vídeo durante a última década, vários algoritmos de renderização de imagem clássicos foram recentemente adaptado para rodar em GPUs. Isto tornou possível para vários métodos de ray tracing com base técnicas de iluminação global executarem mais rapidamente, atingindo níveis de desempenho que são, em alguns casos, adequado para aplicações em tempo real. No entanto, apesar desses avanços, a rasterização ainda é a técnica mais utilizada na indústria de computação gráfica 3D para aplicações em tempo real, devido à sua eficiência na geração de imagens com qualidade visual razoável. Como a aplicação de técnicas foto realistas usando ray tracing em tempo real ainda está fora do alcance de hardware de hoje, houve várias tentativas de combinar rasterização e ray tracing, para obter o melhor dos dois mundos.

Esta dissertação pretende explorar os benefícios de uma abordagem que combina rasterização e ray tracing para estudar a viabilidade desta abordagem utilizando como caso de estudo a criação de sombras sólidas. Houve duas obras relevantes nesta área em particular no passado. O primeiro objectivo destes métodos foi identificar pixéis problemáticos num shadow map, ou seja pixéis que de acordo com o método não pode ser determinado com certeza estar na sombra ou luz. Um desses métodos garante a regularidade dos restantes pixéis, enquanto a outra baseia-se numa abordagem heurística. O conjunto de pixéis em dúvida foi então enviado para um ray tracer para avaliar o seu estado de sombra.

Neste trabalho, nós também exploramos uma extensão para um desses métodos, incluindo informações sobre adjacência do triângulo, sendo assim capaz de garantir o estatuto de sombra mais pixéis com base apenas nos dados fornecidos pelo shadow map. Para averiguar a utilidade de tal abordagem, avaliamos todos os três métodos contra uma solução de ray tracing, analisando os pipeline de gráficos, a sobrecarga em termos de esforço computacional em frente da rasterização, o número de pixéis enviados para o ray tracer, e também o impacto visual dos pixéis que são incorrectamente assumidos como classificados com precisão pelo shadow map.

Contents

List of Figures

List of Tables

List of Acronyms

AABB	Axis-Aligned Bounding Box
PCF	Percentage Closer Filtering
FOV	Field Of View
BGSM	Big Geometry Shadow Map
SGSM	Small Geometry Shadow Map
NSM	Normal Shadow Mapping
CRSM	Conservative Rasterization Shadow Mapping
CRSMA	Conservative Rasterization Shadow Mapping with Adjacency
VSM	Variance Shadow Mapping
ESM	Exponential Shadow Mapping
PFL	Pixels Facing the Light
PnFL	Pixels not Facing the Light

1 Introduction

1.1 Context

Currently, image generation in Computer Graphics is divided into several types of rendering techniques, of which the most prevalent are based on rasterization and ray tracing. These techniques have different guiding principles when it comes to image generation.

The rasterization process produces images of reasonable quality with great efficiency, even considering scenarios with a large number of triangles. However, the quality of the images produced by rasterization is not comparable to the quality obtained through algorithms based on ray tracing.

Hybrid rendering, including other techniques besides rasterization and ray tracing, has been around for a long time. A famous engine that uses hybrid rendering is RenderMan by Pixar, an engine that uses an implementation of the REYES architecture (Cook, Carpenter, & Catmull, 1987), used for the creation of special effects in the film industry, most known for its role in "*Toy Story*" and "*Finding Nemo*".

For the production of the animation film "*Cars*" (Christensen, Fong, Laur, & Batali, 2006), RenderMan was extended to use ray tracing, because the characters in the film required high quality reflections, which was not possible in the rasterization system they had. After this change, Pixar was able to include other effects to film like *ambient occlusion* e *precise shadows*.

With advances in graphics hardware, ray tracing algorithms have been ported to work on the GPU to speed up image synthesis. Ray tracing engines have been created recently, such as NVIDIA's OptiX™ (NVIDIA Corporation, 2008), AMD's FireRays™ (Trudeau, 2015) , and Intel's Embree™ (Wald, Woop, Benthin, Johnson, & Ernst, 2014).

However, despite these advances the possibility of real-time full ray tracing is still far from current graphical capabilities. In this context, proposals that combine rasterization with ray tracing emerged (Beister, Ernst, & Stamminger, 2005) (Hertel, Hormann, & Westermann, 2009) (Cabeleira, 2010), and some in depth studies have been performed (Moderno, 2011). The main idea behind these works is to combine the best of both worlds, the performance of rasterization, and the details that ray tracing is able to provide. Most of the work developed in this field has been

mostly about only one aspect like shadows, while some tried to implement all possible ray tracing effects into rasterization using CPU and GPU parallel processing (Cabeleira, 2010).

When combining ray tracing and rasterization a common approach is the use of *deferred rendering*, where the rasterization process gathers the information of the scene and ray tracing uses that information to create visual effects. The rasterization process can have three purposes: to produce an initial image; collect information regarding the scenario (positions, normals, and other attributes) and identify possible areas of intervention for the subsequent steps with ray tracing.

An example of such approach is the work in (Hertel, Hormann, & Westermann, 2009), where ray tracing is used to compute the transition between light and shadow in the use of shadow maps. This rendering technique produces high quality hard shadows when compared to rasterization.

1.2 Motivation and Objectives

Both methods of producing rendered images, rasterization and ray tracing, have their benefits and flaws: rasterization is designed to produce high quality objects, using the proprieties of the object and local approximations of the light, but is incapable of processing visual effects that involve the entire environment, not having easy access to the scenes proprieties, like geometry and placing of the objects, resulting in more complex algorithms to apply little details, that add a new level a detail; ray tracing is designed to produce images with high quality illumination. Due to its implementation based on the studied behaviour of the light in the real world, obtaining visual complex effects like global illumination, reflections and refractions result in more elegant implementations. However, this comes with a heavy performance impact and high rendering times, since the light effects must take into account the entire scene, and this is not commonly supported directly in hardware graphical pipelines.

The objective of this dissertation is to study the feasibility of hybrid rendering algorithm in the context of hard shadows, combining the rasterization with ray tracing. The solutions will be analysed from a qualitative and quantitative perspective, comparing with both pure rasterization and ray tracing solutions.

Furthermore, this thesis presents an extension to the work in (Hertel, Hormann, & Westermann, 2009) using geometric adjacency information, and evaluate its usefulness considering the performance impact.

Since the implementation of this dissertation will use the ray tracing engine OptiX™ by NVIDIA, in conjunction with the University of Minho's 3D rendering engine Nau3D, another objective of this dissertation is to test and improve Nau's support with OptiX™ engine and enhancing the project's capability of rendering high quality 3D environments in real-time.

1.3 Outline

Besides this chapter, this document will explain the principal concepts used in the shadow mapping and ray tracing, as well as the ideas of conservative rasterization, relevant for the hybrid method that we'll present.

The second chapter covers the basic shadow mapping technique and highlight some of its issued, and will describe some of the techniques which are used in the methods relevant for this study.

The third chapter provides a state of the art focusing on hard shadows. Improvements to standard shadow mapping are presented as well as the hybrid methods that are the basis for this work.

Chapter 4 describes the extension using geometric adjacency information, comparing it with the original algorithm (Hertel, Hormann, & Westermann, 2009). The pipeline and its implementation are also discussed in here.

The analysis and results are presented in chapter 5, where a comparison is performed between the basic shadow mapping approach, the existing hybrid methods and the new proposed extension, considering the ray traced shadows as ground truth. The analysis covers the required pipelines, as well as the ability to determine (correctly) which pixels require the ray tracing pass.

The sixth chapter presents performance results, where a comparison of the hybrid methods is done within a specific test pipeline, where the time required by these methods to produce the necessary materials in those steps is analysed. Also, at the end of the chapter, the full execution times of the pipeline and the frame rate are compared to the pure ray tracing methods, to determine if the hybrid methods provide significant impacts in these statistics.

The final chapter will present the conclusions of this study regarding hybrid methods for the production of hard shadows and report on the feasibility of the proposed extension. Time is always an issue, and in the end we still feel that some avenues could be explored. These ideas will be presented in a section dedicated to the future work.

2 Techniques and Basic Algorithms

Shadow is the named called to an area not exposed directly to a light source, due to an obstruction of the light by an object, creating a dark silhouette of the obscuring object in the area. The area around the shadow where the light's obstruction is partial is called the penumbra. Figure 1, demonstrates this, where the umbra (Latin for shadow) is the inner region of the shadow, where the light source is completely blocked, and the penumbra is the transition region of the shadow, where the light source is partially obstructed, creating a gradient.

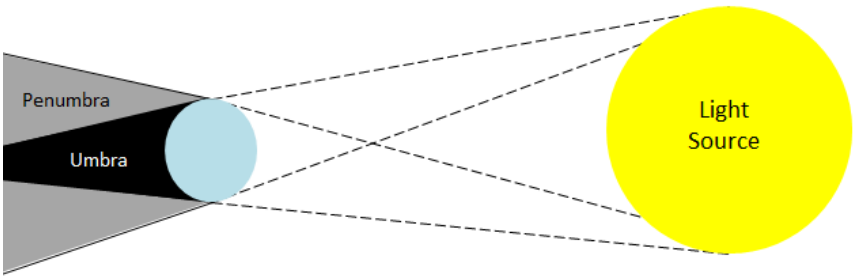


Figure 1 – A diagram showing shadow formation.

The recreation of this natural phenomenon in rendered images is extremely important, because it establishes a sense of depth, facilitating the determination of spatial relation between objects. As seen in Figure 2.

Figure 2 – Shadows provide a valuable clue for the position of the objects.

Regarding the production of shadows, there are methods that compute only the umbra. These are called hard shadow methods. All the hybrid methods on this study fall into this category (Beister, Ernst, & Stamminger, 2005; Hertel, Hormann, & Westermann, 2009). Some methods simulate the penumbra by a filtering process (Donnelly & Lauritzen, 2006; Annen, Mertens, Seidel, Flerackers, & Kautz, 2008). This results in a blurred shadow that, although not physically correct, can provide a more visually appealing shadow representation. The filtering process can introduce some artefacts in some situations as will be shown later. Ray traced shadows can provide a good approximation of

the penumbra, through extensive sampling of area lights. However, this comes with a performance penalty when compared to previous methods.

2.1 Shadow Mapping

The most popular methods in rasterization to generate projected shadow are based on shadow maps (Williams, 1978). This is a 2-pass algorithm, where the first pass will render the depth values of all the objects from the light's point of view into a 2D texture (shadow map). In the second pass of the algorithm, the scene is rendered from the camera location, and each pixel is transformed to light-space to retrieve the depth value stored in the shadow map, and compare it to the actual distance from the corresponding point to the light. If the depth value stored in the shadow map is less than the actual distance to the light, this means that there's another object occluding the point; therefore this pixel is in shadow. Otherwise the pixel is lit.

Since shadows can be determined directly by using the light's point-of-view, there is no need for auxiliary structures or information about the scene but this technique possesses severe aliasing problems, where shadows lose quality due to sampling issues (Nealen, 2002). The most common problems of this technique are Perspective Aliasing and Projection Aliasing (other problems emerge when dealing with large scale environments) (Abrantes, 2009).

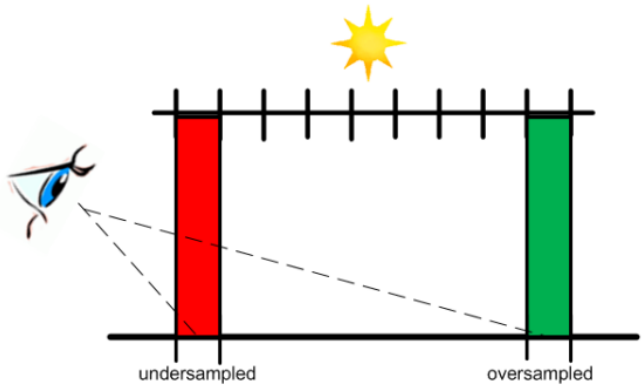


Figure 3 – While distant shadow have great detail, nearby shadows will not have sufficient resolution, resulting in pixelated shadows due to undersampling; (Abrantes, 2009).

Perspective Aliasing refers to the discrepancies between the perspectives of the eye and the light. In the eye's perspective, the objects closer to the eye are larger than distant objects, creating inconsistency in the sampling, causing the closest shadows to be less detailed and the distant

shadows get more detail than required. Figure 3 shows a diagram where the green area represents oversampling of the shadow and the red area represents undersampling of the shadow.

Projection Aliasing refers to the problem that appears when the surface of an object is almost parallel to the light's direction, making the normal of that surface perpendicular to the light direction. This causes the same depth value of the shadow map to become associated to a large number of the pixels on the surface, as shown in Figure 4, resulting in discontinuous shadows along the surface. This becomes more apparent when there's camera movement, since the shadows will flicker with each movement.

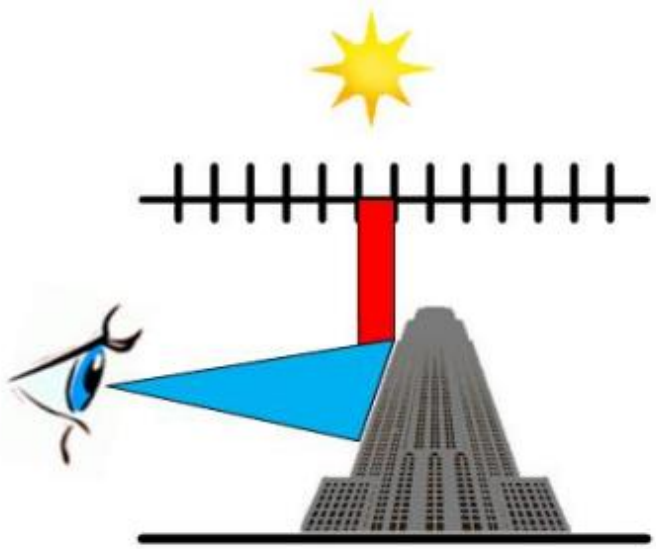


Figure 4 – The shadow map resolution for the shadow in the surface is too small, applying the same pixel (red area) of the shadow map to surface seen by the observer (blue area); (Abrantes, 2009).

2.1.1 Percentage Close Filtering

A method to reduce the aliasing problems is to take multiple samples, making multiple shadow map comparisons per pixel and averaging them together. This technique is called Percentage Closer Filtering (PCF) (Reeves, Salesin, & Cook, 1987), because it computes the percentage of the surface that is closer to the light, and therefore not in shadow. As opposed to the basic algorithm, which is only capable of binary shadow results, PCF produces blurred shadows. The most simple and efficient implementation of the PCF is to create a filter kernel with the compared depth values. When the pixel is processed, the intensity of the shadow will result in the percentage of shadowed

elements inside the kernel. This results in a smooth transition between light and shadow, known as soft shadows. Figure 5 show how the PCF works in relation to the standard shadow map.

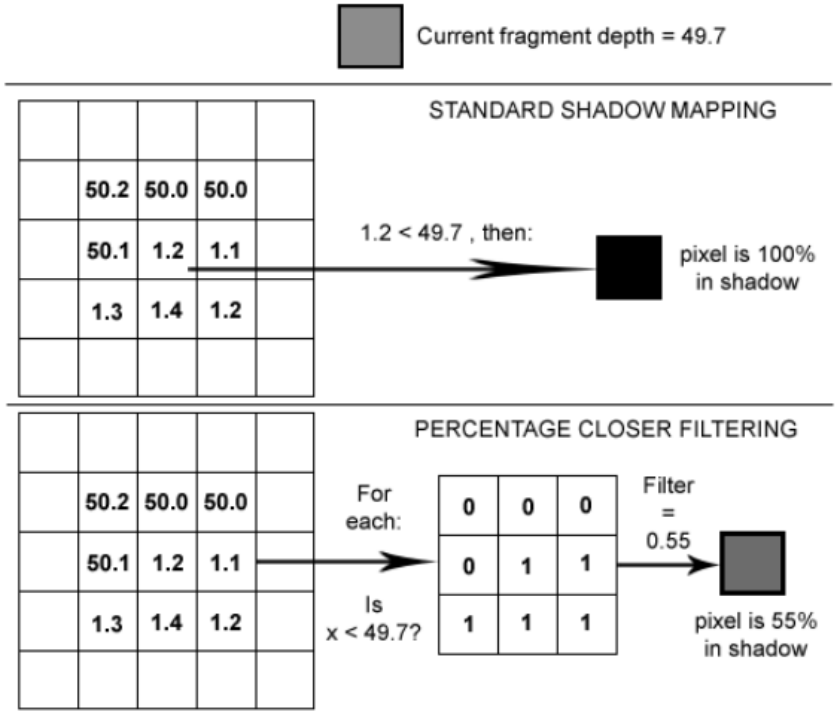


Figure 5 – Standard Shadow Map process (top) vs. Shadow Maps with Percentage Closer Filtering (bottom); (Abrantes, 2009).

Although the main usage of this technique is to fix sampling issues, it can also be used to correct pixels based on the information of the surrounding pixels, creating more precise hard shadow in lower shadow map resolutions, as described in section 3.4.1.

2.2 Conservative Rasterization

When programmable graphics hardware came to the market, research and engineering work was focused to ”port” specific algorithms, so they could run on GPUs. The argument for this was that the GPU’s performance grew faster than that of CPU’s, so in the long run GPUs would have superior performance in future. However several algorithms, like collision detection (Myszkowski, Okunev, & Kunii, 1995) , occlusion culling (Koltun, Cohen-Or, & Chrysanthou, 2001) and visibility testing for shadow acceleration (Lloyd, Wendt, Govindaraju, & Manocha, 2004), that discretize their continuous problem domain don’t achieve the same results with the use of GPU’s standard

rasterization. Figure 6-(a) shows an example of this problem, standard rasterization can't detect that the two triangles overlap geometrically, while in Figure 6-(b) conservative rasterization can detect the overlap and produce the correct result.

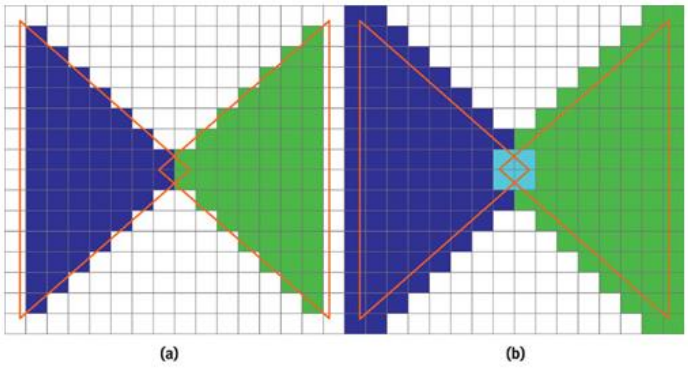


Figure 6 – A comparison of standard (a) and conservative rasterization (b); (Hasselgren, Akenine-Möller, & Ohlsson, 2005).

Although the conservative rasterization algorithm was already detailed in (Akenine-Moller & Aila, 2005), this algorithm was designed as hardware implementation, whereas the methods presented in (Hasselgren, Akenine-Möller, & Ohlsson, 2005) can be easily implemented in shaders. The solution presented in (Hasselgren, Akenine-Möller, & Ohlsson, 2005) consists in modifying the polygons before the rasterization process, where a pixel cell (the rectangular region around a pixel in the pixel grid) moves along the border, applying one of the following transformations, depending on the approach taken to the rasterization process:

- An overestimated conservative rasterization (Figure 7-a), all the pixels caught in the pixel cell are added, dilating the edge;
- An underestimated conservative rasterization (Figure 7-b), all the pixels caught in the pixel cell are erased, eroding the edge.

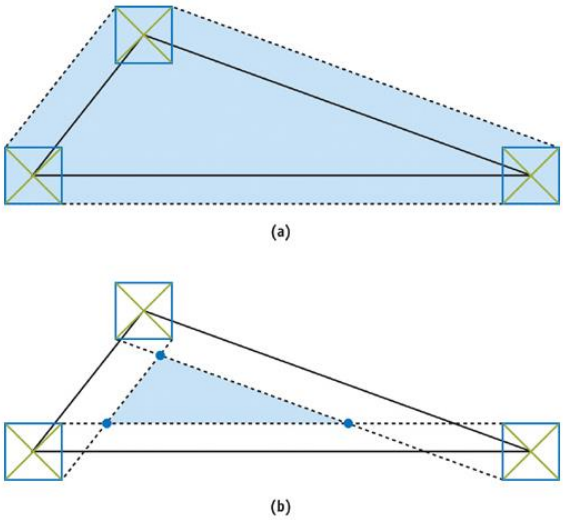


Figure 7 – (a): overestimated conservative rasterization; (b): underestimated conservative rasterization; (Hasselgren, Akenine-Möller, & Ohlsson, 2005).

Although (Hasselgren, Akenine-Möller, & Ohlsson, 2005) presents two implementations of the algorithm; this work will focus on the second implementation, since the first implementation requires multiple output vertices for each input vertex to calculate the optimal bounding box, the second implementation obtains the bounding polygon by intersecting the bounding triangle with an Axis-Aligned Bounding Box (AABB), as showed in Figure 8.

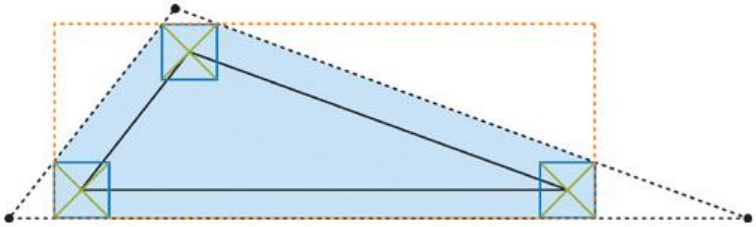


Figure 8 – The Optimal Bounding Polygon for a given triangle; (Hasselgren, Akenine-Möller, & Ohlsson, 2005).

In order to dilate/erode the edges of a triangle, the vertex information must be available, so the implementation requires a geometry shader. The calculations shown below are done in screen space. This is to reduce the complexity of the calculations to dilate/erode, since x and y coordinates represent the position of the triangle in the screen, and the z coordinate will indicate if the triangle is visible in the screen.

2.2.1 Implementation

Given a triangle composed of vertices v_0, v_2 and v_4, the first step of the algorithm is to calculate the edges of the original triangle in screen space.

$$\begin{aligned} e_0 &= v_2 - v_0 \\ e_1 &= v_4 - v_2 \\ e_2 &= v_0 - v_4 \end{aligned} \tag{1}$$

After all of the edges are calculated, we must calculate the perpendicular vector for each edge (pe_N), as if they were lines.

$$pe_{\mathrm{N}} = |(-e_N.x), (e_N.y)| \tag{2}$$

These perpendicular edges represent the directions in which the triangle will be eroded or dilated from, as showed in the diagrams of Figure 9, the orange triangle represents the original triangle and the blue triangle represents the modified triangle.

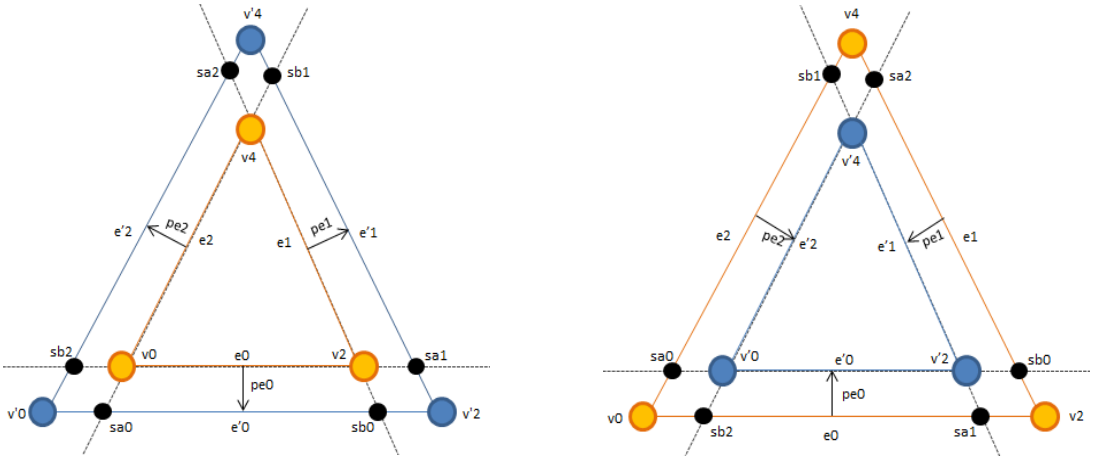


Figure 9 – Transformation of the original triangle (orange) to the new triangle (blue); (Left) Dilation; (Right) Erosion.

But the calculations done now only apply to the edges, in the shader the transformation applies to the vertex, and since each vertex belongs to two edges, we must calculate the shifted positions of each vertex following the direction of pe that that vertex belongs to, the following equations show an example for vertex v_0, but these calculations apply for all vertices.

$$sa_0 = v_0 \pm pe_0 * pD$$
$$sb_2 = v_0 \pm pe_2 * pD \quad (3)$$

Where pD represents diagonal length of the pixel cell. The sa_0 is the shifted position of vertex v_0 following the direction pe_0 and sb_2 is the shifted position of $v0$ following the direction pe_2. The $\pm$ means that functions changes depending on the transformation, for erosion the $+$ sign is used and for dilation the $-$ sign is used.

After calculating all shifted positions of the vertex, we can calculate the intersections of the new edges to form the new triangle form by the vertices v'_0, v'_2 and v'_4. The following section of code shows line intersection formula using four points.

```
vec3 lineLineIntersection(vec3 s1, vec3 e1, vec3 s2, vec3 e2,
                          vec3 n, vec3 p) {
          vec3 result;

          float a1 = e1.y - s1.y;
          float b1 = s1.x - e1.x;
          float c1 = a1 * s1.x + b1 * s1.y;

          float a2 = e2.y - s2.y;
          float b2 = s2.x - e2.x;
          float c2 = a2 * s2.x + b2 * s2.y;

          float delta = a1*b2 - a2*b1;
          result = vec3(b2*c1 - b1*c2, a1*c2 - a2*c1, 0) / delta;

          // now result is the 2D coordinate in x,y
          // we still need to compute z
          // Ax + By + Cz + D = 0 => z = -(Ax + By + D) / C
          if (n.z != 0) {
            float d = -dot(n, p);
            result.z = -(n.x * result.x + n.y * result.y + d) /  n.z;
          }
          else {// this would only happen with a triangle
                    //which is perpendicular to the screen
            result.z = 0;
          }
          return result;

}
```

There might be cases where the triangle is not visible in screen space. This is because the triangle is perpendicular to the camera. Therefore, if the z normal component is 0, this process is skipped.

2.3 Ray Traced Shadows

Considering point or directional lights, shadows are easily created using ray tracing based algorithms (Glassner, 1989). For every pixel in the scene, a ray is sent from its world position to the light source.

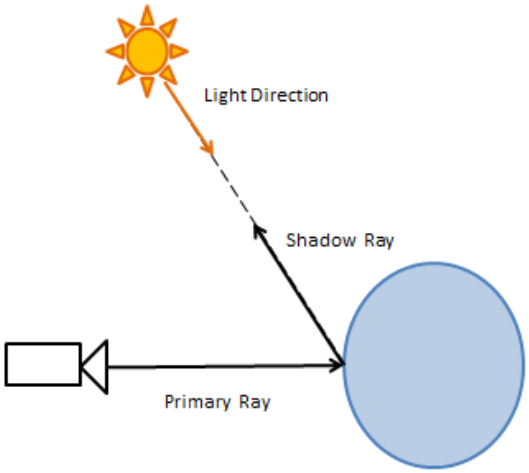


Figure 10 – Diagram for shadow ray in ray tracing.

When a ray from the camera (primary ray) intersects an object, a new ray is created in the intersected point (shadow ray). This ray is sent in the direction of the light source. If the shadow ray intersects any object during its trajectory, then the intersected point is in shadow, if not the intersected point is being illuminated. Figure 10 illustrates the process to obtain and image using ray tracing.

This process of creating projected shadows is simple to implement and achieves great detail in the shadow's contours.

When considering area light sources, the reproduction of shadows with needs to determine the percentage of the light source visible for each point in the scenery. That task can be approximated by using multiple samples of the light. This requires several shadow rays in different directions within the light source. The percentage of rays that reaches the light provides give an approximation of the area of visible light. This allows the creation of penumbra effects, as seen in Figure 11.

In general, larger number of samples provides better results in the production of penumbra and umbra regions, however, as a direct consequence, the render time of the image will be larger.

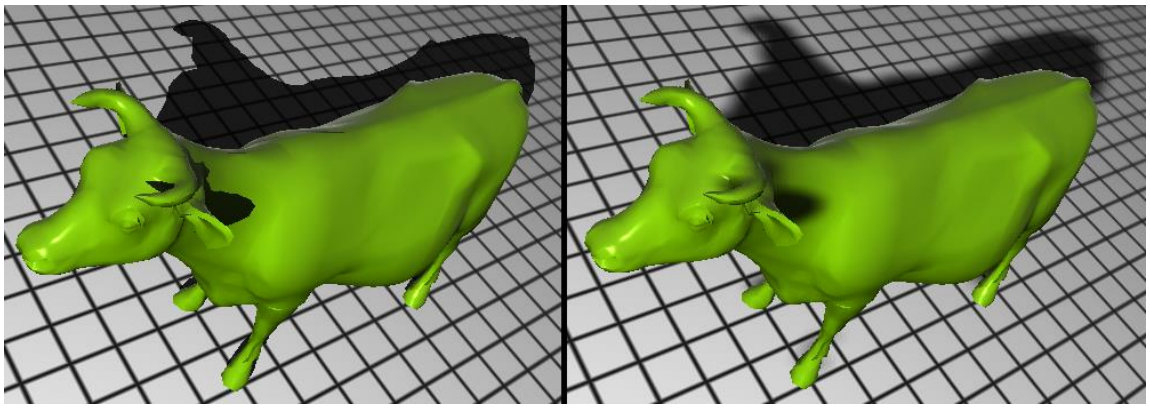


Figure 11 – A comparison of hard shadows (left) and soft shadows (right); (NVIDIA Corporation, 2008).

3 State of the Art

3.1 Trapezoidal Shadow Mapping

Trapezoidal Shadow Mapping (Martin & Tiow-Seng, Anti-aliasing and Continuity with Trapezoidal Shadow Maps, 2004) is a new approach of calculating shadows using trapezoidal shadow maps which are derived from trapezoidal approximations of the eye's frustum as seen from the light. It addresses the resolution problems of the standard shadow mapping, resulting in enhanced shadow map resolution for both static and dynamic objects from near and far, with no constraint on the relative positions and motions of the eye and the light. The approach is efficient as only the eight corners of the eye's frustum plus the centres of the near and far plane, rather the scene, are needed to compute a good trapezoidal approximation, thus it scales well to large scenes. Figure 12 shows an example of this approach.

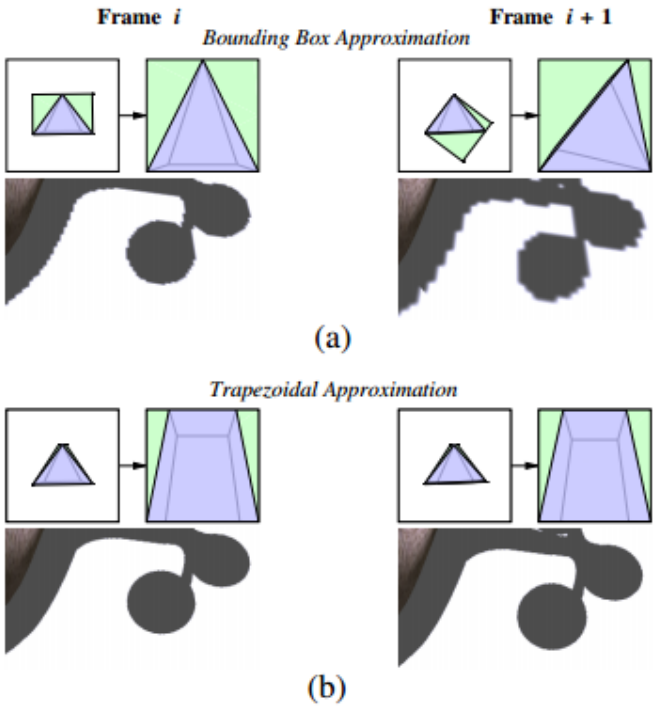


Figure 12 –Bounding Box Approximation (a) vs. Trapezoidal Approximation (b); (Martin & Tiow-Seng, Anti-aliasing and Continuity with Trapezoidal Shadow Maps, 2004)

3.1.1 Increasing Shadow Map Resolution

A shadow map can viewed as a simple construct that contains two types of positions of the eye's frustum: the positions within and the positions outside the frustum. It is clear that only the former is useful to determine whether pixels are in shadow or not. Increasing the shadow map resolution is to minimize the positions outside the frustum, which are collectively termed as wastage. In other words, a good way to address the resolution problem is to better utilize the shadow map for the area within the eye frustum in Figure 13 (denoted as E). This requires the calculation of an additional normalization matrix N to transform the post-perspective space of the light to an N-space in general, where N refers to the trapezoidal space and bounding box space, respectively in Figure 13. The shadow map is constructed by transforming the pixel into the N-space, rather than into the post-perspective space of the light, for depth comparison.

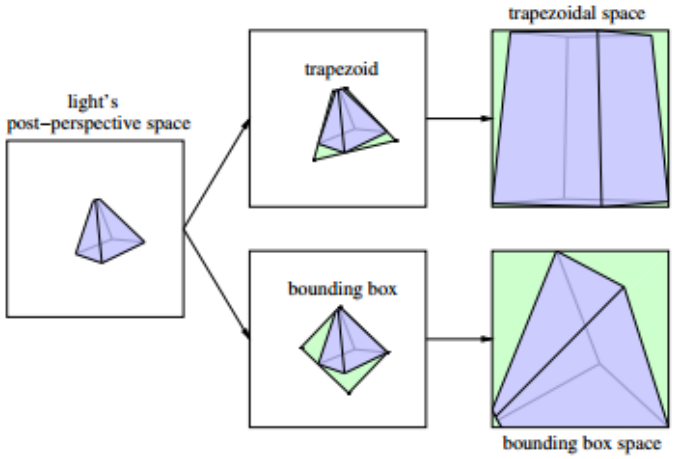


Figure 13 - Left: The eye frustum as seen from the light; Middle: example for trapezoidal (top) and bounding box (bottom) approximations; Right: Wastage obtained in the shadow map with approximations; (Martin & Tiow-Seng, Anti-aliasing and Continuity with Trapezoidal Shadow Maps, 2004).

3.1.2 Constructing the Trapezoid

Since the trapezoid is recognized to be most similar shape to E, the aim of (Martin & Tiow-Seng, Anti-aliasing and Continuity with Trapezoidal Shadow Maps, 2004) work is to construct a trapezoid

T to approximate E, with the constraint that each such consecutive approximation results in a smooth transition of the shadow map resolution.

The first step is to find two parallel lines in post-perspective space (L) of the light to contain the base and the top edge of the required trapezoid (T). The aim is to choose the parallel lines such that there is a smooth transition when the eye moves between frames. The algorithm to determine those lines follows these steps:

1. Transform the eye's frustum into L of the light to obtain E;
2. Compute the central line l, which passes through the centres of the near and far plane of E;
3. Calculate the 2D convex hull of E (which can contain up to six vertices on its boundary);
4. Calculate the top line (l_t) that is orthogonal to l and touches the boundary of the convex hull of E. It intersects l at a point closer to the center of the near plane and far plane of E.
5. Calculate the base line (l_b) which is parallel to (and different from) the top line (l_t) (i.e., orthogonal to l too) and touches the boundary of the convex hull of E.

The calculation of the centre line l is important because it allows recalculating the l_t and l_b accordingly to the eye's movement, creating a smooth transition between frames.

3.1.3 Side Lines

The following step is to calculate the side lines of the trapezoid. Assuming the eye is more interested in objects and their shadows within the first δ distance from the near plane. That is, the focus region of the eye is the eye's frustum truncated at δ distance from the near plane. Let p be a point of δ distance away from the near plane with its corresponding point p_L lying on l in L, as seen in Figure 14. Let the distance of p_L from the top line be δ'. The trapezoid has to contain E, so that N_t maps p_L to some point in T.

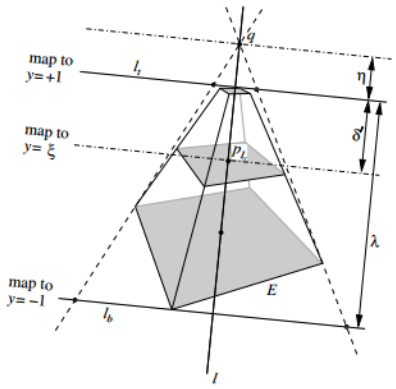


Figure 14 – A 1D homogenous perspective projection problem to compute q; (Martin & Tiow-Seng, Anti-aliasing and Continuity with Trapezoidal Shadow Maps, 2004).

To do this, we must calculate a transformation matrix N_T which maps the four corners of the trapezoid (t_0,t_1,t_2 and t_3) to the front of the unit cube. This can be done with the following constraints:

$$\begin{aligned}(-1,-1,1,1)^T &= N_T * t_0\\ (+1,-1,1,1)^T &= N_T * t_1\\ (+1,+1,1,1)^T &= N_T * t_2\\ (-1,+1,1,1)^T &= N_T * t_3\end{aligned} \tag{4}$$

A straightforward way to determine N_T with these restrictions is to apply rotation, translation, shearing, scaling, and normalization operations to the trapezoid to map it to the front side of the unit cube. This is achieved by calculating the eight matrices: T_1, R, T_2, H, S_1, N, T_3 and S_2.

As the first step, T_1 transforms the centre of the top edge to the origin (Figure 15-(a)).

$$u = \frac{(t_2 + t_3)}{2}$$

$$T_1 = \begin{bmatrix}1 & 0 & 0 & -x_u\\ 0 & 1 & 0 & -y_u\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{5}$$

Then, the trapezoid T is rotated by applying R around the origin in such a way, that the top edge is collinear with the x-axis (Figure 15-(b)):

$$u = \frac{(t_2 - t_3)}{|t_2 - t_3|}$$

$$R = \begin{vmatrix} x_u & y_u & 0 & 0 \\ y_u & -x_u & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (6)$$

After the rotation, the intersection i of the tow side lines containing the two side edges (t_0, t_3) and (t_1, t_2) is transformed, by applying T_2, to the origin (Figure 15-(c)):

$$u = R * T_1 * i$$

$$T_2 = \begin{vmatrix} 1 & 0 & 0 & -x_u \\ 0 & 1 & 0 & -y_u \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (7)$$

As a next step, the trapezoid has to be sheared with H, so that it is symmetrical to the y-axis (Figure 15-(d)), i.e. that the line passing through the centre of the bottom edge and centre of the top edge is collinear with the y-axis:

$$u = \frac{T_2 * R * T_1 * (t_2 + t_3)}{2}$$

$$H = \begin{vmatrix} 1 & -x_u/y_u & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (8)$$

Now, the trapezoid is scaled by applying S_1(Figure 15-(e)), so that the angle between the two side lines containing the two side edges (t_0, t_3) and (t_1, t_2) is 90 degrees, and so that the distance between the top edge and the x-axis is 1:

$$u = H * T_2 * R * T_1 * t_2$$

$$S_1 = \begin{vmatrix} 1/x_u & 0 & 0 & 0 \\ 0 & 1/y_u & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (9)$$

The following transformation N (Figure 15-(f)) transforms the trapezoid to a rectangle:

$$N = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{vmatrix} \tag{10}$$

Then, the rectangle is translated along the y-axis until its centre is coincident with the origin. This is done by applying T_3 (Figure 15-(g)). After this transformation the rectangle is symmetrical to the x-axis as well:

$$u = N * S_1 * H * T_2 * R * T_1 * t_0$$
$$v = N * S_1 * H * T_2 * R * T_1 * t_2$$

$$T_3 = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\left(\frac{y_u}{w_u} + \frac{y_v}{w_V}\right)/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{11}$$

As a last step the rectangle has to be scaled with S_2 (Figure 15-(h)) along the y-axis so that it covers the front side of the unit cube:

$$u = T_3 * N * S_1 * H * T_2 * R * T_1 * t\_0$$

$$S_2 = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & -w_u/y_u & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{12}$$

Now the trapezoidal transformation N_T (Figure 15-(i)) can be computed as follows:

$$N_T = S_2 * T_3 * N * S_1 * H * T_2 * R * T_1 \tag{13}$$

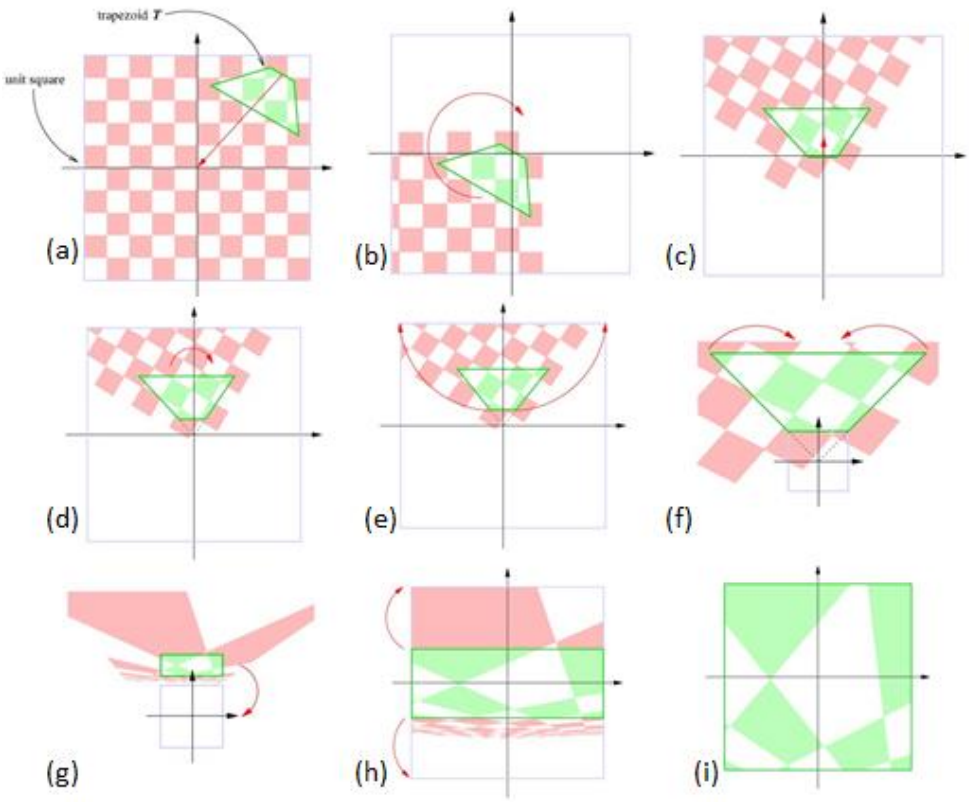


Figure 15 – TSM transformations matrixes: (a) T_1; (b) R; (c) T_2; (d) H; (e) S_1; (f) N; (g) T_3; (i) The final matrix N_T; (Martin & Tiow-Seng, Trapezoidal Shadow Maps (TSM) - Recipe, 2011).

3.1.4 Focus Region on Shadow Maps

Figure 16 shows one of the problems of the trapezoidal transformation N_t to T, given a trapezoid containing four triangles, like shown in (a). N_t has the effect of stretching the top edge of the into a unit length. In this case, the top edge is relatively short compared to the base edge, and therefore the stretching results in pushing all the showed triangles towards the bottom of the unit square as showed in (b). This means that the region near the top edge (in other words, close to the near plane) eventually occupies a major part of the shadow map, which results in over-sampling of objects near the eye, sacrificing resolution of the other objects (such as the second to fourth triangles from the top show).

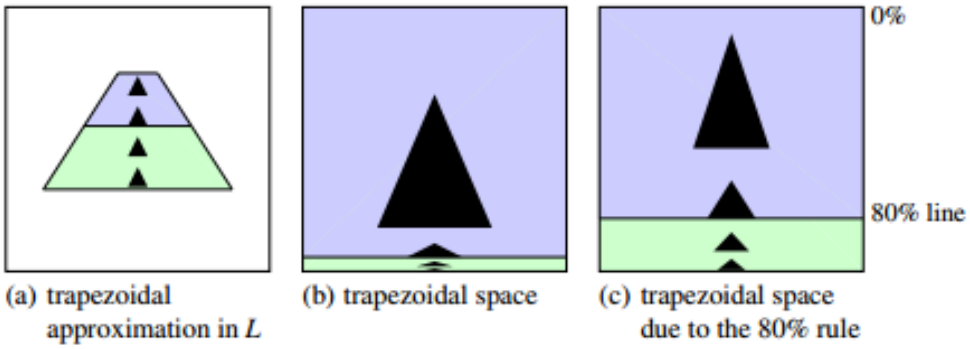


Figure 16 - For the trapezoid in (a), its corresponding T is shown in (b). In this case, we obtain an over-sampling for a small region of E. (c) For a different trapezoid computed with the 80% rule (having the same top and base lines), its trapezoidal transformation maps the focus region (the upper part of the trapezoid) to within the first 80% in the shadow map; (Martin & Tiow-Seng, Anti-aliasing and Continuity with Trapezoidal Shadow Maps, 2004).

The 80% rule allows to all p_L with the distance δ from the top line, when applied the trapezoidal transformation N_t, to be mapped to some point within the 80% line in T. This increases the resolution of the far objects, while maintaining a high detail for objects near the eye.

3.1.5 Shadow Rendering

After calculating the trapezoidal transformation matrix N_t the rendering steps for the shadow mapping process must now envelop the shadow map with the trapezoid. In order to due this they must apply the following transformations in the vertex shader:

$$\text{vertexPosition} = \text{N}_\text{T} * PVM * position \quad (14)$$

After this alteration to the vertex positions, the next steps follow the same process as the regular shadow mapping algorithm.

3.1.6 Conclusion

Trapezoidal Shadow Maps shows that it is practical and maps well to graphics hardware. It is a reasonable heuristic to generate shadow maps of good resolution, but the issues on over-sampling and under-sampling remain for various situations such as in the duelling frusta case where the trapezoidal approximation does not have any particular advantage over other approximations.

3.2 Parallel Split Shadow Maps

Parallel split shadow maps (Zhang, Sun, Xu, & Lun, 2006) is a method to reduce the amount of aliasing error in shadow maps by producing an optimized distribution of shadow map texels. In large-scale environments the shadow map might not have enough resolution in order to create detailed shadows in the scene. PSSM splits the shadow maps in continuous discrete layers, in order to maintain the detail despite the distance of camera, as shown in Figure 17.

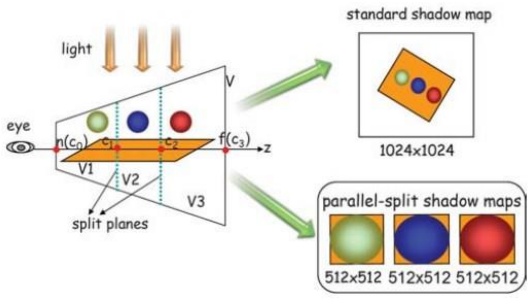


Figure 17 - Split the view frustum into three parts, and shadow maps with the same resolution are generated for the split parts; (Zhang, Sun, Xu, & Lun, 2006).

The processing steps of the PSSMs scheme are outlined in the following:

1. Split the view frustum into multiple depth parts;
2. Split the light's frustum into multiple smaller ones, each of which covers one split part also the objects potentially casting shadows into the part;
3. Render a shadow map for each split part;
4. Render scene shadows for whole scene.

3.2.1 View Frustum Split

The "splitting" is basically dividing the view frustum in to smaller continuous planes, to which a shadow map will be attributed to each of those planes. These planes are continuous because the "split" occurs according certain intervals along the z axis. Figure 17 shows an example of m split planes along the z axis.

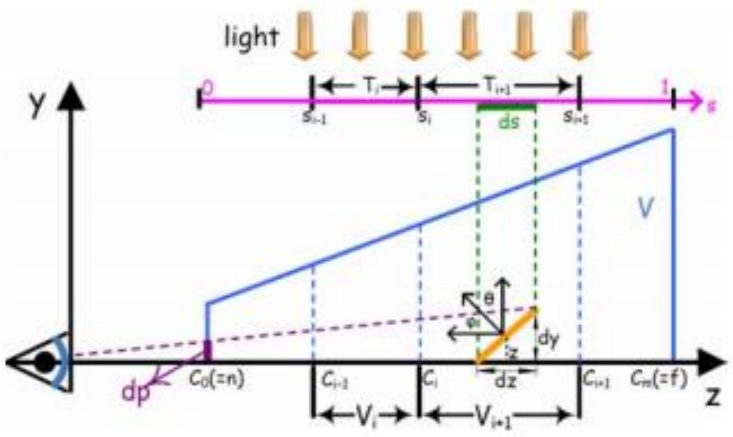


Figure 18 - Along the z axis the view frustum is split into parts by using the split planes at $\{C\_i \mid 0 \le i \le m\}$; (Zhang, Sun, Xu, & Lun, 2006).

In Figure 18, the light beams through a texel with the size $ds \times ds$ in normalized texture space (i.e $s \in [0,1]$) falls on a surface with the length dz in world space. The size of the view beams dp on the screen projected form the surface is approximately ndy/z. φ and θ denote the angles between the surface normal and vector to the screen and the shadow map plane respectively.

Shadow map under-sampling occurs when dp is larger than the pixel size of the screen, this can happen when perspective aliasing (dz/zds) or projection aliasing ($\cos\varphi / \cos\theta$) becomes large. Projection aliasing usually happens when the light's direction is almost parallel to the surface. Since

projections aliasing is heavily related to the scene's geometry details, the only solution to the problem is to increase the sampling density, inevitably increasing the scene analysis time. On the other hand, perspective aliasing comes from the perspective foreshortening effect, which can be reduced by applying a global transformation to warp the shadow map texels.

(Zhang, Sun, Xu, & Lun, 2006) work presents three split schemes (Figure 19) in order to deal with different distribution of perspective aliasing errors:

- Logarithmic split scheme;
- Uniform split scheme;
- Practical split scheme.

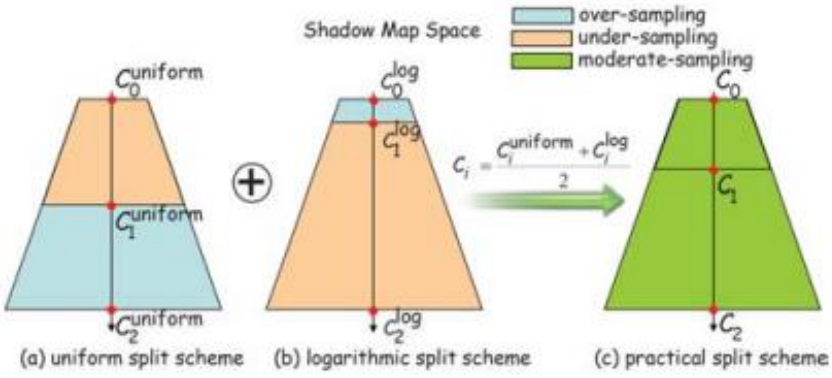


Figure 19 - Different types of spilt schemes; (Zhang, Sun, Xu, & Lun, 2006).

3.2.2 Uniform Split Scheme

The simplest split scheme is to place the split planes uniformly along the z axis:

$$C_i^{uniform} = n + (f - n)i/m \tag{15}$$

This split scheme results in under-sampling at the points near the view and over-sampling in points further from the view. In contrast to the logarithmic split scheme, uniform split scheme results in the theoretically worst aliasing distribution.

3.2.3 Logarithmic Split Scheme

Logarithmic split scheme is based on optimal distribution of perspective errors, which theoretically evens distribution of perspective aliasing errors over the whole depth range. This scheme assumes that the shadow map accurately covers the view frustum and no piece of the resolution is wasted on invisible parts of the scene.

$$C_i^{log} = n(f/n)^{i/m} \tag{16}$$

The main drawback of this split scheme is that the lengths of split parts near the viewer are too small, so few objects can be included in these split parts. This is because of the main assumption that the shadow map covers the view frustum, which requires that every $z \in [n, f]$ must be mapped to a unique $s \in [0,1]$ in the normalized texture space, but this can't be satisfied in practice, resulting in under-sampling for parts further from the viewer and over-sampling for parts nearer the viewer.

3.2.4 Practical Split Scheme

Practical split scheme consists in combining the previous two split schemes to produce a moderate result between the theoretically optimal and worst sampling densities in the extreme cases, the objects near the viewer and the objects further form the viewer.

In Figure 19, practical split scheme produces moderate sampling for both near and far split parts. Combining equations 14 and 15, the split positions C_i can be calculated by the following equation

$$C_i = \frac{\left(C_i^{log} + C_i^{uniform}\right)}{2} + \delta_{bias}, \forall 0 \leq i \leq m \tag{17}$$

The variable δ_{bias} is a non-negative bias that can be used to accurately adjust clip positions (Tadamura, Qin, Jiao, & Nakamae, 1999), if necessary for the application.

3.2.5 PSSMs and Scene Rendering

After the lights frustum is split into W_i, each split part V_i is rendered to a shadow map T_i in the W_i space. These shadow maps can have a fixed size, which is helpful for the uniform and practical splits.

With the PSSMs generated in previous step, shadows shadow effects can be now synthesized into the virtual scene. Like standard shadow mapping, each pixel should be transformed into the light space when determining if the pixel is shadowed or not. The differences here are:

1. The correct shadow map T_i must be selected;
2. The pixel has to be transformed into W_i rather W.

After this, for each rasterized fragment, the sampling must come from the appropriate shadow map based on the depth value of the fragment. Since the coordinates are measured in clip space, C_i is transformed to C_i^{clip} below:

$$C_i^{clip} = \frac{f}{f-n}\left(n - \frac{1}{C_i}\right) \in [0,1] \tag{18}$$

Then, the pixel in the fragment buffer with the depth value z^{clip}, if the $z^{clip} \in [C_{index-1}^{clip}, C_{index}^{clip}]$ the shadow map T_{index} is selected. Consequently, this fragment is transformed into the split light's frustum W_{index} for the depth comparison.

3.2.6 Conclusion

While parallel-split shadow maps is an intuitive and simple implementation of shadow rendering, and using the practical split scheme, the quality of the shadows becomes superior to standard shadow map, without requiring a scene analysis. In fact, PSSM are the only feasible way to deal with large scenarios.

3.3 Other Shadow Mapping algorithms

Although this work is focused on the production of hard shadows, some soft shadow references are included in here for completeness.

3.3.1 Variance Shadow Mapping

Variance Shadow Mapping (Donnelly & Lauritzen, 2006) is a technique that calculates, besides the usual depth value, the depth-squared values. These values will then be used to calculate the probability of each point being lit or not. But due to the fact that the lower bound of brightness is an approximate value derived from using only one single occluding object, if a scene has a high depth complexity, there might be light leaking artefacts (areas appearing lit instead of shadowed), as shown by Figure 20. In most cases this issue can be dealt with using a shadow threshold.

Figure 20 – Variance Shadow Map light leaking example; (Donnelly & Lauritzen, 2006).

3.3.2 Exponential Shadow Mapping

Exponential Shadow Mapping (Annen, Mertens, Seidel, Flerackers, & Kautz, 2008) is a technique that allows an efficient (pre)-filtering to the shadow by using a single-term approximation with a simple exponential. The shadow mapping process only stores the nearest surface it encounters during the creation of the shadow map, projecting shadow on points that are behind the stored depth sample. With this assumption, the exponential behaves like a step function.

Even thought for many pixels the assumption works, for large kernel sizes, the less likely that artefacts can be avoided, as shown by Figure 21. This makes exponential shadow mapping's performance highly dependent on the scene and kernel size. Again thresholding helps to provide a less incorrect result.

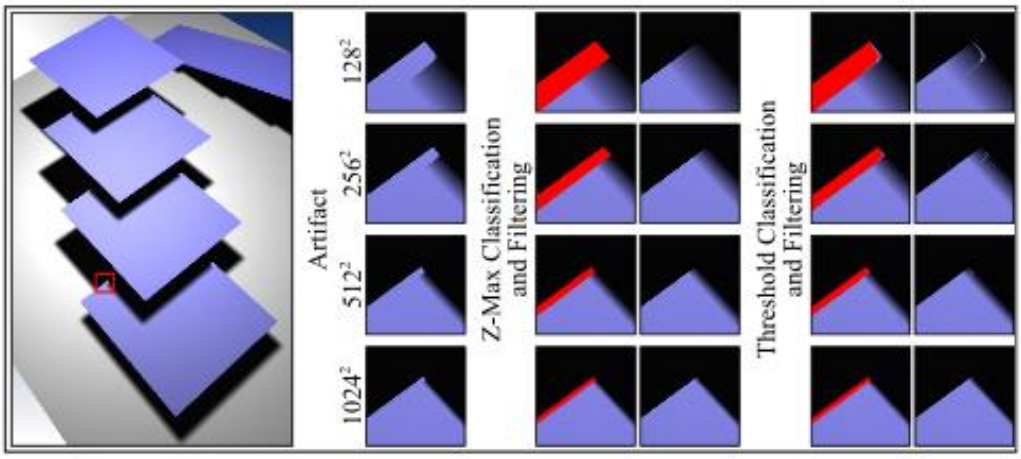


Figure 21 – Exponential Shadow Mapping artefacts; (Annen, Mertens, Seidel, Flerackers, & Kautz, 2008).

3.4 Hybrid Methods

There are been developments to combine ray traced shadow effects with rasterization techniques the most common methods use shadow mapping, due to its very efficient algorithm performance, to perform a first pass of the scene, indicating the areas in shadow with the shadow map, followed by a selective ray tracing algorithm to determine the shadow of the pixel that have a unreliable shadow status. This greatly reduces the number of ray being cast and intersection calculations, going better results when compared to shadow maps initial results, with reduced performance times when compared to pure ray tracing solution.

3.4.1 Hybrid GPU-CPU Renderer

The Hybrid GPU-CPU Renderer (Beister, Ernst, & Stamminger, 2005) presented an algorithm to render shadows by mixing shadow mapping and ray tracing. The algorithm creates a shadow map with a bilinear PCF. Then for each pixel in the interpolated result and if the result is 0 or 1 then the four surrounding pixels will agree and the pixel will be lit (1) or shadowed (0). If the result is between 0 and 1, then the four surrounding pixels aren't in agreement the state of the pixel, so the pixel is marked, as showed in Figure 22. The ray tracer will then calculate the shadowing for each uncertain pixel marked in the shadow map.

This algorithm is also adaptable to the type of light source used in the scenery. If the light source is a point light, the method describe before is used. But if the light source is an area light, the light source will be replaced by eight point lights, one in the centre of the area and the other seven will surround this area. In this case, the agreement will be done using the eight shadow maps, the results of this can be seen in Figure 23.

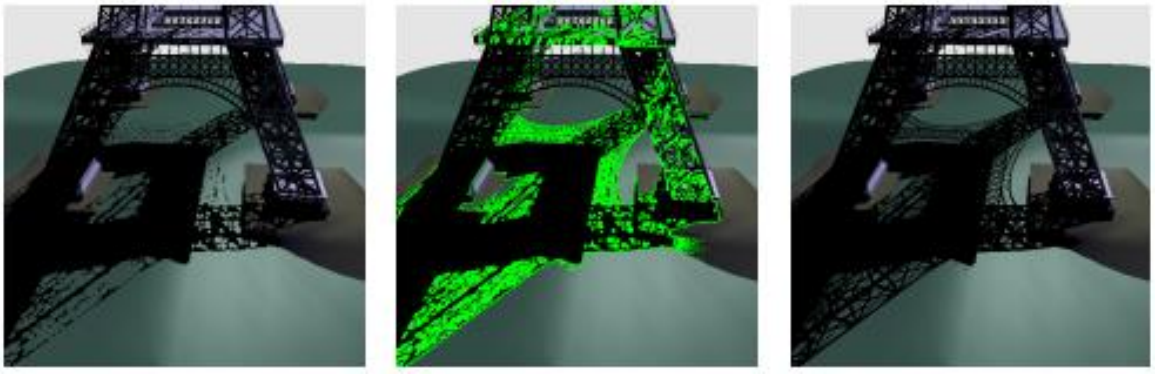


Figure 22 – Shadow Demonstration. Left: Pixels that agreed to be shadowed; Middle: Green pixels indicate the marked pixels for the ray-tracer; Right: Shadows computed for the marked pixels by the ray-tracer; (Beister, Ernst, & Stamminger, 2005).

This algorithm is robust if the shadow map resolution is adequate to the tessellation of the scene, but this is difficult because the scene has to be carefully modeled using a constant tessellation parameter for all objects. Errors may still occur if the four pixels are in agreement are incorrect, with four pixels indicated that a point is lit, yet the point should be in shadow.

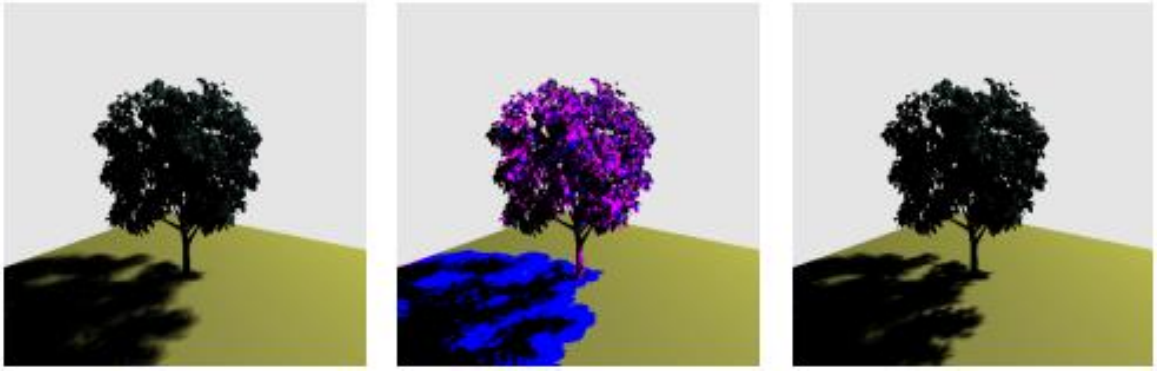


Figure 23 – Soft Shadow Demonstration. Left: Soft shadows by the eight shadow maps; Middle: Blue pixels marked for the ray-tracer; Right: Soft shadows computed by the ray-tracer; (Beister, Ernst, & Stamminger, 2005).

3.4.2 Hybrid GPU Rendering Pipeline for Alias-Free Hard Shadows

The Hybrid GPU Rendering Pipeline for Alias-Free Hard Shadows (Hertel, Hormann, & Westermann, 2009) is used to create alias-free shadows. It creates a conservative rasterization shadow map (CRSM) that similarly to the normal shadow map, but in this case a triangle will be saved in a pixel if it overlaps said pixel in any plane, not only in the centre. This is done as shown in chapter 2.

The CSM consists of a two layer texture where one layer contains all the expanded triangles in the scene (BGSM), and the other layer contains all of the shrunken triangles in the scene (SGSM). In the scene rendering step, the pixel will be analysed using both layers in the CRSM. If both layers are in agreement with the state of the pixel, the pixel will be lit or shadowed in the final image. If the layers disagree on the state of the pixel, the pixel will be classified as "uncertain". Afterwards, for each "uncertain" pixel discover is sent to a ray tracer to verify the state of the pixel. This ray-tracer will use the information of the triangle saved by the pixel and a kD-tree in order to speed up intersection tests. The information of the depth at which the triangle is found will allow for the ray-tracer to only start testing for intersections from there, as there should be no other triangle between this point and the light source.

As can be seen in Figure 24, there are many areas classified as "uncertain" that commonly produce correct results using shadow maps, namely the triangle junctions for triangles in light. The performance of this algorithm is highly dependent on the geometry tessellation hence for highly tessellated models a large number of light rays will be required.

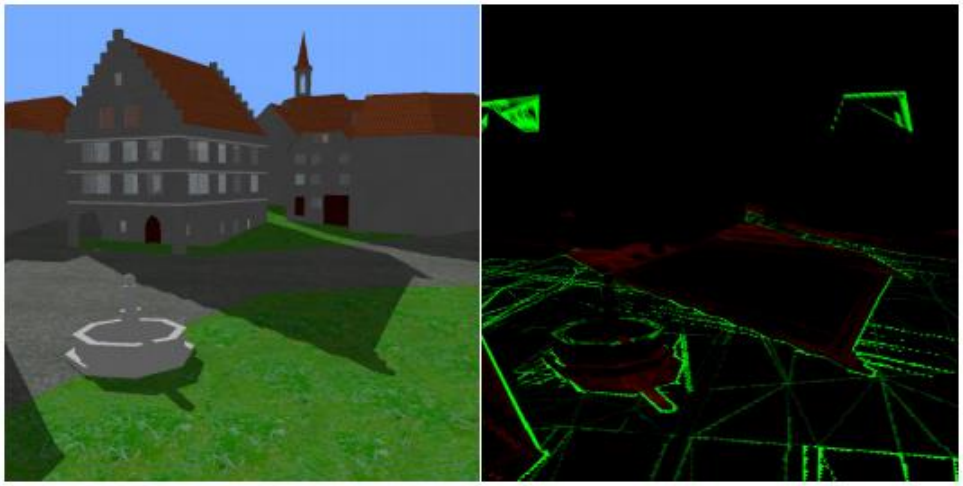


Figure 24 – Example of the "uncertain" areas (green) when using the Hybrid GPU Rendering Pipeline for Alias-Free Hard Shadows; (Hertel, Hormann, & Westermann, 2009).

3.4.3 Conclusion

The hybrid methods demonstrated here use a variety of shadow techniques to detect problems in the shadow map texture so that the ray tracer segment of the process can correct them. However, the shadow mapping still produces visual errors that cannot be so easily detected. The following chapters demonstrate that even with detection methods present before, the final image still possess errors in the image.

The algorithm presented (Hertel, Hormann, & Westermann, 2009) also raises a significant point of analyses: the use of adjacency information. Many of these "uncertain" pixels are located in the connections of triangles in the models, as showed by Figure 24, which causes a significant amount of pixels to be sent to the ray tracer to correct. The next following chapter will present an alternative algorithm that shrinks the triangles and maintains the connections of the triangles, which will be analysed alongside the original version.

4 Conservative Rasterization Shadow Mapping with Adjacency

In the work of (Hertel, Hormann, & Westermann, 2009), the shadow map is created with two layers. One layer views the triangles of the scene larger than normal, therefore each triangle is expanded a certain pixel size λ, which results in the big geometry shadow map (BGSM). The other layer of the shadow map views the triangles in the scene smaller than normal, therefore each triangle is shrunk a certain pixel size λ, which results in the small geometry shadow map (SGSM).

The pixel size λ can be calculated by the following equation:

$$\lambda = \sqrt{\left(\frac{\eta}{l_{sm}}\right)^2 + \left(\frac{\eta}{h_{sm}}\right)^2} \tag{19}$$

Where l_{sm} and h_{sm} are the length and height of the shadow map layer, and η indicates the number of pixels each triangle will expand/shrink. η can be altered by the user depending on many pixels the user wishes to expand/shrink. In this document η was set to 1 pixel.

When the shadows are rendered in to the scene, the shades will look to the corresponding texel in both layers and reach one of the following conclusions.

- If both texels agree that the pixel is in light ($t = 1$), the pixel in the scene will be in light;
- If both texels agree that the pixel is in shadow ($t = 0$), the pixels in the scene will be in shadow;
- If both texels disagree on the state of the pixel, the pixel is classified as "uncertain" and a ray is created to trace the final result.

Although (Hertel, Hormann, & Westermann, 2009) work allows for an effective method of determining "uncertain" pixels in scene, the method could go further since many of the "uncertain" pixels are not that uncertain. Figure 25 shows the results of conservative rasterization on a simple plane consisting of two triangles. In the BGSM (a), the triangles are individually expanded, resulting in overlapping in the centre of the plane and the unbinding of the edges of triangles. Since this is a shadow mapping, the overlap is not an issue and the edges can be easily clipped in the fragment shader to reduce further inconsistencies. In the SGSM (b), the triangles are shrunken

individually resulting in the tearing of the geometry in SGSM. When comparing the SGSM with the BGSM, these tears will be considered in light in the SGSM and be shadowed in the BGSM, and as result of the test, will considered as "uncertain" pixels and create rays to be traced to determine if they are shadowed or not. This is an unnecessary test, since even normal shadow map method would have considered shadowed.

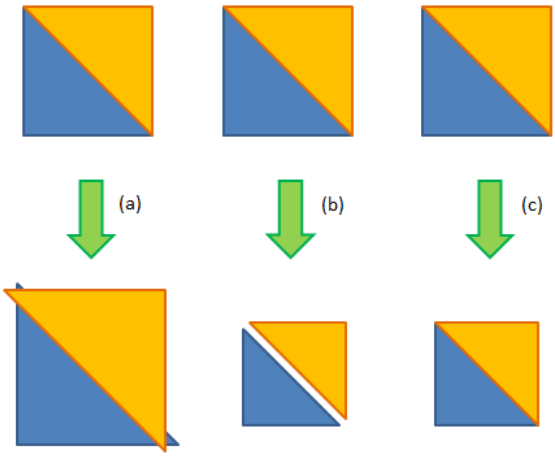


Figure 25 – Geometry Transformations on a pair of triangles that form a plane: (a) Expansion of the triangles in BGSM; (b) Shrinking of triangles in SGSM, without adjacency information; (c) Shrinking of triangles in SGSM, with adjacency information.

This work seeks to reduce the amount of "uncertain" pixels obtained by adding the adjacency information to the work of (Hertel, Hormann, & Westermann, 2009) to produce a more accurate SGSM (c).

OpenGL gives the geometry shader the information of a triangle and all triangles that share an edge with it, as showed in Figure 26.

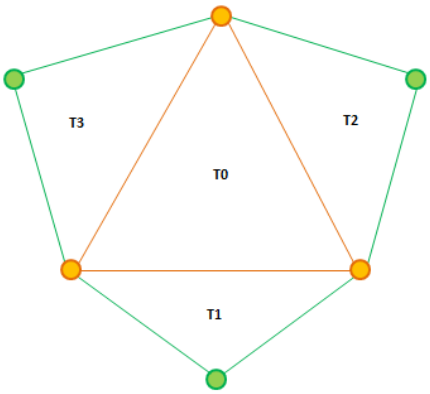


Figure 26 – Central triangle (orange) and the adjacent triangles (green) provided to the geometry shader.

The geometry shader receives an array of vertices, $vertex_{adj} = \{v_0, v_1, v_2, v_3, v_4, v_5\}$. As shown in Figure 26, the central triangle T_0, is formed by $\{v_0, v_2, v_4\}$, T_1 is formed by $\{v_0, v_1, v_2\}$, T_2 and T_3 are formed by $\{v_2, v_3, v_4\}$ and $\{v_4, v_5, v_0\}$ respectively.

As in conservative rasterization, we can calculate the edges of the new triangles. Following the example of equation (1) in section 2.3, the other edges can be calculated like this:

$$\begin{aligned} e_{1T1} &= v_1 - v_0;\ e_{2T1} = v_2 - v_1; \\ e_{1T2} &= v_3 - v_2;\ e_{2T2} = v_4 - v_3; \\ e_{1T3} &= v_5 - v_4;\ e_{2T2} = v_0 - v_5; \end{aligned} \tag{20}$$

These edges are useful to determine if the adjacent triangle exists and determine the facing of the triangles relative to the light. This can be done by the following algorithm for any adjacent triangle ($x \in \{1,2,3\}$):

```
bool txLight //Indicator if the Tx is facing the light/exists

if ( !(length(e1Tx) < 0.001 )){ //Tx exist
          //Calculate the normal of Tx
          normalTx = normalize( cross(vec3(e1Tx), vec3(e2Tx) ));
          if (dot(normalTx, lightDir) > 0)
                    txLight = true; //Tx is facing the light
          else
                    txLight = false; //Tx is not facing the light
          }
else { //Tx doesn't exist
                    txLight = true;
}
```

The first "if" condition is to determine that if the T_x vertex exists. In OpenGL, if adjacent triangle doesn't exist, it is common to set the respective vertex to the previous vertex, allowing for testing in the geometry shader. For example, if v_1 doesn't exist, $vertex_{adj} = \{v_0, v_0, v_2, v_3, v_4, v_5\}$. The second "if" condition is to determine the facing of the adjacent triangle, which will effect of the shrinking of the central triangle, to do this we calculate the normal of the triangle T_x and produce the dot product of the normal and the light direction. If it's positive, the triangle T_x is facing the light, therefore it is in light. Otherwise, the T_x is not facing the light, therefore it is in shadow. Any adjacent triangle that doesn't exist will be considered to be in light, since in practical terms the shadow doesn't exist because the triangle that would cast it doesn't exist. Using this assumption simplifies the shrinking of the central triangle.

We are only altering the triangles that are in shadow, since these triangles are the ones that the shadow map technique uses to project the shadows into the scene, so we can assume that the central triangle is in shadow. Since each central triangle has three adjacent triangles, at most, we'll have to account the following scenarios:

- None of the adjacent triangles is in shadow;
- One of the adjacent triangles is in shadow;
- Two of the adjacent triangles are in shadow;
- All of the adjacent triangles are in shadow;

For all of these cases, the shrunken edges produced in normal conservative rasterization must be calculated, since these edges are used to create the new triangle.

4.1 0-in-Shadow

The simplest case is when none of the adjacent triangles is in shadow (Figure 27). Basically this case is treated like it was a single triangle, following the same process as in normal conservative rasterization, explained in section 2.3.

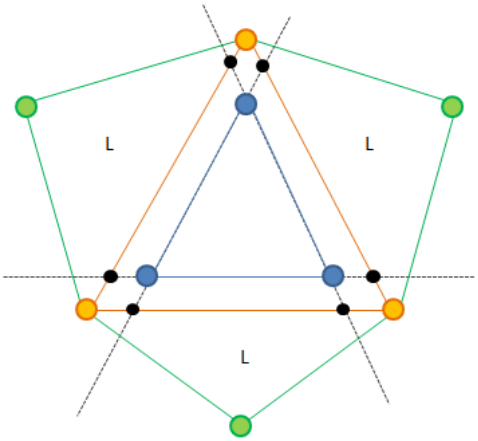


Figure 27 – 0-in-shadow case: the original triangle (orange) is shrunk using normal conservative rasterization (blue).

This mostly occurred in isolated triangles in the scene or in particular geometry constructions where only one triangle is in shadow, while the others are in light. Although this is a rare and mostly unrealistic scenario, it is a possibility and therefore it must be dealt with.

4.2 1-in-Shadow

The idea of this case is to shrink only the edges of the triangles that are in light to maintain the connection of the shadowed triangles. To do this we must calculate the intersection between the new lines formed by the shrinking of the lighted edges, and the old line of the shadowed edge, as demonstrated in Figure 28.

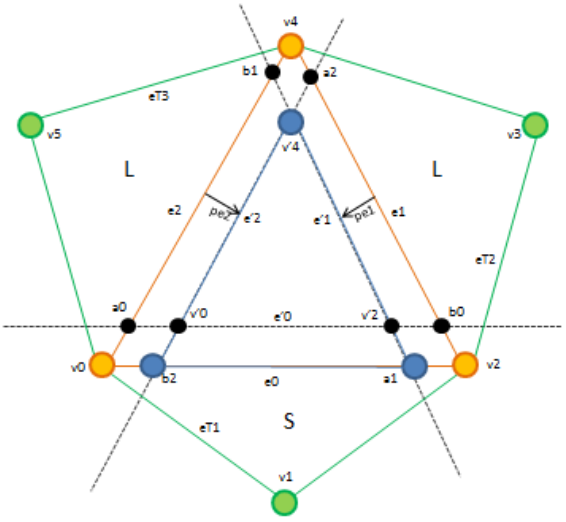


Figure 28 - 1-in-shadow case: The new triangle (blue) is composed by $\{b_2, a_1, v_4'\}$.

Note that b_n and a_n in Figure 28 are the intersection points of the new edges.

Before we can handle the adjacency, the algorithm must determine the shrunken triangle, in order to use the new vertices v'_x. After this process, v'_4 is already taken of. To determine the vertices b_2 and a_1, we'll have to find the following line intersections:

- The intersection point of the old edge e_0 with the new edge e'_2, which will give us $b2$;
- The intersection point of the old edge e_0 with the new edge e'_1, which will give us $a1$;

Using the *lineLineIntersection* function described in section 2.3, these points can be easily determined:

$$\begin{aligned} b_2 &= lineLineIntersection(v_0, v_2, v_4', v_0', n, v_0); \\ a_1 &= lineLineIntersection(v_0, v_2, v_2', v_4', n, v_2); \end{aligned} \tag{21}$$

After obtaining the new vertices, the geometry shader emits the proper shrunken triangle that maintains the connection with the adjacent shadowed triangle. Figure 29 shows all possible scenarios where only one of the adjacent triangles is in shadow.

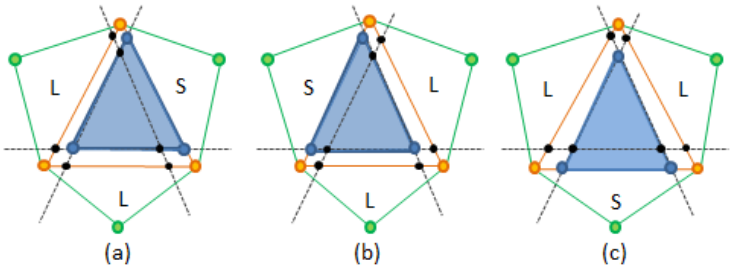


Figure 29 – Three simplified scenarios where the 1-in-Shadow occurs.

These are the easiest cases; there is only the need to calculate two more additional points in order to create the appropriate shrunken triangle.

4.3 2-in-Shadow

In these cases, two of the adjacent triangles are in shadow, while the other is in light. The obvious solution is to shrink the edge of the triangle in light. In the example showed in Figure 30, which would result in a new triangle formed by the vertices $\{v_0, a_1, b_1\}$.

But due to the restrictions of the adjacency information in OpenGL, the geometry shader only has access to six vertices at a time, and $v0$ could be part of a triangle that could be in light, but in the current group of vertices we don't have that information. The only safe assumption we can do is consider that v_0 is in light in another group of $vertex_{adj}$, so we'll cut out v_0 of the set and creating two new triangles in the geometry shader. This will result in the blue and light blue triangles in Figure 30.

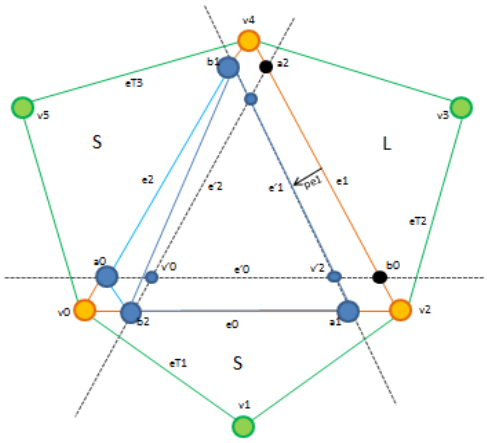


Figure 30 – 2-in-Shadow case: The new geometry is formed by the triangle $\{b_2, a_1, b_1\}$ and the triangle $\{b_1, a_0, b_2\}$.

Like before we'll need to determine the intersections of the old and new edges:

- The intersection point of the old edge e_2 with the new edge e'_1, which will give us b_1;
- The intersection point of the old edge e_0 with the new edge e'_1, which will give us a_1;
- The intersection point of the new edge e'_2 with the old edge e_0, which will give us b_2;
- The intersection point of the new edge e'_0 with the old edge e_2, which will give us a_0;

Using the *lineLineIntersection* function described in section 2.3, these points can be easily determined:

$$\begin{aligned} b_1 &= lineLineIntersection(v_4, v_0, sa_1, sb_1, n, v_4); \\ a_1 &= lineLineIntersection(v_0, v_2, sa_1, sb_1, n, v_2); \\ b_2 &= lineLineIntersection(v_0, v_2, sa_2, sb_2, n, v_0); \\ a_0 &= lineLineIntersection(v_4, v_0, sa_0, sb_0, n, v_0); \end{aligned} \tag{22}$$

After obtaining the new vertices, the geometry shader emits two triangles as depicted in Figure 31: a large triangle, containing the central triangle (dark blue), and a smaller triangle, to cut the corner between the adjacent triangles that are in shadow (light blue).

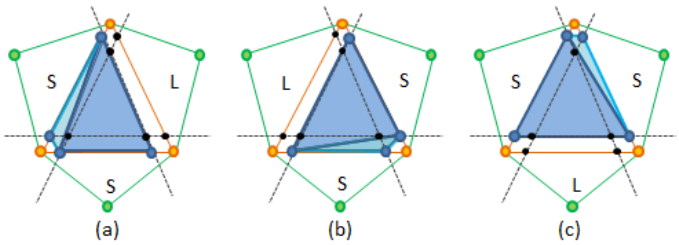


Figure 31 – Three simplified scenarios where the 2-in-Shadow occurs.

These cases are more complex than the previous ones, as they required the emission additional geometry to maintain the connections. In a large number of triangles this case is more taxing on the GPU.

4.4 3-in-Shadow

In this case all the adjacent triangles are in shadow, therefore there is no actual edge to shrink. However the issue that occurred in the 2-in-shadow cases still occurs, we don't know if any of vertices in the original triangle is in light at another $vertex_{adj}$. Following the same assumption as before, the only thing we can do is to clip the corners of the original triangle, as it's showed in Figure 32. In order to do so, we must calculate all the following intersections:

- The intersection point of the new edge e'_0 with the old edge e_2, which will give us a_0;
- The intersection point of the new edge e'_0 with the old edge e_1, which will give us b_0;
- The intersection point of the new edge e'_1 with the old edge e_0, which will give us a_1;
- The intersection point of the new edge e'_1 with the old edge e_2, which will give us b_1;
- The intersection point of the new edge e'_2 with the old edge e_1, which will give us a_2;
- The intersection point of the new edge e'_2 with the old edge e_0, which will give us b_2;

Using the *lineLineIntersection* function described in section 2.3, these points can be easily determined:

$$\begin{aligned} a_0 &= lineLineIntersection(v_4, v_0, v'_0, v'_2, n, v_0); \\ b_0 &= lineLineIntersection(v_2, v_4, v'_0, v'_2, n, v_2); \\ \\ a_1 &= lineLineIntersection(v_0, v_2, v'_2, v'_4, n, v_2); \\ b_1 &= lineLineIntersection(v_4, v_0, v'_2, v'_4, n, v_4); \\ \\ a_2 &= lineLineIntersection(v_2, v_4, v'_4, v'_0, n, v_4); \\ b_2 &= lineLineIntersection(v_0, v_2, v'_4, v'_0, n, v_0); \end{aligned} \tag{23}$$

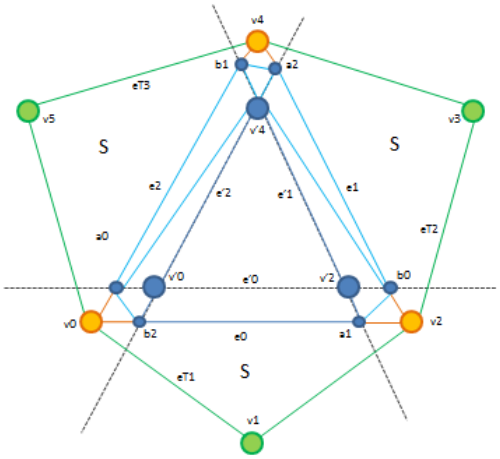


Figure 32 – 3-in-Shadow case: The new geometry is formed by the four triangles: $\{b_1, a_0, b_2\}$, $\{b_2, a_1, b_1\}$, $\{b_1, a_1, a_2\}$ and $\{a_2, a_1, b_0\}$.

After obtaining the new vertices, the geometry shader emits four triangles, depicted by the blue and light blue lines in Figure 32.

This is the heaviest impact case, since it's the most common situation when treating shadowed objects. When a side of an object is shadow, great number of those triangles that are part of the object will fall in this case, and for each of those triangles, four are emitted to shrink and maintain the connectivity of the object.

4.5 Flaws of conservative rasterization process

Despite these upsides, Figure 33 shows an example where the use of conservative rasterization approach fails to completely maintain the connections of all the triangles in the geometry. The plane formed by four triangles in (b) is shrunk with this approach. While each triangle have the information of the adjacent triangles that share an edge, the triangles with the same colour share only a single vertex, and so don't appear in the adjacency list $vertex_{adj}$, which causes the vertex in the middle to disappear, since each triangle will clip it out, resulting in a hole in the geometry. This is due to the restrictions of the adjacency information of only three adjacent triangles, and the resulting assumptions made in 2-in-Shadow and 3-in-Shadow. A way to counteract this problem is to provide the shader all of the adjacency information of the scene before rendering.

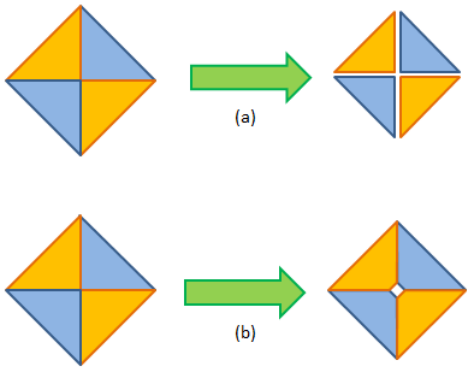


Figure 33 - Errors in the conservative rasterization: (a) Normal conservative rasterization; (b) Conservative rasterization with adjacency.

Still, the use of the adjacency information provides a superior alternative that normal conservative rasterization, where the triangles would a shrunk inwards and causer tears in the geometry. While at (a), rays would have to be traced from all the pixels in the tears, in (b) the only pixels in the holes would have a ray created to be traced.

4.6 Conclusion

This approach to conservative rasterization solves the problem of tearing when shrinking triangles in the geometry. By taking the adjacency information, we can determine which edges of a triangle are safe to shrink to maintain the connectivity of the triangle. Although these transformations will result in more triangles being emitted by the geometry shader, this will not have an impact in the result of the shadow map, since these overlaps will occur in the places where tears would occur normally, filling them in the fragment shader. Regarding performance the computations required to take advantage of the adjacency information will increase the processing time in the geometry shader, on the other hand, this leads to less rays being traced in the following step, as shown in the next chapters.

5 Analysis and Results

In this section, we will analyse the pixel accuracy of the three main algorithms: PCF, CRSM and CRSMA, when faced with the results obtained by the pure ray tracing methods, using OptiX Prime's results as a base of comparison. The tables and figures used in this chapter are small part of large test pool, which were chosen to convey a simplified view to conclusions obtained.

5.1 Test Scenes

The following images will show the scenes and that will be used for testing and the various viewpoints that will be used for said tests.

The first scene consists of a scene with two trees, a lamp, a flower box and a bench, all positioned on top of a plane representing the ground. The scene has a total of 55026 triangles. This scene will be called "Bench". Information of light, camera and field of view of this scene can be observed in Table 1 and Figure 34 shows the image obtained.

Viewport		Coordinates		
		x	y	z
Side	Position	-23,277	18,541	30,143
	Direction	0,397	-0,644774	-0,652
With	Position	-37,034573	35,208973	-8,597797
	Direction	0,605439	-0,732089	0,312232
Against	Position	27,214222	27,875109	27,032139
	Direction	-0,560848	-0,777942	-0,283293
Light Direction		0,744	-0,408	0,527
View Frustum		Far: 400,0	Near: 1,0	FoV: 60°

Table 1 - Information of viewports used for the Bench Scene.

Figure 34 - The side (left), with (centre) and against (right) viewpoints of the first scene; Shadows obtained using OptiX Prime.

The second scene, named "Flowers", will also use the same models as the first scene, but will closely observe the shadows cast by the flowers. The flowers are modelled with very small triangles, allowing the visualization the effect of small geometry on the algorithm. In Table 2 the information of the camera of each viewpoint can be viewed, and Figure 35 shows the image obtained.

Viewport		Coordinates		
		x	y	z
Side	Position	-0,260709	22,376335	6,046194
	Direction	-0,387214	-0,852832	0,350347
With	Position	-17,561422	24,968716	4,010894
	Direction	0,386402	-0,873032	0,297505
Against	Position	-3,263903	24,423452	12,998949
	Direction	-0,239566	-0,958412	-0,155095
Light Direction		0,744	-0,408	0,527
View Frustum		Far: 400,0	Near: 1,0	FoV: 60°

Table 2 - Information of viewports used for the Flower Scene.

Figure 35 – The side (left), against (centre) and with (right) viewpoints of the second scene; Shadows obtained using OptiX Prime.

The third scene, called "Trees", will also use the same models as the second scene, but with focus on an area of the ground where only the shadows of the trees can be seen. Since the trees are constituted by big triangles, this will allow the evaluation of the effect of big triangles on the results. Information of cameras of each viewpoint can be observed in Table 3, and Figure 36 shows the image obtained.

Viewport		Coordinates		
		x	y	z
Side	Position	101,010887	30,839497	-23,4187145
	Direction	-0,235037	-0,802447	0,548369
With	Position	42,947086	24,103859	-27,831772
	Direction	0,415959	-0,784187	0,460467
Against	Position	90,8069972	35,846294	24,138012
	Direction	-0,400057	-0,835954	0,37568
Light Direction		0,744	-0,408	0,527
View Frustum		Far: 400,0	Near: 1,0	FoV: 60°

Table 3 - Information of viewports used for the Trees Scene.

Figure 36 - The with (left), side (centre) and against (right) viewpoints of the third scene; Shadows obtained using OptiX Prime.

The fourth scene, called "Sponza", will use the Crytek's Sponza Palace model (Meinl, 2011), focusing attention on a side area, where the shadows of nearby objects are projected to the ground and the cloth. The light is coming from the second floor on the opposite side form the camera. This scene will allow us to analyse the obtained shadows on complex geometry. The information of the camera and the light is shown in Table 4, and Figure 37 shows the image obtained.

Viewport		Coordinates		
		x	y	z
Main	Position	-83,72924	222,06694	-2,89993
	Direction	0,810612	-0,273794	0,517634
Light Direction		-0,001881	-0,968051	0,250748
View Frustum		Far: 2800,0	Near: 1,0	FoV: 60°

Table 4 - Information of viewports used for the Sponza Scene.

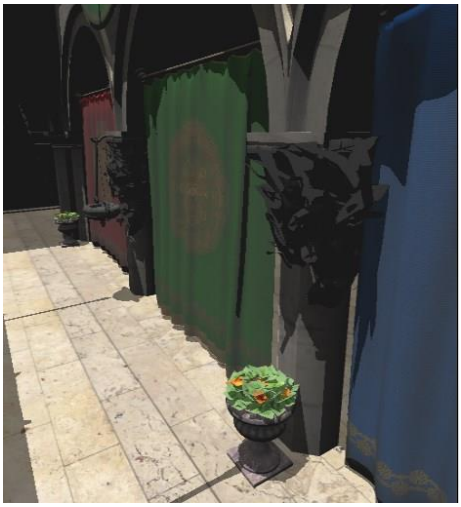


Figure 37 - Sponza Scene; Shadows obtained using OptiX Prime.

Statistics will be collected all scenes, for shadow map resolutions of 512x512, 1024x1024, 2048x2048 and 4096x4096, and considering viewport resolutions of 512x512, 1024x1024 and 1920x1080 (FullHD). The ray tracing results will be used as source of comparison for the NSM and the hybrid methods: PCF, CRSM and CRSMA.

The tables and figures shown in this chapter only show the results for the significant test scenes, mainly the "Flowers" and "Sponza" scenes, the tables and figures for the rest of the test scenes can be found in section A.1 of the Appendix.

5.2 Ray tracing Statistics

To establish a ground truth to compare the other methods NVIDIA's OptiX Prime was used to render all the scenes, under all the viewports.

The collected statistics will show the number of pixels in following states:

- Pixels Facing the Light (PFL), i.e. dot(n,l) > 0;
- Pixels not Facing the Light (PnFL), i.e. dot (n,l) < 0;
- PFL – Pixels in Light;
- PFL – Pixels in Shadow.

The PFL and the PnFL pixels are directly tied to the geometry of the scene, so the number represented by these states will be the same for all the methods. PFL can be divided further into two categories: pixels in light and pixels in shadow, which will be the main source of analysis.

Table 5 shows an example of the data collected for each scene. Both the number of pixels and the percentages are displayed to ease comparisons. Figure 38 shows the image obtained by OptiX Prime.

Scene	Flowers					
Viewport	With					
Pixel Types	Viewport Size					
	512x512		1024x1024		1920x1080	
PFL	259486	98,99%	1037820	98,97%	2061613	99,42%
PnFL	2658	1,01%	10756	1,03%	11987	0,58%
PFL – in Light	153093	59,00%	612204	58,99%	1458469	70,74%
PFL – in Shadow	106393	41,00%	425616	41,01%	603144	29,26%

Table 5 - OptiX Prime results for the With-Flowers scene: PFL represents the Pixels Facing the Light and PnFL represents the Pixels not Facing the Light; The PFLs are then split into the pixels in Light and the pixels in Shadow.

As shown in Table 5, the distribution of the pixels states maintains the same ratio regarding the viewport sizes. The only significant change occurs in the 1920x1080 viewport due to the aspect ratio change.

Figure 38 - OptiX Prime Shadows in the With-Flowers scene using the With viewport; the green pixels represent the PNFL pixels; Viewport Size: 1024x1024.

Table 6 shows an example of the data collected for Sponza scene. Both the number of pixels and the percentages are displayed to ease comparisons. Figure 39 shows the image obtained by OptiX Prime.

Scene	Sponza					
Viewport	Main					
Pixel Types	Viewport Size					
	512x512		1024x1024		1920x1080	
PFL	207377	79,11%	829652	79,12%	1655090	79,82%
PnFL	54767	20,89%	218924	20,88%	418510	20,18%
PFL – in Light	117780	56,80%	471083	56,78%	955576	57,74%
PFL – in Shadow	89597	43,20%	358569	43,22%	699514	42,26%

Table 6 - OptiX Prime results for the Sponza scene: PFL represents the Pixels Facing the Light and PnFL represents the Pixels not Facing the Light; The PFLs are then split into the pixels in Light and the pixels in Shadow.

As shown in Table 6, the distribution of the pixels states maintains the same ratio regarding the viewport sizes. The only significant change occurs in the 1920x1080 viewport due to the aspect ratio change, similarly to the results in Table 5.

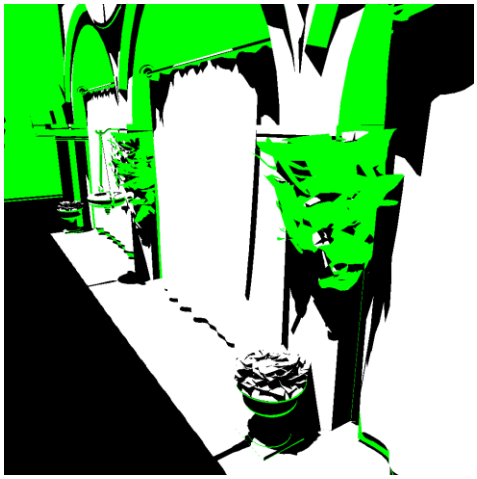


Figure 39 - OptiX Prime Shadows in the Sponza scene; the green pixels represent the PNFL pixels; Viewport Size: 1024x1024.

5.3 Algorithm Precision Testing

In this section we will analyse the pixel precision of the standard shadow map (NSM) method and the following hybrid methods: PCF method used by Beister's method (Beister, Ernst, & Stamminger, 2005), the CRSM used in Hertel's work (Hertel, Hormann, & Westermann, 2009), and the CRSMA presented in the previous chapter. All of the methods are going to be compared to the ray tracing results provided by OptiX Prime.

5.3.1 Standard Shadow Mapping errors

In this section we'll discuss the errors obtained with normal shadow mapping when compared with the results of OptiX Prime. We will only compare the PFL-Light and PFL-Shadow parameters, since the total pixels in the PFL is always the same.

As shown in the Table 7, the number of is errors in the NSM method is quite significant, as well as the placement of these errors, as shown in Figure 40.

Scene	Flowers			
Viewport	With			
Pixel State	Pixel Precision			
	Correct		Incorrect	
Light	597624	96,93%	18934	3,07%
Shadow	406682	96,54%	14580	3,46%
Total	1004306	96,77%	33514	3,23%

Table 7 – Normal Shadow Mapping errors in the With-Flowers scene with a 1024x1024 viewport and 1024x1024 shadow map

The majority of the errors in this scene are placed in the shadows of the bench and the flower pot. The placement of errors corresponds to the contours of the real shadows (Moderno, 2011).

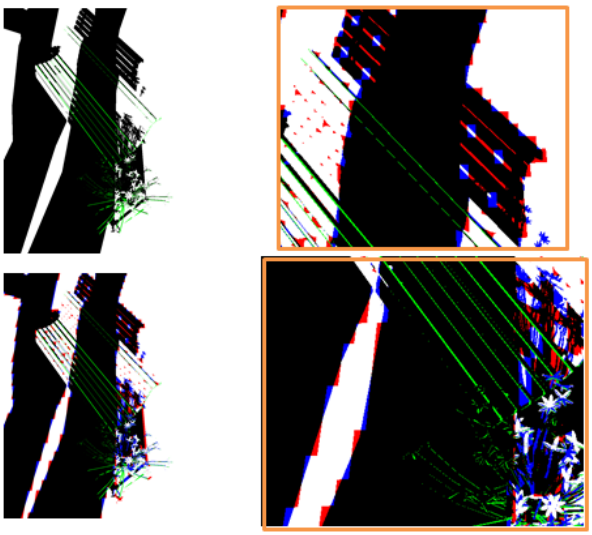


Figure 40 – With-Flowers scene with a 1024x1024 viewport and 1024x1024 shadow map; (Top-Left) OptiX Prime results; (Bottom-Left) NSM results; (Right) Significant errors obtained by NSM; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Table 8 shows the errors obtained in the Sponza scene, the number of is errors in the NSM method is quite significant, as well as the placement of these errors, as shown in Figure 41.

Scene	Sponza			
Viewport	Main			
Pixel State	Pixel Precision			
	Correct		Incorrect	
Light	458890	93,53%	31768	6,47%
Shadow	326793	96,40%	12192	3,60%
Total	785683	94,70%	43960	5,30%

Table 8 – Normal Shadow Mapping errors in the Sponza scene with a 1024x1024 viewport and 1024x1024 shadow map

The great majority the errors in the light pixels are in the flowers pot an in the surrounding areas, while the errors in the shadow pixels are spread around the image, but are more visible in cloth, specifically in the contours of the shadows (Moderno, 2011).

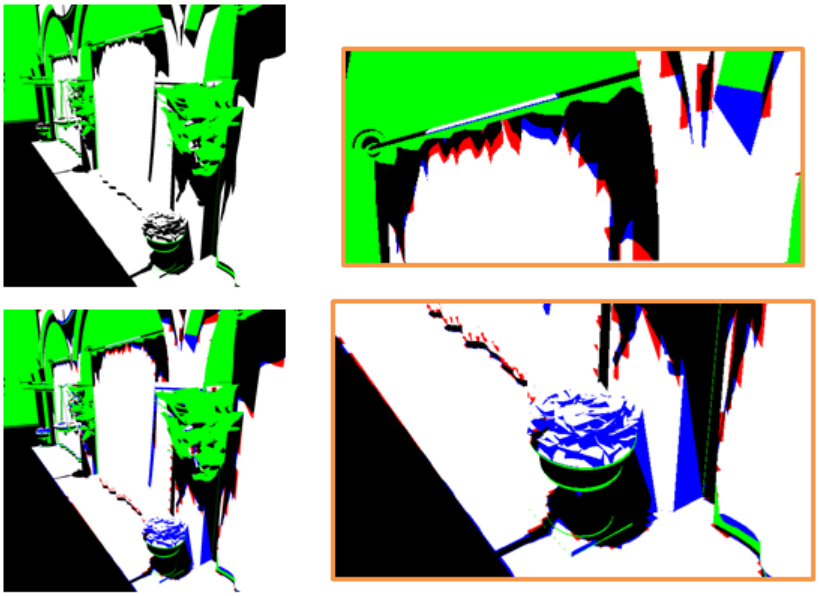


Figure 41 – Sponza scene with a 1024x1024 viewport and 1024x1024 shadow map; (Top-Left) OptiX Prime results; (Bottom-Left) NSM results; (Right) Significant errors obtained by NSM; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels

5.3.2 Viewport Size Variance

Taking consideration the errors found using a fixed viewport, the following table (Table 9) shows how much the change of the viewport, since it does impact on the errors of the shadow map.

Scene	Flowers					
Viewport	With					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
C-Light	149396	96,95%	597624	96,93%	1430029	97,74%
C-Shadow	101695	96,49%	406682	96,54%	570054	95,25%
I-Light	4698	3,05%	18934	3,07%	33090	2,26%
I-Shadow	3697	3,51%	14580	3,46%	28440	4,75%

Table 9 – NSM viewport results for the With-Flowers scene, using a fixed shadow map size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

Although the 512x512 and 1024x1024 viewports achieve the similar percentages, when the viewport changed to 1920x1080, the amount of errors in the shadow pixels increased by approximately 1,29%, due to the change of the aspect ratio to 1,78, while the errors in the light pixels decreased by 0,81%, resulting a 0,50% reduction of the total number of incorrect pixels. All the other scenes provide similar data

Since these percentage changes are very small, with 1% increase of the incorrect pixels as shown in Figure 42, and the only differences between the viewports in the number of pixels each contains, we can conclude that the variance in the viewport doesn't have a great impact in the pixel precision of the NSM process.

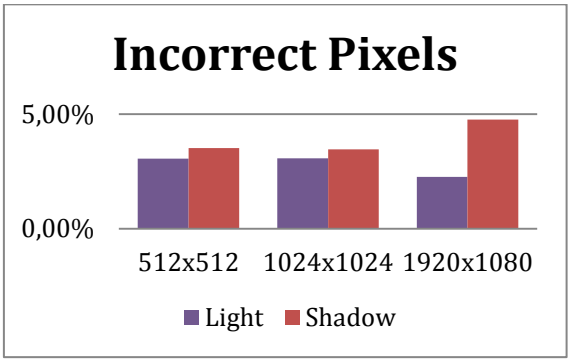


Figure 42 - Viewport Variation results in the With-Flowers scene, using a fixed shadow map with 1024x1024 sizes.

The next table (Table 10) shows how the changing of the viewport size affects the errors obtained in the Sponza scene.

Scene	Sponza					
Viewport	Main					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
C-Light	114745	93,53%	458890	93,53%	938410	95,34%
C-Shadow	81660	96,42%	326793	96,40%	653681	97,44%
I-Light	7935	6,47%	31768	6,47%	45827	4,66%
I-Shadow	3035	3,58%	12192	3,60%	17165	2,56%

Table 10 – NSM viewport results for the Sponza scene, using a fixed shadow map size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

As shown by Table 10, the 512x512 and 1024x1024 viewports achieve similar percentages, and the change to the 1920x1080 viewport caused the amount of errors in the light pixel to decreased by 1,81% and the amount of errors in the shadow pixels by 1,04%, as shown by Figure 43. Like before, these changes are very small and don't cause significant impact in the pixel precision.

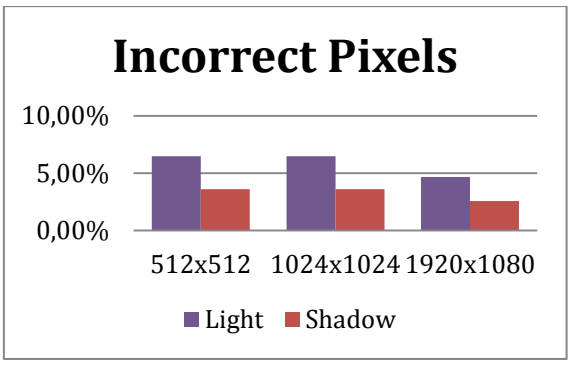


Figure 43 - Viewport Variation results in the Sponza scene, using a fixed shadow map with 1024x1024 sizes.

5.3.3 Shadow Map Size Variance

Now, we'll analyse the changes in the NSM errors with the change of the shadow map resolution with a fixed viewport size. As shown in Table 11 and Figure 44, the error tends to decrease as the number of pixels of the shadow map increase.

Scene	Flowers							
Viewport	With							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
C-Light	582232	95,89%	597624	96,93%	603076	98,14%	606507	98,70%
C-Shadow	400649	93,04%	406682	96,54%	414179	97,84%	417657	98,65%
I-Light	24967	4,11%	18934	3,07%	11437	1,86%	7959	1,30%
I-Shadow	29972	6,96%	14580	3,46%	9128	2,16%	5697	1,35%

Table 11 - NSM shadow map results in the With-Flowers scene, using a fixed viewport size of 1024x1024; I represents the incorrect pixels and C represents the correct pixels.

This is because as the shadow map increases, more texels are stored in the shadow map, increasing the shadow precision. The major changes are in the shadow pixels, where the change from 512x512

to 4096x4096, the decrease in errors is 5,61%, which results in a substantial difference. Similar results are obtained for the other combinations of scenes/viewports.

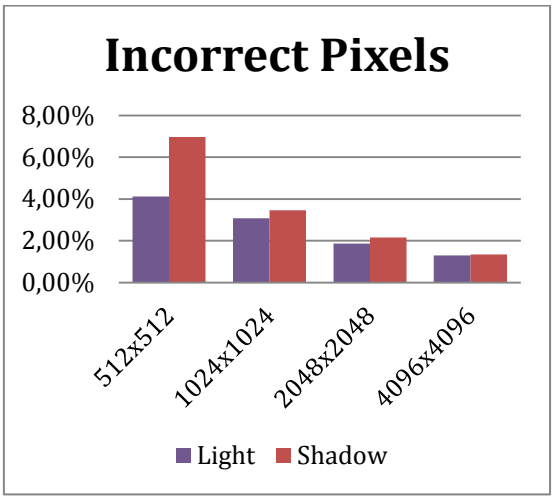


Figure 44 – Shadow Map Size variation results in the With-Flowers scene using a fixed viewport of 1024x1024.

The next table (Table 12) shows how the changing of the shadow map sizes affects the errors obtained in the Sponza scene.

Scene	Sponza							
Viewport	Main							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
C-Light	446750	91,59%	458890	93,53%	465559	94,68%	468229	95,35%
C-Shadow	317531	92,88%	326793	96,40%	332391	98,37%	335718	99,16%
I-Light	41030	8,41%	31768	6,47%	26170	5,32%	22843	4,65%
I-Shadow	24332	7,12%	12192	3,60%	5523	1,63%	2853	0,84%

Table 12 - NSM shadow map results in the Sponza using a fixed viewport size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

The amount of errors tend to decrease, but the only significant changes are in the shadow pixels as shown in Figure 45, the change from a 512x512 shadow map to a 4096x4096 shadow map caused a 6,28% decrease of errors, while the light pixels there was only a 3,76% decrease, making these light pixels the majority of the problem in the NSM method in this scene.

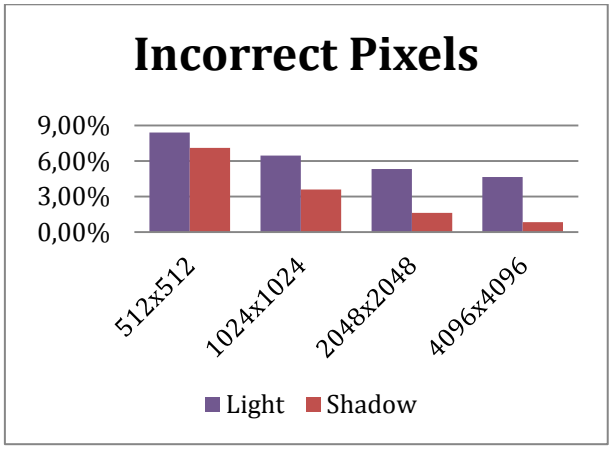


Figure 45 – Shadow Map Size variation results in the Sponza scene, using a fixed viewport of 1024x1024.

5.4 Hybrids Methods errors

In this section we compare the final images obtained in the PCF, CRSM and CRSMA with the images obtained using a pure ray tracing approach with OptiX Prime. This is to determine the amount of correct and incorrect pixels in the images, since ray tracing produces the most accurate images. The following sub-sections will analyse certain scenarios of shadows, where the differences between the three methods provide detailed information into their effectiveness.

From here onwards, the tables will split the PFL into two groups: the "certain" pixels and the "uncertain" pixels. Since the "uncertain" pixels are them sent to the ray tracer to fix them, is it safe to assume that these pixels will have the same state (light/shadow) as the pixels in the pure ray tracing, so these pixels are correct when compared to the OptiX Prime. The "certain" pixels will be split into the correct pixels, which when compared to the respective pixel in the OptiX Prime the states are the same in both, and the incorrect pixels, where the opposite occurs.

5.4.1 Uncertain Pixels Detection

In the original work where these hybrid methods were introduced, the triangle ID is stored in the shadow map to allow a single ray intersection. This approach improves performance, but it induces errors in the image as shown in the Dave's work (Moderno, 2011). By using the OptiX Prime API, the

uncertain pixels are going to be corrected by using the same ray-intersection algorithm used in OptiX Ray Tracing algorithm, which increases the pixel precision.

The following table shows the results in a simple scene, the "Trees" scene, where there is no PnFL in the scene. Since each method has a different algorithm to detect uncertain pixels, we need to quantify the amount of pixels that are found. The following table (Table 13) show the number of "certain" and "uncertain" pixels in the image.

Scene	Trees					
Viewport	Against					
Pixel State	Method					
	PCF		CRSM		CRSMA	
Certain	787486	75,10%	323513	30,85%	558753	53,29%
Uncertain	261090	24,90%	725063	69,15%	489823	46,71%

Table 13 – Certain and Uncertain pixels found in the Against-Trees scene for each method; results obtained with a 1024x1024 viewport and a 1024x1024 shadow map.

PCF is able to obtain a higher number of "certain" pixels than the other two, achieving a difference of 48,13% to CRSM's "certain" pixels and a difference of 21,88% to CRSMA's "certain" pixels. This is the positive side of PCF, since it only detect the uncertain pixels in the border of the shadows, as shown in Figure 46, which result in less rays to be traced.

CRSM method has more "uncertain" pixels than CRSMA, causing a significant difference of 26,25%. This is because of the tears in the shadows, as shown in Figure 47, where the majority of the "uncertain" pixels come from the triangulation of the shadows. In contrast, CRSMA doesn't have the tears but instead has small holes, as shown in Figure 48, which are the flaws of the adjacency information as explained in the previous chapter. This is the positive effect of the use of adjacency information, as it leads to fewer rays to be sent to the ray tracer.

It should be noted, that while the CRSMA has less triangulation tears, both the CRSM and CRSMA make the shadow of the lamp pole disappear, while PCF doesn't. This is because in the SGSM and SGSMA the triangles shrunken to the point that in the shadow map they disappear, while the BGSM they still exist.

Figure 46 – PCF's "uncertain" pixel detection. The purple pixels represent the "uncertain" detected. Viewport size 1024x1024 and Shadow Map size 2048x2048.

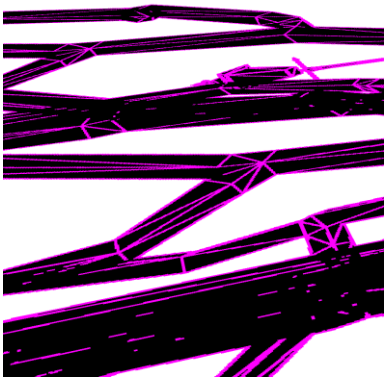


Figure 47 – CRSM's "uncertain" pixel detection. The purple pixels represent the "uncertain" detected. Viewport size 1024x1024 and Shadow Map size 2048x2048.

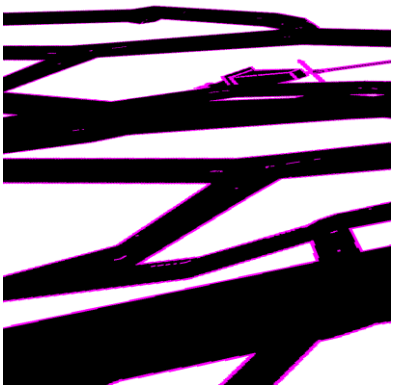


Figure 48 – CRSMA's "uncertain" pixel detection. The purple pixels represent the "uncertain" detected. Viewport size 1024x1024 and Shadow Map size 2048x2048.

5.4.2 Error Analysis

In order to analyse the accuracy of these hybrid methods, we need to separate the "uncertain" pixels from the correct pixels in the image, since these pixels will be corrected by the ray tracer. These "certain" pixels are the pixels which the shadow mapping has no problem in ascertain the pixel state, so this will evaluate the precision of the shadow maps of each method. Table 14, shows these statics in the Flowers scene, using the With viewport.

Scene	Flowers					
Viewport	With					
Method	**PCF**					
Pixel State	Certain - Correct		Uncertain		Certain -Incorrect	
Light	498196	81,40%	113828	18,60%	2220	0,36%
Shadow	328074	77,05%	97722	22,95%	2400	0,56%
Total	821650	79,17%	211550	20,38%	4620	0,45%
Method	**CRSM**					
Pixel State	Certain - Correct		Uncertain		Certain -Incorrect	
Light	463211	75,66%	148993	24,34%	0	0,00%
Shadow	38214	8,98%	387402	91,02%	0	0,00%
Total	501425	48,32%	536395	51,68%	0	0,00%
Method	**CRSMA**					
Pixel State	Certain - Correct		Uncertain		Certain -Incorrect	
Light	463211	75,66%	148993	24,34%	0	0,00%
Shadow	234120	55,01%	191496	44,99%	0	0,00%
Total	697331	67,19%	340489	32,81%	0	0,00%

Table 14 – Pixels Precision results in the With-Flowers scene with a 1024x1024 viewport and a 1024x1024 shadow map.

PCF obtains the highest number of incorrect pixels, resulting in 0,45% of the image is incorrect, despite having 60,56% less "uncertain" pixels than CRSM and having 37,87% less "uncertain" pixels than CRSMA. Although the percentage of total errors is low, the errors in the image are very noticeable, as shown in Figure 49.

Contrasting with PCF, CRSM and CRSMA obtained no incorrect pixels in the image, as shown by Figure 50, while CRSMA less "uncertain" pixels than CRSM. Also note that the majority of the pixels sent to the ray tracer by the CRSM are in the shadow area, while the CRSMA, with the use of adjacency, only sends half of those pixels, resulting on a difference of 46%.

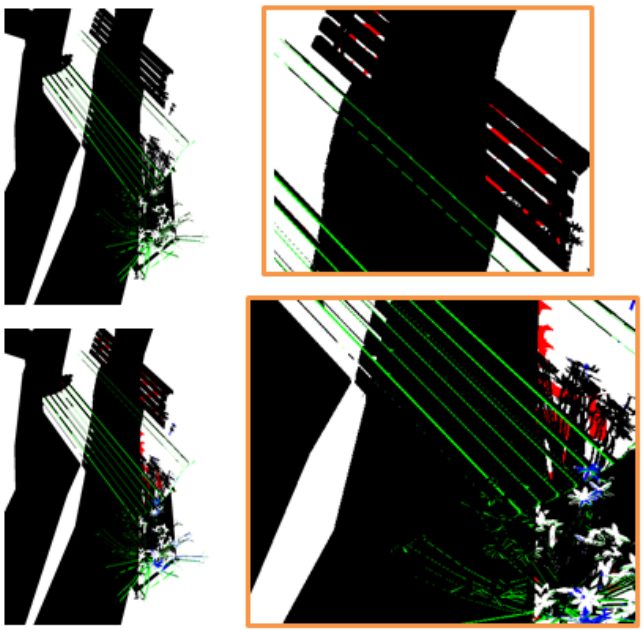


Figure 49 – With-Flowers scene with a 1024x1024 viewport and 1024x1024 shadow map; (Top-Left) OptiX Prime results; (Bottom-Left) PCF results; (Right) Significant errors obtained by NSM; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

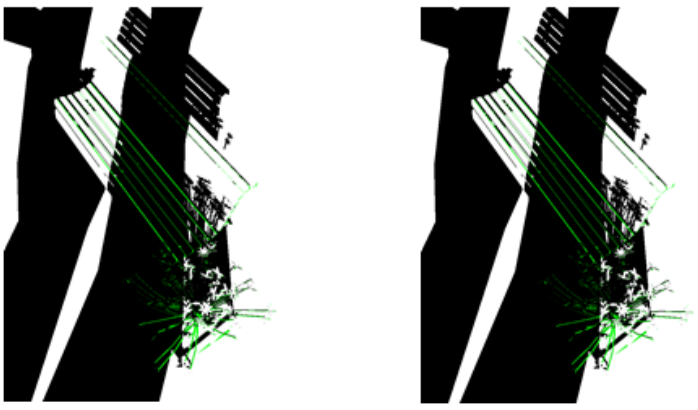


Figure 50 – With-Flowers scene with a 1024x1024 viewport and 1024x1024 shadow map; (a) OptiX Prime results; (b) CRSM and CRSMA results.

The following table (Table 15) presents the errors the hybrid methods produce in the Sponza scene.

Scene	Sponza					
Viewport	Main					
Method	**PCF**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	312081	64,37%	172742	35,63%	14026	2,89%
Shadow	257270	74,61%	87550	25,39%	285	0,08%
Total	555040	66,90%	260292	31,37%	14311	1,72%
Method	**CRSM**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	210396	44,45%	262962	55,55%	2276	0,48%
Shadow	117473	32,97%	238812	67,03%	0	0,00%
Total	325593	39,24%	501774	60,48%	2276	0,27%
Method	**CRSMA**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	210396	44,45%	262962	55,55%	2276	0,48%
Shadow	138589	38,90%	217696	61,10%	0	0,00%
Total	346709	41,79%	480658	57,94%	2276	0,27%

Table 15 – Pixels Precision results in the Sponza scene with a 1024x1024 viewport and a 1024x1024 shadow map.

Like the results of the Flowers scene in Table 14, PCF obtains the highest amount of incorrect pixels, achieving the 1,72% incorrect pixels, while both CRSM and CRSMA only achieve the 0,27% mark. The errors of all methods are quite significant, with Figure 51 demonstrating PCF's error position and Figure 52 demonstrating the CRSM and CRSMA errors. These errors are the result of the front culling applied when creating the shadow map, causing triangles in the scene to disappear, since pure ray tracing doesn't require to any type of culling, the rays will intersect the non-existing triangles causing differences in the hybrid and pure ray tracing images. This leads to an incorrect classification of light pixels in both the BGSM and SGSM, instead of classifying them as "uncertain". Another important factor is the lack of incorrect pixels in the shadow in the CRSM and CRSMA. This shows that conservative rasterization is able to resolve the problems in the shadow, but not all the problems in the light pixels.

The amount of "uncertain" pixels discovered by CRSM and CRSMA is quite similar, with only a 5,93% difference, as opposed to the 18,87% difference form Table 14. This means that for more complex scenes, the use of adjacency information produces diminishing returns to the detection algorithm.

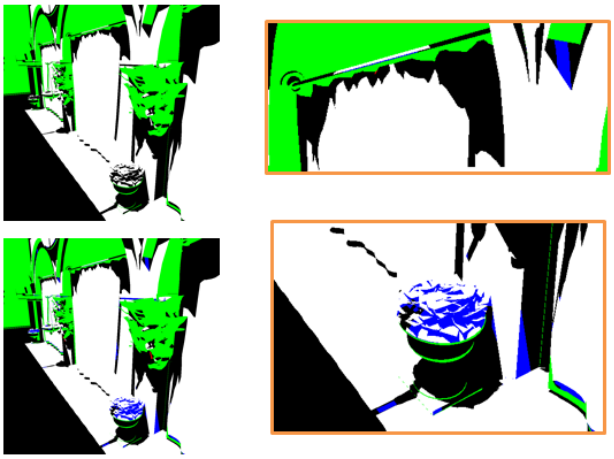


Figure 51 – Sponza scene with a 1024x1024 viewport and 1024x1024 shadow map; (Top-Left) OptiX Prime results; (Bottom-Left) PCF results; (Right) Significant errors obtained by NSM; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

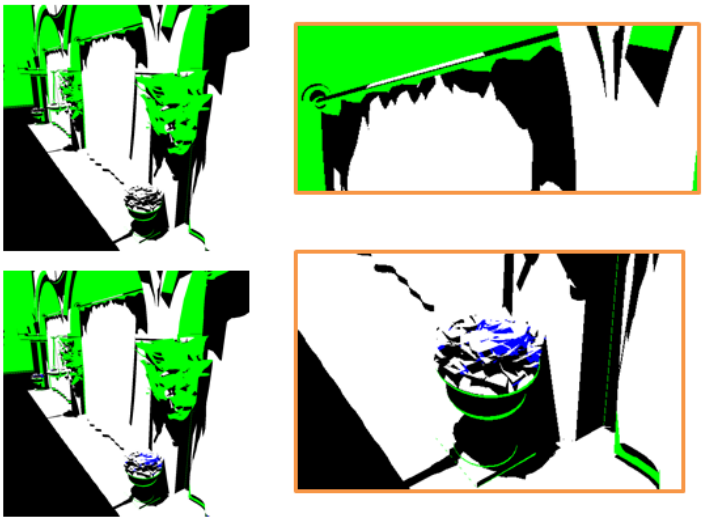


Figure 52 – Sponza scene with a 1024x1024 viewport and 1024x1024 shadow map; (Top-Left) OptiX Prime results; (Bottom-Left) CRSM and CRSMA results; (Right) Significant errors obtained by NSM; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

5.4.2.1 Viewport Variance

Table 16 shows the effect of the viewport size in the amount of correct/incorrect pixels in these hybrid methods. As expected, these percentile changes are similar to the ones obtained in the NSM testing, concluding that the viewport size doesn't affect the pixel precision of the hybrid methods, just like the NSM.

Scene	Flowers					
Viewport	With					
Method	**PCF**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	205461	79,18%	821650	79,17%	1669064	80,96%
Uncertain	52862	20,37%	211550	20,38%	384002	18,63%
Certain-I	1163	0,45%	4620	0,45%	8547	0,41%
Method	**CRSM**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	125343	48,30%	501425	48,32%	1191500	57,79%
Uncertain	134143	51,70%	536395	51,68%	870113	42,21%
Certain-I	0	0,00%	0	0,00%	0	0,00%
Method	**CRSMA**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	174382	67,20%	697331	67,19%	1386322	67,24%
Uncertain	85104	32,80%	340489	32,81%	675291	32,76%
Certain-I	0	0,00%	0	0,00%	0	0,00%

Table 16 – Viewport Variation of the pixel precision results using a fixed Shadow Map Size of 1024x1024 for the With-Flowers scene; C - Correct and I - Incorrect.

5.4.2.2 Shadow Map Variance

Table 17 shows the effect of the shadow map size in the amount of correct/incorrect pixels in these hybrid methods. As expected, PCF obtains better results as the shadow map increases, as the NSM, the more pixels are available in the shadow map, the more accurate it becomes. Figure 53, shows this increase in correct pixels in the PCF.

For CRSM and CRSMA the increase of the shadow map resolution does reduce the amount of "uncertain" pixels detected, as shown in Figure 53. This is due to the λ variable being dependent of

the shadow map size, as the shadow map increases, the λ will reduce the shrinking/expanding effect in the shadow maps, reducing the "uncertain" pixels in the image. The negative effect of the λ is to cause incorrect pixels to appear in the image.

Scene	Flowers							
Viewport	With							
Method	**PCF**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	719491	69,33%	821650	79,17%	909691	87,65%	957219	92,23%
Uncertain	312383	30,10%	211550	20,38%	124541	12,00%	77954	7,51%
Certain-I	5946	0,57%	4620	0,45%	3588	0,35%	2647	0,26%
Method	**CRSM**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	420355	40,50%	501425	48,32%	651146	62,74%	795903	76,69%
Uncertain	617464	59,50%	536395	51,68%	386674	37,26%	241903	23,31%
Certain-I	1	0,00%	0	0,00%	0	0,00%	14	0,00%
Method	**CRSMA**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	458953	44,22%	697331	67,19%	808757	77,93%	888952	85,66%
Uncertain	578866	55,78%	340489	32,81%	229063	22,07%	148854	14,34%
Certain-I	0	0,00%	0	0,00%	0	0,00%	14	0,00%

Table 17 – Shadow Map Variation of the pixel precision results, using a fixed viewport of 1024x1024 for the With-Flowers scene; C - Correct and I - Incorrect.

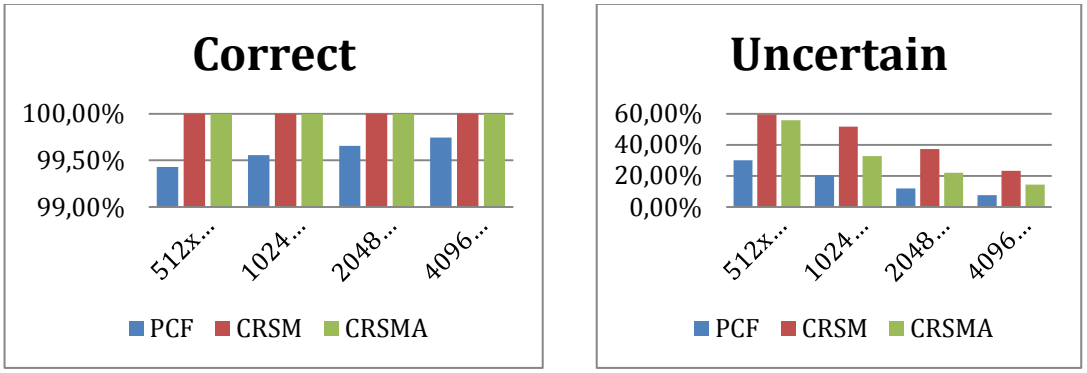


Figure 53 – Shadow Map size Variation results using a fixed viewport of 1024x1024 for the With-Flowers scene; (Left) Correct pixel percentages (Certain-C and Uncertain); (Right) Uncertain pixel percentages.

Table 18 shows the effect of the shadow map size in the amount of correct/incorrect pixels by hybrid methods in the Sponza. Due to the complexity of the scene, the number of errors obtained by the hybrid methods is substantially higher than the in the other scenes.

Scene	Sponza							
Viewport	Main							
Method	**PCF**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	358134	43,17%	555040	66,90%	709905	85,57%	754311	90,92%
Uncertain	458780	55,30%	260292	31,37%	101968	12,29%	57013	6,87%
Certain-I	12729	1,53%	14311	1,72%	17770	2,14%	18319	2,21%
Method	**CRSM**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	383395	46,21%	325593	39,24%	516196	62,22%	596407	71,89%
Uncertain	444708	53,60%	501774	60,48%	306366	36,93%	224780	27,09%
Certain-I	1540	0,19%	2276	0,27%	7081	0,85%	8456	1,02%
Method	**CRSMA**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	399318	48,13%	346709	41,79%	535883	64,59%	631836	76,16%
Uncertain	428785	51,68%	480658	57,94%	286679	34,55%	189351	22,82%
Certain-I	1540	0,19%	2276	0,27%	7081	0,85%	8456	1,02%

Table 18 – Shadow Map Variation of the pixel precision results, using a fixed viewport of 1024x1024 for the Sponza scene; C - Correct and I - Incorrect.

Like in the Flowers scene, the increase of the shadow map size causes PCF to obtain more correct pixels and less uncertain pixels, as Figure 54 shows. However the number of incorrect pixels increases significantly, as the difference between the 512x512 shadow map and the 4096x4096 is of about 0,68% more incorrect pixels.

Figure 54 also shows the results of the increase of the shadow map size for the CRSM and CRSM methods, as it does reduce the number of uncertain pixels, but the number of incorrect pixels obtained by the shadow map increases, as shown in Table 18, resulting in a difference between the 512x512 shadow map and 4096x4096 shadow of about 0,83% more incorrect pixels. The negative effect of the λ is more significant than in the other scenes, as the incorrect pixels by the CRSM and CRSMA reach the 1% mark, but it's still an improvement compared to PCF's 2,21% of incorrect pixels.

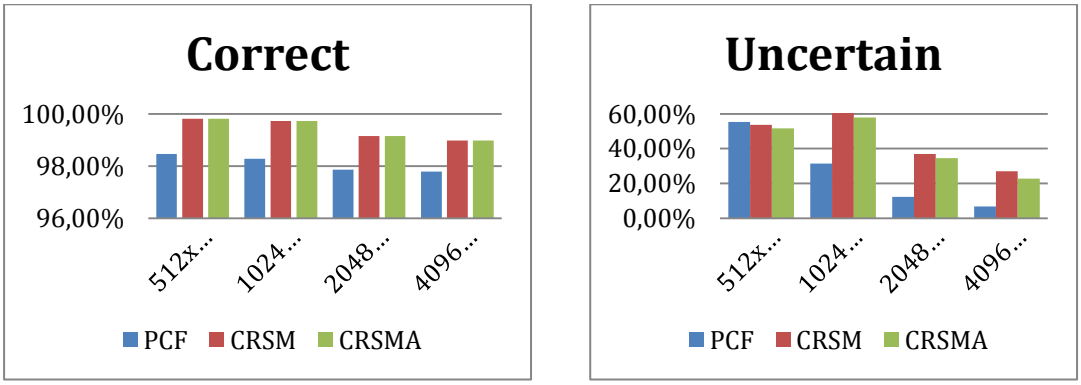


Figure 54 – Shadow Map size Variation results using a fixed viewport of 1024x1024 for the Sponza scene; (Left) Correct pixel percentages (Certain-C and Uncertain); (Right) Uncertain pixel percentages.

5.5 Conclusion

All three hybrid methods provide a significant improvement, regarding pixel accuracy, over NSM shadows, even though PCF's "uncertain" pixels detection algorithm does not eliminate all the incorrect pixels in the image, it still provides an significant reduction, Where the NSM obtain a 3%-5% total incorrect pixels while PCF only obtains 0,5%-2% total incorrect pixels. The low number of uncertain pixels also has the advantage of creating fewer rays to be tracing reducing the OptiX Prime ray tracing time, as detailed further in the next chapter.

The CRSM and the CRSMA provide the best visual results over PCF's, reaching the 0%-1% total incorrect pixels marks, but they produce more "uncertain" pixels, which in turn will increase the OptiX Prime ray tracing time. The use of adjacency information to cover the "tears" in the shadows does provide significant decreases in simple scenes; in more complex scenes however these benefits are too small to compensate the overhead cost in the geometry shader, which the next chapter demonstrate in more detail.

6 Performance Testing

In this section, we'll analyse the performance statistics, mainly the execution time and frame rate, of the three main algorithms: PCF, CRSM and CRSMA. The tests were performed on a laptop with a GeForce GTX770M, with 4GB RAM. The results were obtained using a specific pipeline with the following steps:

1. Render the shadows from the point of view of the light(s) and create the shadow map;
2. Project the shadows in the shadow map and save the result in a colour texture and save the "uncertain pixels" in a colour-coded map of the scene;
3. Analyse colour-coded map, and for each "uncertain" pixel create a ray and store it in a ray Buffer;
4. Send the ray buffer to the OptiX Prime to trace the rays and store the results in a hits buffer;
5. Render the final image using the "certain" pixels in the colour texture and the ray tracing results in the hits buffer.

Figure 55 present a diagram of the test pipeline used, and what passes and material are associated with each step.

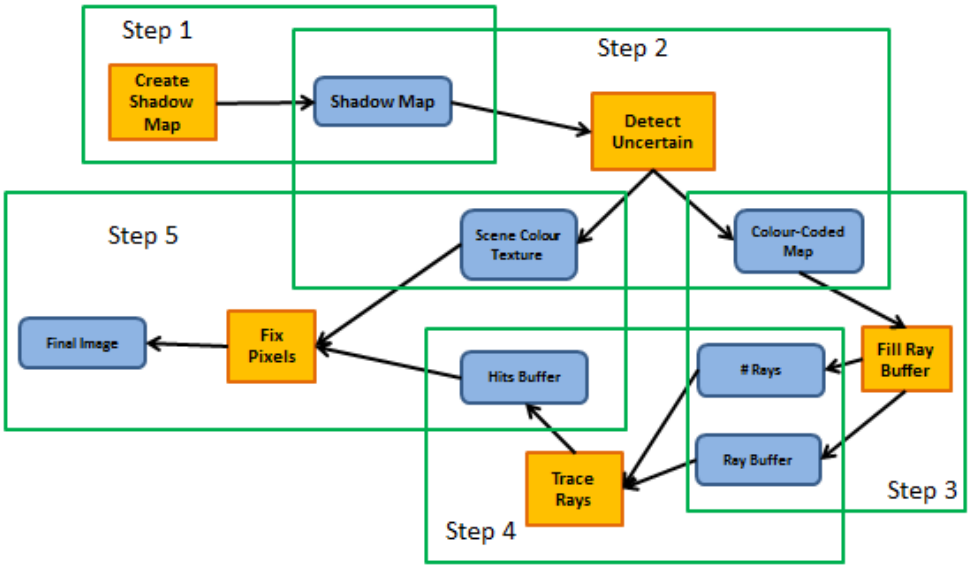


Figure 55 – Pipeline Box Diagram; The Orange Boxes represent pipeline passes while the Blue boxes represent materials (buffers, textures, variables) used and produced by the pipeline passes.

The CRSM and the CRSMA create a two layer shadow map, in order to access the SGSM and the BGSM, so in practical terms, the Detect Uncertain step reads from two shadow maps instead of just the one from PCF's pipeline.

The results presented in this chapter were achieved in the rendering of the Sponza scene, detailed in the previous chapter. The results for the other test scenes can be found in section A.2 of the Appendix.

6.1 OptiX Prime Ray Testing

Before we proceed to Sponza scene results, we will analyse the time required by OptiX Prime to perform trace rays. Since OptiX Prime is an external API used in this pipeline, we to take in consideration the time required to setup the tracing engine, and how the number of rays affects the time need to an obtain a full image.

Note that this solution is itself hybrid, since a rasterization pass is used to provide the shadow ray information to OptiX. The following table (Table 19) shows the execution time of the OptiX Prime API to produce an image of the Sponza scene in the most common resolutions sizes. These results provide a practical comparison stage for the step 3 of the test pipeline. The number of rays traced by the OptiX Prime is equal to the screen size, as shown in the "#Rays" line.

OptiX Prime Execution Time						
Width x Height	800 x 600	1024 x 600	1024 x 768	1280 x 720	1280 x 768	1152 x 864
# Rays	480000	614400	786432	921600	983040	995328
Time (ms)	8,31	10,16	12,39	14,06	14,98	15,17
Width x Height	1280 x 800	1360 x 768	1366 x 768	1280 x 960	1440 x 900	1280 x 1024
# Rays	1024000	1044480	1049088	1228800	1296000	1310720
Time (ms)	15,41	15,73	15,81	18,20	19,18	18,98
Width x Height	1600 x 900	1400 x 1050	1680 x 1050	1600 x 1200	1920 x 1080	
# Rays	1440000	1470000	1764000	1920000	2073600	
Time (ms)	21,50	21,00	25,49	26,89	28,80	

Table 19 – OptiX Prime time results (milliseconds) to render the Sponza scene, with common screen display sizes.

As shown in Table 19, the execution time of the OptiX Prime tends to follow a linear curve as the image resolution increases, as shown in Figure 56.

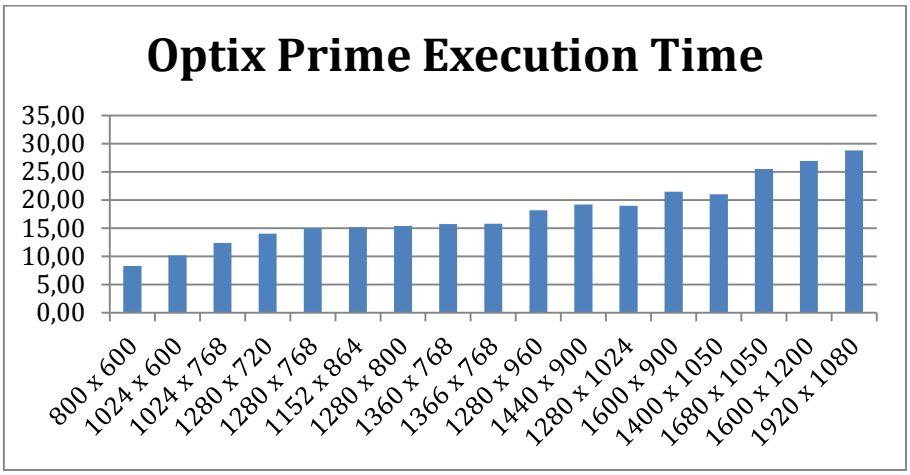


Figure 56 - Graphical representation of the OptiX Prime time results to render the Sponza scene (milliseconds/screen size).

6.2 Execution Time of the Create Shadow Map Step

In this section, we'll analyse the execution time that the hybrid methods need to produce the shadow maps (Figure 57).

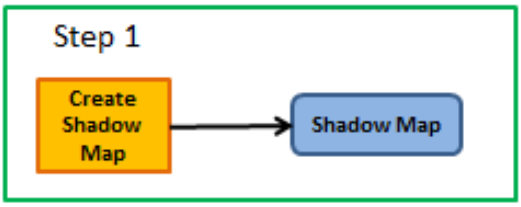


Figure 57 – Create Shadow Map step diagram.

Table 20 shows clearly that conservative rasterization process impacts time required to make the shadow map, as PCF is faster to produce a single shadow map, while CRSM and CRSMA take longer because they create a two layer shadow map.

Execution Time (ms): Create Shadow Map step			
Shadow Map Size	PCF	CRSM	CRSMA
512x512	0,14	1,53	10,84
1024x1024	0,16	1,74	12,98
2048x2048	0,26	2,42	17,84
4096x4096	0,63	3,04	24,30

Table 20 - Execution times of the create shadow map step for the Sponza scene.

The time results also show that the use of that adjacent information, with the respective computations demonstrated in chapter 4, causes the create shadow map step in CRSMA to take more time to produce the shadow maps than CRSM. As shown in Table 20, for the 512x512 shadow map CRSMA takes approximately 11 milliseconds more than CRSM. This difference between becomes more pronounced as the shadow map increases, as shown in Figure 58.

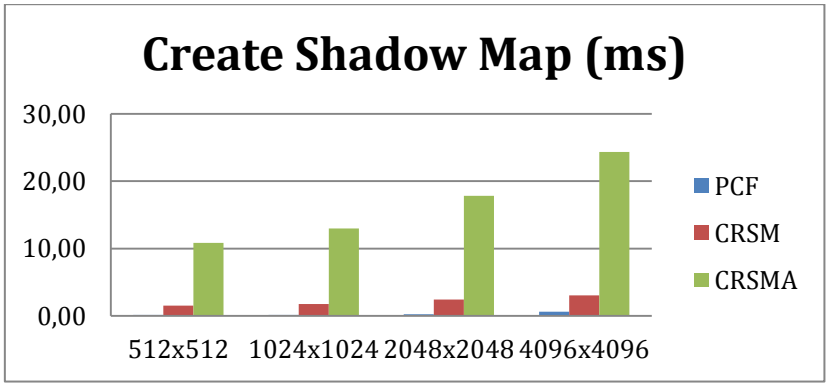


Figure 58 – Graphical representation of the execution times of the Create Shadow Map step for the Sponza scene.

6.3 Execution Time of the Uncertain Detection Step

In this section, we'll analyse the execution time that the hybrid methods need to detect uncertain pixels, and the time required to produce the colour-coded map and the scene colour texture (Figure 59).

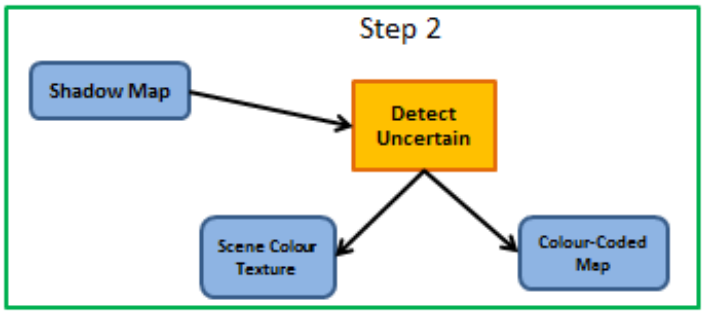


Figure 59 - Detect Uncertain step diagram.

As shown in Table 21, all PCF and CRSM and CRSMA achieve similar results. This proves that the reading of the two layer shadow map spends the same amount of time as the regular shadow map, and that the detection algorithm doesn't have much impact in the execution time.

Execution Time (ms): Detect Uncertain step				
Viewport Size	Shadow Map Size	PCF	CRSM	CRSMA
512 x 512	512x512	0,64	0,65	0,65
	1024x1024	0,64	0,65	0,65
	2048x2048	0,65	0,65	0,65
	4096x4096	0,65	0,64	0,77
1024 x 1024	512x512	2,17	2,17	2,18
	1024x1024	2,17	2,17	2,17
	2048x2048	2,17	2,17	2,18
	4096x4096	2,17	2,22	2,23
1920 x 1080	512x512	4,04	4,04	4,04
	1024x1024	4,03	4,05	4,04
	2048x2048	4,04	4,04	4,04
	4096x4096	4,03	4,05	4,05

Table 21 - Execution times of the Detect Uncertain step for the Sponza scene.

6.4 Execution Time of the Fill Ray Buffer Step

In this section, we'll analyse the execution time that the hybrid methods need to produce the ray buffer (Figure 60).

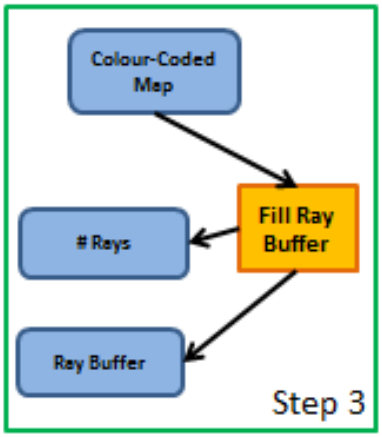


Figure 60 – Fill Ray Buffer step diagram.

As shown in Table 22 and Figure 61, the increase of the shadow map resolution decreases the time required to fill the ray buffer, as this is tied to the number of uncertain pixels found during the previous step. Although there are cases where the numbers of uncertain pixels are similar, the fact that the color-coded map is the same size as viewport, the time it takes to fill the buffer for the same amount of rays, might be different because of the viewport size.

Execution Time (ms): Fill Ray Buffer step							
Viewport Size	Shadow Map Size	PCF		CRSM		CRSMA	
		#Rays	Time	#Rays	Time	#Rays	Time
512 x 512	512x512	114669	2,07	171006	2,72	163432	2,62
	1024x1024	65058	1,40	125406	2,29	120168	2,20
	2048x2048	25400	0,87	76565	1,69	71613	1,58
	4096x4096	14245	0,59	56138	1,24	47295	1,16
1024 x 1024	512x512	458780	7,39	684174	10,15	653842	9,73
	1024x1024	260292	4,79	501774	8,31	480658	7,84
	2048x2048	101968	2,61	306366	5,92	286679	5,47
	4096x4096	57013	1,65	224780	4,39	189351	3,83
1920 x 1080	512x512	1029018	14,85	1349756	20,08	1273137	18,24
	1024x1024	614562	10,34	988492	15,96	937996	14,53
	2048x2048	240378	5,09	520567	9,62	484444	8,55
	4096x4096	98918	2,36	325214	6,20	294871	5,45

Table 22 - Execution times of the fill ray buffer step for the Sponza scene; the # Rays is equal to the number of "uncertain" pixels found in the Detect Uncertain step.

As expected, PCF's low number of "uncertain" pixels make it so that the ray buffer is filled much faster than the other two methods, by an average of approximately 0,5 milliseconds for the 512x512 viewport, and average of 3 milliseconds for the 1024x1024 viewport, and an average of 5 milliseconds for the 1920x1080 viewport.

The differences between CRSM and CRSMA are very small, with an average difference of 0,10 milliseconds for the 512x512, and average difference of 1,5 difference of the rest of the viewports, which isn't sufficient to compensate the CRSMA's overhead in the create shadow map step.

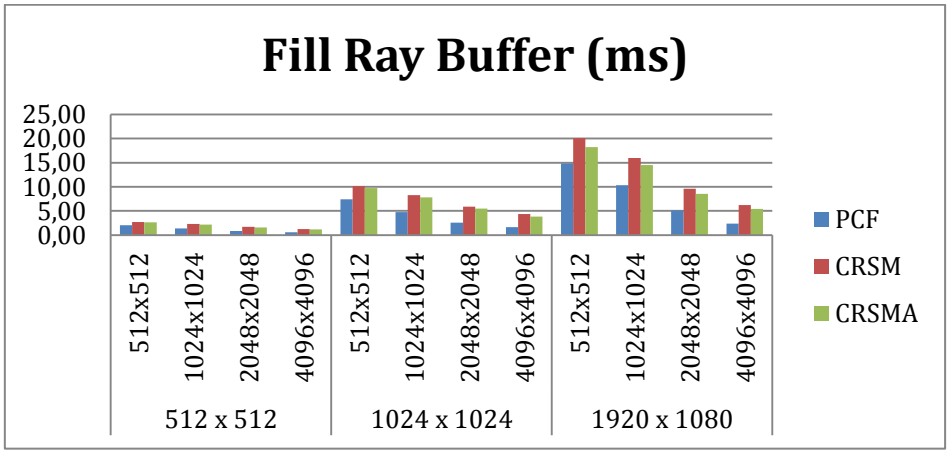


Figure 61 - Graphical representation of the execution times of the Fill Ray Buffer step for the Sponza scene.

6.5 Execution Time of the Traced Ray Step

In this section, we'll analyse the execution time that the OptiX Prime requires to trace the ray buffer created in the previous step (Figure 62).

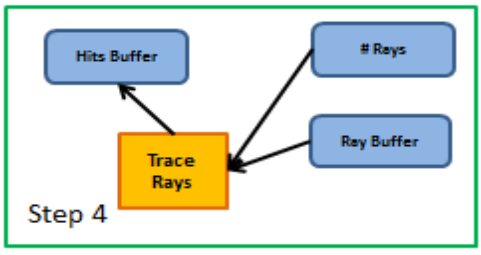


Figure 62 – Trace Rays Step diagram

Table 23 and Figure 63 show that the number of rays reduces for each viewport as the shadow map increases, causing the time required to trace the rays created from the "uncertain" pixels to reduce, similarly like in the previous step. However there are differences in the times, because unlike the previous step, the OptiX Prime API is not affected by the viewport size, only by the number of rays needed to trace, therefore, some combinations of viewport and shadow map sizes produce very similar results.

Execution Time (ms): Trace Rays step							
Viewport Size	Shadow Map Size	PCF		CRSM		CRSMA	
		#Rays	Time	#Rays	Time	#Rays	Time
512 x 512	512x512	114669	2,38	171006	3,08	163432	3,03
	1024x1024	65058	1,68	125406	2,60	120168	2,55
	2048x2048	25400	1,13	76565	2,01	71613	1,92
	4096x4096	14245	0,89	56138	1,67	47295	1,56
1024 x 1024	512x512	458780	6,69	684174	9,08	653842	8,89
	1024x1024	260292	4,42	501774	7,39	480658	7,16
	2048x2048	101968	2,47	306366	5,32	286679	4,99
	4096x4096	57013	1,70	224780	4,21	189351	3,70
1920 x 1080	512x512	1029018	12,79	1349756	16,53	1273137	15,40
	1024x1024	614562	8,55	988492	13,01	937996	12,27
	2048x2048	240378	4,28	520567	7,99	484444	7,48
	4096x4096	98918	2,37	325214	5,62	294871	5,22

Table 23 - Execution times of the trace rays step for the Sponza scene; the # Rays is equal to the number of "uncertain" pixels found in the Detect Uncertain step.

Like in the previous step, PCF's low number of rays makes it the fastest to obtain the needed ray tracer results, having on average of 1 milliseconds advantage in the 512x512 viewport, and average advantage of 3,5 milliseconds for the other view ports. Also like in the previous section, CRSM and CRSMA differences are too small to compensate the use of adjacency information, with an average improvement of 0,5 milliseconds of the ray tracing time needed to correct.

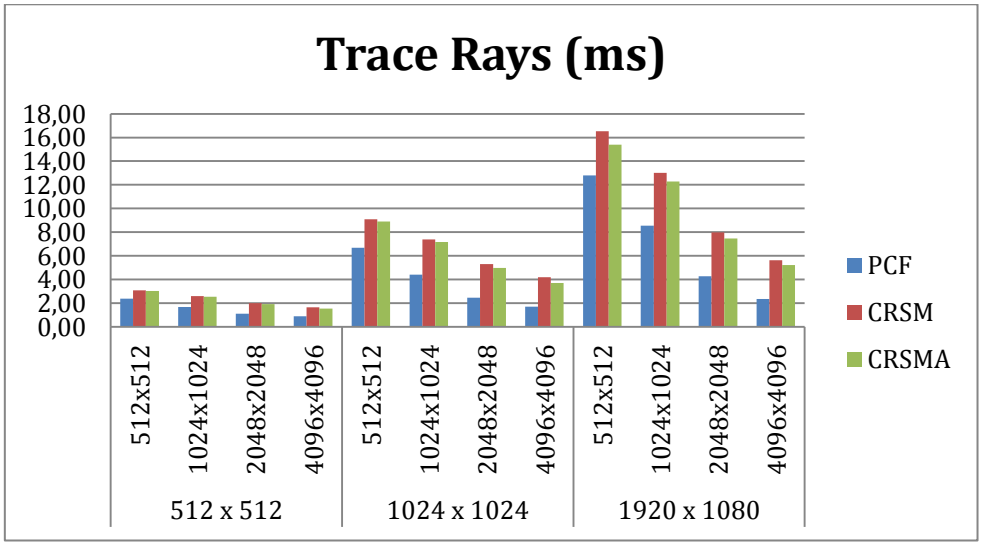


Figure 63 - Graphical representation of the execution times of the Trace Rays step for the Sponza scene.

6.6 Execution Time of the Fix Pixels Step

In this section, we'll analyse the execution time that the OptiX Prime requires to fix the pixels in the scene's shadows to produce the final image (Figure 64).

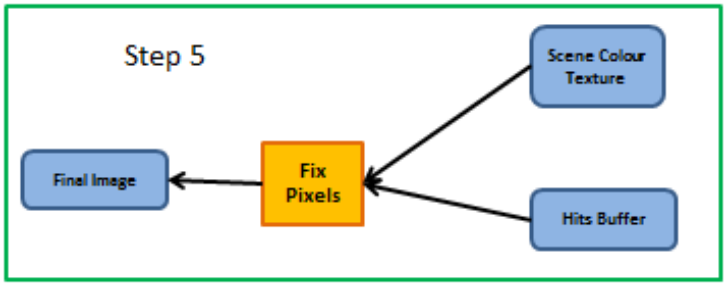


Figure 64 – Fix Pixels step diagram

Like in the step 2 of the pipeline, the hybrid methods achieve the same amount of time results, as Table 24 shows, due to the fact that in this step we are basically applying to the colour texture the results in the hits buffer specifically in the positions of the "uncertain" pixels detected before.

Execution Time (ms): Fix Pixels step			
Viewport Size	PCF	CRSM	CRSMA
512 x 512	0,07	0,07	0,07
1024 x 1024	0,29	0,29	0,29
1920 x 1080	0,57	0,57	0,57

Table 24 - Execution times of the fix pixels step for the Sponza scene.

6.7 Execution Time of the Entire Pipeline

Finally, we'll analyse the combined times of each time in order to evaluate the hybrid methods in the entire pipeline. The results are present in Table 25.

Execution Time (ms): Pipeline						
Viewport Size	Shadow Map Size	OptiX Prime	NSM	PCF	CRSM	CRSMA
512 x 512	512x512	4,96	0,50	5,30	8,05	17,21
	1024x1024		0,53	3,95	7,35	18,45
	2048x2048		0,65	2,98	6,84	22,06
	4096x4096		1,11	2,83	6,66	27,86
1024 x 1024	512x512	15,32	0,90	16,68	23,22	31,93
	1024x1024		0,94	11,83	19,90	30,44
	2048x2048		1,07	7,80	16,12	30,77
	4096x4096		1,57	6,44	14,15	34,35
1920 x 1080	512x512	28,73	1,47	32,39	42,75	49,09
	1024x1024		1,50	23,65	35,33	44,39
	2048x2048		1,59	14,24	24,64	38,48
	4096x4096		1,88	9,96	19,48	39,59

Table 25 - Execution times of the test pipeline for the Sponza scene.

As evidenced by the values in Table 25, PCF method is the fastest of three hybrids, having an average difference of 4 milliseconds and 12 milliseconds difference over the CRSM and CRSMA respectively for the 512x512 viewport, and a 10 millisecond and 20 milliseconds difference over the CRSM and CRSMA in the other viewports. The CRSM provides faster times than the CRSMA due to the lack of geometry manipulations in the create shadow map step, giving it and advantage of approximately 15 milliseconds over the CRSMA. These results show that, as of now, the use of the adjacency information to cover the "tears" in the shadows is not a viable solution. The resulting in the following

section will collaborate in this assessment with the frame rate analysis of the hybrid methods. Figure 65 provides a graphical representation of these results, providing visual confirmation to this analysis.

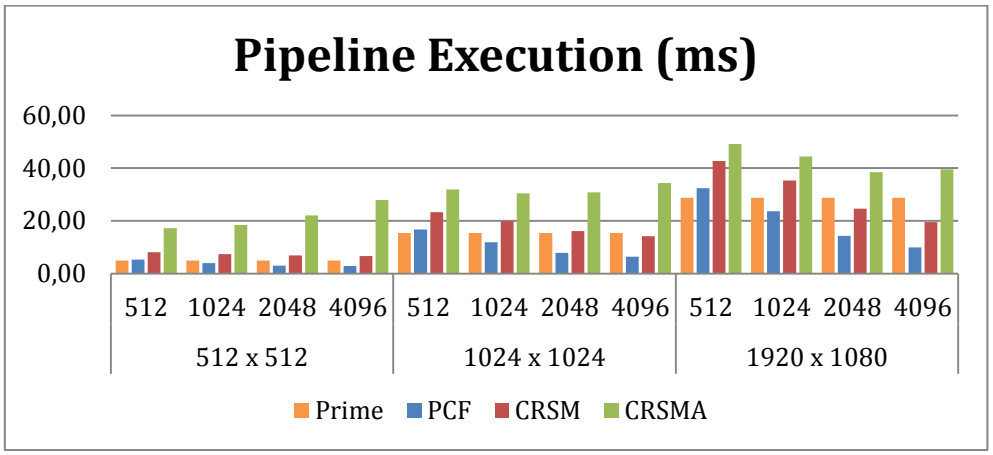


Figure 65 - Graphical representation of the execution times of the Pipeline for the Sponza scene.

6.8 Frame Rate for the Entire Pipeline

In this section, we'll analyse the frame rate achieved with these hybrid methods and compare them with the frame rate for OptiX Prime and NSM pipeline.

Frame Rate (FPS)						
Viewport Size	Shadow Map Size	OptiX Prime	NSM	PCF	CRSM	CRSMA
512 x 512	512	136,01	384,62	89,77	63,56	39,30
	1024		384,17	114,89	70,75	37,97
	2048		364,70	137,61	78,81	34,26
	4096		297,89	145,01	70,82	28,41
1024 x 1024	512	55,37	316,46	34,46	24,82	20,21
	1024		325,41	48,60	29,60	22,21
	2048		313,87	73,51	38,12	23,88
	4096		257,00	87,99	40,50	22,47
1920 x 1080	512	31,03	277,55	17,92	13,48	12,81
	1024		249,94	25,60	17,10	15,06
	2048		239,58	44,90	26,19	19,43
	4096		230,79	63,47	32,50	20,69

Table 26 – Frame rate of the test pipeline for the Sponza scene.

As shown by Table 26 and Figure 66, NSM provides the highest frame rate possible in the Sponza scene. Within the hybrids methods, PCF provides the highest frame rate in all the viewports, only surpassing the OptiX Prime in the 2048x2048 and 4096x4096 shadow map sizes.

The CRSM provides faster frame rates than CRSMA as expected because of the geometric transformations of CRSMA, just like in the previous section. These methods also provide improvements as the shadow map size increases, but rarely provide superior times than OptiX Prime. The only case of improvement is with the 1920x1080 viewport and the 4096x4096 shadow map, where the difference between the CRSM and OptiX Prime is approximately 1millisecond, which is not significant enough to use CRSM over OptiX Prime, as far as performance is concerned.

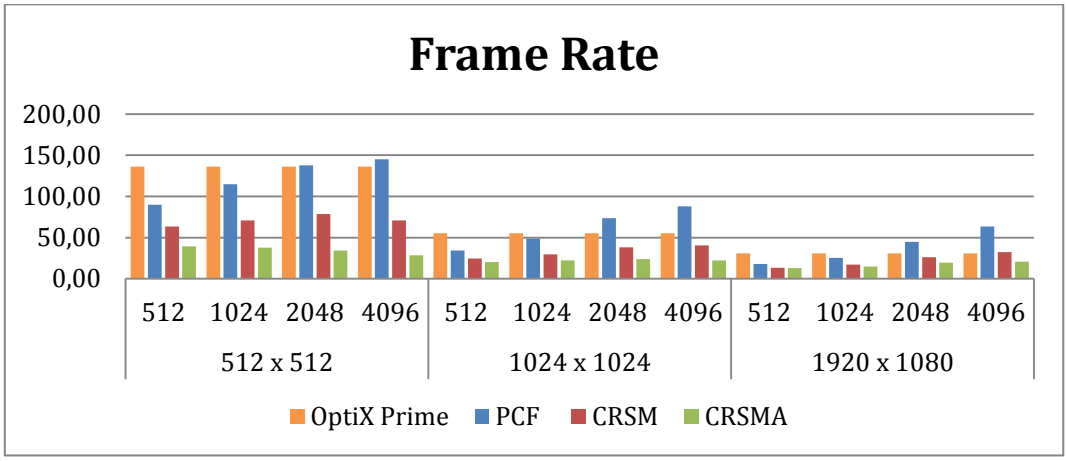


Figure 66 - Graphical representation of the frame rate (FPS), shown in Table 26. The first number represents the shadow map size, while the second represent the viewport dimensions.

6.9 Conclusion

In accordance to the test pipeline, the PCF obtains the fastest times, having a slight advantage over the CRSM, due to the low number of "uncertain" pixels, However, taking in consideration the results of the previous chapter, it produces significant visual errors. The CRSM provides a compromise between relatively performance and a low number of visual errors, always stands in the middle of the PCF's and CRSMA's times. CRSMA's time results show that the use of adjacency information to reduce the number of uncertain rays does provide a slight decrease in the OptiX Prime's ray tracing time, but

these improvements don't compensate the time overhead in the geometry shader, which increases the execution time of CRSMA method quite drastically.

In regard to the frame rate obtained by these hybrid methods, the hybrid methods achieve worst frame rates than the OptiX Prime, the only exception to this is the PCF when the shadow map size is 2048x2048 or above. Taking in consideration the pixels analyses results of the previous chapter, by increasing the shadow map size in the Sponza scene, it begins to introduce visual errors, so the OptiX Prime ray tracing engine provides superior results.

7 Conclusions and Future Work

Shadow mapping (Williams, 1978) is probably the most known and explored method to produce shadows. Conceptually it is not an error free algorithm due to perspective aliasing, perspective mismatch, the discrete nature of the z-buffer among other issues.

Several proposals have partially solved this issues resulting in methods that have far less errors than the standard approach while maintaining high performance even with complex scenes. Trapezoidal shadow maps (Martin & Tiow-Seng, Trapezoidal Shadow Maps (TSM) - Recipe, 2011) and parallel split shadow maps (Zhang, Sun, Xu, & Lun, 2006) are two examples. These methods explore the shadow map useful area, thereby providing better results.

Other approaches such as Percentage Closer Filtering (Reeves, Salesin, & Cook, 1987) use heuristics to improve the shadows using a kernel filter. More recent developments such as VSM (Donnelly & Lauritzen, 2006) and ESM (Annen, Mertens, Seidel, Flerackers, & Kautz, 2008) extend the filtering idea reinterpreting the shadow map such that it can be freely filtered. All these later methods provide some sort of soft shadows, where VSM and ESM do create an penumbra, whereas PCF only blurs the shadowing results, not taking into account the distance between the occluder and the occlude.

Ray tracing (Glassner, 1989) is capable of producing error free hard shadows with an amazingly elegant algorithm. However its performance is far from the previous rasterization based methods.

Today, with the versatility of the graphics hardware it is possible to implement many rendering paradigms in the GPU. Rasterization and ray tracing are two of the most popular paradigms and this thesis focus on the combination of both using as a case study the production of hard shadows.

Two previous works have already attempted that, but no comparative evaluation has been performed so far, both between these methods, as well as against more ray tracing intensive solutions. This work explored the feasibility of these methods, evaluating their quality and performance.

The essence behind these approaches is a method to determine the correctness of the shadow status of each pixel. While CRSM (Hertel, Hormann, & Westermann, 2009) ensures an error free method, at least theoretically, PCF (Beister, Ernst, & Stamminger, 2005) provides a heuristic that is able to determine the shadow correctness for most pixels. However, in the tested scenes a significant number of incorrect pixels are not detected by this heuristic, and as shown these can have a significant visual

impact. Both these methods use a selective ray tracing aid, seeking to reduce the ray tracer load. Pixels which cannot be classified as correct by the method are then classified by a ray tracer.

To explore the accuracy of the detection of possibly incorrect pixels in the shadow map an extension to CRSM (Hertel, Hormann, & Westermann, 2009) using adjacency information is also presented (CRSMA). The main purpose of this extension is to ascertain the shadow status of the pixels in the gaps between adjacent triangles created when the triangles are shrunk. This results in more pixels being classified as correct, which in turn reduces even further the number of pixels that must be sent to the ray-tracer to determine their shadow status.

The hybrid methods analysed in this dissertation provide a significant visual improvement to standard shadow mapping producing less errors. The results show that even PCF with its heuristic approach can significantly improve the shadow map results, while both CRSM and CRSMA provide an almost error free solution. The advantage of the usage of adjacency information varies a lot from scene to scene, having less impact on scenes with smaller triangles, yet it consistently requires less intervention from the ray tracer.

From a performance point of view, being able to ascertain the correctness of the shadow status for a large number of pixels results in a faster ray tracing step, since as shown the performance of the ray tracer varies in a linear fashion with the number of rays.

However, determining the correctness of the shadow status of each pixel requires previous work preparing a shadow map that helps with this evaluation. Having to shrink and extend triangles takes its time and the usage of adjacency information to fill in the gaps has a huge performance penalty, rendering this extension useless with current hardware. CRSM is able to perform faster than the OptiX solution but it requires a large shadow map.

The main disadvantage of using OptiX is the memory cost, since it requires a duplicate of the scene. Note that, as mentioned before, the OptiX solution is itself hybrid.

Regarding future work there are two avenues that may be worth exploring:

Test CRSM and CRSMA with NVIDIA Maxwell architecture (NVIDIA Corporation, 2014) which offers hardware support for conservative rasterization;

Test with other ray tracing engines, namely Embree™ (Wald, Woop, Benthin, Johnson, & Ernst, 2014) and AMD's FireRays™ (Trudeau, 2015). In particular Embree could prove to be a useful ally for the hybrid rasterization approach since it works exclusively on the CPU, effectively leaving the GPU free for other effects.

References

Abrantes, J. (2009). *Analysis and Proposal of a Shadow Mapping Remix Approach (Masters Thesis).* School of Engineering, Department of Informatics. Braga: University of Minho.

Akenine-Moller, T., & Aila, T. (2005). Conservative and Tiled Rasterization Using a Modified Triangle Setup. *Journal of Graphics Tools, 10*, 1-8.

Annen, T., Mertens, T., Seidel, H.-P., Flerackers, E., & Kautz, J. (2008). Exponential Shadow Maps. *Proceedings of Graphics Interface 2008. GI '08*, pp. 155-161. Windsor, Ontario, Canada: Canadian Information Processing Society.

Beister, M., Ernst, M., & Stamminger, M. (2005). A Hybrid GPU-CPU Renderer. In G. Greiner, J. Hornegger, H. Niemann, & M. Stamminger (Ed.), *Vision, Modeling, and Visualization 2005*, (pp. 415-420). Berlin: Akademische Verlagsgesellschaft Aka GmbH.

Cabeleira, J. P. (2010). *Combining Rasterization and Ray Tracing Techniques to Approximate Global Illumination in Real-Time (Masters Thesis).* Instituto Superior Técnico, Department of Computer Science and Engineering. Lisboa: Technical University of Lisbon.

Christensen, P., Fong, J., Laur, D., & Batali, D. (2006). Ray Tracing For The Movie "Cars". *IEEE Symposium on Interactive Ray Tracing*, 1-6.

Cook, R., Carpenter, L., & Catmull, E. (1987). The Reyes Image Rendering Architecture. *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87* (pp. 95-102). New York, NY, USA: ACM.

Donnelly, W., & Lauritzen, A. (2006). Variance Shadow Maps. *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (pp. 161-165). Redwood City, California: ACM.

Glassner, A. S. (1989). *An Introduction to Ray Tracing.* London, UK: Academic Press Ltd.

Hasselgren, J., Akenine-Möller, T., & Ohlsson, L. (2005). Conservative Rasterization. In *GPU Gems 2* (pp. 677-690). Addison-Wesley Professional.

Hertel, S., Hormann, K., & Westermann, R. (2009). A Hybrid GPU Rendering Pipline for Alias-Free Hard Shadows. *Proceedings of Eurographics 2009 Area* (pp. 59–66). München, Germany: Eurographics Association.

Koltun, V., Cohen-Or, D., & Chrysanthou, Y. (2001). Hardware-accelerated from-region visibility using a dual ray space. *Eurographics Workshop on Rendering* (pp. 204-214). The Eurographics Association.

Lloyd, B., Wendt, J., Govindaraju, N., & Manocha, D. (2004). CC Shadow Volumes. *In Proceedings of the 2nd EG Symposium on Rendering.* Eurographics Association.

Martin, T., & Tiow-Seng, T. (2004). Anti-aliasing and Continuity with Trapezoidal Shadow Maps. *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques* (pp. 153-160). Norrköping, Sweden: Eurographics Association.

Martin, T., & Tiow-Seng, T. (2011, May 2). *Trapezoidal Shadow Maps (TSM) - Recipe*. (School of Computing, National University of Singapore) Retrieved September 15, 2015, from Anti-aliasing and Continuity with Trapezoidal Shadow Maps: http://www.comp.nus.edu.sg/~tants/tsm/TSM_recipe.html

Meinl, F. (2011, July 27). *McGuire Graphics Data - Meshes.* Retrieved December 5, 2015, from McGuire Graphics Data: http://graphics.cs.williams.edu/data/meshes.xml

Moderno, D. (2011). *Shadow Mapping and Ray-Tracing (Master Thesis).* School of Engineering, Department of Informatics. Braga: University of Minho.

Myszkowski, K., Okunev, O. G., & Kunii, T. L. (1995). Fast Collision detection between complex solids using rasterization graphics hardware. *The Visual Computer 11, 9*, 497-512.

Nealen, A. V. (2002). *Shadow Mapping and Shadow Volumes: Recent Developments in Real-Time (Project Report).* University of British Columbia.

NVIDIA Corporation. (2008). *NVIDIA® OptiX™ Ray Tracing Engine*. Retrieved October 25, 2013, from https://developer.nvidia.com/optix

NVIDIA Corporation. (2014, November 19). *Don't be conservative with Conservative Rasterization*. (NVIDIA Corporation) Retrieved December 18, 2015, from NVIDIA Developer Zone: https://developer.nvidia.com/content/dont-be-conservative-conservative-rasterization

Reeves, W., Salesin, D., & Cook, R. (1987). Rendering Antialiased Shadows with Depth Maps. *SIGGRAPH Comput. Graph., 21*(4), 283-291.

Tadamura, K., Qin, X., Jiao, G., & Nakamae, E. (1999). Rendering optimal solar shadows using plural sunlight depth buffers. *Proceedings of the International Conference on Computer Graphics.* Washington, DC, USA: IEEE Computer Society.

Trudeau, J. (2015, August 14). *AMD FireRays Library Accelerates Ray Tracing*. (AMD) Retrieved 11 25, 2015, from http://developer.amd.com/community/blog/2015/08/14/amd-firerays-library/

Wald, I., Woop, S., Benthin, C., Johnson, G. S., & Ernst, M. (2014). Embree: A Kernel Framework for Efficient CPU Ray Tracing. *ACM Trans. Graph., 33*, 143:1-143:8.

Williams, L. (1978). Casting Curved Shadows on Curved Surfaces. *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 270-274). New York, NY, USA: ACM.

Zhang, F., Sun, H., Xu, L., & Lun, L. (2006). Parallel-split Shadow Maps for Large-scale Virtual Environments. *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications* (pp. 311-318). Hong Kong, China: ACM.

A Appendix

A.1 Pixel Analysis and Results

A.1.1 OptiX Prime

Scene	Bench					
Viewport	Against					
Pixel Types	Viewport Size					
	512x512		1024x1024		1920x1080	
PFL	255894	97,62%	1023625	97,62%	2045839	98,66%
PnFL	6250	2,38%	24951	2,38%	27761	1,34%
PFL – in Light	151055	59,03%	604274	59,03%	1296685	63,38%
PFL – in Shadow	104839	40,97%	419351	40,97%	749154	36,62%

Table 27 - OptiX Prime results for the Bench scene with the Against viewport; PFL represents the Pixels Facing the Light and PnFL represents the Pixels not Facing the Light; The PFLs are then split into the pixels in Light and the pixels in Shadow.

Figure 67 - OptiX Prime Shadows in the Bench scene using the Against viewport; the green pixels represent the PNFL pixels; Viewport Size: 1024x1024.

Scene	Bench					
Viewport	Side					
Pixel Types	Viewport Size					
	512x512		1024x1024		1920x1080	
PFL	248797	94,91%	995221	94,91%	2014185	97,13%
PnFL	13347	5,09%	53355	5,09%	59415	2,87%
PFL – in Light	181422	72,92%	725643	72,91%	1493410	74,14%
PFL – in Shadow	67375	27,08%	269578	27,09%	520775	25,86%

Table 28 - OptiX Prime results for the Bench scene with the Side viewport; PFL represents the Pixels Facing the Light and PnFL represents the Pixels not Facing the Light; The PFLs are then split into the pixels in Light and the pixels in Shadow.

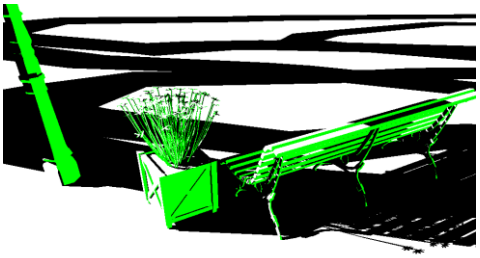


Figure 68 - OptiX Prime Shadows in the Bench scene using the Side viewport; the green pixels represent the PNFL pixels; Viewport Size: 1024x1024.

Scene	Bench					
Viewport	With					
Pixel Types	Viewport Size					
	512x512		1024x1024		1920x1080	
PFL	261769	99,86%	1047046	99,85%	2071919	99,92%
PnFL	375	0,14%	1530	0,15%	1681	0,08%
PFL – in Light	170302	65,06%	681183	65,06%	1421847	68,62%
PFL – in Shadow	91467	34,94%	365863	34,94%	650072	31,38%

Table 29 - OptiX Prime results for the Bench scene with the With viewport; PFL represents the Pixels Facing the Light and PnFL represents the Pixels not Facing the Light; The PFLs are then split into the pixels in Light and the pixels in Shadow.

Figure 69 - OptiX Prime Shadows in the Bench scene using the With viewport; the green pixels represent the PNFL pixels; Viewport Size: 1024x1024.

Scene	Flowers					
Viewport	Against					
Pixel Types	Viewport Size					
	512x512		1024x1024		1920x1080	
PFL	246986	94,22%	987818	94,21%	2005998	96,74%
PnFL	15158	5,78%	60758	5,79%	67602	3,26%
PFL – in Light	130891	53,00%	523454	52,99%	1399264	69,75%
PFL – in Shadow	116095	47,00%	464364	47,01%	606734	30,25%

Table 30 - OptiX Prime results for the Flowers scene with the Against viewport; PFL represents the Pixels Facing the Light and PnFL represents the Pixels not Facing the Light; The PFLs are then split into the pixels in Light and the pixels in Shadow.

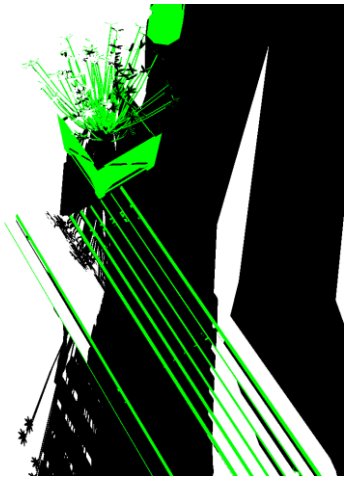


Figure 70 - OptiX Prime Shadows in the Flowers scene using the Against viewport; the green pixels represent the PNFL pixels; Viewport Size: 1024x1024.

Scene	Flowers					
Viewport	Side					
Pixel Types	Viewport Size					
	512x512		1024x1024		1920x1080	
PFL	254287	97,00%	1017330	97,02%	1993921	96,16%
PnFL	7857	3,00%	31246	2,98%	79679	3,84%
PFL – in Light	159907	62,88%	639789	62,89%	1314683	65,93%
PFL – in Shadow	94380	37,12%	377541	37,11%	679238	34,07%

Table 31 - OptiX Prime results for the Flowers scene with the Side viewport; PFL represents the Pixels Facing the Light and PnFL represents the Pixels not Facing the Light; The PFLs are then split into the pixels in Light and the pixels in Shadow.

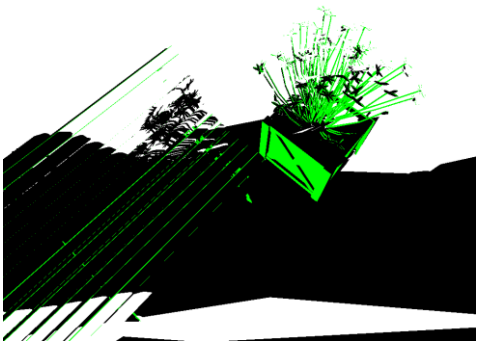


Figure 71 - OptiX Prime Shadows in the Flowers scene using the Side viewport; the green pixels represent the PNFL pixels; Viewport Size: 1024x1024.

Scene	Trees					
Viewport	Against					
Pixel Types	Viewport Size					
	512x512		1024x1024		1920x1080	
PFL	262144	100,00%	1048576	100,00%	2073600	100,00%
PnFL	0	0,00%	0	0,00%	0	0,00%
PFL – in Light	114468	43,67%	457662	43,65%	957000	46,15%
PFL – in Shadow	147676	56,33%	590914	56,35%	1116600	53,85%

Table 32 - OptiX Prime results for the Trees scene with the Against viewport; PFL represents the Pixels Facing the Light and PnFL represents the Pixels not Facing the Light; The PFLs are then split into the pixels in Light and the pixels in Shadow.

Figure 72 - OptiX Prime Shadows in the Trees scene using the Against viewport; the green pixels represent the PNFL pixels; Viewport Size: 1024x1024.

Scene	Trees					
Viewport	Side					
Pixel Types	Viewport Size					
	512x512		1024x1024		1920x1080	
PFL	262144	100,00%	1048576	100,00%	2073445	99,99%
PnFL	0	0,00%	0	0,00%	155	0,01%
PFL – in Light	168949	64,45%	675786	64,45%	1320293	63,68%
PFL – in Shadow	93195	35,55%	372790	35,55%	753152	36,32%

Table 33 - OptiX Prime results for the Trees scene with the Side viewport; PFL represents the Pixels Facing the Light and PnFL represents the Pixels not Facing the Light; The PFLs are then split into the pixels in Light and the pixels in Shadow.

Figure 73 - OptiX Prime Shadows in the Trees scene using the Side viewport; the green pixels represent the PNFL pixels; Viewport Size: 1024x1024.

Scene	Trees					
Viewport	With					
Pixel Types	Viewport Size					
	512x512		1024x1024		1920x1080	
PFL	262144	100,00%	1048576	100,00%	2073600	100,00%
PnFL	0	0,00%	0	0,00%	0	0,00%
PFL – in Light	163824	62,49%	655342	62,50%	1331188	64,20%
PFL – in Shadow	98320	37,51%	393234	37,50%	742412	35,80%

Table 34 - OptiX Prime results for the Trees scene with the With camera; PFL represents the Pixels Facing the Light and PnFL represents the Pixels not Facing the Light; The PFLs are then split into the pixels in Light and the pixels in Shadow.

Figure 74 - OptiX Prime Shadows in the Bench scene using the Side viewport; the green pixels represent the PNFL pixels; Viewport Size: 1024x1024.

A.1.2 Standard Shadow Mapping

Scene	Bench			
Viewport	Against			
Pixel State	Pixel Precision			
	Correct		Incorrect	
Light	593185	97,36%	16085	2,64%
Shadow	403266	97,32%	11089	2,68%
Total	996451	97,35%	27174	2,65%

Table 35 - Normal Shadow Mapping errors in the Bench scene with the Against viewport, with a 1024x1024 viewport and 1024x1024 shadow map.

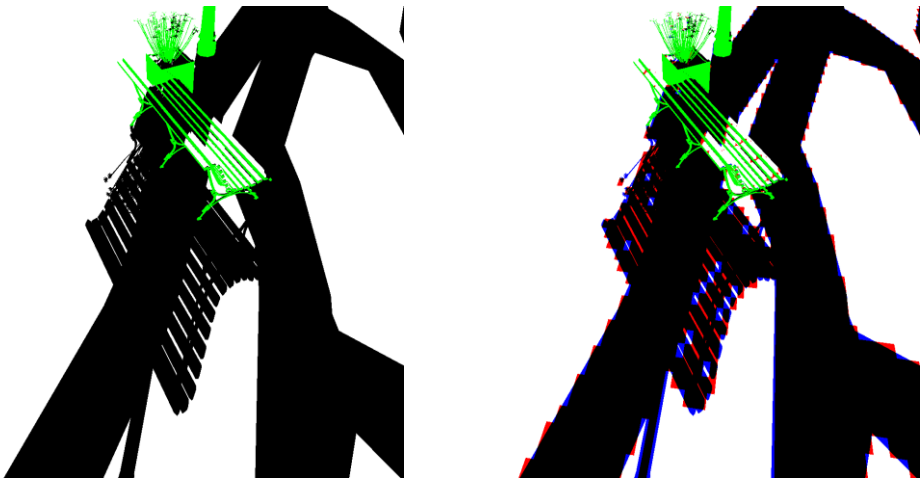


Figure 75 - Against-Bench scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) OptiX Prime results; (Right) NSM results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Bench					
Viewport	Against					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
C-Light	148289	97,35%	593185	97,36%	1266825	97,98%
C-Shadow	100801	97,33%	403266	97,32%	723093	96,03%
I-Light	4038	2,65%	16085	2,64%	26061	2,02%
I-Shadow	2766	2,67%	11089	2,68%	29860	3,97%

Table 36 - NSM viewport results in the Bench scene with the Against viewport, using a fixed shadow map size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

Scene	Bench							
Viewport	Against							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
C-Light	581478	96,54%	593185	97,36%	595705	98,83%	600553	99,13%
C-Shadow	398503	94,59%	403266	97,32%	412292	97,96%	414070	99,11%
I-Light	20848	3,46%	16085	2,64%	7059	1,17%	5281	0,87%
I-Shadow	22796	5,41%	11089	2,68%	8569	2,04%	3721	0,89%

Table 37 - NSM shadow map results in the Bench scene with the Against viewport, using a fixed viewport size of 1024x1024; I represents the incorrect pixels and C represents the correct pixels.

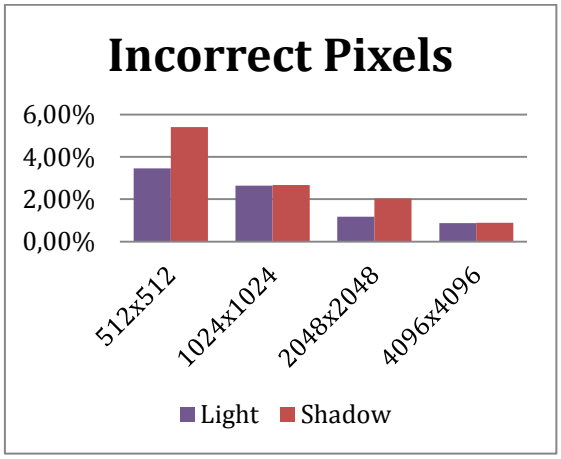


Figure 76 - Shadow Map Size variation results in the Against-Bench scene using a fixed viewport of 1024x1024.

Scene	Bench			
Viewport	Side			
Pixel State	Pixel Precision			
	Correct		Incorrect	
Light	718232	98,76%	8984	1,24%
Shadow	260594	97,23%	7411	2,77%
Total	978826	98,35%	16395	1,65%

Table 38 - Normal Shadow Mapping errors in the Bench scene with the Side viewport, with a 1024x1024 viewport and 1024x1024 shadow map.

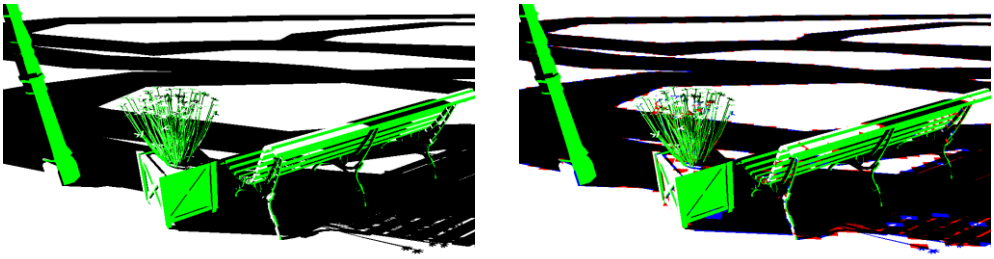


Figure 77 - Side-Bench scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) OptiX Prime results; (Right) NSM results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Bench					
Viewport	Side					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
C-Light	179521	98,72%	718232	98,76%	1474955	98,82%
C-Shadow	65039	97,16%	260594	97,23%	503144	96,46%
I-Light	2336	1,28%	8984	1,24%	17631	1,18%
I-Shadow	1901	2,84%	7411	2,77%	18455	3,54%

Table 39 - NSM viewport results in the Bench scene with the Side viewport, using a fixed shadow map size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

Scene	Bench							
Viewport	Side							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
C-Light	712690	97,81%	718232	98,76%	720739	99,16%	722975	99,39%
C-Shadow	253592	95,14%	260594	97,23%	263452	98,17%	265154	99,00%
I-Light	15986	2,19%	8984	1,24%	6126	0,84%	4424	0,61%
I-Shadow	12953	4,86%	7411	2,77%	4904	1,83%	2668	1,00%

Table 40 - NSM shadow map results in the Bench scene with the Side viewport, using a fixed viewport size of 1024x1024; I represents the incorrect pixels and C represents the correct pixels.

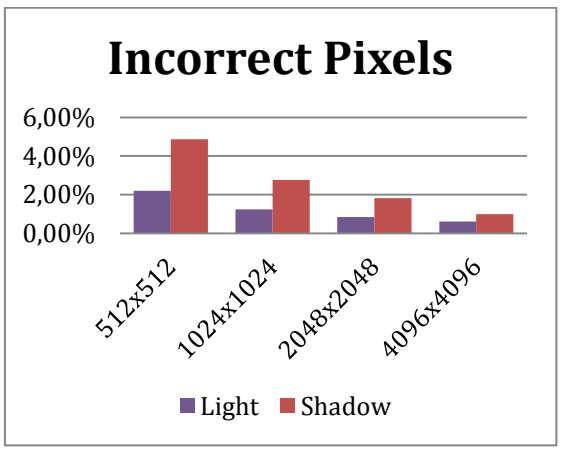


Figure 78 - Shadow Map Size variation results in the Side-Bench scene using a fixed viewport of 1024x1024.

Scene	Bench			
Viewport	With			
Pixel State	Pixel Precision			
	Correct		Incorrect	
Light	672574	98,66%	9109	1,34%
Shadow	356754	97,64%	8609	2,36%
Total	1029328	98,31%	17718	1,69%

Table 41 - Normal Shadow Mapping errors in the Bench scene with the With viewport, with a 1024x1024 viewport and 1024x1024 shadow map.

Figure 79 - With-Bench scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) OptiX Prime results; (Right) NSM results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Bench					
Viewport	With					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
C-Light	168196	98,69%	672574	98,66%	1399990	98,53%
C-Shadow	89226	97,69%	356754	97,64%	629226	96,64%
I-Light	2241	1,31%	9109	1,34%	20846	1,47%
I-Shadow	2106	2,31%	8609	2,36%	21857	3,36%

Table 42 - NSM viewport results in the Bench scene with the With viewport, using a fixed shadow map size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

Scene	Bench							
Viewport	With							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
C-Light	665562	97,82%	672574	98,66%	676652	99,17%	678718	99,55%
C-Shadow	350996	95,74%	356754	97,64%	360197	98,76%	362790	99,33%
I-Light	14867	2,18%	9109	1,34%	5666	0,83%	3073	0,45%
I-Shadow	15621	4,26%	8609	2,36%	4531	1,24%	2465	0,67%

Table 43 - NSM shadow map results in the Bench scene with the With viewport, using a fixed viewport size of 1024x1024; I represents the incorrect pixels and C represents the correct pixels.

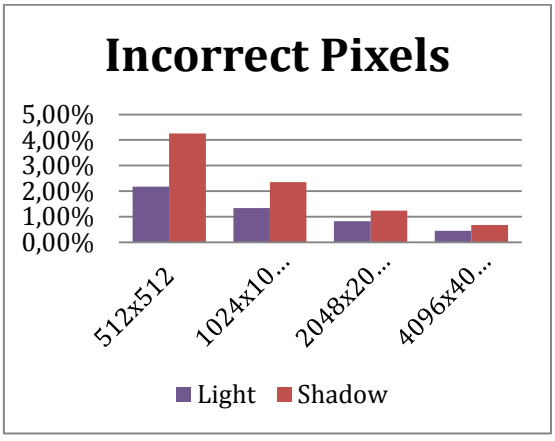


Figure 80 - Shadow Map Size variation results in the With-Bench scene using a fixed viewport of 1024x1024.

Scene	Flowers			
Viewport	Against			
Pixel State	Pixel Precision			
	Correct		Incorrect	
Light	510737	97,14%	15023	2,86%
Shadow	449341	97,25%	12717	2,75%
Total	960078	97,19%	27740	2,81%

Table 44 - Normal Shadow Mapping errors in the Flowers scene with the Against viewport, with a 1024x1024 viewport and 1024x1024 shadow map.

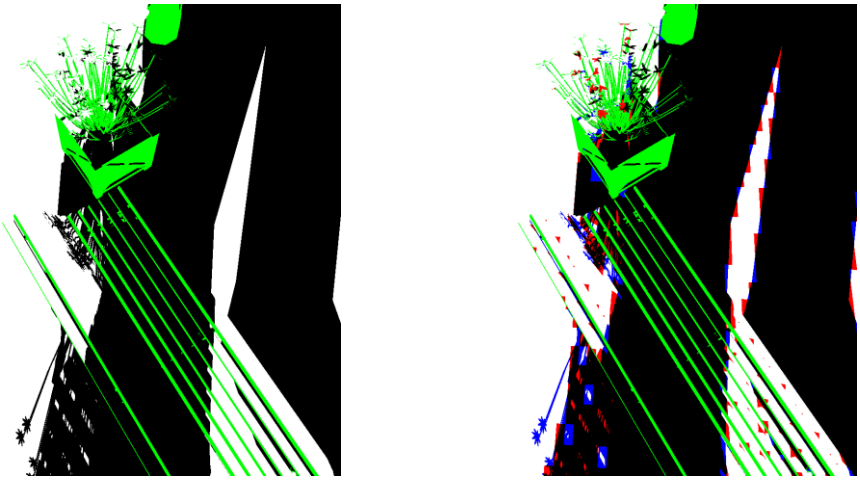


Figure 81 - Against-Flowers scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) OptiX Prime results; (Right) NSM results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Flowers					
Viewport	Against					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
C-Light	127712	97,13%	510737	97,14%	1373497	98,20%
C-Shadow	112327	97,25%	449341	97,25%	581599	95,76%
I-Light	3768	2,87%	15023	2,86%	25134	1,80%
I-Shadow	3179	2,75%	12717	2,75%	25767	4,24%

Table 45 - NSM viewport results in the Flowers scene with the Against viewport, using a fixed shadow map size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

Scene	Flowers							
Viewport	Against							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
C-Light	498481	95,99%	510737	97,14%	515356	98,12%	518487	98,66%
C-Shadow	443563	94,67%	449341	97,25%	454500	98,25%	457336	98,93%
I-Light	20801	4,01%	15023	2,86%	9864	1,88%	7028	1,34%
I-Shadow	24973	5,33%	12717	2,75%	8098	1,75%	4967	1,07%

Table 46 - NSM shadow map results in the Flowers scene with the Against Camera, using a fixed viewport size of 1024x1024; I represents the incorrect pixels and C represents the correct pixels.

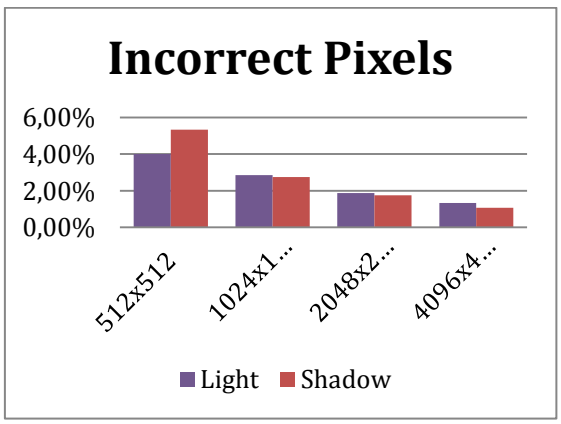


Figure 82 - Shadow Map Size variation results in the Against-Flowers scene using a fixed viewport of 1024x1024.

Scene	Flowers			
Viewport	Side			
Pixel State	Pixel Precision			
	Correct		Incorrect	
Light	624443	97,50%	16042	2,50%
Shadow	361499	95,93%	15346	4,07%
Total	985942	96,91%	31388	3,09%

Table 47 - Normal Shadow Mapping errors in the Flowers scene with the Side viewport, with a 1024x1024 viewport and 1024x1024 shadow map.

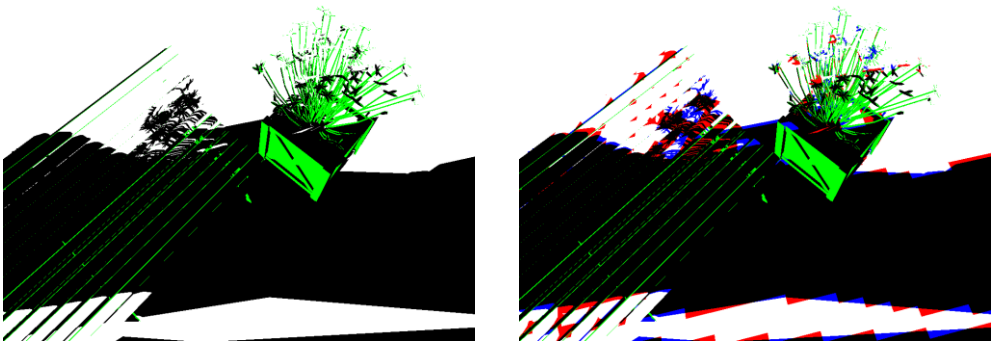


Figure 83 - Side-Flowers scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) OptiX Prime results; (Right) NSM results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Flowers					
Viewport	Side					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
C-Light	156075	97,50%	624443	97,50%	1284602	97,86%
C-Shadow	90381	95,93%	361499	95,93%	651151	95,58%
I-Light	3999	2,50%	16042	2,50%	28087	2,14%
I-Shadow	3832	4,07%	15346	4,07%	30081	4,42%

Table 48 - NSM viewport results in the Flowers scene with the Side viewport, using a fixed shadow map size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

Scene	Flowers							
Viewport	Side							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
C-Light	603129	97,40%	624443	97,50%	631302	98,10%	634106	98,83%
C-Shadow	361436	90,79%	361499	95,93%	365319	97,73%	370053	98,49%
I-Light	16105	2,60%	16042	2,50%	12222	1,90%	7488	1,17%
I-Shadow	36660	9,21%	15346	4,07%	8487	2,27%	5683	1,51%

Table 49 - NSM shadow map results in the Flowers scene with the Side viewport, using a fixed viewport size of 1024x1024; I represents the incorrect pixels and C represents the correct pixels.

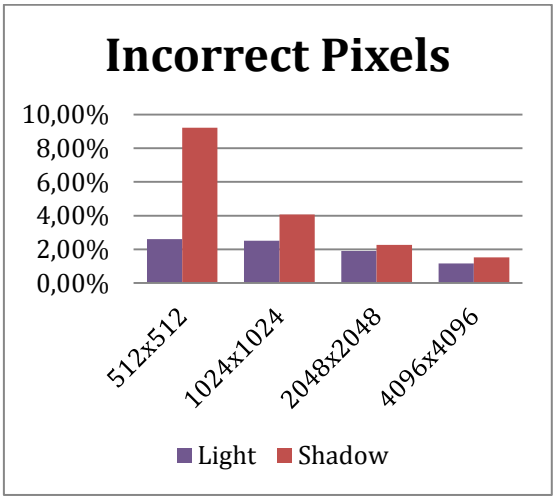


Figure 84 - Shadow Map Size variation results in the Side-Flowers scene using a fixed viewport of 1024x1024.

Scene	Trees			
Viewport	Against			
Pixel State	Pixel Precision			
	Correct		Incorrect	
Light	444767	96,78%	14797	3,22%
Shadow	576117	97,81%	12895	2,19%
Total	1020884	97,36%	27692	2,64%

Table 50 - Normal Shadow Mapping errors in the Trees scene with the Against viewport, with a 1024x1024 viewport and 1024x1024 shadow map.

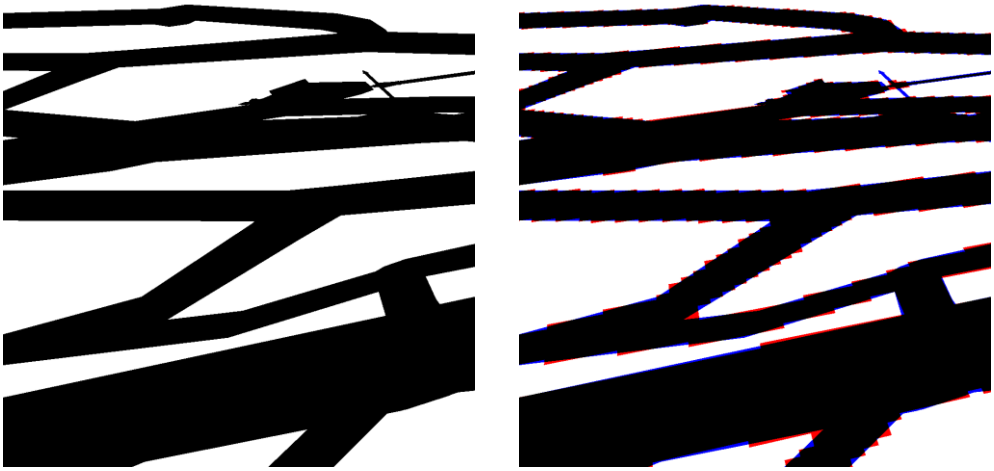


Figure 85 - Against-Trees scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) OptiX Prime results; (Right) NSM results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Trees					
Viewport	Against					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
C-Light	111240	96,81%	444767	96,78%	925202	96,96%
C-Shadow	144015	97,81%	576117	97,81%	1087640	97,16%
I-Light	3661	3,19%	14797	3,22%	28960	3,04%
I-Shadow	3228	2,19%	12895	2,19%	31798	2,84%

Table 51 - NSM viewport results in the Trees scene with the Against viewport, using a fixed shadow map size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

Scene	Trees							
Viewport	Against							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
C-Light	426075	94,44%	444767	96,78%	451089	98,42%	454397	99,19%
C-Shadow	565841	94,71%	576117	97,81%	583662	98,89%	587222	99,45%
I-Light	25073	5,56%	14797	3,22%	7252	1,58%	3692	0,81%
I-Shadow	31587	5,29%	12895	2,19%	6573	1,11%	3265	0,55%

Table 52 - NSM shadow map results in the Trees scene with the Against viewport, using a fixed viewport size of 1024x1024; I represents the incorrect pixels and C represents the correct pixels.

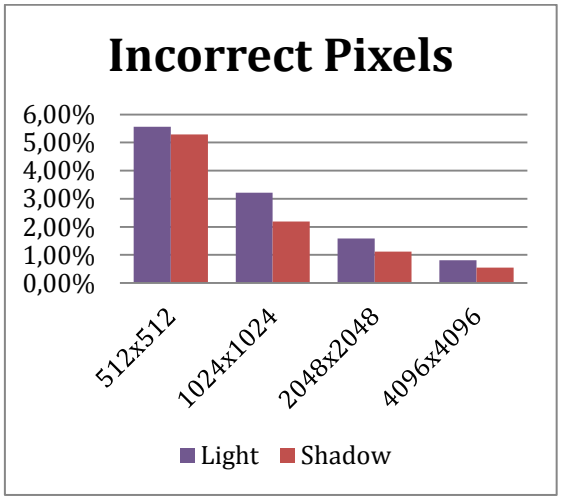


Figure 86 - Shadow Map Size variation results in the Against-Trees scene using a fixed viewport of 1024x1024.

Scene	Trees			
Viewport	Side			
Pixel State	Pixel Precision			
	Correct		Incorrect	
Light	662529	98,49%	10162	1,51%
Shadow	362628	96,47%	13257	3,53%
Total	1025157	97,77%	23419	2,23%

Table 53 - Normal Shadow Mapping errors in the Trees scene with the Side viewport, with a 1024x1024 viewport and 1024x1024 shadow map.

Figure 87 - Side-Trees scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) OptiX Prime results; (Right) NSM results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Trees					
Viewport	Side					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
C-Light	165613	98,48%	662529	98,49%	1293046	97,88%
C-Shadow	90643	96,45%	362628	96,47%	725120	96,38%
I-Light	2552	1,52%	10162	1,51%	28032	2,12%
I-Shadow	3336	3,55%	13257	3,53%	27247	3,62%

Table 54 - NSM viewport results in the Trees scene with the Side viewport, using a fixed shadow map size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

Scene	Trees							
Viewport	Side							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
C-Light	654341	97,06%	662529	98,49%	670402	99,18%	673197	99,54%
C-Shadow	352965	94,27%	362628	96,47%	367260	98,56%	369699	99,30%
I-Light	19825	2,94%	10162	1,51%	5530	0,82%	3091	0,46%
I-Shadow	21445	5,73%	13257	3,53%	5384	1,44%	2589	0,70%

Table 55 - NSM shadow map results in the Trees scene with the Side viewport, using a fixed viewport size of 1024x1024; I represents the incorrect pixels and C represents the correct pixels.

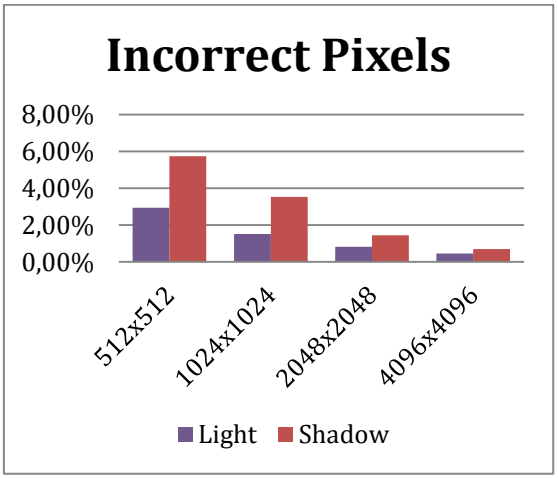


Figure 88 - Shadow Map Size variation results in the Side-Trees scene using a fixed viewport of 1024x1024.

Scene	Trees			
Viewport	With			
Pixel State	Pixel Precision			
	Correct		Incorrect	
Light	642100	97,93%	13590	2,07%
Shadow	379644	96,63%	13242	3,37%
Total	1021744	97,44%	26832	2,56%

Table 56 - Normal Shadow Mapping errors in the Bench scene with the With viewport, with a 1024x1024 viewport and 1024x1024 shadow map.

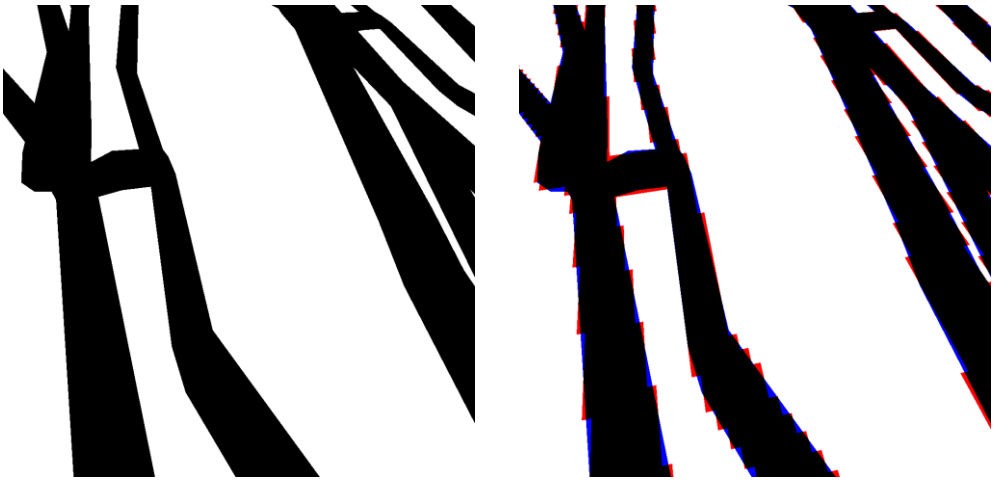


Figure 89 - With-Trees scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) OptiX Prime results; (Right) NSM results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Trees					
Viewport	With					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
C-Light	160517	97,92%	642100	97,93%	1293783	97,51%
C-Shadow	94916	96,63%	379644	96,63%	709377	94,99%
I-Light	3404	2,08%	13590	2,07%	33035	2,49%
I-Shadow	3307	3,37%	13242	3,37%	37405	5,01%

Table 57 - NSM viewport results in the Trees scene with the With viewport, using a fixed shadow map size of 1024x1024; I represents the incorrect pixels and C represent the correct pixels.

Scene	Trees							
Viewport	With							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
C-Light	628509	96,17%	642100	97,93%	649170	98,95%	652024	99,47%
C-Shadow	368176	93,21%	379644	96,63%	386339	98,43%	389740	99,16%
I-Light	25058	3,83%	13590	2,07%	6895	1,05%	3494	0,53%
I-Shadow	26833	6,79%	13242	3,37%	6172	1,57%	3318	0,84%

Table 58 - NSM shadow map results in the Trees scene with the With viewport, using a fixed viewport size of 1024x1024; I represents the incorrect pixels and C represents the correct pixels.

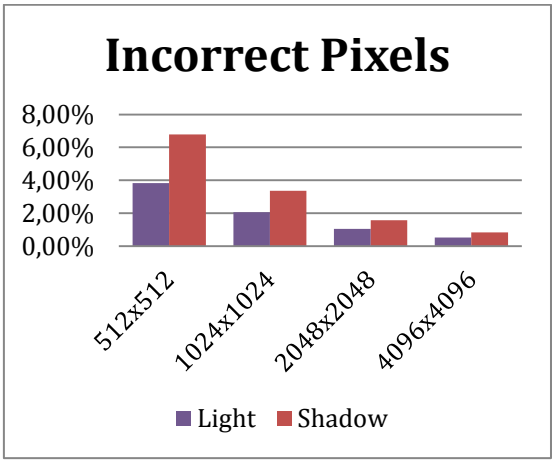


Figure 90 - Shadow Map Size variation results in the With-Trees scene using a fixed viewport of 1024x1024.

A.1.3 Hybrid Methods

Scene	Bench					
Viewport	Against					
Method	**PCF**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	520628	86,23%	83108	13,77%	1892	0,31%
Shadow	317692	75,66%	102197	24,34%	2430	0,58%
Total	833998	81,47%	185305	18,10%	4322	0,42%
Method	**CRSM**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	477355	79,00%	126919	21,00%	0	0,00%
Shadow	34580	8,25%	384771	91,75%	0	0,00%
Total	511935	50,01%	511690	49,99%	0	0,00%
Method	**CRSMA**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	477355	79,00%	126919	21,00%	0	0,00%
Shadow	182163	43,44%	237188	56,56%	0	0,00%
Total	659518	64,43%	364107	35,57%	0	0,00%

Table 59 - Pixels Precision results in the Bench scene with the Against viewport, with a 1024x1024 viewport and a 1024x1024 shadow map.

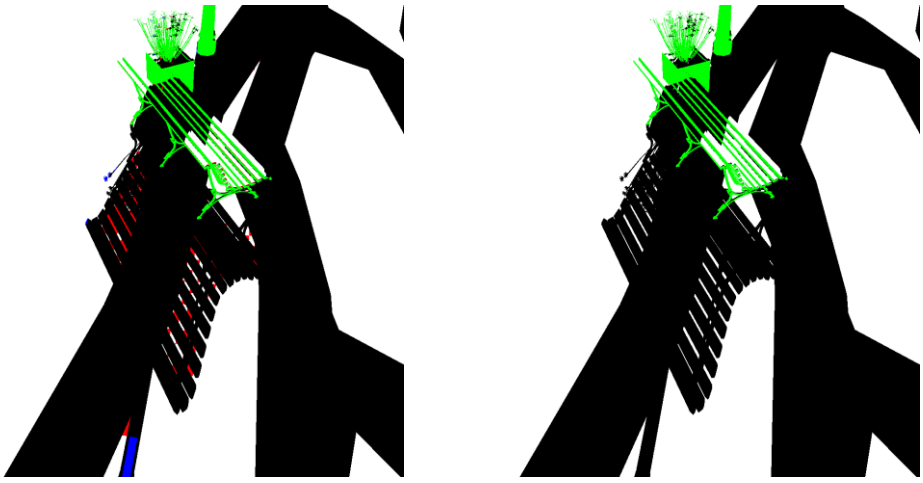


Figure 91 - Against-Bench scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) PCF results; (Right) CRSM/CRSMA results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Bench					
Viewport	Side					
Method	**PCF**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	676275	93,22%	49224	6,78%	1032	0,14%
Shadow	213259	79,07%	56463	20,93%	1176	0,44%
Total	887326	89,16%	105687	10,62%	2208	0,22%
Method	**CRSM**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	646787	89,13%	78856	10,87%	0	0,00%
Shadow	21062	7,81%	248516	92,19%	0	0,00%
Total	667849	67,11%	327372	32,89%	0	0,00%
Method	**CRSMA**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	646787	89,13%	78856	10,87%	0	0,00%
Shadow	129768	48,14%	139810	51,86%	0	0,00%
Total	776555	78,03%	218666	21,97%	0	0,00%

Table 60 - Pixels Precision results in the Bench scene with the Side viewport, with a 1024x1024 viewport and a 1024x1024 shadow map.

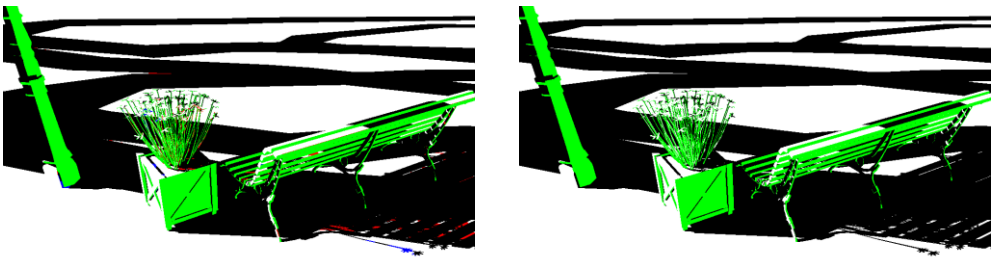


Figure 92 - Side-Bench scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) PCF results; (Right) CRSM/CRSMA results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Bench					
Viewport	With					
Method	**PCF**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	608023	89,29%	72903	10,71%	675	0,10%
Shadow	296469	80,98%	69651	19,02%	932	0,25%
Total	902885	86,23%	142554	13,61%	1607	0,15%
Method	**CRSM**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	566053	83,10%	115136	16,90%	6	0,00%
Shadow	22585	6,17%	343272	93,83%	0	0,00%
Total	588632	56,22%	458408	43,78%	6	0,00%
Method	**CRSMA**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	566053	83,10%	115135	16,90%	5	0,00%
Shadow	205275	56,11%	160583	43,89%	0	0,00%
Total	771323	73,67%	275718	26,33%	5	0,00%

Table 61 - Pixels Precision results in the Bench scene with the With viewport, with a 1024x1024 viewport and a 1024x1024 shadow map.

Figure 93 - With-Bench scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) PCF results; (Right) CRSM/CRSMA results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Flowers					
Viewport	Against					
Method	**PCF**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	420456	80,46%	102101	19,54%	1918	0,37%
Shadow	384027	82,54%	81234	17,46%	2815	0,61%
Total	799750	80,96%	183335	18,56%	4733	0,48%
Method	**CRSM**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	386569	73,85%	136885	26,15%	0	0,00%
Shadow	27666	5,96%	436698	94,04%	0	0,00%
Total	414235	41,93%	573583	58,07%	0	0,00%
Method	**CRSMA**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	386569	73,85%	136885	26,15%	0	0,00%
Shadow	270110	58,17%	194254	41,83%	0	0,00%
Total	656679	66,48%	331139	33,52%	0	0,00%

Table 62 - Pixels Precision results in the Flowers scene with the Against viewport, with a 1024x1024 viewport and a 1024x1024 shadow map.

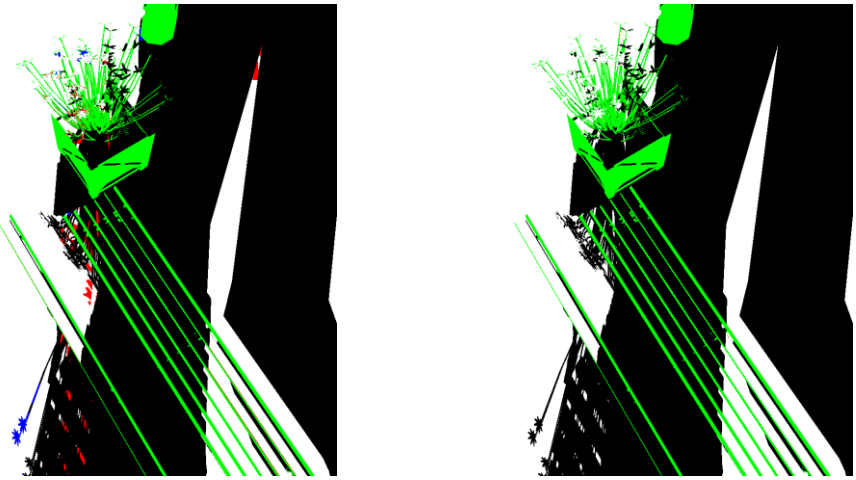


Figure 94 - Against-Flowers scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) PCF results; (Right) CRSM/CRSMA results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Flowers					
Viewport	Side					
Method	**PCF**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	532193	83,31%	106608	16,69%	1899	0,30%
Shadow	298371	78,82%	80158	21,18%	2887	0,76%
Total	825778	81,17%	186766	18,36%	4786	0,47%
Method	**CRSM**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	513380	80,24%	126410	19,76%	1	0,00%
Shadow	24412	6,47%	353128	93,53%	0	0,00%
Total	537791	52,86%	479538	47,14%	1	0,00%
Method	**CRSMA**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	513380	80,24%	126409	19,76%	0	0,00%
Shadow	216081	57,23%	161460	42,77%	0	0,00%
Total	729461	71,70%	287869	28,30%	0	0,00%

Table 63 - Pixels Precision results in the Flowers scene with the Side viewport, with a 1024x1024 viewport and a 1024x1024 shadow map.

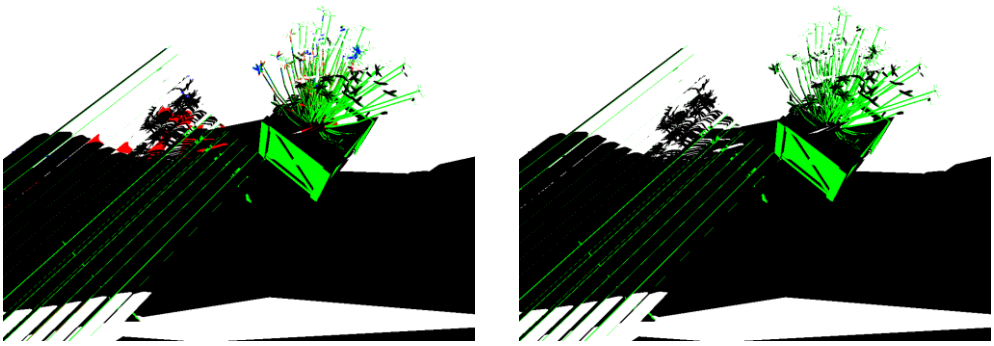


Figure 95 - Side-Flowers scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) PCF results; (Right) CRSM/CRSMA results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Trees					
Viewport	Against					
Method	**PCF**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	334384	72,96%	123899	27,04%	1079	0,24%
Shadow	453102	76,76%	137191	23,24%	458	0,08%
Total	785949	74,95%	261090	24,90%	1537	0,15%
Method	**CRSM**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	269766	58,94%	187896	41,06%	0	0,00%
Shadow	53747	9,10%	537167	90,90%	0	0,00%
Total	323513	30,85%	725063	69,15%	0	0,00%
Method	**CRSMA**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	269766	58,94%	187896	41,06%	0	0,00%
Shadow	288987	48,91%	301927	51,09%	0	0,00%
Total	558753	53,29%	489823	46,71%	0	0,00%

Table 64 - Pixels Precision results in the Trees scene with the Against viewport, with a 1024x1024 viewport and a 1024x1024 shadow map.

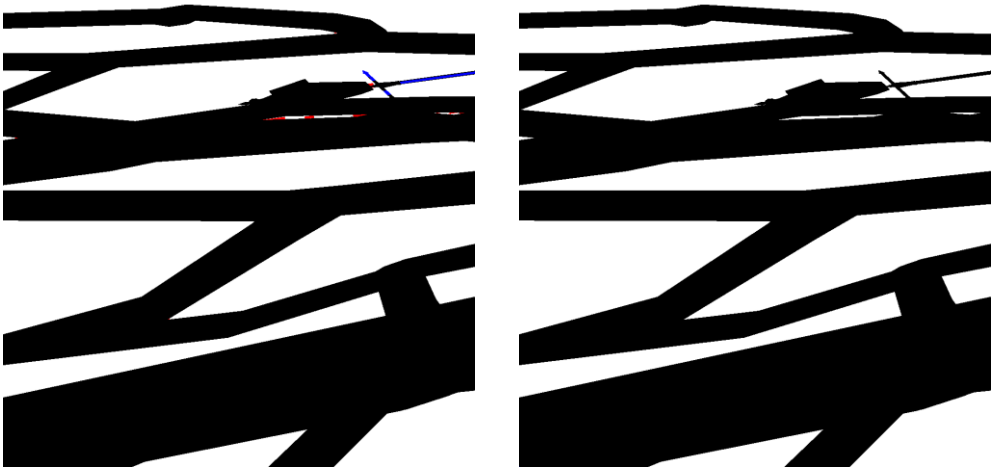


Figure 96 - Against-Trees scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) PCF results; (Right) CRSM/CRSMA results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Trees					
Viewport	Side					
Method	**PCF**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	573225	84,86%	102252	15,14%	0	0,00%
Shadow	267271	71,64%	105828	28,36%	309	0,08%
Total	840187	80,13%	208080	19,84%	309	0,03%
Method	**CRSM**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	525398	77,75%	150388	22,25%	0	0,00%
Shadow	14479	3,88%	358311	96,12%	0	0,00%
Total	539877	51,49%	508699	48,51%	0	0,00%
Method	**CRSMA**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	525398	77,75%	150387	22,25%	0	0,00%
Shadow	124392	33,37%	248399	66,63%	1	0,00%
Total	649789	61,97%	398786	38,03%	1	0,00%

Table 65 - Pixels Precision results in the Trees scene with the Side viewport, with a 1024x1024 viewport and a 1024x1024 shadow map.

Figure 97 - Side-Trees scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) PCF results; (Right) CRSM/CRSMA results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

Scene	Trees					
Viewport	With					
Method	**PCF**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	529645	80,90%	125021	19,10%	0	0,00%
Shadow	260053	66,02%	133857	33,98%	676	0,17%
Total	789022	75,25%	258878	24,69%	676	0,06%
Method	**CRSM**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	463670	70,75%	191672	29,25%	0	0,00%
Shadow	5198	1,32%	388036	98,68%	0	0,00%
Total	468868	44,71%	579708	55,29%	0	0,00%
Method	**CRSMA**					
Pixel State	Certain - Correct		Uncertain		Certain - Incorrect	
Light	463670	70,75%	191672	29,25%	0	0,00%
Shadow	107191	27,26%	286043	72,74%	0	0,00%
Total	570861	54,44%	477715	45,56%	0	0,00%

Table 66 - Pixels Precision results in the Trees scene with the With viewport, with a 1024x1024 viewport and a 1024x1024 shadow map.

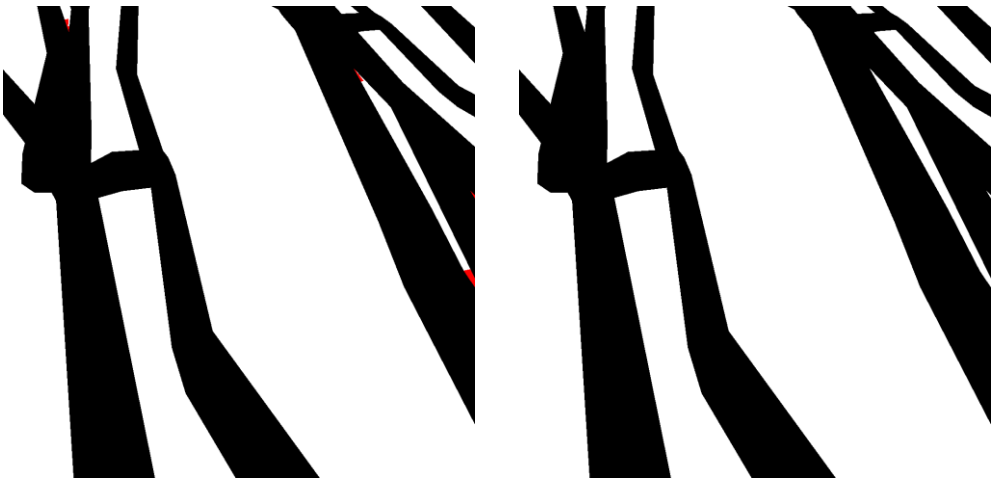


Figure 98 - With-Trees scene with a 1024x1024 viewport and 1024x1024 shadow map; (Left) PCF results; (Right) CRSM/CRSMA results; Blue pixels represent the incorrect light pixels, and red pixels represent the incorrect shadow pixels.

A.1.3.1 Viewport Variance

Scene	Bench					
Viewport	Against					
Method	**PCF**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	208483	81,47%	833998	13,77%	1620174	79,19%
Uncertain	46328	18,10%	185305	81,47%	417252	20,40%
Certain-I	1083	0,42%	4322	0,42%	8413	0,41%
Method	**CRSM**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	127973	50,01%	511935	50,01%	1020542	49,88%
Uncertain	127921	49,99%	511690	49,99%	1025297	50,12%
Certain-I	0	0,00%	0	0,00%	0	0,00%
Method	**CRSMA**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	164840	64,42%	659518	64,43%	1210372	59,16%
Uncertain	91054	35,58%	364107	35,57%	835467	40,84%
Certain-I	0	0,00%	0	0,00%	0	0,00%

Table 67 - Viewport Variation of the pixel precision results using a fixed Shadow Map Size of 1024x1024 for the Against-Bench scene; C - Correct and I - Incorrect.

Scene	Bench					
Viewport	Side					
Method	**PCF**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	221867	89,18%	887326	6,78%	1778089	88,28%
Uncertain	26391	10,61%	105687	89,16%	231527	11,49%
Certain-I	539	0,22%	2208	0,22%	4569	0,23%
Method	**CRSM**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	167011	67,13%	667849	67,11%	1354639	67,25%
Uncertain	81786	32,87%	327372	32,89%	659544	32,74%
Certain-I	0	0,00%	0	0,00%	2	0,00%
Method	**CRSMA**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	194127	78,03%	776555	78,03%	1544128	76,66%
Uncertain	54670	21,97%	218666	21,97%	470055	23,34%
Certain-I	0	0,00%	0	0,00%	2	0,00%

Table 68 - Viewport Variation of the pixel precision results using a fixed Shadow Map Size of 1024x1024 for the Side-Bench scene; C - Correct and I - Incorrect.

Scene	Bench					
Viewport	With					
Method	**PCF**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	225808	86,26%	902885	10,71%	1723748	83,20%
Uncertain	35573	13,59%	142554	86,23%	342530	16,53%
Certain-I	388	0,15%	1607	0,15%	5641	0,27%
Method	**CRSM**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	147118	56,20%	588632	56,22%	1166972	56,32%
Uncertain	114648	43,80%	458408	43,78%	904947	43,68%
Certain-I	3	0,00%	6	0,00%	0	0,00%
Method	**CRSMA**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	192854	73,67%	771323	73,67%	1352739	65,29%
Uncertain	68913	26,33%	275718	26,33%	719180	34,71%
Certain-I	2	0,00%	5	0,00%	0	0,00%

Table 69 - Viewport Variation of the pixel precision results using a fixed Shadow Map Size of 1024x1024 for the With-Bench scene; C - Correct and I - Incorrect.

Scene	Flowers					
Viewport	Against					
Method	**PCF**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	199959	80,96%	799750	19,54%	1643701	81,94%
Uncertain	45824	18,55%	183335	80,96%	354853	17,69%
Certain-I	1203	0,49%	4733	0,48%	7443	0,37%
Method	**CRSM**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	103578	41,94%	414235	41,93%	1138259	56,74%
Uncertain	143408	58,06%	573583	58,07%	867738	43,26%
Certain-I	0	0,00%	0	0,00%	0	0,00%
Method	**CRSMA**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	164235	66,50%	656679	66,48%	1320774	65,84%
Uncertain	82751	33,50%	331139	33,52%	685213	34,16%
Certain-I	0	0,00%	0	0,00%	10	0,00%

Table 70 - Viewport Variation of the pixel precision results using a fixed Shadow Map Size of 1024x1024 for the Against-Flowers scene; C - Correct and I - Incorrect.

Scene	Flowers					
Viewport	Side					
Method	**PCF**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	206436	81,18%	825778	16,69%	1649401	82,72%
Uncertain	46670	18,35%	186766	81,17%	334768	16,79%
Certain-I	1181	0,46%	4786	0,47%	9752	0,49%
Method	**CRSM**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	134424	52,86%	537791	52,86%	1104398	55,39%
Uncertain	119862	47,14%	479538	47,14%	889523	44,61%
Certain-I	1	0,00%	1	0,00%	0	0,00%
Method	**CRSMA**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	182341	71,71%	729461	71,70%	1401555	70,29%
Uncertain	71946	28,29%	287869	28,30%	592366	29,71%
Certain-I	0	0,00%	0	0,00%	0	0,00%

Table 71 - Viewport Variation of the pixel precision results using a fixed Shadow Map Size of 1024x1024 for the Side-Flowers scene; C - Correct and I - Incorrect.

Scene	Trees					
Viewport	Against					
Method	**PCF**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	196503	74,96%	785949	27,04%	1496253	72,16%
Uncertain	65257	24,89%	261090	74,95%	573361	27,65%
Certain-I	384	0,15%	1537	0,15%	3986	0,19%
Method	**CRSM**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	80881	30,85%	323513	30,85%	586961	28,31%
Uncertain	181263	69,15%	725063	69,15%	1486638	71,69%
Certain-I	0	0,00%	0	0,00%	1	0,00%
Method	**CRSMA**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	139694	53,29%	558753	53,29%	946991	45,67%
Uncertain	122450	46,71%	489823	46,71%	1126609	54,33%
Certain-I	0	0,00%	0	0,00%	0	0,00%

Table 72 - Viewport Variation of the pixel precision results using a fixed Shadow Map Size of 1024x1024 for the Against -Trees scene; C - Correct and I - Incorrect.

Scene	Trees					
Viewport	Side					
Method	**PCF**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	210059	80,13%	840187	15,14%	1530001	73,79%
Uncertain	52007	19,84%	208080	80,13%	542168	26,15%
Certain-I	78	0,03%	309	0,03%	1276	0,06%
Method	**CRSM**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	134955	51,48%	539877	51,49%	940847	45,38%
Uncertain	127189	48,52%	508699	48,51%	1132598	54,62%
Certain-I	0	0,00%	0	0,00%	0	0,00%
Method	**CRSMA**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	162416	61,96%	649789	61,97%	1090849	52,61%
Uncertain	99728	38,04%	398786	38,03%	982596	47,39%
Certain-I	0	0,00%	1	0,00%	0	0,00%

Table 73 - Viewport Variation of the pixel precision results using a fixed Shadow Map Size of 1024x1024 for the Side-Trees scene; C - Correct and I - Incorrect.

Scene	Trees					
Viewport	With					
Method	**PCF**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	197243	75,24%	789022	19,10%	1439163	69,40%
Uncertain	64727	24,69%	258878	75,25%	625022	30,14%
Certain-I	174	0,07%	676	0,06%	9415	0,45%
Method	**CRSM**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	117215	44,71%	468868	44,71%	874549	42,18%
Uncertain	144929	55,29%	579708	55,29%	1199051	57,82%
Certain-I	0	0,00%	0	0,00%	0	0,00%
Method	**CRSMA**					
Pixel State	Viewport Size					
	512x512		1024x1024		1920x1080	
Certain-C	142704	54,44%	570861	54,44%	918812	44,31%
Uncertain	119440	45,56%	477715	45,56%	1154788	55,69%
Certain-I	0	0,00%	0	0,00%	0	0,00%

Table 74 - Viewport Variation of the pixel precision results using a fixed Shadow Map Size of 1024x1024 for the With-Trees scene; C - Correct and I - Incorrect.

A.1.3.2 Shadow Map Variance

Scene	Bench							
Viewport	Against							
Method	**PCF**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	712204	69,58%	833998	81,47%	914675	89,36%	956649	93,46%
Uncertain	305954	29,89%	185305	18,10%	106105	10,37%	65707	6,42%
Certain-I	5467	0,53%	4322	0,42%	2845	0,28%	1269	0,12%
Method	**CRSM**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	506460	49,48%	511935	50,01%	671142	65,57%	790877	77,26%
Uncertain	517165	50,52%	511690	49,99%	352483	34,43%	232739	22,74%
Certain-I	0	0,00%	0	0,00%	0	0,00%	9	0,00%
Method	**CRSMA**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	560117	54,72%	659518	64,43%	809721	79,10%	878205	85,79%
Uncertain	463508	45,28%	364107	35,57%	213904	20,90%	145411	14,21%
Certain-I	0	0,00%	0	0,00%	0	0,00%	9	0,00%

Table 75 - Shadow Map Variation of the pixel precision results, using a fixed viewport of 1024x1024 for the Against-Bench scene; C - Correct and I - Incorrect.

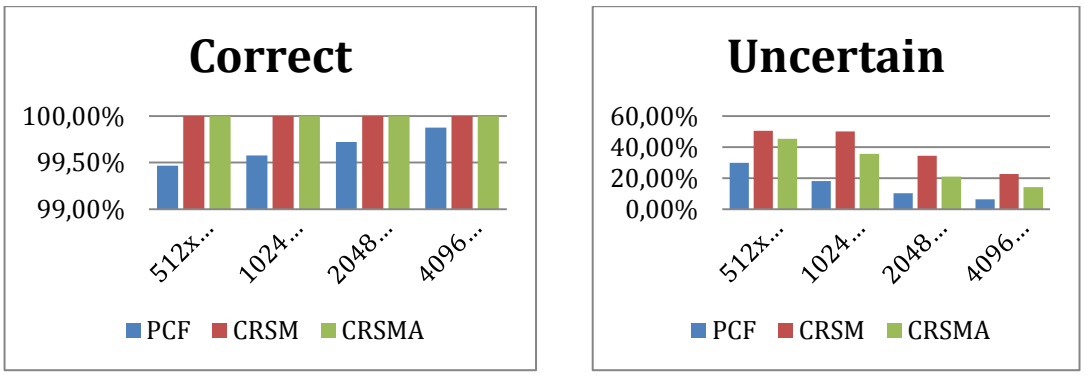


Figure 99 – Shadow Map size Variation results using a fixed viewport of 1024x1024 for the Against-Bench scene; (Left) Correct pixel percentages (Certain-C and Uncertain); (Right) Uncertain pixel percentages.

Scene	Bench							
Viewport	Side							
Method	**PCF**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	816713	82,06%	887326	89,16%	930029	93,45%	953564	95,81%
Uncertain	174391	17,52%	105687	10,62%	62711	6,30%	39873	4,01%
Certain-I	4117	0,41%	2208	0,22%	2481	0,25%	1784	0,18%
Method	**CRSM**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	669997	67,32%	667849	67,11%	762997	76,67%	845311	84,94%
Uncertain	325224	32,68%	327372	32,89%	232224	23,33%	149860	15,06%
Certain-I	0	0,00%	0	0,00%	0	0,00%	50	0,01%
Method	**CRSMA**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	726928	73,04%	776555	78,03%	860120	86,43%	903529	90,79%
Uncertain	268293	26,96%	218666	21,97%	135101	13,57%	91642	9,21%
Certain-I	0	0,00%	0	0,00%	0	0,00%	50	0,01%

Table 76 - Shadow Map Variation of the pixel precision results using a fixed viewport of 1024x1024 for the Side-Bench scene; C - Correct and I - Incorrect.

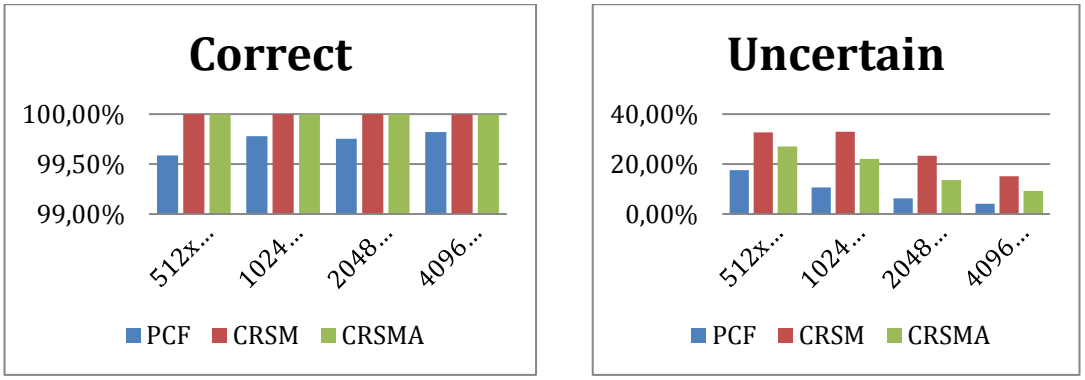


Figure 100 – Shadow Map size Variation results using a fixed viewport of 1024x1024 for the Side-Bench scene; (Left) Correct pixel percentages (Certain-C and Uncertain); (Right) Uncertain pixel percentages.

Scene	Bench							
Viewport	With							
Method	**PCF**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	800563	76,46%	902885	86,23%	967494	92,40%	1003203	95,81%
Uncertain	241344	23,05%	142554	13,61%	78419	7,49%	42953	4,10%
Certain-I	5139	0,49%	1607	0,15%	1133	0,11%	890	0,09%
Method	**CRSM**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	595297	56,85%	588632	56,22%	736608	70,35%	869006	83,00%
Uncertain	451748	43,15%	458408	43,78%	310435	29,65%	178031	17,00%
Certain-I	1	0,00%	6	0,00%	3	0,00%	9	0,00%
Method	**CRSMA**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	665555	63,57%	771323	73,67%	901443	86,09%	969072	92,55%
Uncertain	381490	36,43%	275718	26,33%	145600	13,91%	77965	7,45%
Certain-I	1	0,00%	5	0,00%	3	0,00%	9	0,00%

Table 77 - Shadow Map Variation of the pixel precision results using a fixed viewport of 1024x1024 for the With-Bench scene; C - Correct and I - Incorrect.

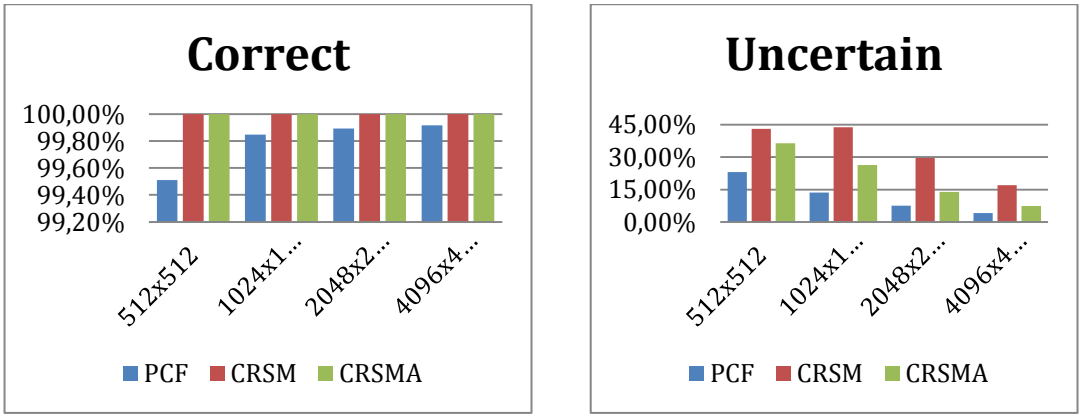


Figure 101 – Shadow Map size Variation results using a fixed viewport of 1024x1024 for the With-Bench scene; (Left) Correct pixel percentages (Certain-C and Uncertain); (Right) Uncertain pixel percentages.

Scene	Flowers							
Viewport	Against							
Method	**PCF**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	731924	74,10%	799750	80,96%	865525	87,62%	917970	92,93%
Uncertain	249400	25,25%	183335	18,56%	119301	12,08%	67168	6,80%
Certain-I	6494	0,66%	4733	0,48%	2992	0,30%	2680	0,27%
Method	**CRSM**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	415735	42,09%	414235	41,93%	568621	57,56%	727695	73,67%
Uncertain	572083	57,91%	573583	58,07%	419197	42,44%	260059	26,33%
Certain-I	0	0,00%	0	0,00%	0	0,00%	64	0,01%
Method	**CRSMA**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	527911	53,44%	656679	66,48%	765507	77,49%	843197	85,36%
Uncertain	459907	46,56%	331139	33,52%	222311	22,51%	144557	14,63%
Certain-I	0	0,00%	0	0,00%	0	0,00%	64	0,01%

Table 78 - Shadow Map Variation of the pixel precision results, using a fixed viewport of 1024x1024 for the Against-Flowers scene; C - Correct and I - Incorrect.

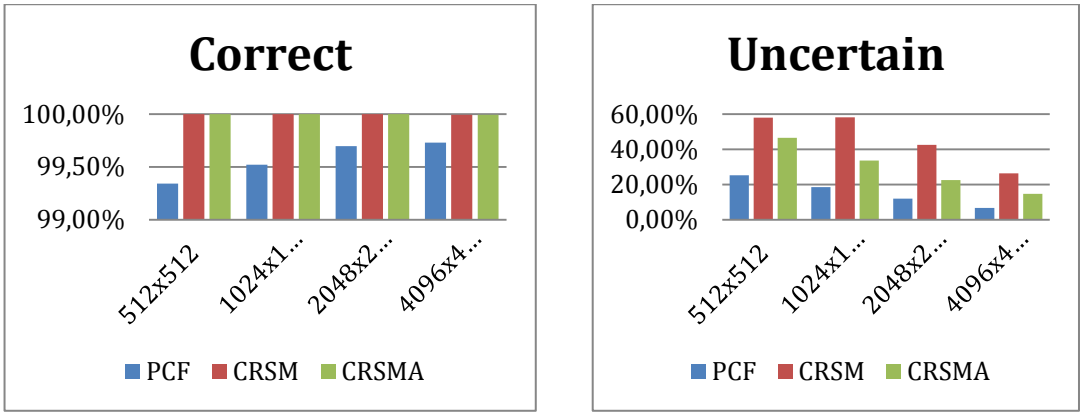


Figure 102 – Shadow Map size Variation results using a fixed viewport of 1024x1024 for the Against-Flowers scene; (Left) Correct pixel percentages (Certain-C and Uncertain); (Right) Uncertain pixel percentages.

Scene	Flowers							
Viewport	Side							
Method	**PCF**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	776881	76,36%	825778	81,17%	888745	87,36%	939695	92,37%
Uncertain	229184	22,53%	186766	18,36%	125608	12,35%	75324	7,40%
Certain-I	11255	1,11%	4786	0,47%	2977	0,29%	2311	0,23%
Method	**CRSM**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	537659	52,85%	537791	52,86%	664398	65,31%	803043	78,94%
Uncertain	479670	47,15%	479538	47,14%	352932	34,69%	214284	21,06%
Certain-I	1	0,00%	1	0,00%	0	0,00%	3	0,00%
Method	**CRSMA**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	653426	64,23%	729461	71,70%	809094	79,53%	884584	86,95%
Uncertain	363903	35,77%	287869	28,30%	208236	20,47%	132743	13,05%
Certain-I	1	0,00%	0	0,00%	0	0,00%	3	0,00%

Table 79 - Shadow Map Variation of the pixel precision results using a fixed viewport of 1024x1024 for the Side-Flowers scene; C - Correct and I - Incorrect.

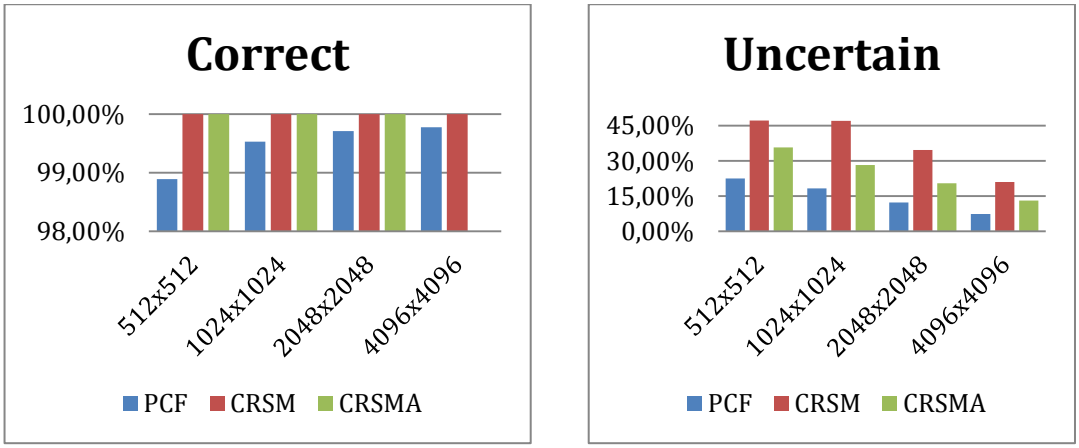


Figure 103 – Shadow Map size Variation results using a fixed viewport of 1024x1024 for the Side-Flowers scene; (Left) Correct pixel percentages (Certain-C and Uncertain); (Right) Uncertain pixel percentages.

Scene	Trees							
Viewport	Against							
Method	**PCF**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	559905	53,40%	785949	74,95%	912800	87,05%	979647	93,43%
Uncertain	480724	45,85%	261090	24,90%	135326	12,91%	68896	6,57%
Certain-I	7947	0,76%	1537	0,15%	450	0,04%	33	0,00%
Method	**CRSM**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	299556	28,57%	323513	30,85%	576188	54,95%	788061	75,16%
Uncertain	749019	71,43%	725063	69,15%	472388	45,05%	260515	24,84%
Certain-I	1	0,00%	0	0,00%	0	0,00%	0	0,00%
Method	**CRSMA**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	330796	31,55%	558753	53,29%	814941	77,72%	931993	88,88%
Uncertain	717779	68,45%	489823	46,71%	233635	22,28%	116583	11,12%
Certain-I	1	0,00%	0	0,00%	0	0,00%	0	0,00%

Table 80 - Shadow Map Variation of the pixel precision results, using a fixed viewport of 1024x1024 for the Against-Trees scene; C - Correct and I - Incorrect.

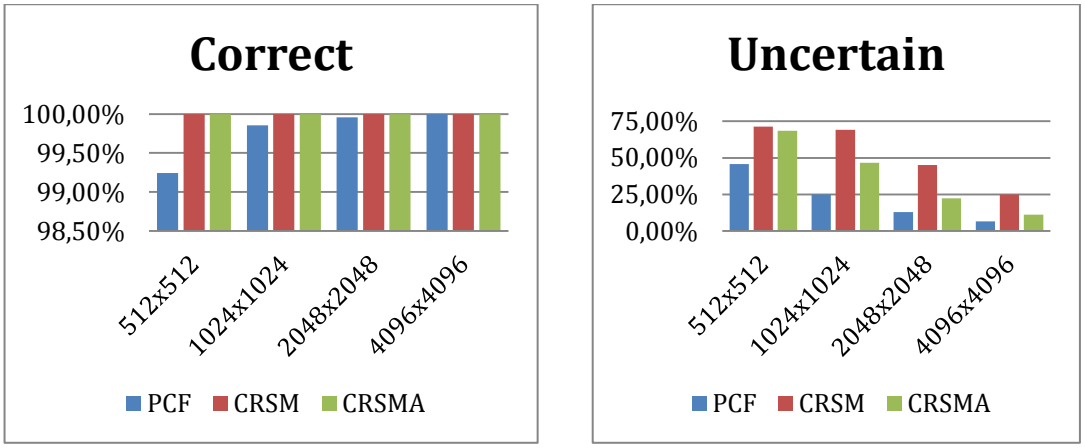


Figure 104 – Shadow Map size Variation results using a fixed viewport of 1024x1024 for the Against-Trees scene; (Left) Correct pixel percentages (Certain-C and Uncertain); (Right) Uncertain pixel percentages.

Scene	Trees							
Viewport	Side							
Method	**PCF**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	653884	62,36%	840187	80,13%	941155	89,76%	994716	94,86%
Uncertain	393303	37,51%	208080	19,84%	107364	10,24%	53841	5,13%
Certain-I	1389	0,13%	309	0,03%	57	0,01%	19	0,00%
Method	**CRSM**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	545915	52,06%	539877	51,49%	694820	66,26%	841050	80,21%
Uncertain	502661	47,94%	508699	48,51%	353756	33,74%	207526	19,79%
Certain-I	0	0,00%	0	0,00%	0	0,00%	0	0,00%
Method	**CRSMA**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	564028	53,79%	649789	61,97%	863908	82,39%	958986	91,46%
Uncertain	484548	46,21%	398786	38,03%	184668	17,61%	89590	8,54%
Certain-I	0	0,00%	1	0,00%	0	0,00%	0	0,00%

Table 81 - Shadow Map Variation of the pixel precision results using a fixed viewport of 1024x1024 for the Side-Trees scene; C - Correct and I - Incorrect.

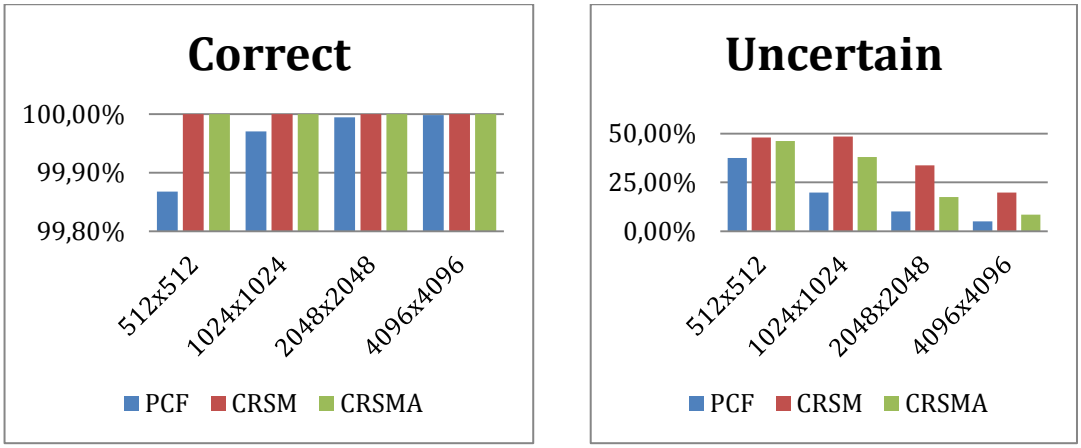


Figure 105 – Shadow Map size Variation results using a fixed viewport of 1024x1024 for the Side-Trees scene; (Left) Correct pixel percentages (Certain-C and Uncertain); (Right) Uncertain pixel percentages.

Scene	Trees							
Viewport	With							
Method	**PCF**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	548902	52,35%	789022	75,25%	916714	87,42%	981878	93,64%
Uncertain	495147	47,22%	258878	24,69%	131810	12,57%	66668	6,36%
Certain-I	4527	0,43%	676	0,06%	52	0,00%	30	0,00%
Method	**CRSM**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	480596	45,83%	468868	44,71%	642687	61,29%	808423	77,10%
Uncertain	567980	54,17%	579708	55,29%	405889	38,71%	240153	22,90%
Certain-I	0	0,00%	0	0,00%	0	0,00%	0	0,00%
Method	**CRSMA**							
Pixel State	Shadow Map Size							
	512x512		1024x1024		2048x2048		4096x4096	
Certain-C	485761	46,33%	570861	54,44%	828779	79,04%	940158	89,66%
Uncertain	562815	53,67%	477715	45,56%	219797	20,96%	108418	10,34%
Certain-I	0	0,00%	0	0,00%	0	0,00%	0	0,00%

Table 82 - Shadow Map Variation of the pixel precision results using a fixed viewport of 1024x1024 for the With-Trees scene; C - Correct and I - Incorrect.

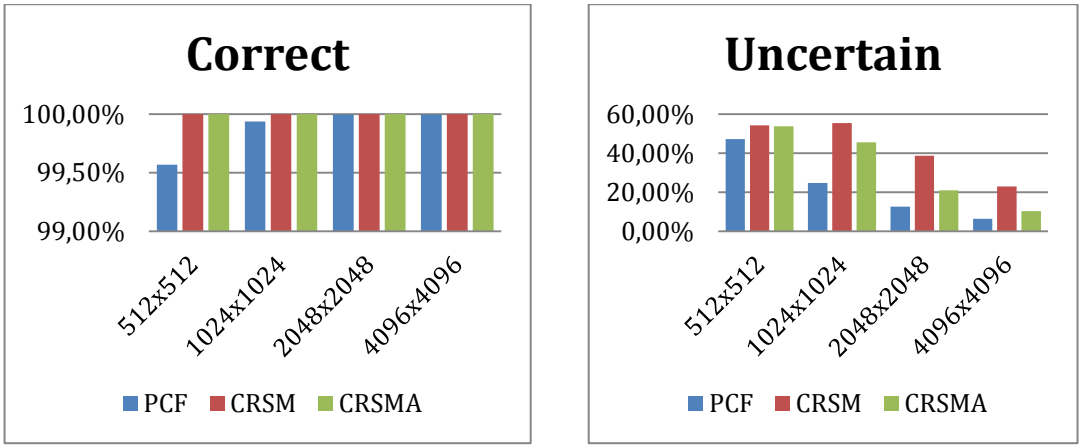


Figure 106 – Shadow Map size Variation results using a fixed viewport of 1024x1024 for the With-Trees scene; (Left) Correct pixel percentages (Certain-C and Uncertain); (Right) Uncertain pixel percentages.

A.2 Performance Testing

A.2.1 Bench Scene

Execution Time (ms): Create Shadow Map step			
Shadow Map Size	PCF	CRSM	CRSMA
512x512	0,01	0,15	0,76
1024x1024	0,01	0,16	0,86
2048x2048	0,01	0,18	0,92
4096x4096	0,04	0,18	0,99

Table 83 - Execution times of the create shadow map step for the With-Bench scene.

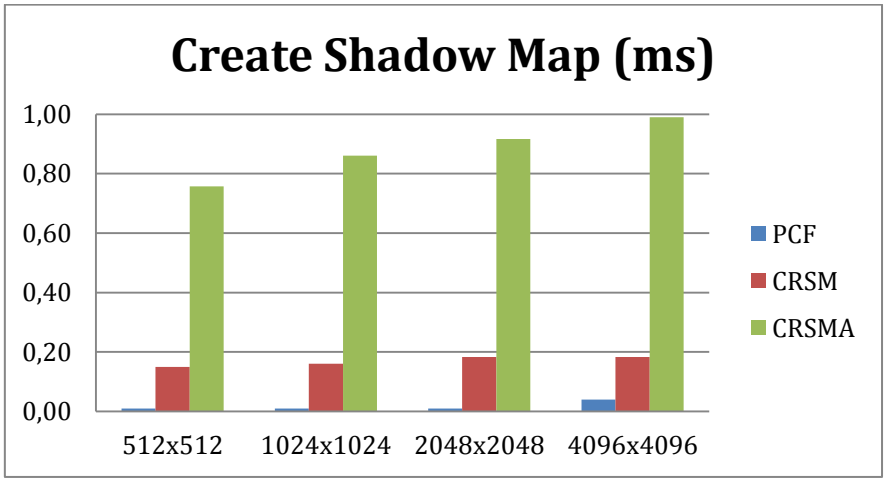


Figure 107 – Graphical representation of the execution times of the Create Shadow Map step for the With-Bench scene.

Execution Time (ms): Detect Uncertain step				
Viewport Size	Shadow Map Size	PCF	CRSM	CRSMA
512 x 512	512x512	0,27	0,27	0,27
	1024x1024	0,27	0,27	0,27
	2048x2048	0,27	0,27	0,27
	4096x4096	0,27	0,26	0,27
1024 x 1024	512x512	0,95	0,95	0,95
	1024x1024	0,95	0,95	0,95
	2048x2048	0,95	0,95	0,95
	4096x4096	0,95	0,95	0,95
1920 x 1080	512x512	1,77	1,77	1,77
	1024x1024	1,77	1,77	1,77
	2048x2048	1,77	1,77	1,77
	4096x4096	1,77	1,77	1,77

Table 84 - Execution times of the Detect Uncertain step for the With-Bench scene.

Execution Time (ms): Fill Ray Buffer step							
Viewport Size	Shadow Map Size	PCF		CRSM		CRSMA	
		#Rays	Time	#Rays	Time	#Rays	Time
512 x 512	512x512	60366	1,13	138190	1,94	132793	1,93
	1024x1024	35573	0,82	114648	1,72	68913	1,56
	2048x2048	19632	0,62	77614	1,49	36347	1,13
	4096x4096	10764	0,52	44513	1,27	19508	0,85
1024 x 1024	512x512	241344	3,91	451748	7,6	381490	7,57
	1024x1024	142554	2,68	458408	6,55	275718	5,73
	2048x2048	78419	1,88	310435	5,3	145600	3,73
	4096x4096	42953	1,47	178031	3,93	77965	2,53
1920 x 1080	512x512	601005	9,15	1051937	14,16	1057880	14,24
	1024x1024	342530	5,83	904947	12,56	719180	12,05
	2048x2048	190290	3,84	673444	10,34	352098	8,14
	4096x4096	100823	2,71	407141	7,62	179835	5,28

Table 85 - Execution times of the fill ray buffer step for the With-Bench scene; the # Rays is equal to the number of "uncertain" pixels found in the Detect Uncertain step.

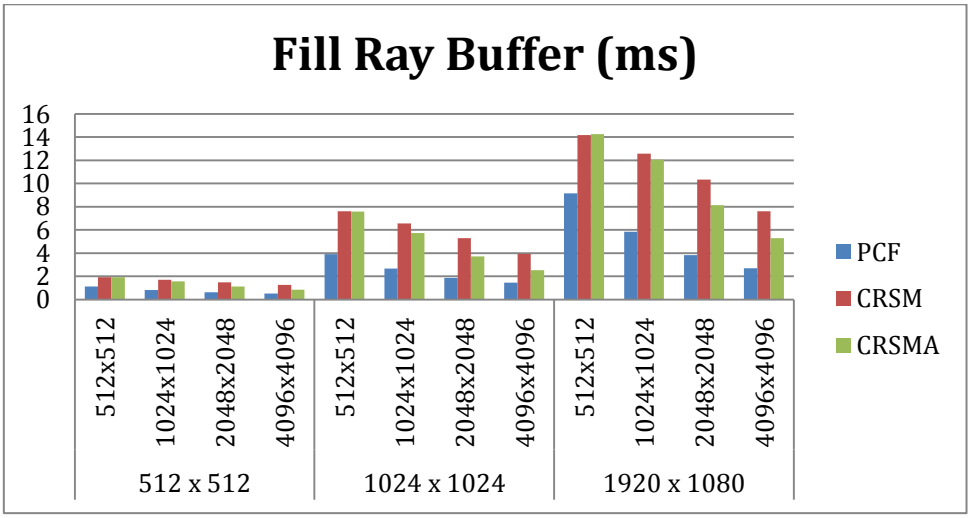


Figure 108 - Graphical representation of the execution times of the Fill Ray Buffer step for the With-Bench scene.

Execution Time (ms): Trace Rays step							
Viewport Size	Shadow Map Size	PCF		CRSM		CRSMA	
		#Rays	Time	#Rays	Time	#Rays	Time
512 x 512	512x512	60366	2,21	138190	2,78	132793	2,80
	1024x1024	35573	2,03	114648	2,67	68913	2,51
	2048x2048	19632	1,81	77614	2,41	36347	2,23
	4096x4096	10764	1,49	44513	2,21	19508	2,00
1024 x 1024	512x512	241344	3,94	451748	6,44	381490	6,42
	1024x1024	142554	3,28	458408	5,74	275718	5,06
	2048x2048	78419	2,59	310435	4,67	145600	3,84
	4096x4096	42953	2,13	178031	3,71	77965	3,16
1920 x 1080	512x512	601005	6,28	1051937	9,86	1057880	9,85
	1024x1024	342530	4,5	904947	8,94	719180	8,67
	2048x2048	190290	3,67	673444	7,41	352098	6,10
	4096x4096	100823	2,66	407141	5,48	179835	4,46

Table 86 - Execution times of the trace rays step for the With-Bench scene; the # Rays is equal to the number of "uncertain" pixels found in the Detect Uncertain step.

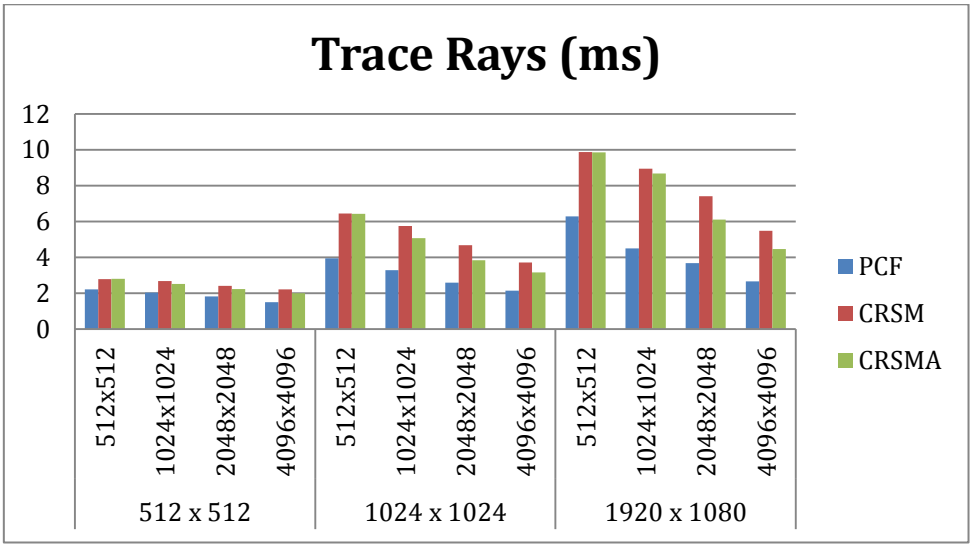


Figure 109 - Graphical representation of the execution times of the Trace Rays step for the With-Bench scene.

Execution Time (ms): Fix Pixels step			
Viewport Size	PCF	CRSM	CRSMA
512 x 512	0,07	0,07	0,07
1024 x 1024	0,29	0,29	0,29
1920 x 1080	0,57	0,57	0,57

Table 87 - Execution times of the fix pixels step for the With-Bench scene.

Execution Time (ms): Pipeline						
Viewport Size	Shadow Map Size	OptiX Prime	NSM	PCF	CRSM	CRSMA
512 x 512	512x512	4,53	0,18	3,69	5,21	5,83
	1024x1024		0,18	3,20	4,89	5,27
	2048x2048		0,19	2,78	4,42	4,62
	4096x4096		0,22	2,39	3,99	4,18
1024 x 1024	512x512	10,78	0,51	9,10	15,43	15,99
	1024x1024		0,52	7,21	13,69	12,89
	2048x2048		0,52	5,72	11,39	9,73
	4096x4096		0,55	4,88	9,06	7,92
1920 x 1080	512x512	17,22	0,96	17,78	26,51	27,19
	1024x1024		0,96	12,68	24,00	23,92
	2048x2048		0,96	9,86	20,27	17,50
	4096x4096		0,98	7,75	15,62	13,07

Table 88 - Execution times of the test pipeline for the With-Bench scene.

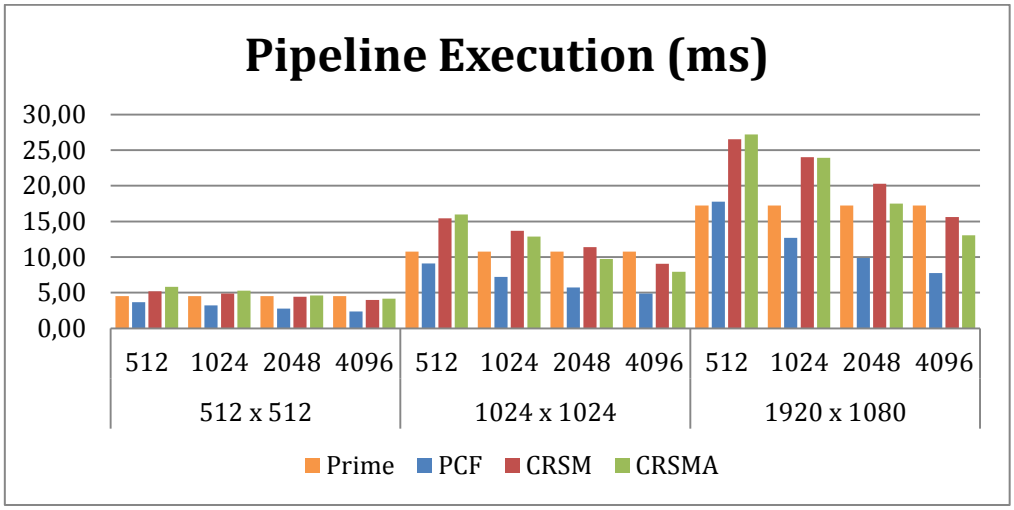


Figure 110 - Graphical representation of the execution times of the Pipeline for the With-Bench scene.

Frame Rate (FPS)						
Viewport Size	Shadow Map Size	OptiX Prime	NSM	PCF	CRSM	CRSMA
512 x 512	512x512	200,46	981,35	153,02	103,54	96,44
	1024x1024		946,07	185,67	111,84	112,31
	2048x2048		968,99	212,54	130,38	135,52
	4096x4096		903,34	244,74	152,23	152,60
1024 x 1024	512x512	86,44	722,02	63,97	35,97	35,18
	1024x1024		728,33	84,65	41,03	46,17
	2048x2048		686,81	108,45	52,08	64,63
	4096x4096		652,32	131,56	68,82	83,04
1920 x 1080	512x512	54,59	535,62	32,11	20,47	20,24
	1024x1024		527,15	47,17	23,16	23,76
	2048x2048		513,87	64,27	28,44	34,81
	4096x4096		514,67	84,88	39,19	50,17

Table 89 – Frame rate of the test pipeline for the With-Bench scene.

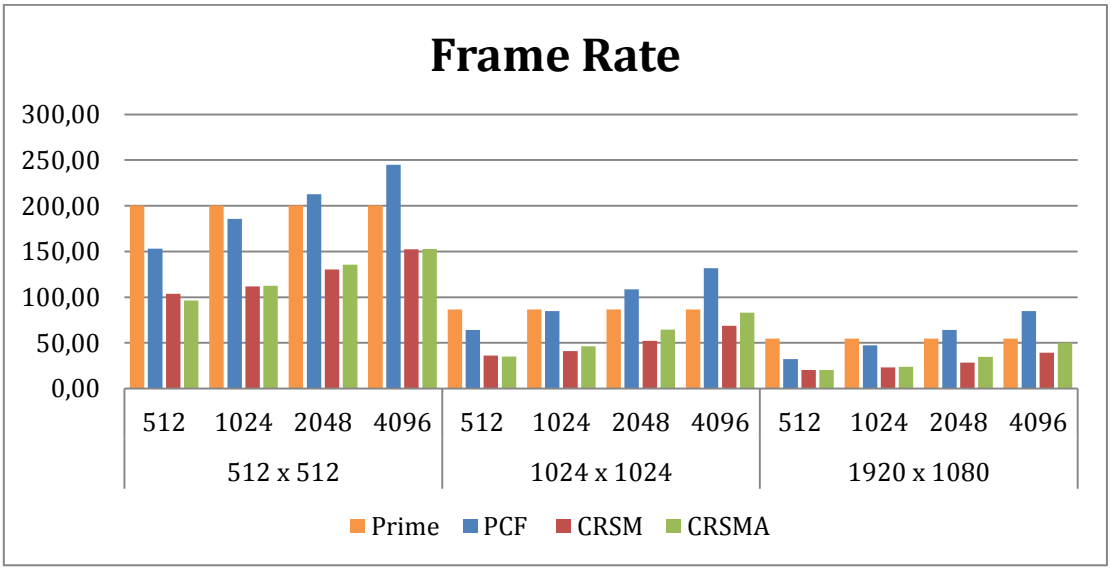


Figure 111 - Graphical representation of the frame rate (FPS) obtained in the With-Bench scene. The first number represents the shadow map size, while the second represent the viewport dimensions.

A.2.2 Flowers Scene

Execution Time (ms): Create Shadow Map step			
Shadow Map Size	PCF	CRSM	CRSMA
512x512	0,01	0,12	0,76
1024x1024	0,01	0,14	0,86
2048x2048	0,01	0,14	0,91
4096x4096	0,01	0,11	0,97

Table 90 - Execution times of the create shadow map step for the Side-Flowers scene.

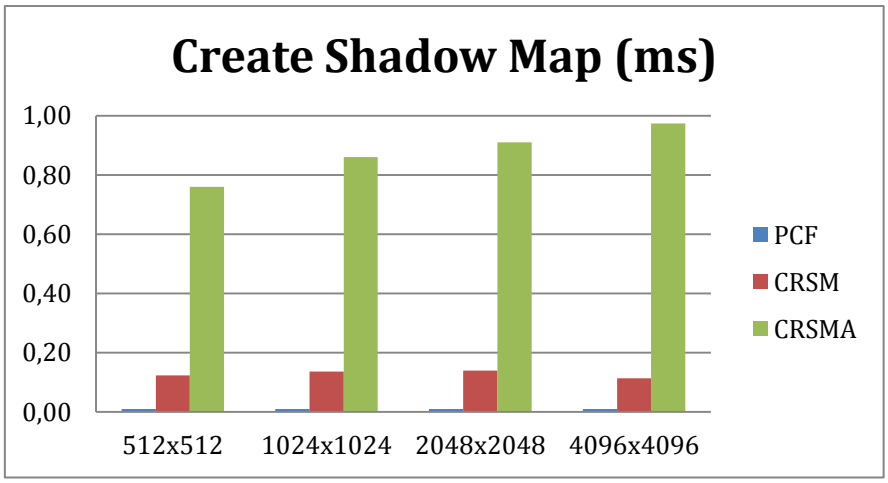


Figure 112 – Graphical representation of the execution times of the Create Shadow Map step for the Side-Flowers scene.

Execution Time (ms): Detect Uncertain step				
Viewport Size	Shadow Map Size	PCF	CRSM	CRSMA
512 x 512	512x512	0,34	0,35	0,35
	1024x1024	0,34	0,35	0,35
	2048x2048	0,34	0,35	0,35
	4096x4096	0,35	0,34	0,35
1024 x 1024	512x512	1,16	1,16	1,16
	1024x1024	1,16	1,16	1,16
	2048x2048	1,16	1,16	1,16
	4096x4096	1,16	1,16	1,16
1920 x 1080	512x512	2,04	2,04	2,04
	1024x1024	2,04	2,04	2,04
	2048x2048	2,04	2,04	2,04
	4096x4096	2,04	2,04	2,04

Table 91 - Execution times of the Detect Uncertain step for the Side-Flowers scene.

Execution Time (ms): Fill Ray Buffer step							
Viewport Size	Shadow Map Size	PCF		CRSM		CRSMA	
		#Rays	Time	#Rays	Time	#Rays	Time
512 x 512	512x512	57250	1,04	129145	1,89	118686	1,90
	1024x1024	46670	0,89	119862	1,84	71946	1,64
	2048x2048	31402	0,78	88192	1,61	52024	1,27
	4096x4096	18842	0,64	53546	1,39	33169	0,98
1024 x 1024	512x512	229184	3,76	516668	7,36	474857	7,48
	1024x1024	186766	3,19	479538	7,01	287869	6,12
	2048x2048	125608	2,57	352932	5,81	208236	4,49
	4096x4096	75324	1,95	214284	4,23	132743	3,23
1920 x 1080	512x512	499844	7,65	946373	13,35	942120	13,29
	1024x1024	334768	5,50	889523	12,79	592366	11,27
	2048x2048	238631	4,52	694762	10,75	405949	8,01
	4096x4096	144211	3,38	448651	7,96	264530	5,80

Table 92 - Execution times of the fill ray buffer step for the Side-Flowers scene; the # Rays is equal to the number of "uncertain" pixels found in the Detect Uncertain step.

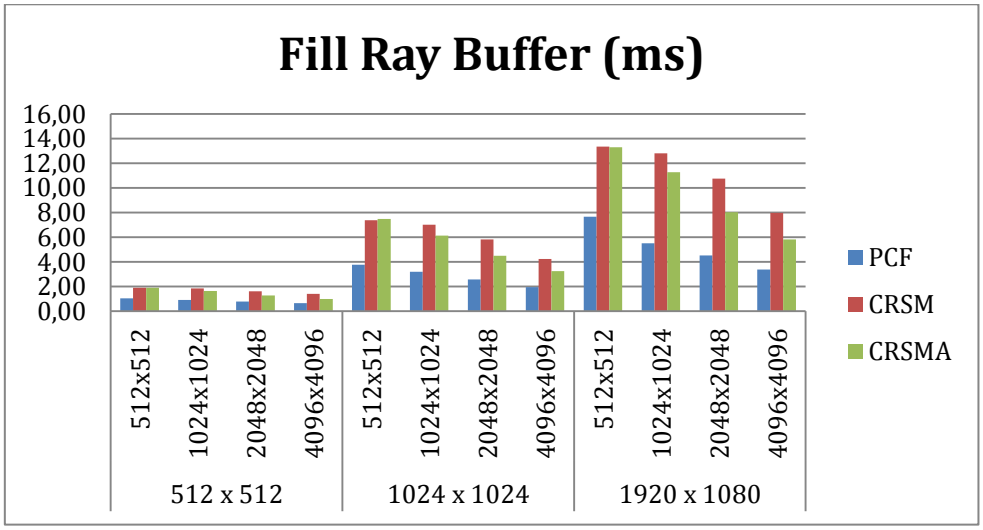


Figure 113 - Graphical representation of the execution times of the Fill Ray Buffer step for the Side-Flowers scene.

Execution Time (ms): Trace Rays step							
Viewport Size	Shadow Map Size	PCF		CRSM		CRSMA	
		#Rays	Time	#Rays	Time	#Rays	Time
512 x 512	512x512	57250	2,74	129145	5,11	118686	5,13
	1024x1024	46670	2,61	119862	5,00	71946	4,57
	2048x2048	31402	2,19	88192	4,49	52024	3,91
	4096x4096	18842	2,40	53546	4,02	33169	3,46
1024 x 1024	512x512	229184	6,21	516668	12,31	474857	12,35
	1024x1024	186766	5,51	479538	11,97	287869	10,45
	2048x2048	125608	4,78	352932	10,21	208236	8,66
	4096x4096	75324	4,49	214284	8,05	132743	7,16
1920 x 1080	512x512	499844	7,77	946373	15,99	942120	15,92
	1024x1024	334768	6,56	889523	15,69	592366	14,34
	2048x2048	238631	6,37	694762	13,58	405949	11,36
	4096x4096	144211	5,55	448651	10,70	264530	9,19

Table 93 - Execution times of the trace rays step for the Side-Flowers scene; the # Rays is equal to the number of "uncertain" pixels found in the Detect Uncertain step.

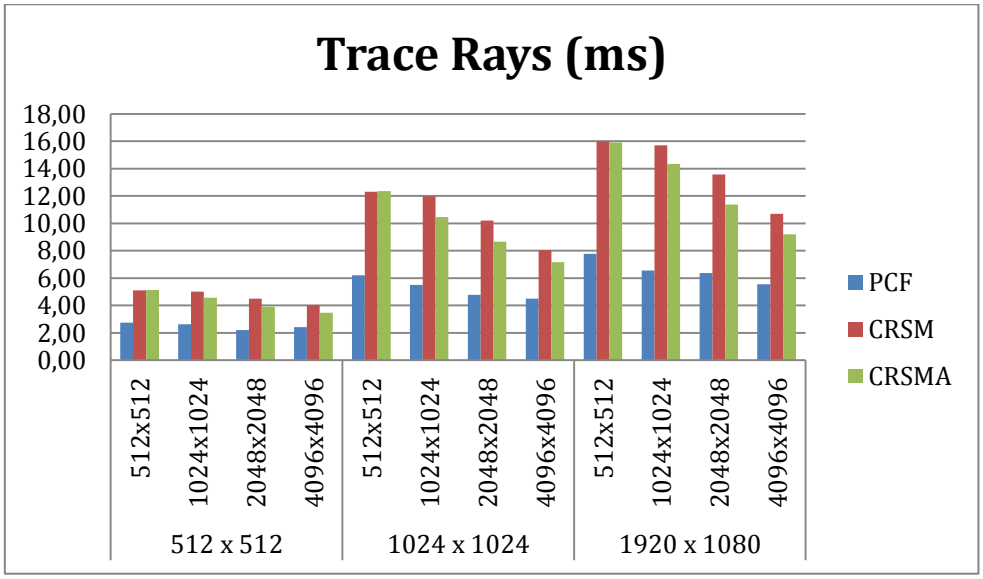


Figure 114 - Graphical representation of the execution times of the Trace Rays step for the Side-Flowers scene.

Execution Time (ms): Fix Pixels step			
Viewport Size	PCF	CRSM	CRSMA
512 x 512	0,07	0,07	0,07
1024 x 1024	0,29	0,29	0,29
1920 x 1080	0,57	0,57	0,57

Table 94 - Execution times of the fix pixels step for the Side-Flowers scene.

Execution Time (ms): Pipeline						
Viewport Size	Shadow Map Size	OptiX Prime	NSM	PCF	CRSM	CRSMA
512 x 512	512x512	8,81	0,19	4,20	7,54	8,21
	1024x1024		0,19	3,92	7,40	7,49
	2048x2048		0,20	3,39	6,66	6,51
	4096x4096		0,20	3,47	5,93	5,83
1024 x 1024	512x512	21,03	0,56	11,43	21,24	22,04
	1024x1024		0,57	10,16	20,57	18,88
	2048x2048		0,57	8,81	17,61	15,51
	4096x4096		0,57	7,90	13,84	12,81
1920 x 1080	512x512	29,63	1,01	18,04	32,07	32,58
	1024x1024		1,01	14,68	31,23	29,08
	2048x2048		1,01	13,51	27,08	22,89
	4096x4096		1,02	11,55	21,38	18,57

Table 95 - Execution times of the test pipeline for the Side-Flowers scene.

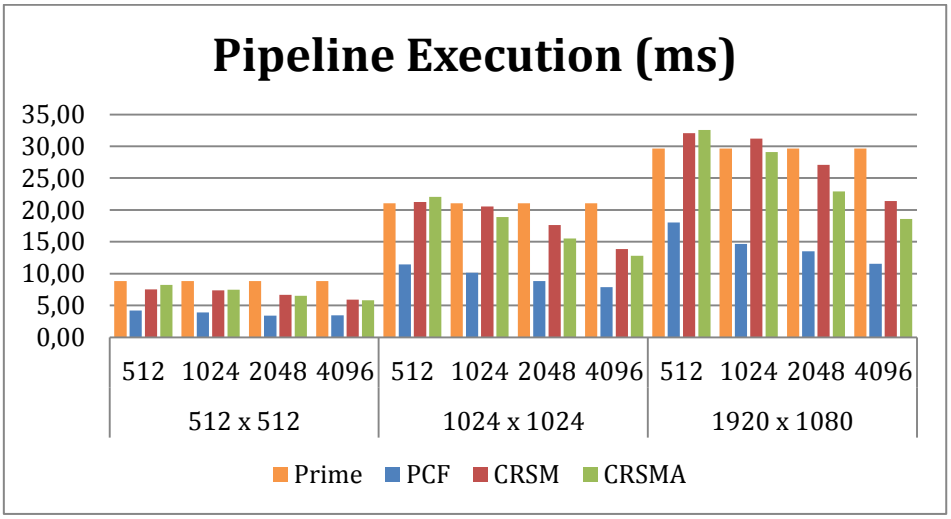


Figure 115 - Graphical representation of the execution times of the Pipeline for the Side-Flowers scene.

Frame Rate (FPS)						
Viewport Size	Shadow Map Size	OptiX Prime	NSM	PCF	CRSM	CRSMA
512 x 512	512x512	108,76	1007,05	141,80	83,79	78,90
	1024x1024		998,00	154,49	86,49	88,55
	2048x2048		986,19	180,93	98,35	103,41
	4096x4096		915,75	181,82	116,32	117,98
1024 x 1024	512x512	45,83	672,49	56,25	30,28	29,67
	1024x1024		717,88	63,86	31,66	35,56
	2048x2048		660,50	76,30	37,94	44,64
	4096x4096		642,67	87,47	49,73	56,15
1920 x 1080	512x512	32,56	503,78	33,91	19,26	19,00
	1024x1024		517,06	43,29	20,02	21,92
	2048x2048		514,14	49,70	23,51	29,06
	4096x4096		503,27	60,76	31,14	37,45

Table 96 – Frame rate of the test pipeline for the Side-Flowers scene.

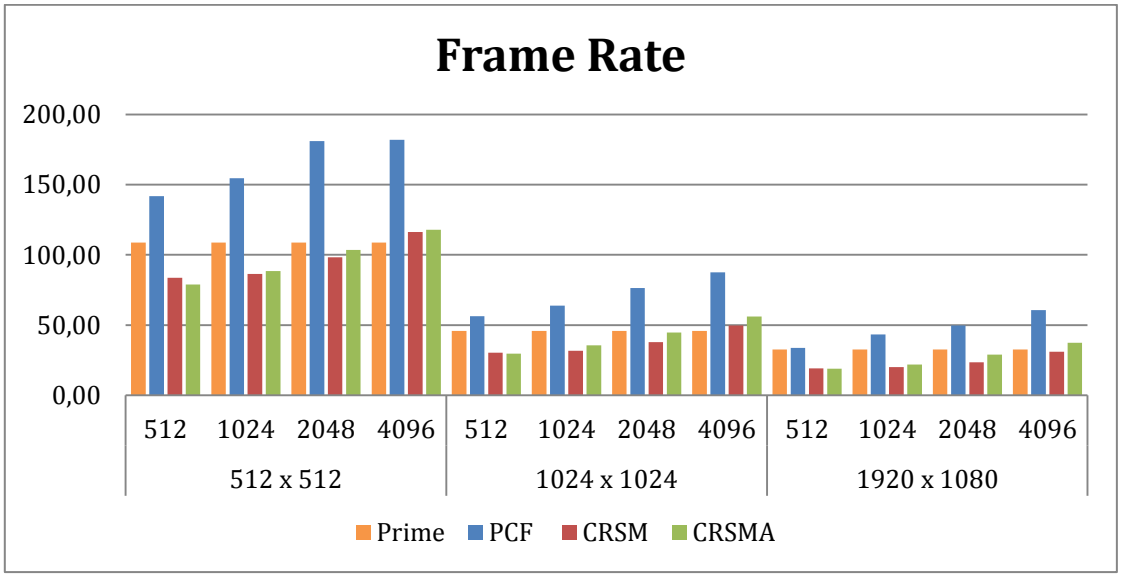


Figure 116 - Graphical representation of the frame rate (FPS) obtained in the Side-Flowers scene. The first number represents the shadow map size, while the second represent the viewport dimensions.

A.2.3 Trees Scene

Execution Time (ms): Create Shadow Map step			
Shadow Map Size	PCF	CRSM	CRSMA
512x512	0,01	0,13	0,76
1024x1024	0,01	0,15	0,86
2048x2048	0,01	0,16	0,91
4096x4096	0,03	0,15	0,98

Table 97 - Execution times of the create shadow map step for the Against-Trees scene.

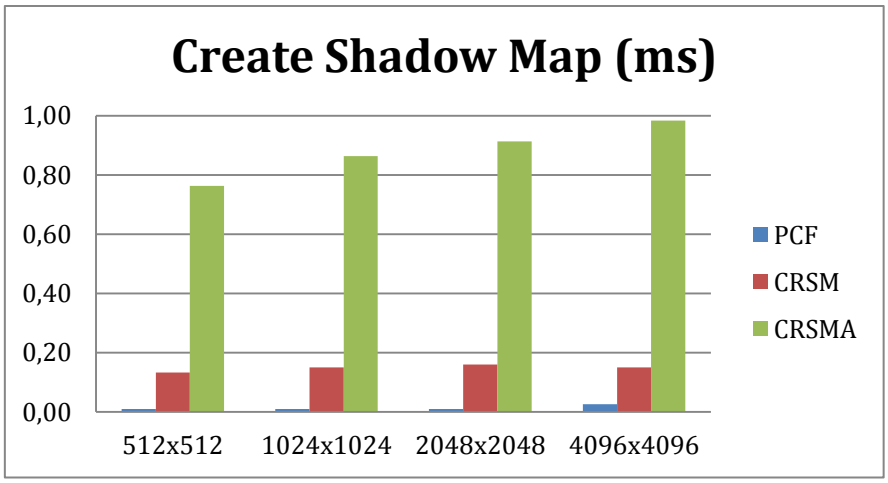


Figure 117 – Graphical representation of the execution times of the Create Shadow Map step for the Against-Trees scene.

Execution Time (ms): Detect Uncertain step				
Viewport Size	Shadow Map Size	PCF	CRSM	CRSMA
512 x 512	512x512	0,24	0,24	0,24
	1024x1024	0,24	0,24	0,24
	2048x2048	0,24	0,24	0,24
	4096x4096	0,24	0,23	0,24
1024 x 1024	512x512	0,24	0,23	0,24
	1024x1024	0,85	0,85	0,85
	2048x2048	0,85	0,85	0,85
	4096x4096	0,85	0,85	0,85
1920 x 1080	512x512	1,65	1,65	1,65
	1024x1024	1,65	1,65	1,65
	2048x2048	1,65	1,65	1,65
	4096x4096	1,65	1,65	1,65

Table 98 - Execution times of the Detect Uncertain step for the Against-Trees scene.

Execution Time (ms): Fill Ray Buffer step							
Viewport Size	Shadow Map Size	PCF		CRSM		CRSMA	
		#Rays	Time	#Rays	Time	#Rays	Time
512 x 512	512x512	261090	2,12	725063	3,21	489823	3,18
	1024x1024	65257	1,53	181263	2,83	122450	2,67
	2048x2048	33839	1,10	118101	2,38	58407	2,02
	4096x4096	17222	0,86	65085	2,16	29122	1,46
1024 x 1024	512x512	480724	8,84	749019	11,00	717779	10,54
	1024x1024	261090	4,80	725063	10,65	489823	9,80
	2048x2048	135326	3,11	472388	8,33	233635	6,47
	4096x4096	68896	2,29	260515	6,06	116583	4,22
1920 x 1080	512x512	1039063	15,82	1770977	24,22	1693958	24,05
	1024x1024	573361	10,08	1486638	20,82	1126609	20,08
	2048x2048	299942	6,44	1001660	16,90	514134	13,30
	4096x4096	154671	4,49	563932	12,06	257146	8,50

Table 99 - Execution times of the fill ray buffer step for the Against-Trees scene; the # Rays is equal to the number of "uncertain" pixels found in the Detect Uncertain step.

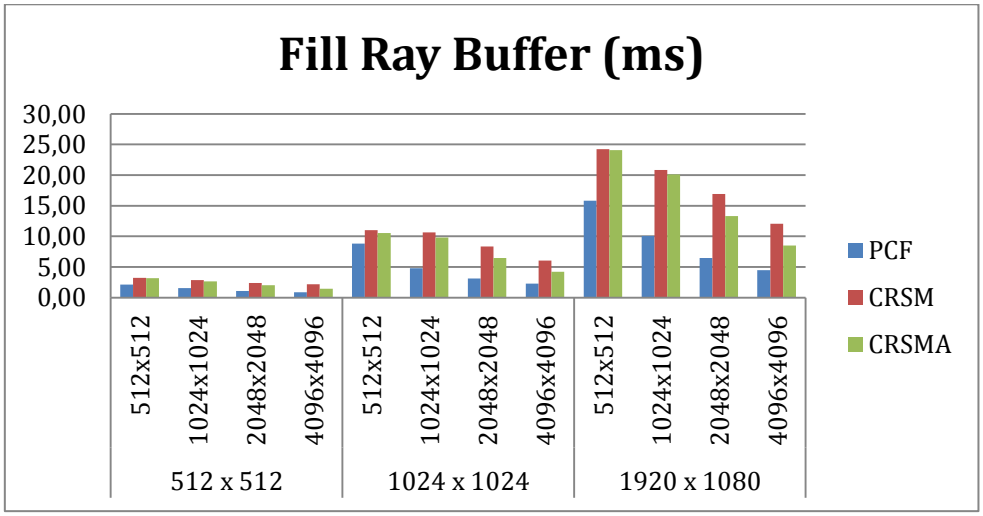


Figure 118 - Graphical representation of the execution times of the Fill Ray Buffer step for the Against-Trees scene.

Execution Time (ms): Trace Rays step							
Viewport Size	Shadow Map Size	PCF		CRSM		CRSMA	
		#Rays	Time	#Rays	Time	#Rays	Time
512 x 512	512x512	261090	1,88	725063	2,88	489823	2,89
	1024x1024	65257	1,35	181263	2,49	122450	2,28
	2048x2048	33839	0,97	118101	1,90	58407	1,59
	4096x4096	17222	0,77	65085	1,37	29122	1,09
1024 x 1024	512x512	480724	6,26	749019	7,50	717779	7,19
	1024x1024	261090	3,40	725063	7,26	489823	6,54
	2048x2048	135326	2,17	472388	5,26	233635	4,08
	4096x4096	68896	1,45	260515	3,36	116583	2,44
1920 x 1080	512x512	1039063	10,35	1770977	15,87	1693958	15,18
	1024x1024	573361	6,72	1486638	13,81	1126609	12,93
	2048x2048	299942	4,13	1001660	10,12	514134	7,96
	4096x4096	154671	2,61	563932	6,44	257146	4,67

Table 100 - Execution times of the trace rays step for the Against-Trees scene; the # Rays is equal to the number of "uncertain" pixels found in the Detect Uncertain step.

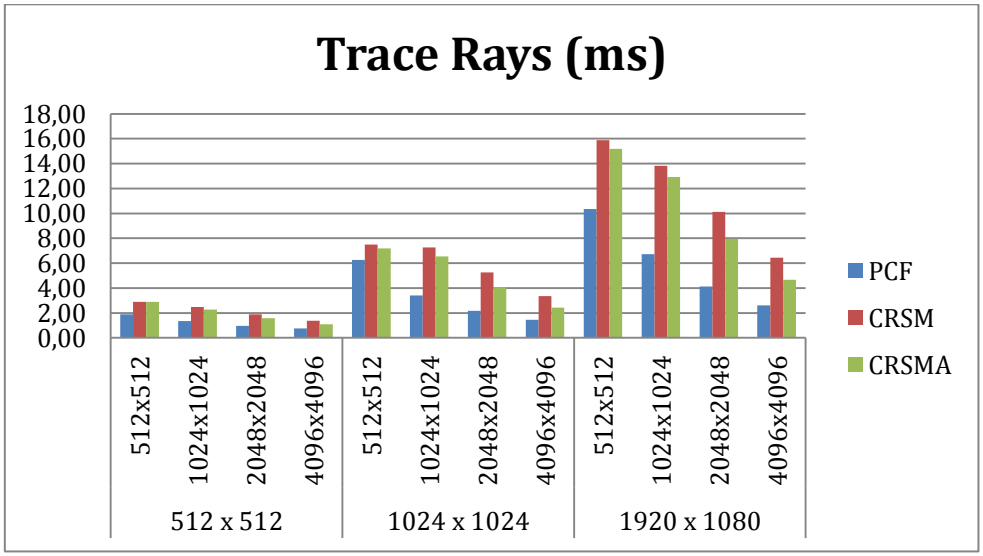


Figure 119 - Graphical representation of the execution times of the Trace Rays step for the Against-Trees scene.

Execution Time (ms): Fix Pixels step			
Viewport Size	PCF	CRSM	CRSMA
512 x 512	0,07	0,07	0,07
1024 x 1024	0,29	0,29	0,29
1920 x 1080	0,57	0,57	0,57

Table 101 - Execution times of the fix pixels step for the Against-Trees scene.

Execution Time (ms): Pipeline						
Viewport Size	Shadow Map Size	OptiX Prime	NSM	PCF	CRSM	CRSMA
512 x 512	512x512	4,98	0,18	4,32	6,53	7,14
	1024x1024		0,18	3,20	5,78	6,12
	2048x2048		0,19	2,39	4,75	4,83
	4096x4096		0,19	1,97	3,98	3,84
1024 x 1024	512x512	13,61	0,49	15,64	19,16	19,02
	1024x1024		0,50	9,35	19,20	18,34
	2048x2048		0,50	6,43	14,89	12,60
	4096x4096		0,50	4,91	10,71	8,78
1920 x 1080	512x512	24,66	0,92	28,40	42,44	42,21
	1024x1024		0,92	19,03	37,00	36,09
	2048x2048		0,94	12,80	29,40	24,39
	4096x4096		0,94	9,35	20,87	16,37

Table 102 - Execution times of the test pipeline for the Against-Trees scene.

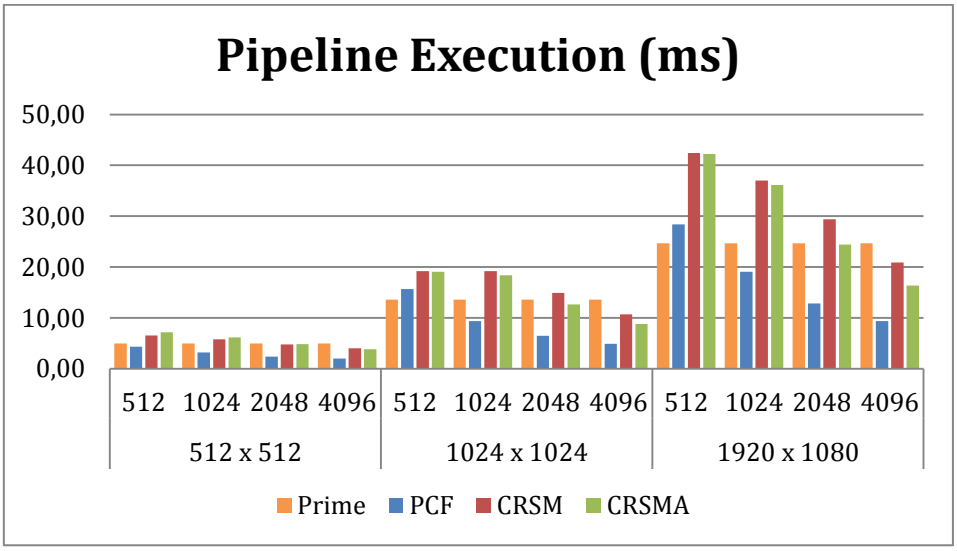


Figure 120 - Graphical representation of the execution times of the Pipeline for the Against-Trees scene.

Frame Rate (FPS)						
Viewport Size	Shadow Map Size	OptiX Prime	NSM	PCF	CRSM	CRSMA
512 x 512	512x512	188,21	970,87	117,80	78,51	74,51
	1024x1024		1013,17	164,88	89,83	91,13
	2048x2048		989,12	223,81	112,49	123,30
	4096x4096		943,40	272,33	147,10	154,30
1024 x 1024	512x512	70,11	748,50	40,03	24,03	23,70
	1024x1024		741,29	61,62	28,72	31,46
	2048x2048		740,19	91,60	38,94	48,33
	4096x4096		684,93	123,20	56,84	72,25
1920 x 1080	512x512	38,78	542,89	19,98	12,50	12,38
	1024x1024		540,25	31,54	14,54	15,63
	2048x2048		522,47	48,78	19,87	25,10
	4096x4096		509,42	69,51	29,77	39,89

Table 103 – Frame rate of the test pipeline for the Against-Trees scene.

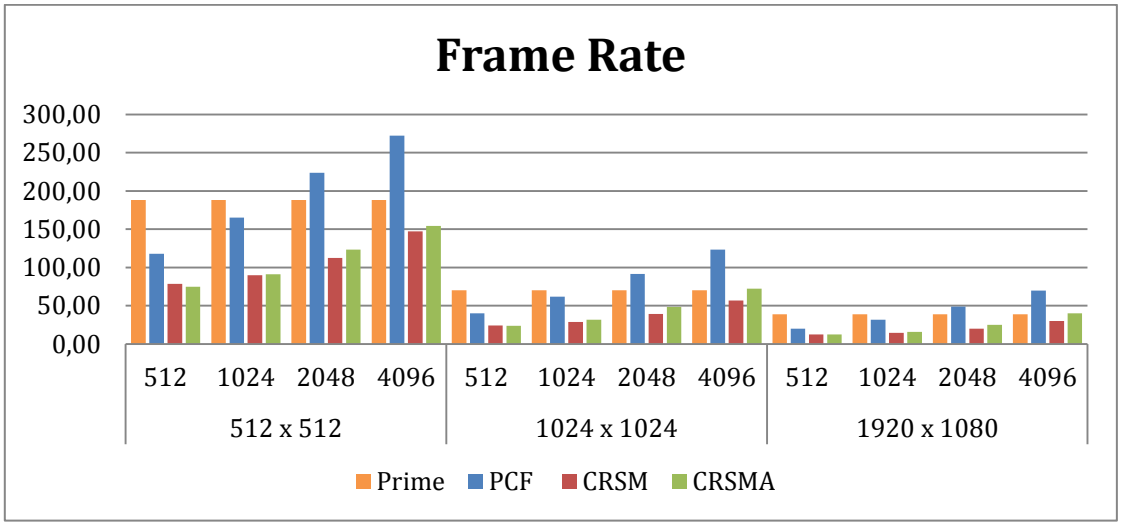


Figure 121 - Graphical representation of the frame rate (FPS) obtained in the Against-Trees scene. The first number represents the shadow map size, while the second represent the viewport dimensions.