Utrecht University

Combining Reprojection and Adaptive Sampling for Real-Time Path Tracing on the GPU

Master Thesis
ICA-3911640

Olaf Schalk

Supervised by

Dr. Ing. J. Bikker
H. van Summeren
Prof. Dr. M.J. van Kreveld

January 19, 2018
version: 1.0

Abstract

Real-time path tracing becomes more and more realistic and is usable for all kinds of graphical applications. Still most implementations of a path tracer tend to see each pixel as equal and they discard all the received results when there is a movement or change in the scene. With our adaptive sampling method, we show that each pixel can be treated differently, based on the variance per pixel received by the path tracer. By using a probability, which indicates how complex a pixel is compared to others, we distribute samples across all the pixels in the screen. Our reprojection method can reuse pixels from previous frames. Not all pixels can be reused, because of material properties, which are dependent on the view position. Therefore, we created an error function, based on the material property and the incoming light energy, which can determine if a reprojection will exceed a specified maximum error. By combining both the methods we can give more priority to pixels with fewer samples and to pixels that are more complex. We demonstrate that our methods can improve a default implementation of a path tracer within the first milliseconds, which is an advantage for real-time applications.

Contents

1 Introduction

Physically based rendering becomes increasingly viable for real-time applications with better hardware and new techniques. It is already used to render animations and photo realistic images, but it is not yet applied in games. The reason for this is that games need to produce at least 24 frames per second, while animations have in principle an unlimited time budget per frame. Also all camera scenery positions within an animation are known, which would be impossible for a game, duo to the interactive nature of these applications.

Most applications discard all the information gained from a physically based rendering technique after rendering a frame. This is a waste, because the next frame is typically similar to the previous frame. Similar points can be reprojected from frame to frame, so you do not have to start with zero information. Within animation rendering reprojection is already in use, by taking two frames and reprojecting points between the two frames. Reprojection would only work when there are no drastic changes between the frames. Most games use smoothly moving cameras, so reprojection could be a good improvement for games.

Physically based rendering techniques also tend to spread their focus evenly over all the pixels. The variance of individual pixels depends on local materials and illumination. By using adaptive sampling it is possible to focus more on pixels who have a higher variance. This can reduce the amount of samples needed to get the same result.

By combining the two techniques some information of a previous frame can be reused by reprojection. All the information that is still missing can be focused on by using the adaptive sampling technique. So this combination will ensure data reuse and focus on the right pixels, which makes it possible to improve the performance of a physically based renderer.

1.1 Thesis goal

This thesis investigates the combination of reprojection and adaptive sampling in the context of interactive applications with small changes in the camera view, to improve image quality.

From the goal follows the research question:

- **By what factor can we improve the efficiency of a real-time path tracer, using reprojection and/or adaptive sampling, without exceeding a specified maximum error?**
- Research subquestions:
  - Does reprojection yield better efficiency?
  - How can we calculate an upper bound for the reprojection error?
  - Does adaptive sampling yield better efficiency?
  - What is the maximum error threshold until it is noticeable from the true value?

1.2 Thesis overview

In chapter 2 related work is discussed which is relevant for this thesis. First the basics of physically based rendering are discussed, with path tracing as the focused rendering technique. Furthermore other research about adaptive sampling and reprojection will be discussed.

The reprojection method in combination with the adaptive sampling method, used in this thesis, is explained in chapter 3. How pixels are weighted during reprojection and how this relates to the viewpoint of the camera is also elaborated in this chapter.

Chapter 4 shows the results of the two individual implementations of reprojection and adaptive sampling. The chapter also shows the results of the combination. The approaches are compared against a default path tracer. The results will be discussed in chapter 5. The final chapter 6 summarizes the thesis. This chapter contains also the conclusions and future research.

2 Related work

2.1 Physically based rendering

Physically based rendering algorithms all simulate the flow of light in the real world, by solving the rendering equation by Kajiya (1986):

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega f(x, \omega_i, \omega_o) L_i(x, \omega_i)(\omega_i \cdot n) d\omega_i \tag{1}$$

$L_o(x, \omega_o)$ is the amount of light coming from point x towards the direction ω_o. This comes from $L_e(x, \omega_o)$ the amount of light emitted from point x towards ω_o, plus the integral over all incoming directions ω_i of the hemisphere Ω on point x, with $L_i(x, \omega_i)$ the incoming radiance which is translated to irradiance by $\omega_i \cdot n$ where n is the surface normal on point x, multiplied by the Bidirectional Scattering Distribution Function (BSDF) $f(x, \omega_i, \omega_o)$.

For most scenes, it is not possible to directly evaluate the rendering equation. Several rendering algorithms use Monte Carlo integration to estimate it:

- Path tracing (Kajiya (1986))
- Bidirectional path tracing (Lafortune and Willems (1993))
- Metropolis light transport (Veach and Guibas (1997))
- Photon mapping (Jensen et al. (2002))

In this thesis we will focus on the path tracing algorithm.

2.1.1 Path tracing

The path tracing algorithm proposed by Kajiya (1986) solves the rendering equation 1, by using a Monte Carlo integration for the integral. The main idea is to trace a ray from a viewpoint to a light source instead of the other way around. Each time the ray hits a non emissive surface it will randomly select a new direction based on the BSDF. Once the ray hits an emissive object the energy will be returned to the pixel, which will define the pixel color. Repeating this procedure for a pixel and averaging the results will lead to the correct result eventually.

The advantage of path tracing is that it is simple to understand and easy to implement, but without any variance reduction techniques, there will be quite a lot of noise before it converges. To reduce the variance, techniques like Next Event Estimation and Importance Sampling are used.

- **Next Event Estimation:** Each time a ray hits a non-emissive surface a shadow ray will be cast to a random light. If the shadow ray is not blocked by another object, it will add the energy of the light to the path. This approach increases the amount of rays to cast, but surfaces that are visible to a light will converge faster to the ground truth.
- **Importance Sampling:** When a ray takes a new direction within a hemisphere, after hitting a surface, it has to return energy to improve the result. The idea behind importance sampling is to not give each direction on the hemisphere the same probability to be chosen, but to give higher probabilities to directions that are expected to yield more energy.

Not only a path tracer converges faster by using variance reduction techniques, but also if the calculations could be done cheaper. This means that you can take more samples per pixel (SPP) within the same amount of time. Each ray in path tracing is calculated individually, this makes path tracing perfect for parallel computations on the CPU and massive parallel computations on GPU, which increases the amount of calculations. To reach real-time path tracing a GPU is needed. With the increasing performance of hardware, we are getting very close to real-time path tracing.

2.2 Adaptive Sampling

Adaptive sampling is an approach which estimates which pixels would benefit most from additional samples. In this case not all the pixels will get the same amount of samples. This way a surface in the scene can get more attention if it converges slower than other surfaces. For instance, a partial refractive surface will need more samples than a diffuse surface to converge.

Recent advances in adaptive sampling for Monte Carlo rendering techniques are discussed in Zwicker et al. (2015). They summarize different adaptive sampling and reconstruction techniques with their advantages and disadvantages. The article categorizes the different techniques into a priori and a posteriori group. Priori techniques use analytical analysis of the light transport equation, which typically requires additional scene information. Posteriori techniques use statistical error estimations based on acquired samples.

For priori techniques the challenge is to combine the analysis of all relevant effects (motion blur, depth of field, glossy indirect illumination) into one framework. These techniques also require storage of all initial samples and have a higher reconstruction overhead. In comparison with the priori techniques, the posteriori techniques get all their information from the renderer, from where the techniques can get the pixel variances and means. The advantage is that a posteriori technique can be build easily on top of a renderer. A disadvantage, as stated by the authors, is that it is still too expensive for real-time applications even when it is implemented on a GPU.

Next we will discuss a multidimensional method that is known as a priori technique in section 2.2.1. Also we will discuss some image space methods that are known as posteriori techniques in section: 2.2.2.

2.2.1 Priori technique

A priori technique is one from Hachisuka et al. (2008). It is a multidimensional adaptive sampling technique and is build further upon the first research into the direction of multidimensional adaptive sampling by Mitchell (1991).

The multidimensional adaptive sampling technique is used to capture more camera effects like:

1. Motion blur
2. Soft shadows
3. Depth of field

A multidimensional domain can be seen as image locations over time for motion blur, or image locations with lens aperture for depth of field.

The approach detects discontinuities based on the radiance received from samples within the multidimensional domain. This differs from an image space method, which only looks for discontinuities within the image space. The first samples are stored within a kd-tree in the multidimensional domain. The leaf with the highest variance will receive a new sample within the leaf, that is placed such that it has the largest distance between all other existing samples. If a leaf exceed the amount of samples it will be median-split along the longest dimension. This will be repeated until the maximum amount of samples is reached.

The multidimensional adaptive sampling technique seems promising for visual camera effects, but as the number of effects increase, so will the amount of dimensions. Here the curse of dimensionality will become a problem, not only for computation time, but also for memory cost.

2.2.2 Posteriori techniques

Image space methods for adaptive sampling require an estimation of variance in the rendered image. Several authors suggest different ways to do this and show how they use the variance to determine which pixels should receive more samples.

One of the earlier image space methods was used for anti-aliasing from Kirk and Arvo (1991). They proposed a method that should remove the bias that adaptive sampling introduces. Therefor they estimated the mean for a pixel, by using multiple samples. When the variance, of all samples that are used to estimate the mean, is higher than a certain threshold, then the pixel will get more samples.

The paper states that it is not always beneficial to remove the source of the bias, because the error is typically small and an unbiased approach will give additional computation cost.

Tamstorf and Jensen (1997) also had the idea to not waste computation time to small differences that a person can not see or a display could not show. Their adaptive sampling is based on the paper of Purgathofer (1987), who uses the probability that the true mean lies around a certain width of the estimated mean, to determine the minimum amount of samples per pixel. In addition Tamstorf and Jensen (1997) takes into account the tone operator (gamma correction). For the maximum allowed error they used $1/256$, corresponding to the color resolution of the final image. Since this approach for adaptive sampling leads to the bias, they presented the amount of bias in the results. Concluding from the paper is that the bias is significant, but that it is not a problem, because the error is invisible on a typical display.

A greedy approach is used by Rousselle et al. (2011), using the pixel values gained from the Monte Carlo samples. With the greedy approach they determine where to cast more rays to reduce the mean squared error. The mean squared error (MSE) can be calculated as followed:

$$MSE = \frac{1}{n}\sum_{i}^{n}(\hat{S}_i - S_i)^2 \tag{2}$$

Where n is the amount of pixels, S_i is the true pixel value and $\hat{S}_i$ is the estimated pixel value.

For each pixel they want to calculate the MSE by using Gaussian smoothing filters at different scales from fine to coarse. One problem for calculating the MSE is that the true pixel value is not known during path tracing, unless a reference image is already created. Therefore they calculate the expected change in MSE by:

$$\Delta MSE \approx \underbrace{\frac{r_{f_f}^2 + r_{f_c}^2}{r_{f_f}^2 - r_{f_c}^2} * (f_c - f_f)^2}_{\text{Approximate bias term}} + Var(f_c) - Var(f_f) \tag{3}$$

With r the scale of the filters, f_c and f_f the coarse and fine filtered pixel value and $Var(f)$ the variance of the filtered pixel. They use the fine filter when $MSE(f_c) - MSE(f_f) \geq 0$. The expected change of MSE is stored into a priority queue. Now the pixels that have a higher chance of decreasing the total MSE will be sampled more. At last each pixel is filtered by a Gaussian filter with the selected scale during the calculation of the expected change in MSE.

Their results show that they reduce the MSE with less samples per pixels than a path tracer that gives each pixel the same amount of samples. For their simplest scene the adaptive sampling and filter selection takes 4.36% of the rendering time on a renderer by Pharr et al. (2016). They see this as the worst case, because it is independent on scene complexity. However it takes different rendering times in other scenes which could be a cause of the adaptive sampling that chooses more "difficult" samples to trace in a scene. In the future work of the paper they propose to implement the greedy error minimization on a real-time GPU ray tracer what could be promising.

Li et al. (2012) uses a different approach to estimate the MSE using Stein's Unbiased Risk Estimator (SURE).

$$SURE(f(S_i)) = ||f(S_i) - S_i||^2 + 2Var(S_i)\frac{df(S_i)}{dS_i} - Var(S_i) \tag{4}$$

Here $f(S_i)$ is a filter function over pixel S_i. Instead of taking the first filter that has a MSE difference greater than zero, they take the filter with the least SURE error. The filter is used at the end of the rendering algorithm.

The adaptive sampling function is created such that regions with higher variance receive more samples. They also guide more samples to darker areas, because human eyes are more sensitive to see errors in dark regions, this is adopted from Overbeck et al. (2009). The adaptive sampling function looks as follows:

$$As(S_i) = \frac{SURE(f(S_i)) + Var(S_i)}{I(f(S_i))^2 + \epsilon} \tag{5}$$

Where $f(S_i)$ is the filtered color using cross bilateral filters (cross bilateral filters proved to give the best results in this paper), $I(f(S_i))^2$ is the squared luminance of the filtered pixel color, and ϵ is a small number

to prevent division by zero. How many samples pixel S_i will receive, when there is a budget of m samples, could be determined by:

$$Samples(As(S_i)) = m * \frac{As(S_i)}{\sum_j As(S_j)} \tag{6}$$

What depends on all the neighboring pixels S_j used in the cross bilateral filter.

The results show a comparison of the SURE-based approach with the greedy error minimization (GEM) of Rousselle et al. (2011). In figure 1 it can be seen that the MSE of GEM is comparable to the SURE-based approach, but GEM tends to send few rays to regions where most of the samples carry zero radiance. The limitation of the SURE-based approach is that it does not preserve details in scenes with high-frequency textures using a low sample budget. This is due to the noise created by environmental lighting and glossy reflections in combination of using a bilateral filter. Just like Rousselle et al. (2011) this paper also wants a GPU implementation of their approach to see the effects on a real-time path tracer.

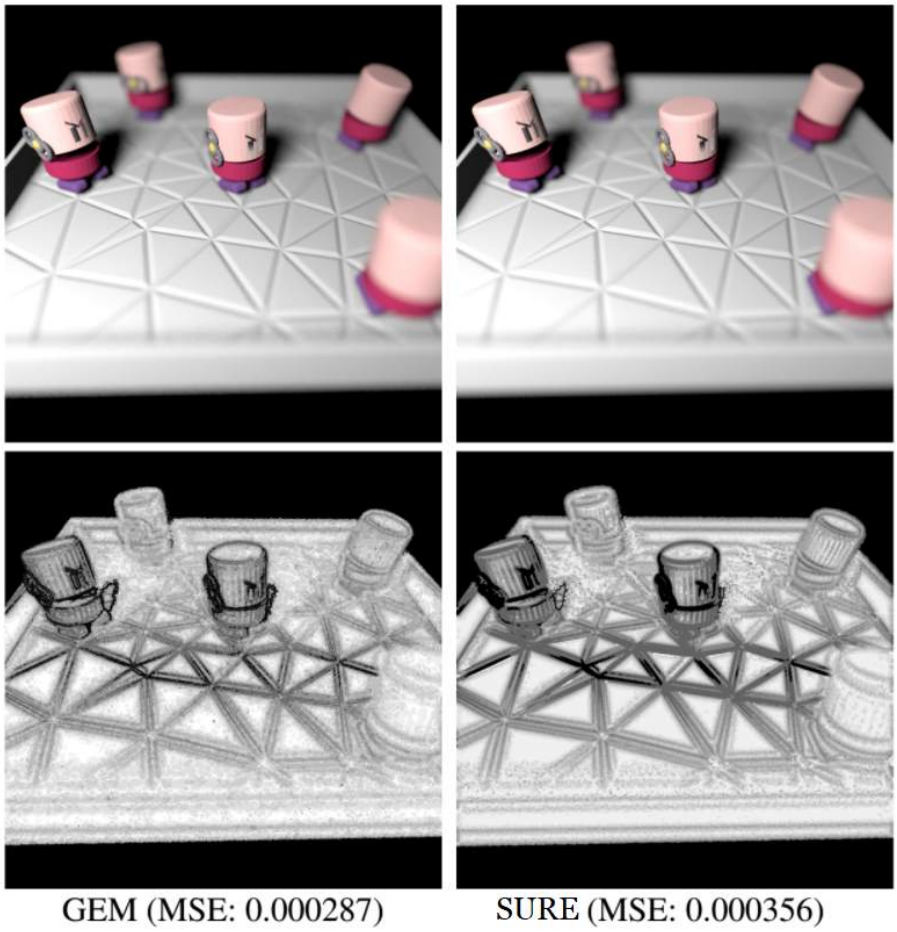


Figure 1: Result of the comparison between GEM by Rousselle et al. (2011) and SURE by Li et al. (2012). A low sample density is represented by the color white. The color black represent a high sample density.

2.3 Reprojection

Each iteration of the path tracing algorithm contributes to the result. This continues until a small change occurs within the scene, for example: an object is added, removed, replaced or the camera viewpoint is changing from position. When this happens the previous result from the path trace algorithm is discarded and it restarts all over again for the few new changes in the scene. By using reprojection, not everything from the previous result is discarded, but previous results are reused as much as possible. On Shadertoy there is a proof of concept of reprojection. Reprojection is mostly used for animations where the camera's and objects only change slightly in position, because if the camera gets a completely different viewpoint, then there is nothing to reproject.

2.3.1 Reprojection in animations

Adelson and Hodges (1993) uses reprojection within a ray-tracer to generate animation frames. The initial frame should be fully ray-traced and the intersectionpoint, normal vector, diffuse color, ID's of the intersected object(s) and the shadowing object(s) should be saved. Their technique consists of four stages:

1. **Reprojection:** Take the samples from the base frame and project them to the new viewpoint. When two samples reproject to the same pixel, the sample with the closest intersectionpoint to the viewpoint will be chosen.
2. **Verification:** Test if the reprojected samples are now occluded, due to a moving object or viewpoint. If the reprojected sample is occluded, it should be discarded.
3. **Enhancing:** All viewpoint dependent phenomena are added: reflection, refraction and specular highlights. These effects cannot be guaranteed to have remained unchanged since the previous frame. The number of ray-casts in this stage will be approximately equal to the amount in a fully ray-traced image.
4. **Filtering:** The samples are filtered down to image resolution using a desired filter.

In a scene with static light sources this approach saves 50 to 90 percent of the rays to cast and 45 to 56 percent in rendering time according to the paper. Scenes that only have diffuse objects save up 60 to 96 percent of the rays needed to be cast. What the exact timings are of this approach is not given, but the reduction in ray casts seems promising.

The research of Zimmer et al. (2015) uses decomposed rendered images with motion vectors to interpolate between animation frames. Not only do they interpolate between frames, but they also denoise and upsample the final result. Decomposition of a rendered image results in disjoint path space components, where each component represents an image buffer. Each component is associated with a set of features that are easily obtained as byproduct of path tracing. The features are extracted from the first non-specular object that is hit and consists of:

- Reflectance value
- Normal vector
- Object ID
- Face ID
- Texture coordinates
- Amount of emitted radiance from visible light

For the interpolation between frames they use their motion vectors. The primary motion vectors are easily obtained by mapping the underlying intersections of visible surface positions forward in time and projecting the 3D motion into screen space. Specular and refraction motion vectors are more challenging. They propose a generalized version of the manifold exploration from (Jakob and Marschner, 2012) to compute the apparent motion of objects observed through a sequence of specular reflections or refractions. In their decomposed framework they interpolate between each separate component using the motion vectors.

The results of Zimmer et al. (2015) shows that interpolating between frames improves the quality of the animation. The disadvantage in this approach is that the reference frames they use to calculate the motion vectors need to be known in advance. For animation rendering this will not be a problem, because all camera positions are known, but for real-time path tracing, where the camera positions are dynamic, the approach will not work.

2.3.2 Reprojection for reflections

Xie et al. (2017) introduce a novel reflection reprojection method to establish the mapping of reflections between individual frames. For their approach they start by generating reflection buffers B that contain the following information for each pixel:

1. The normal vector and depth of an intersected object.
2. The specular factor ranging from 1, for a perfectly specular surface, to 0, for a perfectly diffuse surface.
3. The reflection color.
4. The traveling distance of the reflected ray

They use a mapping function that describes a pixel's offset vector from the reference frame due to the camera movement between the frames. View-independent pixels are validated by a threshold based on the depth to the intersected object $|Distance_{current} - Distance_{previous}| < \epsilon$.

For view-dependent pixels a depth threshold will be invalid, because it varies according to the change in camera position. Therefore they created a distance function for reflective surfaces, which looks at the most similar reflectance point in the buffer B. The distance function is calculated for multiple neighboring pixels and the one that gives the smallest distance, is chosen to use as reprojection value. If non of the other pixels give a smaller distance than the original reprojected pixel, or if one will fall out of the reflection surface, then the pixel will not be reprojected.

Their reflectance reprojection method can handle multiple reflections by extending the reflection buffers and apply multiple passes recursively. Only the authors state that it is not worth the additional cost over the visual addition. For future work the authors would like to see an extension to other view-dependent components, like refraction and glossy reflections.

Another approach that tries to reproject specular and even refractive materials is the one from Lochmann et al. (2014). They use a server to render the images, which are encoded as RGBZ ray trees from Whitted (2005). The server splits the rendered image into diffuse, reflection and refraction including the depths of the ray from the camera. This data is send to the client who first retrieves the diffuse positions and normals and subsequently the specular positions. These are used to create a flow from one frame to the other that will be applied to each node in the ray tree. At last the reprojected images are composed for the result image.

The results show that there is a blurry effect, because of the compressed images. The author states that a more adapted compression and blurring scheme would be required to get the right balance between quality and performance.

2.3.3 Other reprojection techniques

Reprojection is also used for shadows like in Tole et al. (2002). Here they use a Shading Cache that contains recently computed shading values within an objects-space mesh data structure. The shading cache is used to refine the rendered image. Updating the shading cache is done asynchronous from updates of the camera or moving objects. This way they can display moving objects at high frame rates without error in geometry or texture, which will be independent of the shading speed. The main drawback of the approach is that the shading is not correct if an object or camera just moved.

Günther and Grosch (2015) focused more on scene editing, were they try to reuse as much of the previous scene as possible. For their rendering they make use of stochastic progressive photon mapping from (Hachisuka and Jensen, 2009). The equation for a new frame, if something has changed within the scene, is as follows:

$$L_{repair} = (L_{pre} + L_{diff}) * (1 - M) + L_{new} * M \tag{7}$$

Here L_{repair} is the new value for the changed scene, L_{pre} is the previous scene, L_{new} is the new scene, L_{diff} is the difference between the new and previous scene and M is a mask. The value of the mask depends on whether an eye ray hits a modified object and on an error value that is bounded by a threshold. When an eye ray hits a modified object, the pixel will be reseted by setting the mask to one. Otherwise the error value will decide the value of the mask. The error value is calculated as follows:

$$\epsilon = \begin{cases} \frac{L_{diff}}{L_{diff}+L_{pre}} & \text{if } L_{diff} \geq 0 \\ \frac{L_{diff}}{L_{diff}-L_{new}} & \text{else} \end{cases} \tag{8}$$

If $\epsilon > \tau$ then the mask will also be set to one. Otherwise the mask will be set to the error value. Here the τ is a threshold that is manually set between 0 and 1.

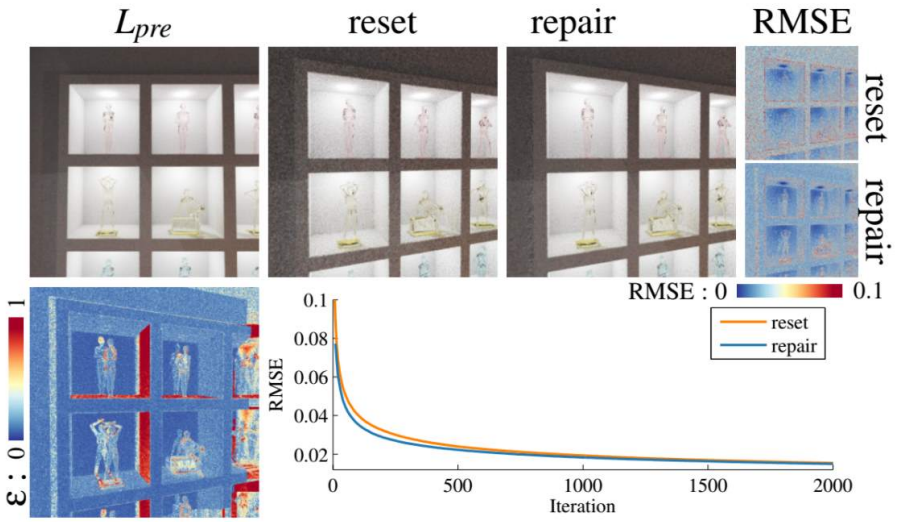


Figure 2: Reprojection result from Günther and Grosch (2015) using 100 iterations of stochastic progressive photon mapping. The RMSE over the amount of iterations is shown in the graph.

For camera movements they establish the correspondence of the old and new frame by reverse reprojection (Nehab et al., 2007). For each pixel they shoot a new ray through the center of the pixel and reproject the first intersection point to the old camera view. If the point is within the old camera view and not occluded the pixel can be reused. When the object has a glossy or glass material, it could be that the reprojection is incorrect, this will increase the error calculated in equation 8 what makes the approach use less of the the previous frame for those pixels.

Within this paper they also try their method on a bidirectional path tracer, but only when removing a glass ball. For future research they recommend more experiments with bidirectional path tracing. The result when using this method for reprojection can be seen in figure 2. Here they use 100 iterations of stochastic progressive photon mapping for L_{pre} and L_{new}. Also they show the RMSE with different amount of iterations for the refreshed frame and the repaired frame. By combining the old and new frame a small reduce in the RMSE can be noticed. Calculating the mask is straightforward and it only uses the byproduct and outcome of a rendering algorithm. This is an advantage for the memory cost and calculation time. The disadvantage of the approach is that it needs the exact same amount of samples of the new frame before it can be combined with the old frame.

A GPU implementation for stable ray-tracing is created by Corso et al. (2017). At each frame they reproject samples by using a forward reprojection. Therefore they store the first intersection position in object space with a transform ID to make it possible to transform it back to world space coordinates. To make sure not too many or few samples are projected to the same pixel, they divide each pixel in $M * M$ subpixels. A corresponding bitmask is used to notify if a subpixel is occupied, which is later used to determine the local sample density. They also use another bitmask to indicate if a subpixel is occluded or not, which will be calculated in a verification phase. When the occupied bitmask for a specific subpixel is 0, the reprojection sample will be copied and the bit will be set to 1, otherwise they look at the occlusion bitmask. If the destination occlusion bit is one and the source occlusion bit is zero, it will be copied, otherwise the sample will be discarded.

After the reprojection is done, they analyze the data and choose where samples should be added or removed due to a desired sampling rate. For a good spatial distribution of samples the pixels are divided into strata. If they have to remove or add samples, they choose from the stratum with the highest or lowest amount of samples. The number of samples to remove or add is decided on the local sample density $d = N/A$, where N is the amount of unoccluded samples in an area of A around the current pixel. A density target and tolerance is set by the user. When the density stays within bounds there are no samples that need to be removed or added.

In the verification phase a ray is cast to the object intersection point from the new camera position to validate if there is an occlusion. When the traveling distance of the ray is different than the distance between the intersection point and the camera position, the subpixel will be set to occluded. At last all the subpixel samples are filtered by a truncated spatial Gaussian filter.

The total overhead of reprojection with this approach is between 1.4x and 1.5x of the overall rendering time. When they take four samples per pixel, the approach can remove most of the spatial aliasing artifacts and provides a fairly stable result. The advantage of the method is that it combines samples when they reproject to the same pixel. This reduce instability after reprojection, because now the pixel value will be averaged. The amount of samples that can be combined is dependent on the size of the buffer, but it is not known if this influence the result.

3 Method

In this chapter we explain our proposed method. We implement and evaluate three approaches:

- Adaptive sampling
- Reprojection
- Combination of adaptive sampling and reprojection

First in chapter 3.1 the implementation details of adaptive sampling are elaborated. Then, in chapter 3.2 the implementation of reprojection is explained including how the introduced error caused by reprojection is determined. And finally how the combination of adaptive sampling and reprojection is handled, is elaborated in chapter 3.3.

3.1 Adaptive sampling

To determine which pixels should get more samples is done by looking at the variance over all the samples that are taken on a pixel. Just like other adaptive image space methods (Rousselle et al., 2011; Li et al., 2012) the variance is taken into account to determine the correct distribution of samples. The variance over all the samples of a pixel can be estimated dynamically as proposed by Welford (1962). This is done by remembering the mean M of each color channel M_r M_g M_b which is updated by:

$$M_{new} = M_{old} + \frac{(S_p - M_{old})}{spp} \tag{9}$$

And remembering the squired distances from the mean M^2 of each color channel, which is updated by:

$$M^2_{new} = M^2_{old} + (S_p - M_{old}) * (S_p - M_{new}) \tag{10}$$

There are now six stored color values for a pixel that can give the total pixel variance by averaging the variance of all color channels:

$$Var_p = \frac{M_r^2 + M_g^2 + M_b^2}{(spp - 1) * 3} \tag{11}$$

A problem with the variance is that it will be larger in regions that receive a lot of light. So just like the method of Li et al. (2012) we scale the variance by the luminance, because human eyes are more sensitive to see errors in dark regions. We also take the squire root of the variance, because we are more interested in the standard deviation instead of the total variance. The standard deviation makes our method less sensitive for outliers.

$$Var_p = \frac{\sqrt{Var_p} + \epsilon}{L(p) + \epsilon} \tag{12}$$

With $L(p)$ the luminance of pixel p and ϵ a small value to prevent division by zero. The ϵ above the fraction makes sure when a pixel does not get any variance and luminance, what is possible with a low amount of samples, still can get samples.

The variance is calculated for all the pixels. This happens until they reach a minimum amount of samples min_{spp}. After they reach min_{spp} a threshold τ is set to one and will linearly decreased to zero. The step size to decrease the threshold is based on a maximum amount of samples max_{spp}. Until the threshold is zero the pixels with $Var_p > \tau$ will gain new samples and update their variance.

At the end when $\tau == 0$ and thus max_{spp} is reached, the amount of samples gained for each pixel will be divided by the max_{spp}. The resulting value is used as a probability to gain a new sample until the path tracing algorithm restarts by a movement or change in the scene. The complete pseudo code can be found in algorithm 1.

We chose for a probability based adaptive sampling method, because it still guarantees that the path tracer converge to a correct value as long $min_{spp} > 0$. If we use a static threshold then only some pixels get new samples, which also update their variance, but after a while the variance will stabilize to the true

variance of the incoming energy on a pixel. Now the pixels who stay above the threshold with their variance, are the only pixels that get new samples.

$itr \leftarrow 0$; // iteration count
$\tau \leftarrow 1$; // adaptive threshold
$min_{spp} \leftarrow 4$;
$max_{spp} \leftarrow 32$;
while $itr \neq END$ **do**
  foreach *Pixel* P *in Pixels* **do**
    // calculate probability
    if $itr == max_{spp}$ **then**
      $P_{variance} \leftarrow \frac{P_{spp}}{max_{spp}}$;
    end
    $ready \leftarrow false$;
    if $itr \geq max_{spp}$ **then** $ready \leftarrow$ `RandomFloat()` $\leq P_{variance}$;
    else $ready \leftarrow P_{variance} \geq \tau \;||\; Pspp < min_{spp}$;
    // distribute samples
    if $ready$ **then**
      $P_{sample} \leftarrow$ `TracePixel();`
      P_{spp}++;
      if $itr \geq max_{spp}$ **then** $continue$;
      // Update variance
      $delta \leftarrow P_{sample} - P_{mean}$;
      P_{mean} += $\frac{delta}{P_{spp}}$;
      $delta2 \leftarrow P_{sample} - P_{mean}$;
      P_{M^2} += $delta * delta2$;
      $P_{variance} \leftarrow \frac{\sqrt{\frac{P_{M^2}}{P_{spp-1}}}+\epsilon}{P_{luminance}+\epsilon}$
    end
  end
  // Update adaptive threshold
  if $itr \geq min_{spp}$ **then**
    τ -= $\frac{1}{max_{spp}-min_{spp}}$;
  end
  itr++;
end

Algorithm 1: Pseudo code for the adaptive sampling algorithm, which will restart when a movement occurs or when there is a change in the scene.

3.2 Reprojection Method

The reprojection method uses forward reprojection where each pixel is copied to a new assigned pixel depending on the camera matrix. This is adapted from Corso et al. (2017), where they also use forward reprojection instead of reverse reprojection as in Nehab et al. (2007), because it is better suited for ray tracing.

Before reprojecting pixels after a movement occurs, three different validations are done to make sure the reprojection is valid:

- Validate if the reprojection is still within the screen limits.
- Validate if the intersection point is still visible and not occluded from the new camera position.
- Validate if the introduced error will not exceed the threshold.

To validate if a pixel is within the screen limits, each pixel should store their first intersection point when the pixel is not yet reprojected. Now the validation is done by taking a direction from the new camera

position towards the stored intersection point and map the direction to a screen pixel. Whenever the pixel is still within the screen limits the reprojection may proceed.

Validate if there is an occlusion is a bit harder. Therefore a ray needs to be cast from the new camera position towards the stored intersection point to find out if the distance between the new traced intersection point and the stored intersection point is smaller than a chosen value ϵ. If the distance is larger, then there is an occlusion, which will result in not reprojecting the pixel. These two validating steps are similar to other reprojection methods like the verification stage of Adelson and Hodges (1993).

Most materials are view dependent, for example refractive or slightly glossy materials, which introduce highlights based on the viewing position. This could make the reprojected data not valid from the new camera angle. Therefore a calculation is done to find out what the view dependency error will be for a certain reprojection. The third validation looks if the view dependency error stays bellow a threshold. How the calculation of the viewpoint dependency error works, is explained in chapter 3.2.1. In Günther and Grosch (2015) they also calculated an error for reprojection, but here the error is based on the final color of the old and new frame instead on material properties.

3.2.1 Viewpoint dependency error calculation

A Lambertian reflectance that defines a perfect diffuse surface gives always the same color result regardless of the viewing position. This would be an ideal situation for reprojections, because changing the viewpoint will not introduce highlights on the material. When the surface is not pure diffuse, it will be viewpoint dependent, which will introduce an error in the reprojected pixels.

In this thesis a microfacet model is used of Walter et al. (2007), using the Beckmann distribution to create rough surfaces. The reflection term of the model is as follows:

$$f_r(i, o, n) = \frac{F(i, h_r)G(i, o, h_r)D(h_r, \alpha)}{4|i \cdot n||o \cdot n|} \tag{13}$$

With $F(i, h_r)$ as the Fresnel term, $G(i, o, h_r)$ as the shadow-masking term, $D(h_r)$ as the Beckmann distribution and h_r as the halfway vector. The h_r can be calculated by taking the normalized vector of $sign(i \cdot n)(i + o)$. Here i, o and n are respectively incoming, outgoing and surface normal directions. An advantage of the model is that it only use one variable α that can be set by the user. The variable α indicates the roughness of the material which has a value between 0 and 1, where 0 is pure specular and 1 is a pure diffuse surface.

A material is considered specular when $\alpha < 0.01$ or if the material is refractive. Materials which are specular are not reprojected, because a small change in perspective for such materials has a high probability to yield a total different outcome. It is possible to verify if a specular point can be reprojected like in Xie et al. (2017), but it introduces more complexity especially when there are multiple specular bounces.

To gain a viewpoint dependency error based on the material we need to store some more data based on the first intersection. So next to the intersection point the following data is needed:

- Outgoing chosen direction L_{t-1} from the intersection point.
- Roughness value α of the material.
- Normal of the surface N on the intersection point.
- The reflection term value fr_{t-1} for the chosen outgoing direction L_{t-1}.
- The diffuse color of the material A.
- Irradiance Ir_{t-1} coming from L_{t-1}.

Whenever a movement occurs the reflection term 13 will be evaluated again with the new incoming direction V_t and the stored first chosen outgoing direction L_{t-1} (see figure 3). Now the two reflection terms can be used to gain a viewpoint dependency error ϵ:

$$\epsilon = \frac{f_r(V_t, L_{t-1}, N)}{f_r(V_{t-1}, L_{t-1}, N)} \tag{14}$$

Now the viewpoint dependency error will always be $\epsilon \in [0, 1]$ which made it easier to discard some of the previous result based on this error. The disadvantage of a relative error is that the difference of dark areas can be quite high while it is not visible for the human eye, and such spaces would introduces less visible errors when they are reprojected. Another flaw of this viewpoint error calculation, is that it does not account for the incoming light energy and the possible outgoing energy. Now the viewpoint dependency error will make no difference between brighter or darker areas. To make the viewpoint dependency error take into account the light energy, the refraction term is multiplied by the irradiance Ir_{t-1} and the diffuse color A. Also an absolute error is calculated instead of a relative error. Now a threshold can be based on the difference in color value. The new viewpoint dependency error will become:

$$\epsilon = abs(f_r(V_t, L_{t-1}, N) - f_r(V_{t-1}, L_{t-1}, N)) * A * Ir_{t-1} \tag{15}$$

This gives an error of $\epsilon \in [0, \infty]$, which cannot be used to discard some of the previous results, but gives a better understanding of how much the reprojected color will deviate from the new viewpoint.

To make the reprojection correct, each sample that contributes to the final pixel color should be reprojected on its own, because the incoming irradiance L_{t-1} is different for each new sample. Reprojection in this way will increase the used memory linear to the amount of samples taken and the reprojection calculation time will also increase linear based on the amount of accepted reprojections. When there is unlimited memory and time this would be no problem, but for a real-time application it will not work yet. Therefore an approximation is made by averaging all the incoming irradiance of the gathered samples on a pixel. Now each pixel holds a reprojection state with the stored data and will reproject all the gathered samples all at ones, based on the error calculated with an approximated irradiance.

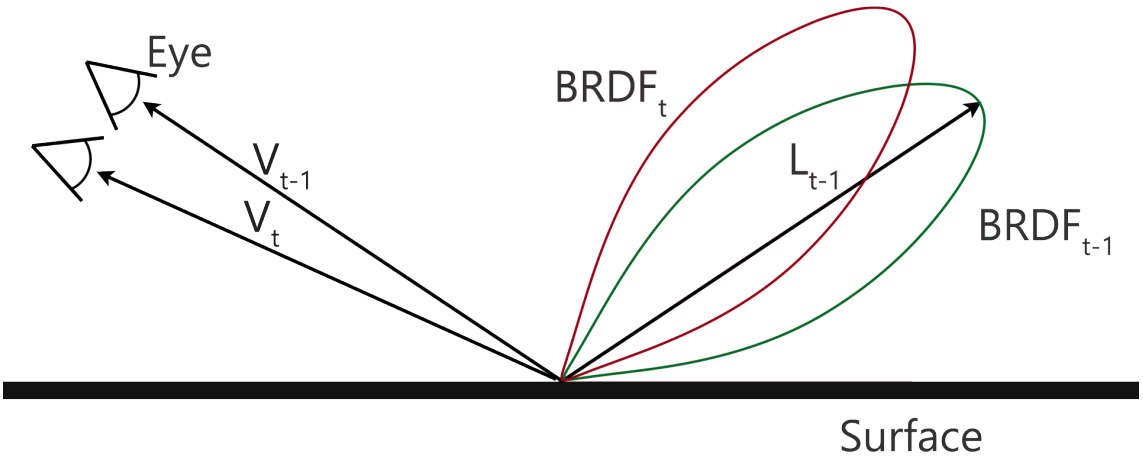


Figure 3: Shows the difference in BRDF based on the viewpoint with the same outgoing direction L_{t-1}.

3.2.2 Aliasing artifacts

Reprojecting pixels that are not only sampled through the center of the pixel may introduce aliasing artifacts. Such artifacts can be seen in figure: 4. The artifacts are created during reprojection, when the stored intersection point is correctly validated and the current pixel in the previous frame had multiple objects within his bounds. When going to a new viewpoint the pixel that has the intersection point within bounds does not necessary need to have the same objects within his bounds as the previous pixel.

If we look at the figure in 5, we see how the problem occurs. The red dot is the first stored intersection point for that pixel in frame $t - 1$. Through the pixel a total of five rays are traced (red and black dots), which determine the color. The color would be more orange than green, because more of the rays hit the orange triangle. After a movement to the right the pixel is still reprojected to frame t. This is caused by the first intersection point, which is still reachable in the new frame. Now the color of the pixel in frame t contains a bit of orange passed by the previous frame, which should be total green.

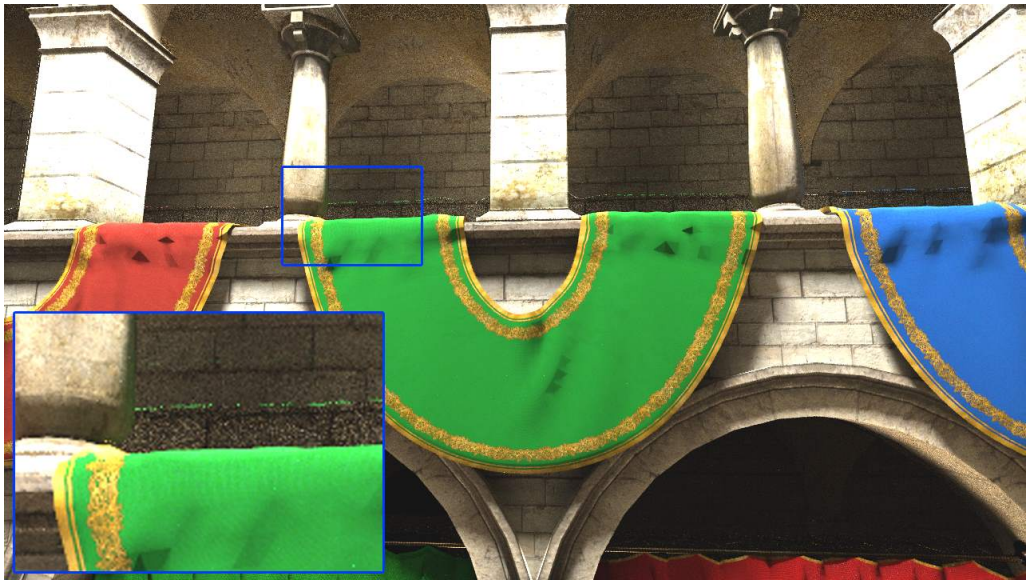


Figure 4: Aliasing artifacts are visible after movement when using reprojection.

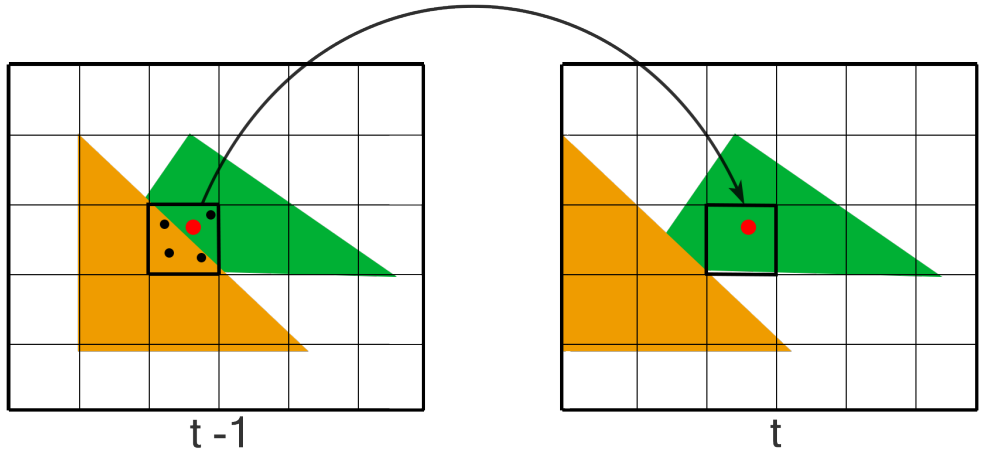


Figure 5: Five samples acquired on a pixels in frame $t-1$ are reprojected to frame t, which introduce an aliasing artifact.

By always trace pixels through their center the problem will be solved, but now aliasing will be visible. As a solution an anti aliasing algorithm can be applied in a post process. Another solution is to accept the aliasing artifacts, which can be reduced by saving more samples per pixel and reprojecting each of them, also the artifacts will fade overtime when more samples are taken.

Another way to reduce the aliasing artifacts is to check if the distance between the stored intersection point and the new found intersection points from rays casted through the same pixel is smaller than a certain maximum distance. Whenever a point exceeds the maximum distance, then the pixel will not be reprojected. This solution only works for far distance aliasing artifacts like in 4, but the artifacts can still occur on places where objects are closer to each other than the maximum accepted distance. The solution may cause a small increase in overhead for reprojection, but it creates a more visual appealing result.

3.3 Combining reprojection and adaptive sampling

When combining the reprojection and adaptive sampling method some advantages come forward. For example the additional buffer in reprojection, to keep track of the amount of samples per pixel, can be reused for adaptive sampling. Another advantage is that pixels who are not reprojected can be focused on by giving them priority with adaptive sampling.

The combination does not deviate much from the explained adaptive sampling 3.1 and reprojection 3.2 method, but when a reprojection occurs we need to make some changes. During the reprojection, the pixels that are reprojected do not only copy the samples per pixel and the color value from their previous pixels, they also copy the calculated variance/probability received from the adaptive sampling method. After the reprojection, pixels with fewer samples than the defined max_{spp} and for which their is not yet a probability calculated, are the only pixels who get new samples. This continues until all pixels reach the max_{spp}. Now each pixel has its own calculated probability which is determined by the adaptive sampling method, and based on this probability new samples are distributed until a new reprojection happens. The pseudo code for the combination method is visible in algorithm 2.

By focusing on pixels with less samples than the defined max_{spp} and without a calculated probability after a reprojection occurs, the difference between reprojected and not reprojected pixels decreases faster. This can result in a more appealing image.

```
itr ← 0 ;                                              // iteration count
τ ← 1 ;                                              // adaptive threshold
ϵ ← 1/256 ;                                  // maximum reprojection error
min_spp ← 4 ;
max_spp ← 32 ;
// restart loop on movement itr ← 0
while itr ≠ END do
    foreach Pixel P in Pixels do
        // calculate probability
        if itr == max_spp then
            P_variance ← P_spp / max_spp;
            P_probabilityCalculated ← True ;
        end
        // apply reprojection if verification is accepted
        if itr == 0 then
            if ReprojectError(P) ≤ ϵ then Reproject(P) ;
        end
        // distribute samples
        ready ← false ;
        if itr ≥ max_spp then
            ready ← RandomFloat() ≤ P_variance;
        else if !P_probabilityCalculated then
            ready ← P_variance ≥ τ || Pspp < min_spp;
        end
        if ready then
            P_sample ← TracePixel();
            P_spp++;
            UpdateVariance(P);
        end
    end
    UpdateAdaptiveThreshold();
    itr++;
end
```

Algorithm 2: Pseudo code for the combination algorithm, which will restart when a movement occurs or when there is a change in the scene.

In the combination of adaptive sampling and reprojection a different distribution for the probability of sampling pixels can happen, when it is compared with the adaptive sampling method. This occurs when the variance of pixels is reprojected while the pixels do not yet have a calculated probability. So after the reprojection these pixels get new samples and update their variance, based on the adaptive threshold which has just been reset. Now it is possible that a pixel can get more samples than the max_{spp}. This

results in incorrect probabilities. We expect that this is no problem, because if the same pixel is reprojected multiple times, the chance increase that it will exceed the maximum error of the reprojection method, what results in a rejection for another reprojection. Also we expect that such cases rarely occur.

4 Results

In the following chapters the results of all the different methods can be found. The different scenes we use for the tests are elaborated in chapter 4.1. The adaptive sampling results are in chapter 4.2, where the spreading of the mean squired error (MSE) is compared against a default path tracer, with different settings for the min_{spp} and max_{spp}. Chapter 4.3 shows the reprojection results against a default path tracer, from which we can deduct if it yields a better efficiency. Also a measurement on the overheat of the reprojection method is elaborated. In the last chapter 4.4 the combination of reprojection and adaptive sampling is tested against the default path tracer, by using the best settings from 4.2.

The default path tracer is implemented on a GPU and it uses next event estimation and importance sampling to accelerate the outcome.

All the results are acquired on a NVidia GeForce GTX 1070 graphics card. It process 6,463 GFLOPS (billion floating point operations per second) and it has a memory bandwidth of 256.3GB/s. All the tests are done with a screen size of 1280x720.

4.1 Scenes

To test the methods we use three different scenes, which are well known for graphics research. Each scene differs in amount of triangles lighting and material complexity. The scenes used in this thesis are:

- Cornell box: Our Cornell box consists of 34 triangles with one surface area light. Each Cornell box can contain a different model inside. We choose to use two glossy boxes. The walls of the Cornell box are also slightly glossy (see figure: 6).
- Sponza: This scene consists of 262267 triangles (see figure: 7). It contains different textures with bump maps and there is an area-light in the center above the hall. In the reference image we see that the floor is a little reflective.
- San Miguel: This scene contains 9980693 triangles (see figure: 8) and is a very detailed terrace scene, which only contains directional light and textures with bump maps and specular maps.

For each scene a reference image is taken to compare with the methods. Each reference image takes 4096 samples per pixel on our default path tracer.

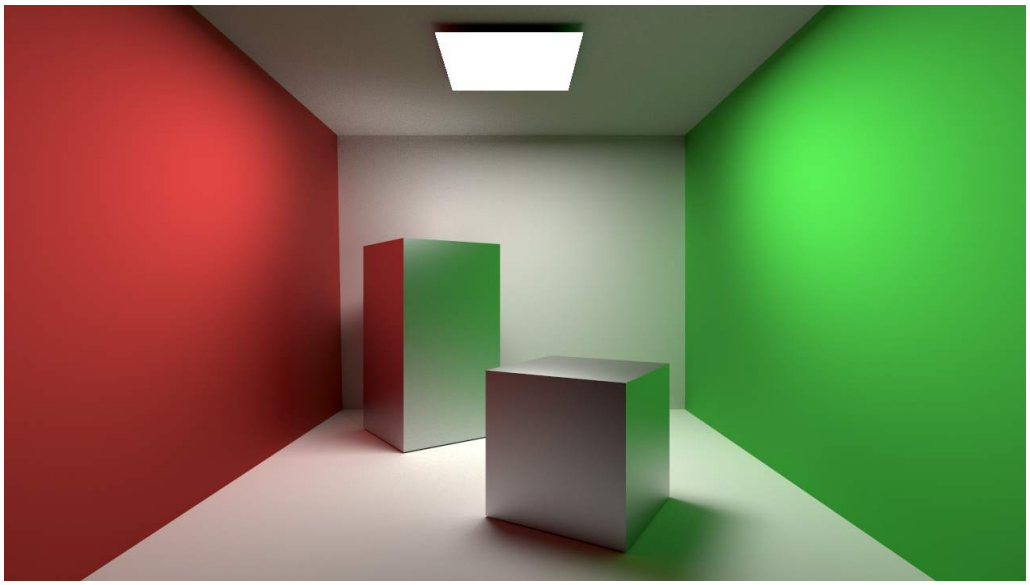


Figure 6: Reference image of the Cornell box scene.

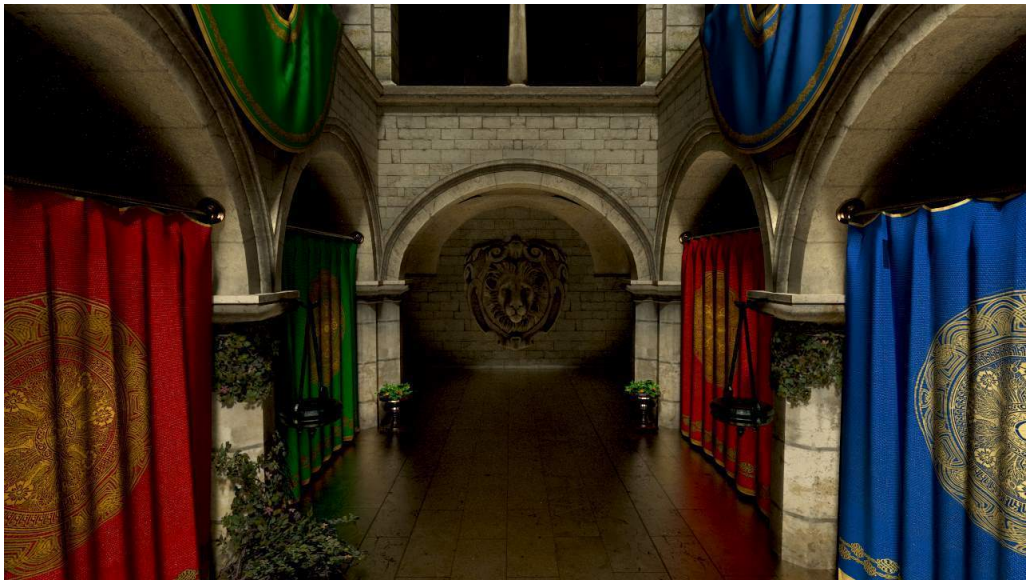


Figure 7: Reference image of the Sponza scene.

Figure 8: Reference image of the San Miguel scene.

4.2 Adaptive sampling results

We test adaptive sampling against the default path tracer by looking at the decrease in variance of the distance to the reference pixels. The distance to the reference pixel is calculated in euclidean distance.

Min	Max	Min probability
2	16	0.125
4	16	0.25
8	16	0.5
2	32	0.063
4	32	0.125
8	32	0.25
16	32	0.5

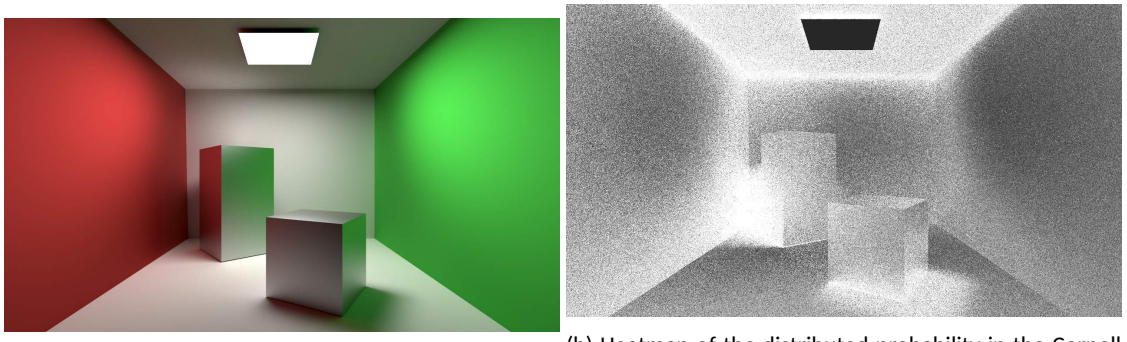
Table 1: Minimum and maximum spp for adaptive sampling

In table 1 the min_{spp} and max_{spp} used for adaptive sampling can be seen, with their minimum probability for a pixel to get a new sample.

4.2.1 Cornell box

Figure 9 shows the reference image with 4096 samples per pixel created by the default path tracer, and the heatmap created by adaptive sampling. The brightness of the pixels on the heatmap indicates the probability to get a new sample. A higher brightness means a higher probability.

On the heatmap we see that the adaptive sampling method targets the darker areas. Next to the darker areas, the method also seems to target caustics in the left top corner, created by the glossy box.



(a) Reference image of the Cornell box, which is compared with the adaptive sampling method.

(b) Heatmap of the distributed probability in the Cornell box. Created by adaptive sampling after 233 ms with 32 spp.

Figure 9

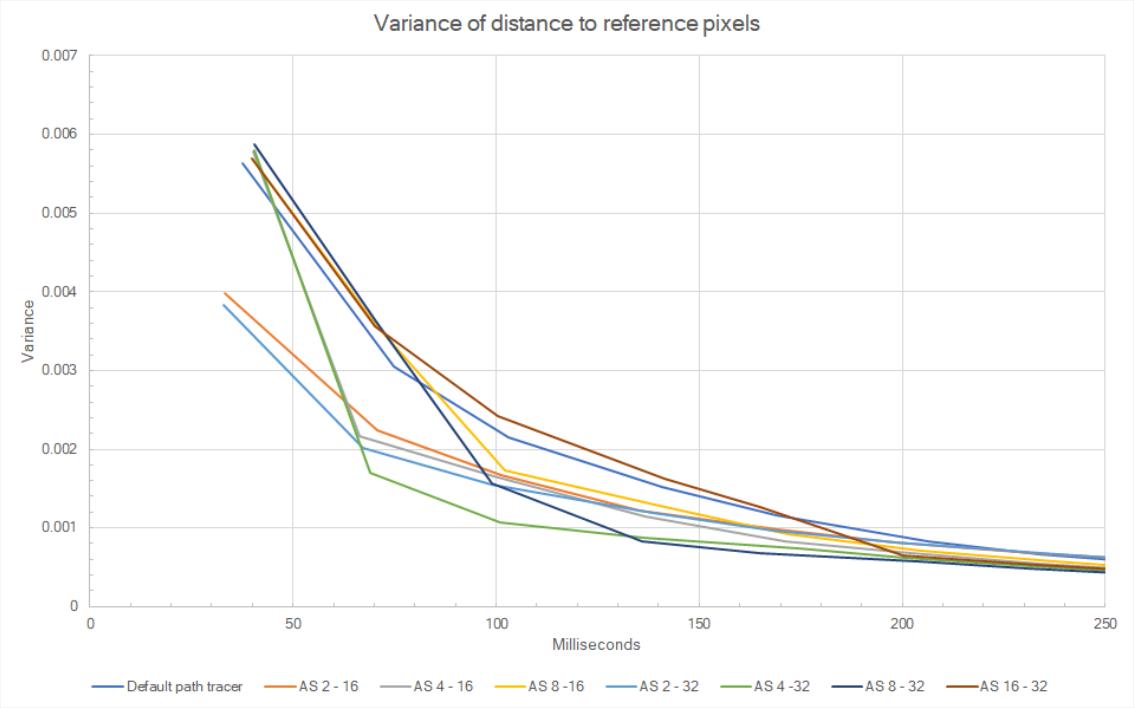


Figure 10

The decrease in variance of the distance to the reference pixels with different settings for the min_{spp} and max_{spp} can be seen in figure 10. Adaptive sampling with $min_{spp} = 4$ and $max_{spp} = 32$, seems to have the most stable outcome in comparison with the default path tracer. This means that the overall distance to the reference image around 70 milliseconds has a better spreading than the default path tracer.

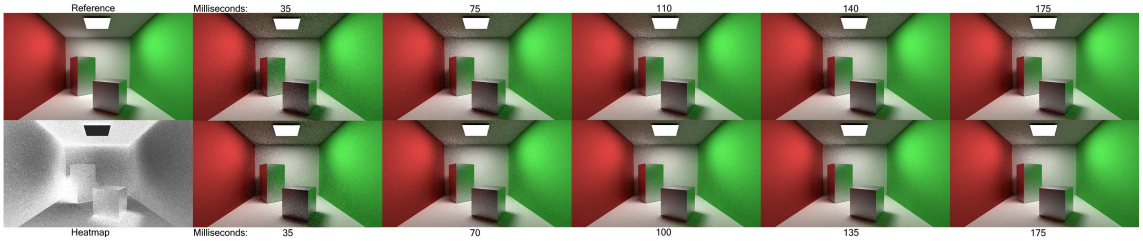
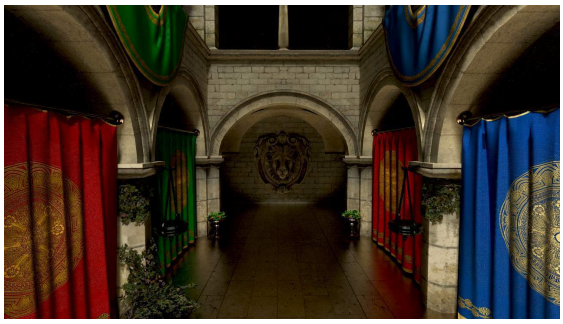


Figure 11: The default path tracer results over time(upper row) compared against the adaptive sampling method results (bottom row).

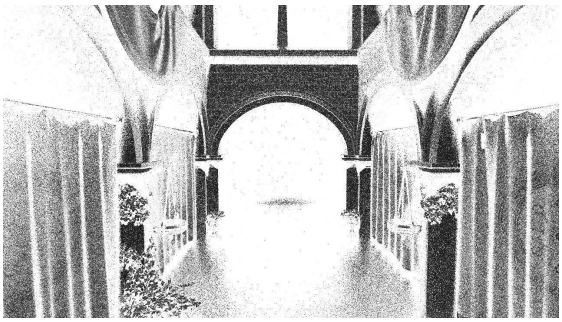
In figure 11, the difference over time between adaptive sampling and the default path tracer is visualized with the reference and heatmap image. The bottom row shows the results of adaptive sampling when we use $min_{spp} = 4$ and $max_{spp} = 32$. The top row shows the results of the default path tracer. When we compare the images there is a small difference visible on spots that are brighter in the heatmap.

4.2.2 Sponza

An increase of the Sponza scene complexity and triangles, naturally creates a longer render time before it generates the heatmap in figure 12, when we compare it with the Cornell box scene. The heatmap shows that areas which receive energy directly from a light source, like the wall above the arc, gain a lower probability to be sampled. This is exactly what the adaptive sampling method should do, because such areas do not need many samples to converge to the correct value. Furthermore the method mainly focuses on the dark areas in this scene.



(a) Reference image of the Sponza scene, which is compared with the adaptive sampling method.



(b) Heatmap of the distributed probability in the Sponza scene. Created by adaptive sampling after 660 ms with 32 spp.

Figure 12

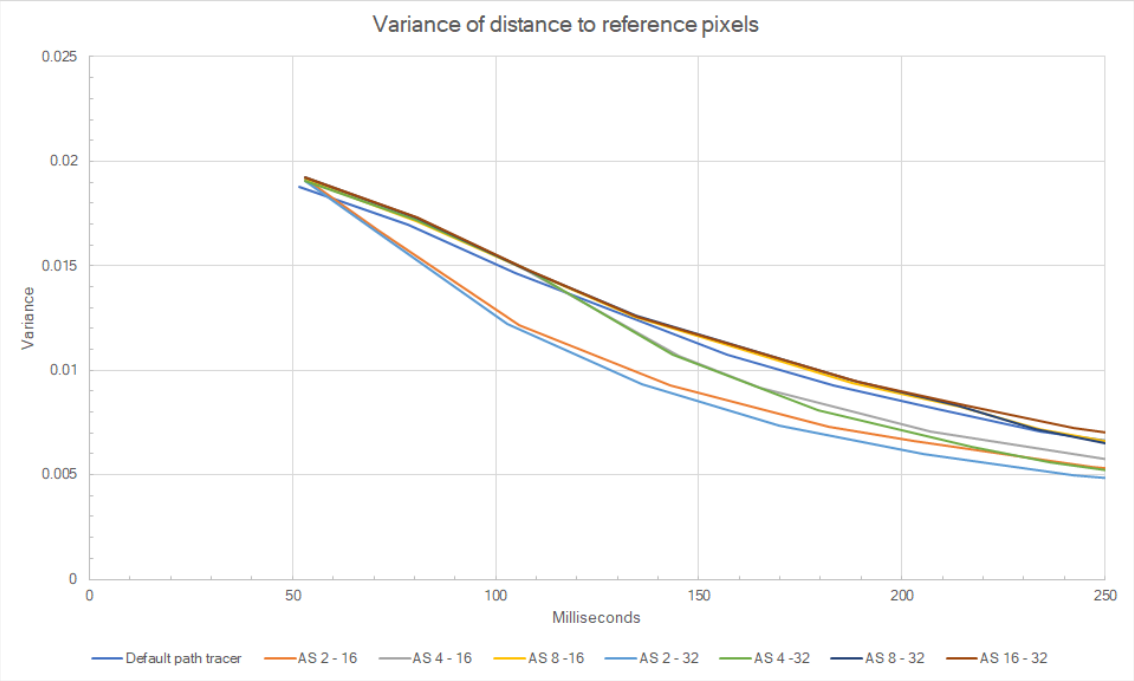


Figure 13

When we look at the graph in figure 13, the best setting for the adaptive sampling method seems to be $min_{spp} = 2$ and $max_{spp} = 32$, which is different than for the Cornell box scene. This is possible due to the amount of samples taken over time and the different probability distributions on the scenes.

4.2.3 San Miguel

In the San Miguel scene we only use a directional light, which is visible in the generated heatmap by the adaptive sampling method (figure 14). All the surfaces that are directly hit by the directional light gain a lower sample probability than surfaces in the shade. The surfaces in the shade are quite dark, which gives them a high sample probability from the adaptive sampling method.

(a) Reference image of the San Miguel scene, which is compared with the adaptive sampling method.

(b) Heatmap of the distributed probability in the San Miguel scene. Created by adaptive sampling after 1150 ms with 32 spp.

Figure 14

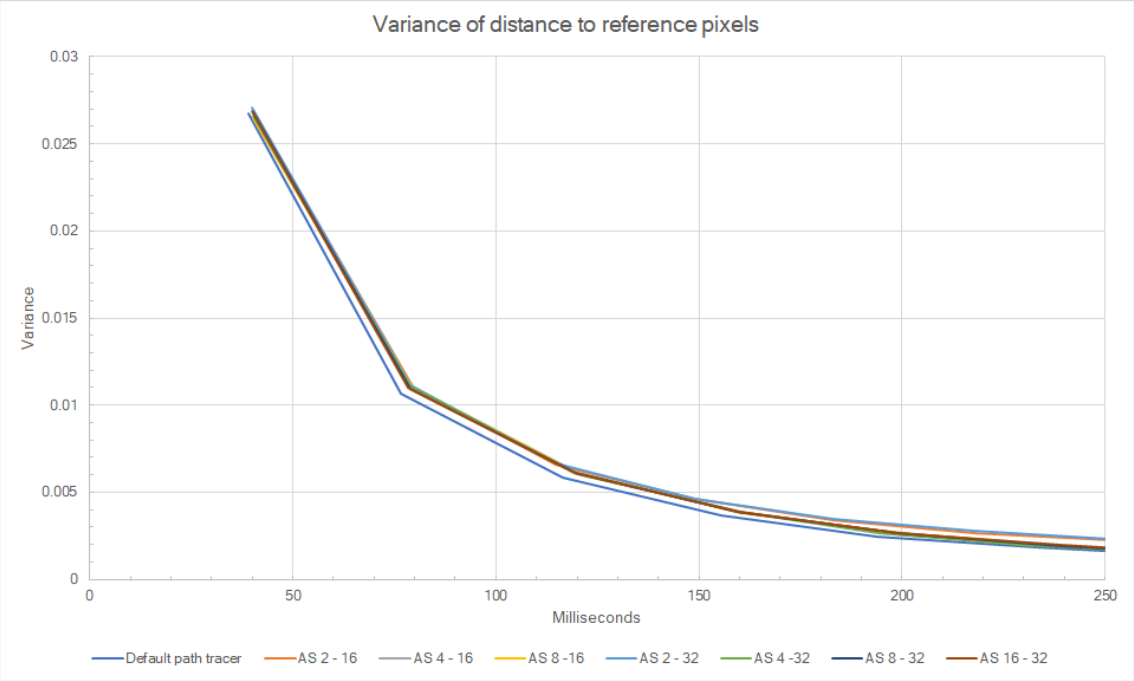


Figure 15

Adaptive sampling does not improves the reduction in variance of distance to the reference pixels with any min_{spp} and max_{spp} combination, compared with the default path tracer (see figure 15). Multiple reasons can be the cause of this, like the increase of triangles, different lighting and the complexity of scene materials. In the next chapter 4.2.4 these different reasons are tested in the Cornell box scene.

4.2.4 Scene property tests

A few different properties can influence the effectivity of the adaptive sampling method. By changing the scene properties and measure the reduce in variance of distance to the reference pixels, we can measure which property influences the method the most. The properties are:

- Scene lighting: Surface light, point light and directional light.
- Amount of triangles: 34, 69461 and 1087726 triangles.
- Material complexity: Soft glossy, high reflective and glass materials.

The default scene contains 34 triangles with soft glossy materials and a surface light, which is used in section 4.2.1. We take for the $min_{spp} \in (2, 4)$ and $max_{spp} = 32$, because these two combinations show the best results obtained in the above chapters.

Results of the default, directional light and high reflective materials can be seen respectively in figures: 16, 17, 18. The other results are in the appendix: A.1.1.

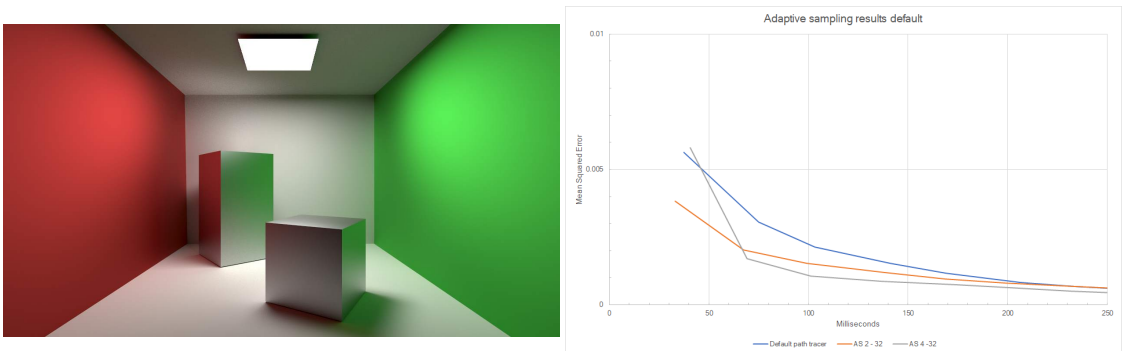


Figure 16: Reference image of the default Cornell box, with the result graph.

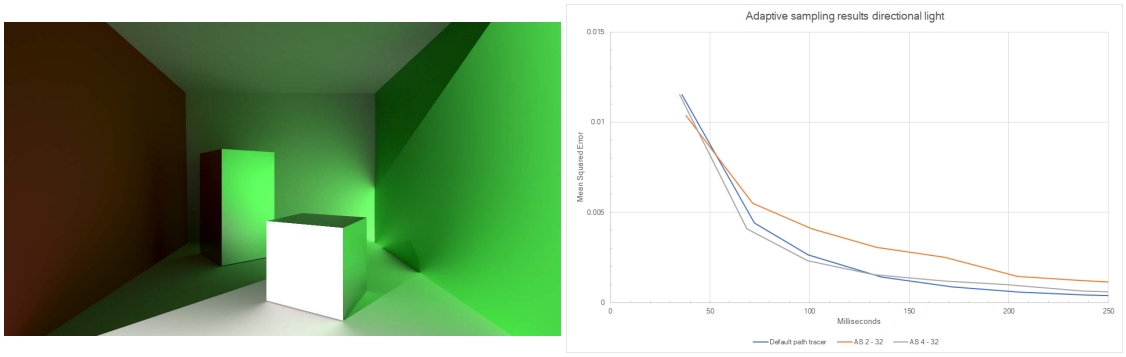


Figure 17: Reference image of the Cornell box with directional light, with the result graph.

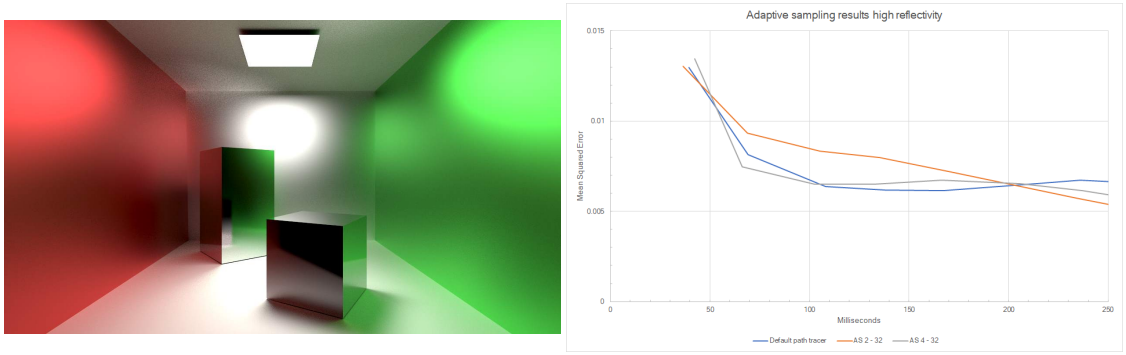


Figure 18: Reference image of the Cornell box with low roughness materials, with the result graph.

From the graphs it is clear that the adaptive sampling method is not effective for scenes with directional light or for scenes that contain a lot of reflective materials. The measurements show that the $min_{spp} = 4$ setting gives a better result than the $min_{spp} = 2$ setting, especially in the scenes where adaptive sampling is not very effective.

4.3 Reprojection results

For testing the reprojection method we first turn the camera around, for one hundred degrees, in differed step sizes, before we measure the mean squired error. For each step a certain amount of samples are taken, 32 and 64 respectively. The step sizes are: 5°, 10°, 20° and 50° with different maximum reprojection errors of $\frac{1}{256}$, $\frac{4}{256}$ and $\frac{8}{256}$. Next to all these possibilities a reprojection can also reproject multiple samples per pixel, so therefore a test is done by reprojecting four samples per pixel instead of one.

The amount of overhead for reprojection is measured for reprojecting one and four samples per pixel. We do this by measuring the render time per frame, while there is a continuous rotation. The average frame time over 250 measurements, while using reprojection, is compared against the average frame time of the default path tracer, which gives the overhead. Extra required memory for the reprojection method for a screen resolution of 1280x720 is 133 MB, which increases linear on the amount of pixels and reprojected samples per pixel.

4.3.1 Cornell box

Figure 19 shows the reference image of the default Cornell box after turning one hundred degrees. Next to the reference image a visualization of the reprojection is shown for the last step of five degrees. In this visualization the pixels who are reprojected, are shown in white. The pixels who are discarded are shown in black. It is clearly visible that the edges of the boxes are not reprojected, this is because of the depth test, which removes aliasing artifacts (see chapter 3.2.2).

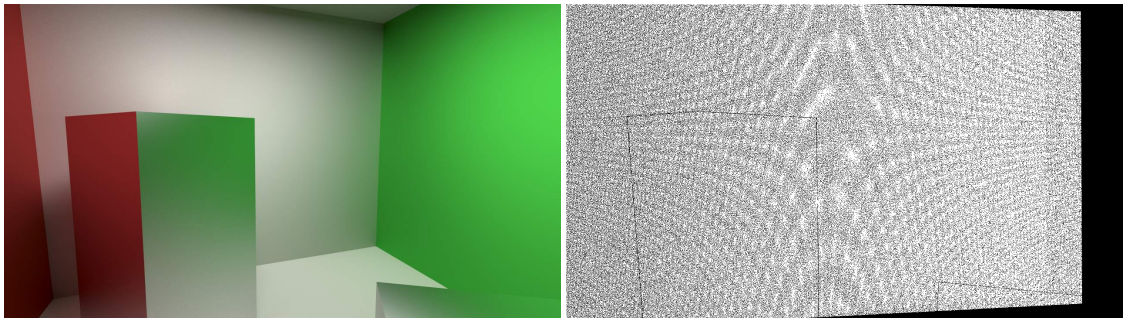


Figure 19: Reference image of the default Cornell box after rotating one hundred degrees, with a reprojection visualization of the last five degrees turn.

All the different possibilities of samples per step and amount of samples reprojected for steps of 5° with a maximum error of $\frac{1}{256}$, give a lower mean squired error than the default path tracer, which can be viewed in figure 20. Around 1500 milliseconds the mean squared error becomes equal for all the combinations, but the most important part are the first milliseconds, because the focus of the thesis is on real-time applications. In the first 100 milliseconds we see the largest difference between the default path tracer and the reprojection method reprojecting four samples per pixel and taking 64 samples per step (see figure 21). The difference between the amount of samples per step taken before rotating is quite like expected, because the mean squared error becomes lower when more samples are reprojected from the last frame.

As for the difference in rotation degrees, it is obvious that the smallest rotation gives the best results. A smaller rotation gives fewer pixels from which we do not have any samples yet. But still the maximum turning degree of 50° improves the quality compared with the default path tracer. All the results of the Cornell box can be viewed in the appendix B.1.

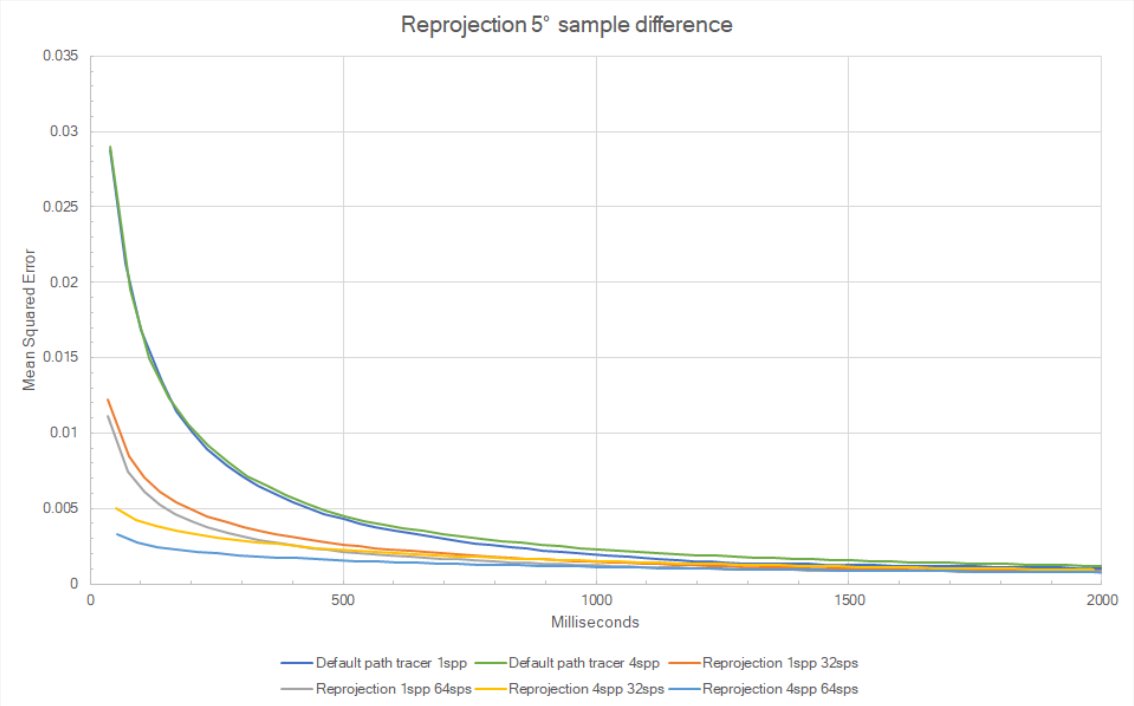


Figure 20: Graph of the difference in mean squared error for different amount of samples per step and amount of samples reprojected. Here we use a step size of 5° and a maximum error of $\frac{1}{256}$.

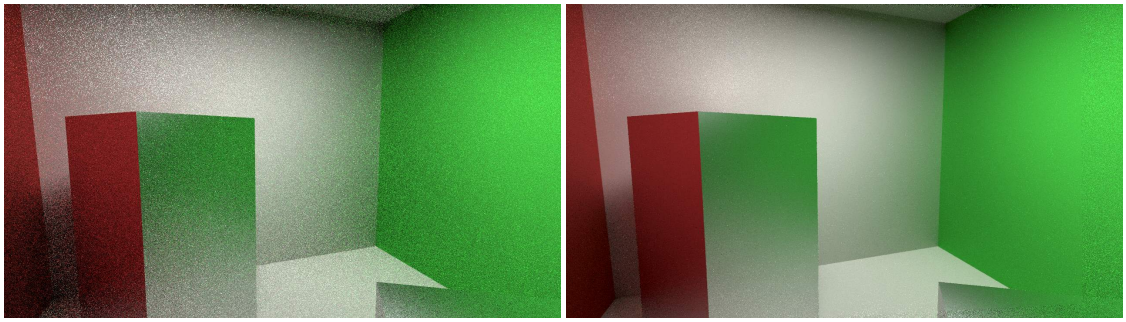


Figure 21: Difference between the default path tracer (left 39ms) and the reprojection method of 64 samples per step while reprojecting four samples per pixel (right 52ms). Both methods take four samples after the final rotation.

In figure 22 the different maximum errors are compared against each other for reprojection with a step size of 5° and 32 samples per step. Only two lines are visible on the graph, one for the default path tracer and one for the maximum error of $\frac{8}{256}$. This means that the maximum error does not affect the reprojection method using these three different error thresholds. Using different step sizes, samples per step and scenes yield the same result for all the maximum errors, which can be seen in the appendix B.

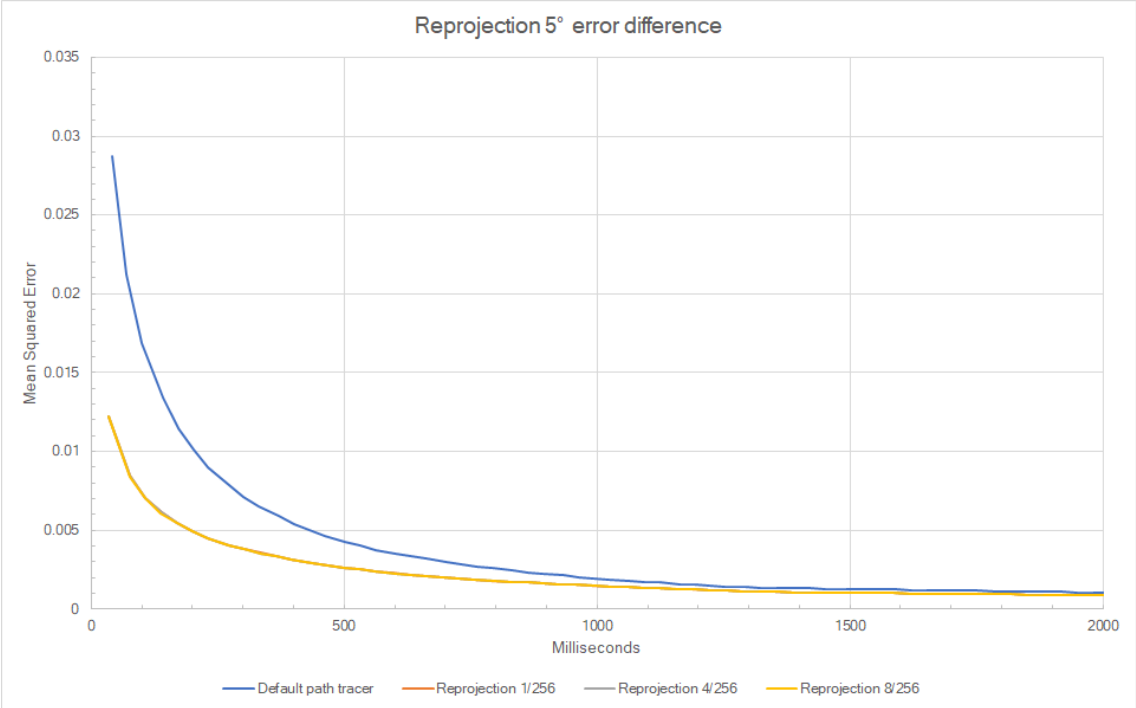


Figure 22: Graph of the mean squired error difference for different maximum errors, using a step size of 5° and 32 samples per step.

The overhead introduced by reprojection in the Cornell box is 26% when we take one sample per pixel, and 25% if we take four samples per pixel. It seems to be quite a lot, but the average measured render time per frame for one sample per pixel is $12,53$ milliseconds instead of $9,30$ milliseconds for the default path tracer. Both frame times are acceptable for a real-time application.

4.3.2 Sponza

The Sponza scene reference image we use for the reprojection tests, with the corresponding visualization of the reprojected pixels are visible in figure 23. Just like in the Cornell box scene, we can see that edges are not reprojected. It is extremely visible from the plant model in the Sponza scene. Most of the pixels are not reprojected for this model, because it is small and complex.

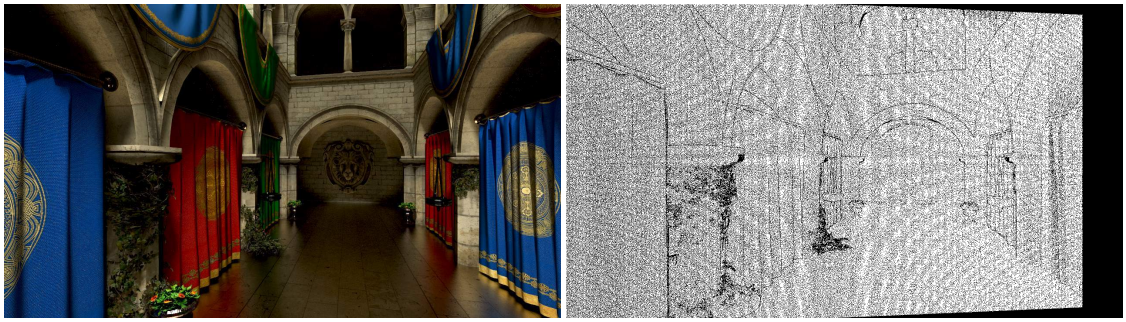


Figure 23: Reference image of the Sponza scene after rotating one hundred degrees, with a reprojection visualization of the last five degrees turn.

The results of different amounts of samples per step, while reprojecting one or four samples, is shown in figure 24. The level of quality is in the same order as for the Cornell box, but the distance between the reprojection method and the default path tracer keeps continuing after 1500 milliseconds. The maximum difference between the default path tracer and the reprojection method is visualized in figure 25, for all the other results of Sponza see appendix B.2.

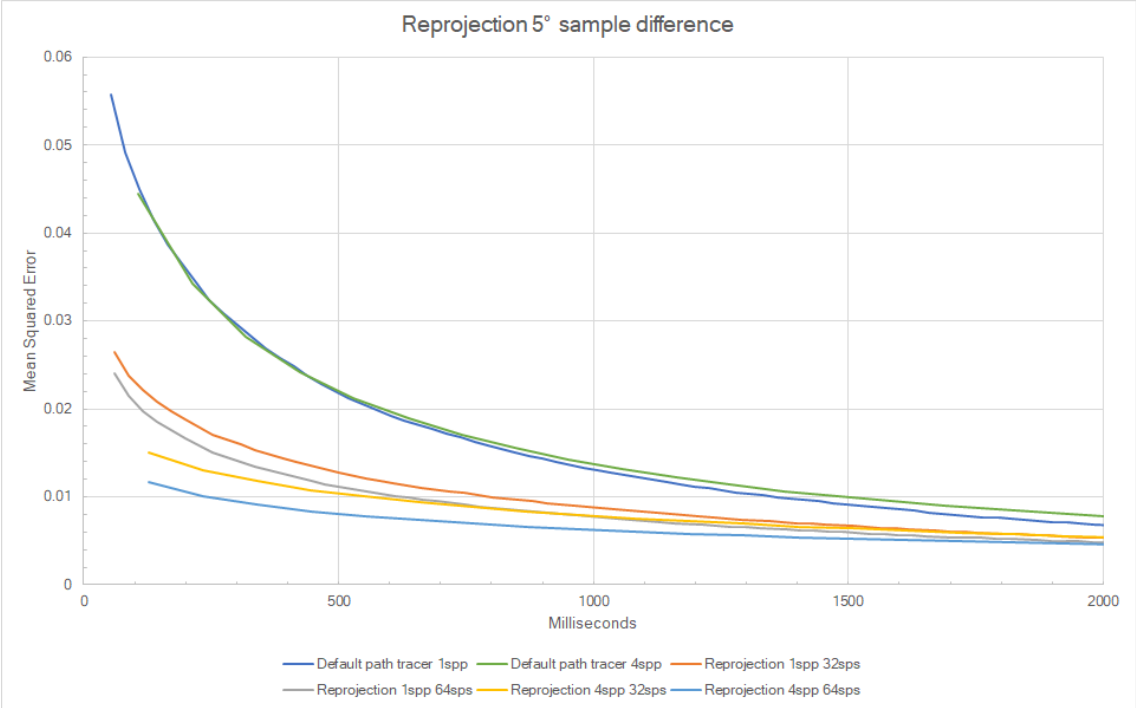


Figure 24: Graph of the difference in mean squared error for different amount of samples per step and amount of samples reprojected, using a step size of 5° and a maximum error of $\frac{1}{256}$.

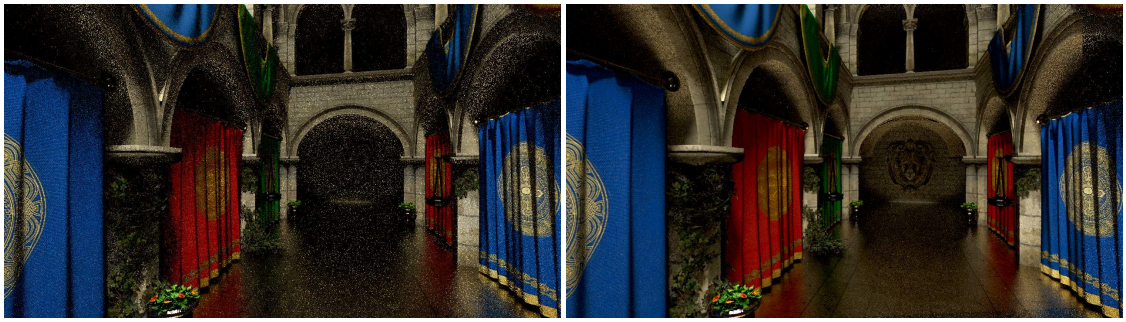


Figure 25: Difference between the default path tracer (left 106ms) and the reprojection method of 64 samples per step while reprojecting four samples per pixel (right 125ms). Both methods take four samples after the final rotation.

The average time for a frame with one sample per pixel without reprojection is $25,99$ milliseconds, which is almost three times slower than for the Cornell box scene. One reason for the increase in frame time is the amount of triangles that are in the scene. The material complexity with textures also causes an increase in frame time. The time for one frame with reprojection is $31,64$ milliseconds, which gives an overhead of 18%. When reprojecting four samples per pixel the overhead becomes 17%. This is a 8% reduce in overhead compared with the Cornell box, what can be a cause of the scene complexity.

4.3.3 San Miguel

The San Miguel scene contains more refractive materials and it has a directional light. The refractive materials can be seen in black in the visualization image of the reprojected pixels in figure 26. Such materials, the fountain water and glasses on the table, are not reprojected because of their material complexity and high view dependency.

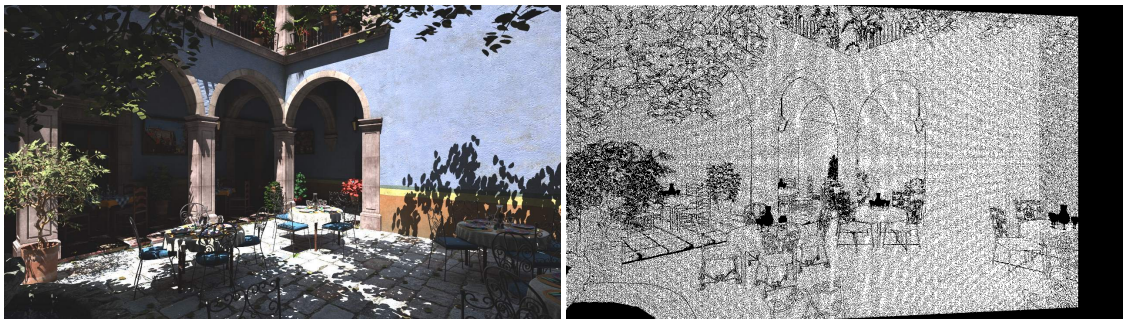


Figure 26: Reference image of the San Miguel scene after rotating one hundred degrees, with a reprojection visualization of the last five degrees turn.

From the results in figure 27, we can see that reprojection still improves the image quality over the default path tracer. But the difference between 32 or 64 samples per step does not seem to matter in this scene. Also the distance between the reprojection method and the default path tracer decreases faster compared to the other scenes. Next to these differences, we see that a rotation of step size 50° does not have any effect on the San Miguel scene (appendix B.3). We try to find the reason for the different result with some scene property tests in chapter 4.3.4.

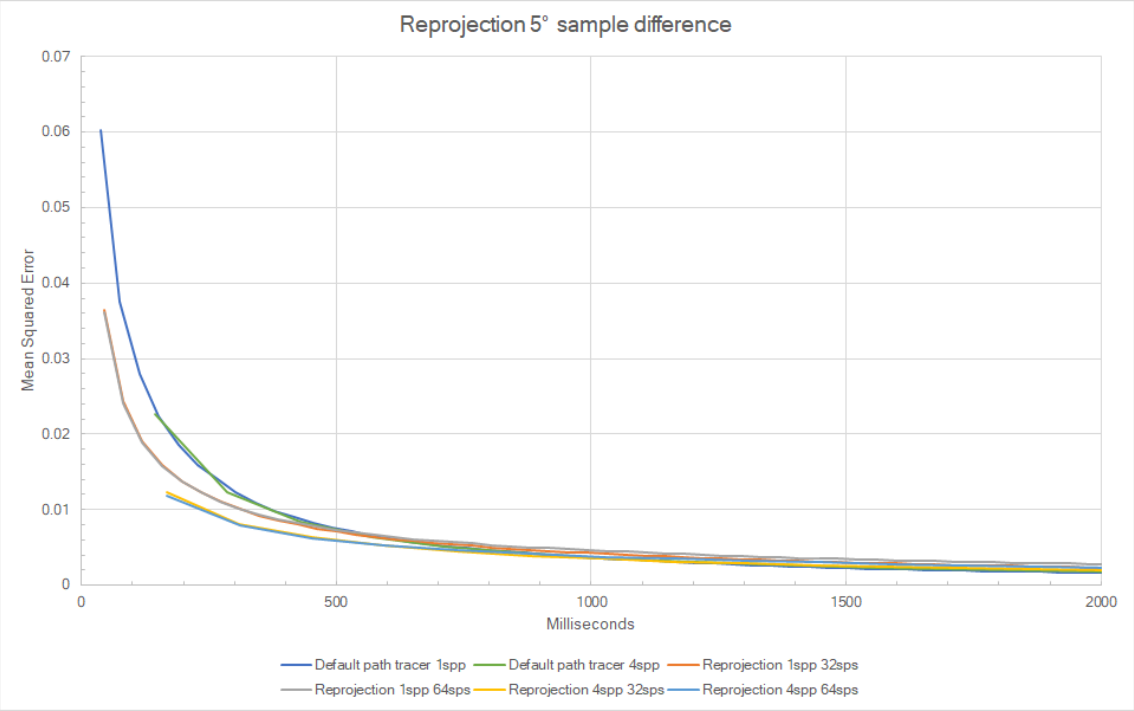


Figure 27: Graph of the difference in mean squared error for different amount of samples per step and amount of samples reprojected, using a step size of 5° and a maximum error of $\frac{1}{256}$.

Figure 28: Difference between the default path tracer (left 145ms) and the reprojection method of 64 samples per step while reprojecting four samples per pixel (right 170ms). Both methods take four samples after the final rotation.

The frame time for one frame with one sample is increased to 36.35 milliseconds without reprojection, because the scene complexity and the amount triangles is increased compared to the Sponza scene. With reprojection it takes 43.01 milliseconds, which gives an overhead of 15%. For reprojecting four samples at once it gives an overhead of 18%. The measured overhead for the San Miguel scene is comparable to the overhead of the Sponza scene.

4.3.4 Scene property tests

Which scene property influences the quality of our reprojection method is tested with the same scenes as in chapter 4.2.4. The following figures 29, 30, 31 show the results of reprojection in the different scenes of the Cornell box, making use of reprojection with 32 samples per step and a maximal error of $\frac{1}{256}$. For all the results see the appendix B.1.5.

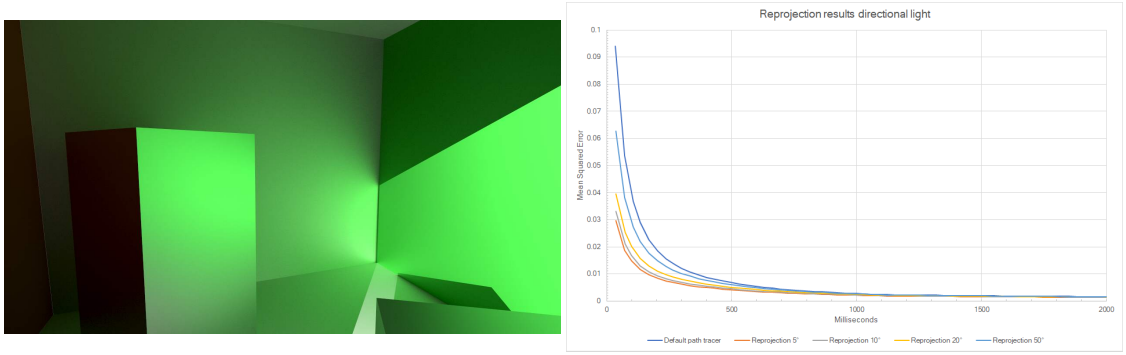


Figure 29: Reference image of the Cornell box with directional light, with the result graph.

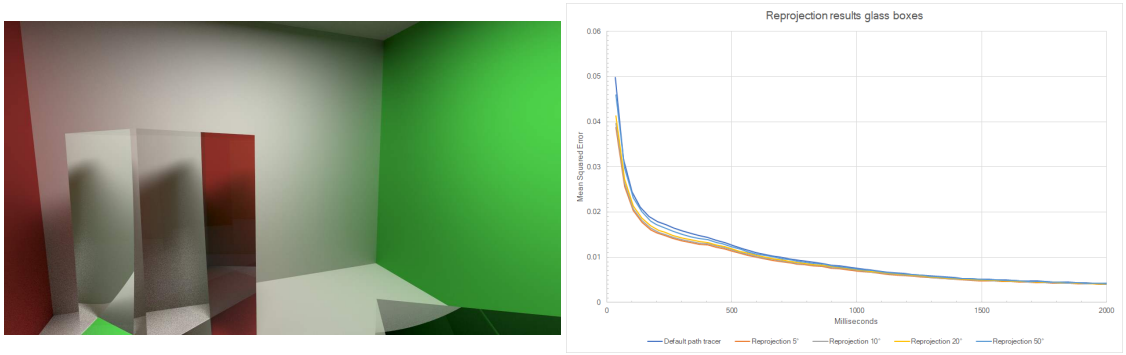


Figure 30: Reference image of the Cornell box with refractive materials, with the result graph.

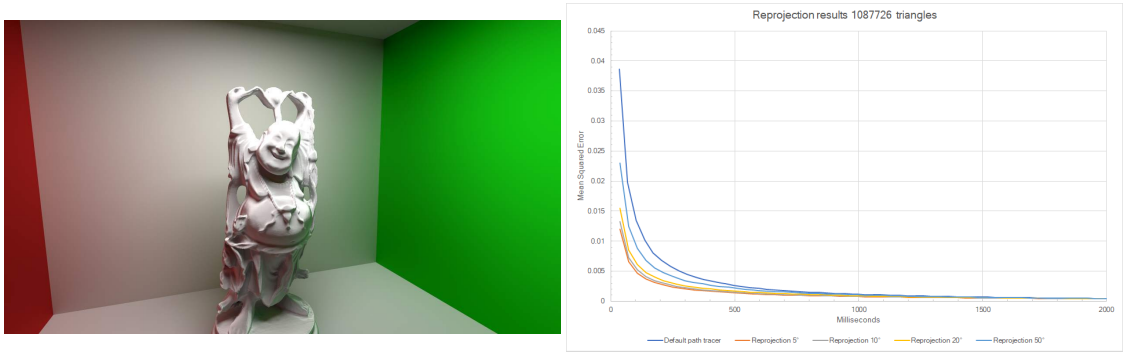


Figure 31: Reference image of the Cornell box with a Buddha model of 1087726 triangles, with the result graph.

From the result we can notice that different lighting or the amount of triangles in the scene does not influence the effectivity of our method. Figure 30 shows that the material complexity does matter, just like the high reflective scene, which is in the appendix B.1.5. This explains why the reprojection method is less effective in the San Miguel scene than in the other scenes, because the San Miguel scene contains more complex materials.

4.4 Combination results

For the combination of the two methods we use a few of the previous tested settings. So we take the best settings for the $min_{spp} \in (2, 4)$ and $max_{spp} = 32$ founded in chapter 4.2. For reprojection we take the best rotation step size 5° with the smallest acceptable reprojection error $\frac{1}{256}$. The amount of samples we take per step is both tested for 32 and 64 samples per step. The amount of samples reprojected is also tested with one or four samples. The same kind of test is preformed like the reprojection method tests in chapter 4.3 and the results are compared with the reprojection method and the default path tracer.

4.4.1 Cornell box

Figure 32 shows the results of all different settings on the combination method, which uses adaptive sampling and reprojection. Just like the results of the reprojection method it improves the default path tracer in the first few milliseconds. Only now, if the methods use $min_{spp} = 2$, the default path tracer catches up around 200 milliseconds, which is less promising than the reprojection method.

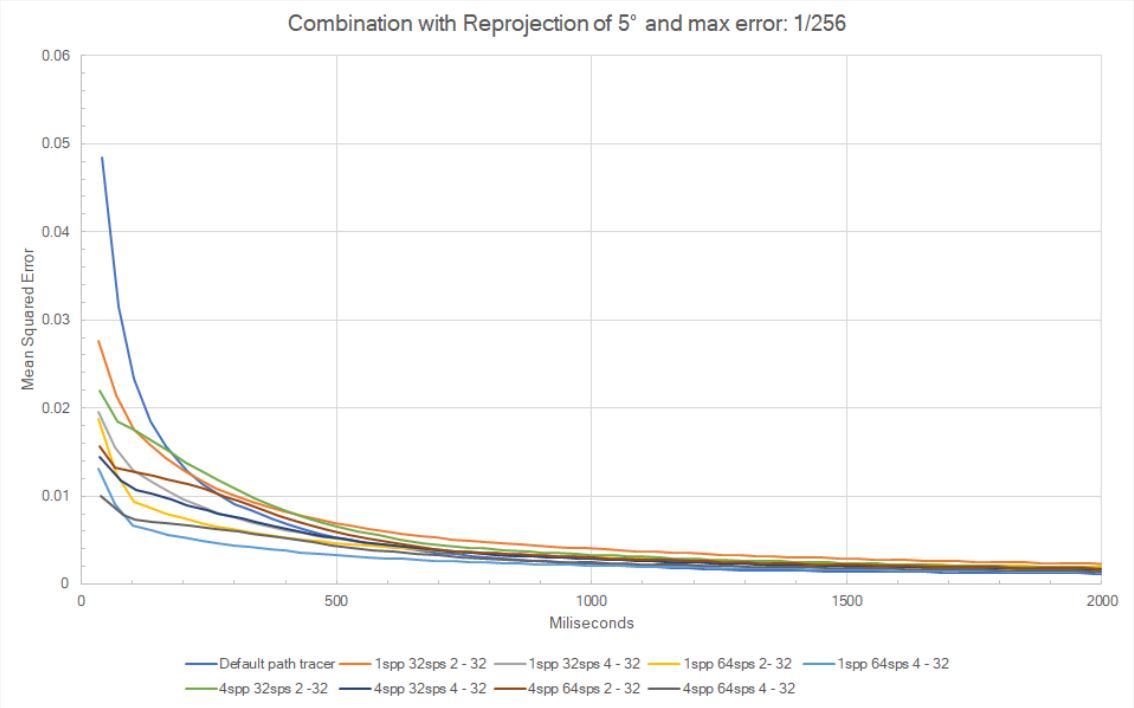


Figure 32: Result graph of the combination method with different settings.

As a comparison of the default path tracer against the reprojection and combination method, we take the settings of 64 samples per step with $min_{spp} = 4$ and $max_{spp} = 32$, while reprojecting one sample at a time. This comparison can be viewed in figure 33. The graph shows that the combination and reprojection method do not give very different results. In figure 34 the image comparison after 100 milliseconds is visualized. Here is a small noise reduction visible between the combination and reprojection method on the right border of the image. In this part of the image, a new part of the scene is introduced that does not have any samples yet that can be reprojected. The combination method uses adaptive sampling to focus on such parts of an image.

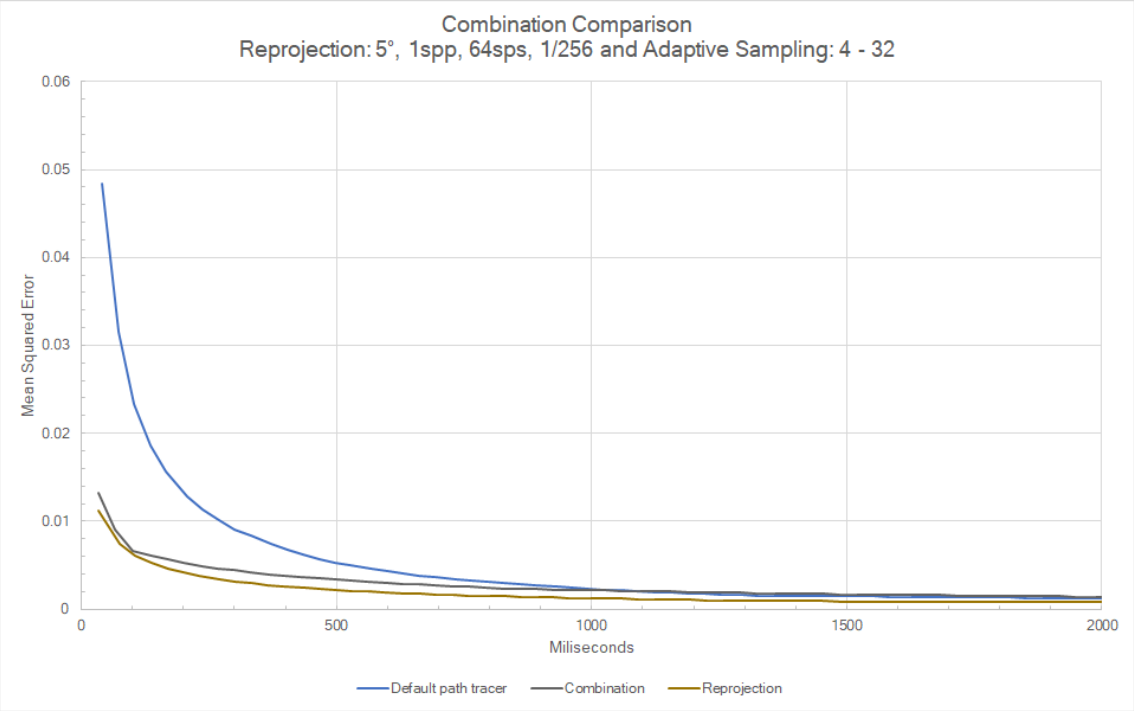


Figure 33: Comparison between the default path tracer, reprojection and the combination method.

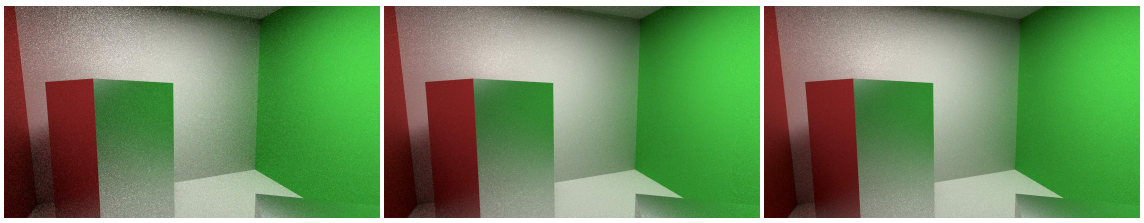


Figure 34: Comparison between the default path tracer (left 109ms), reprojection (center 106ms) and the combination (right 106ms).

4.4.2 Sponza

The results of the combination method in the Sponza scene in figure 35, show a longer advantage over the default path tracer compared to the results of the combination method in the Cornell box scene. Also the results show an order of improvement: 64 samples per step improves on 32 samples per step, a $min_{spp} = 4$ improves on a $min_{spp} = 2$ and reprojecting four samples gives a better result than reprojecting one sample.

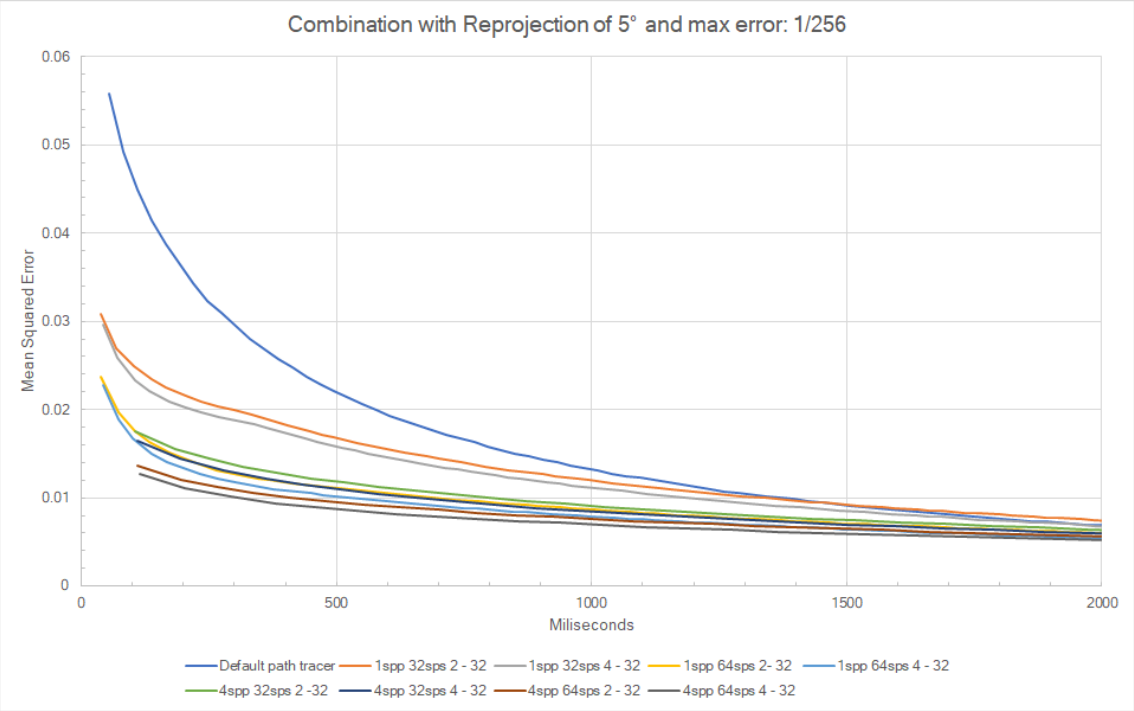


Figure 35: Result graph of the combination method with different settings.

For the comparison of the combination method with the reprojection method and the default path tracer, the same settings are used as for the comparison in the Cornell box scene. For the first 700 milliseconds the combination method seems to give a better result instead of the reprojection method (see figure 36). This is possible, because the Sponza scene is more complex and needs more time per frame, where the adaptive sampling in the combination method gives a reduction in frame time, by choosing pixels that need more samples. In the comparison images in figure 37, which are taken after 200 milliseconds, we see a noise reduction between the default path tracer and the reprojection method. However, when we compare the reprojection method with the combination method, an improvement is hardly visible. Only above the blue curtain on the right, a small noise reduction can be noticed.

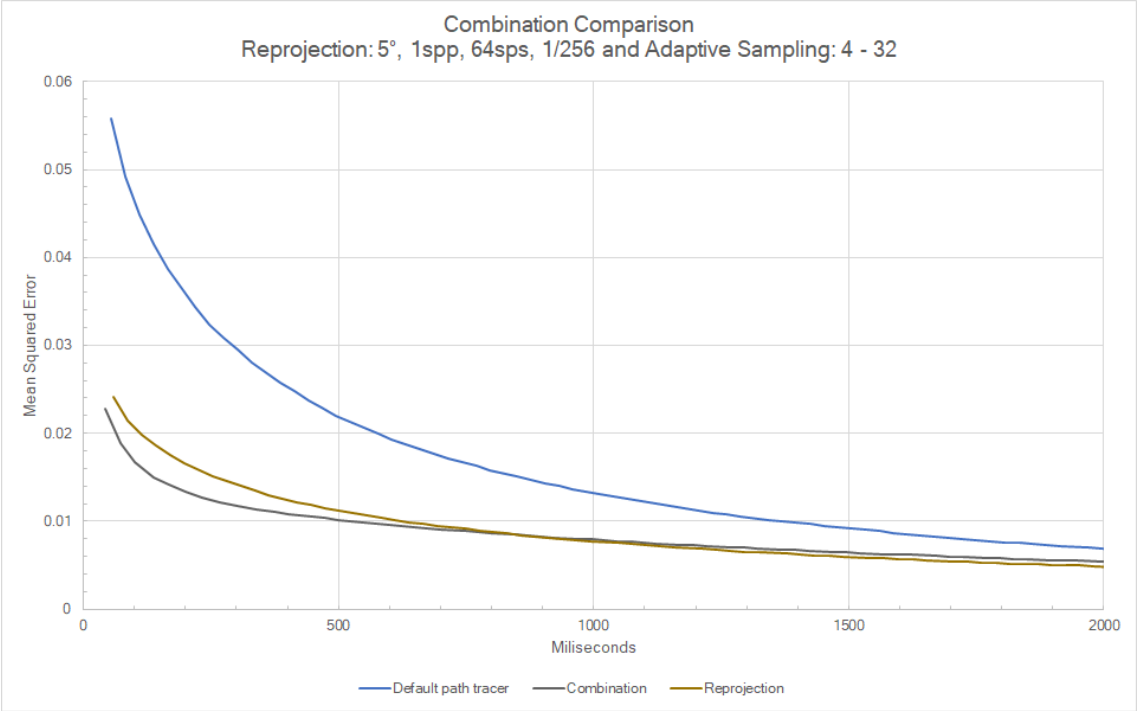


Figure 36: Comparison between the default path tracer, reprojection and the combination method.

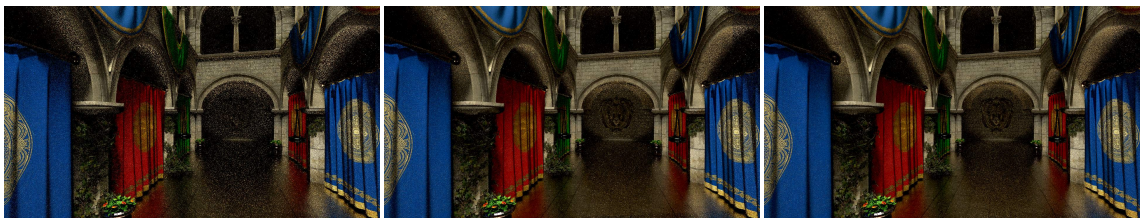


Figure 37: Comparison between the default path tracer (left 221ms), reprojection (center 201ms) and the combination (right 201ms).

4.4.3 San Miguel

In chapters 4.2.4 and 4.3.4 we have seen that directional light influences the adaptive sampling results and complex scene materials influence the reprojection results. The San Miguel scene has both directional light and complex scene materials. The results can be seen in graph 38. All the different settings do not deviate much from each other, but they all give an improvement on the default path tracer for a small amount of milliseconds.

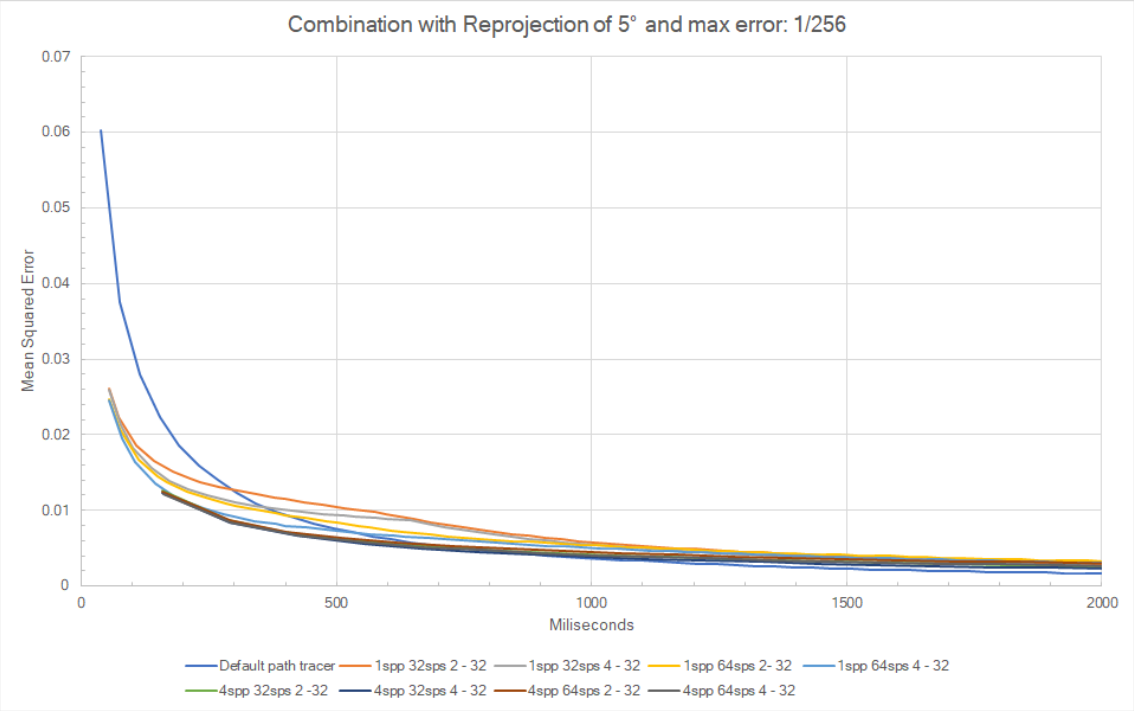


Figure 38: Result graph of the combination method with different settings.

The comparison gives the same results as in the Sponza scene (see figure 39). Here the combination method improves on the reprojection method, but only for a short time, which is the same on all other results on the San Miguel scene. In figure 40, the improvement is hardly visible between the combination method and reprojection method. Still the dark areas get a reduction in noise compared to the default path tracer.

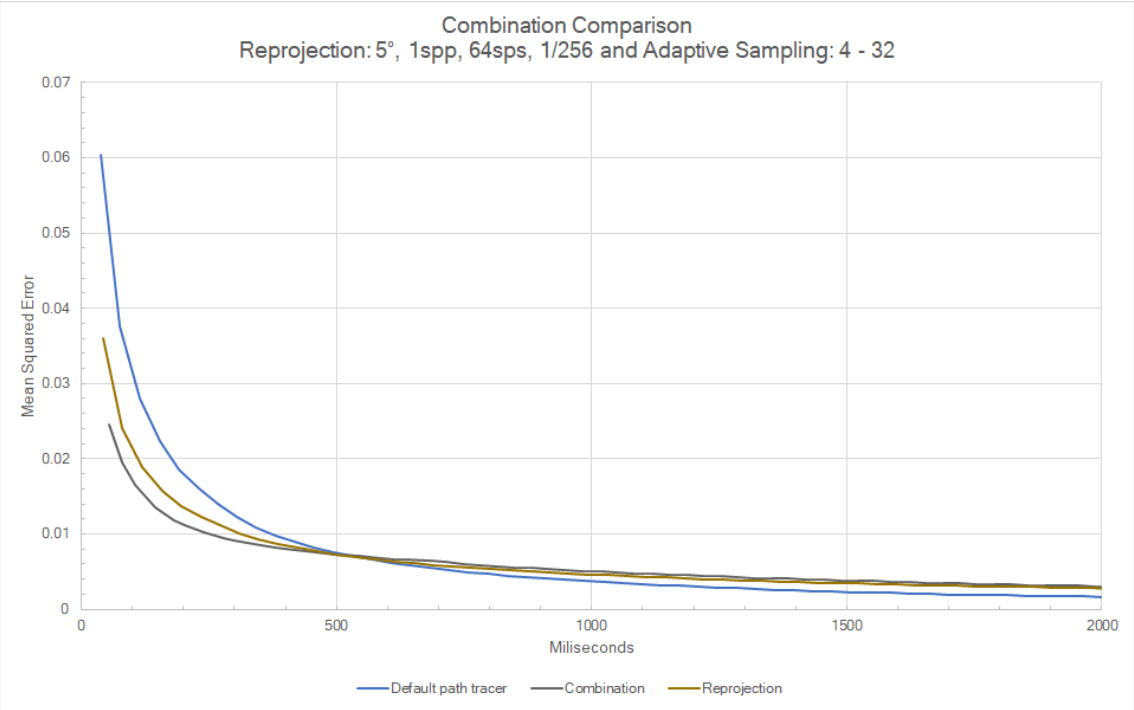


Figure 39: Comparison between the default path tracer, reprojection and the combination method.

Figure 40: Comparison between the default path tracer (left 412ms), reprojection (center 422ms) and the combination (right 414ms).

5 Discussion

Chapter 4 shows all the results of both methods and the combination method, while using different parameters and scenes. From the results of the scene property tests, we know that both methods depend on the scene complexity. Still most of the results give a higher quality measure in a shorter amount of time compared with the default path tracer.

Adaptive sampling improves the quality of the image in the first milliseconds compared with the default path tracer, by resolving pixels who got a higher chance of noise. This only applies for the Sponza scene and Cornell box scene, because the San Miguel scene contains directional lighting, which affects the adaptive sampling. After an amount of milliseconds, the default path tracer catches up with the adaptive sampling method. We, however, are focused on low render times which gives us the preference to the adaptive sampling method. When we use $min_{spp} = 8$, the default path tracer gives a better result than the adaptive sampling method. We expect that a min_{spp} of eight increases the learning time too much before the method has effect, which gives worse results compared to the default path tracer within the first few milliseconds.

The heatmaps of the scenes, which show where the focus of our adaptive sampling method lies, show that we focus on darker areas and ignore brighter spots. This is exactly what we want to see, because brighter spots converge faster, like the stonewall above the arc in the Sponza scene. There seems to be no difference in sampling probability when we look at shaded areas in the heatmap of the San Miguel scene. Here we scale the variance too much by the luminance, which gives the same probability to all shaded areas.

By first resolving pixels who have a higher chance to get noise, the outcome of the adaptive sampling method can be a better source for filtering algorithms. Such algorithms depend on the input, because when there is less high frequency noise the output of a filtering algorithm gets more appealing.

If the render time does not matter, the adaptive sampling method will not improve on the default path tracer. We make an approximation on how much more samples we need, based on the variance known so far. If the approximation is not correct, then the probability of some pixels is too small to get enough samples, in which case the default path tracer preforms better over time.

The reprojection method improves the quality of the image in an early stage, what is practical for small camera movements in real-time applications. Using the material properties as a guideline if we reject or accept a reprojection, like we do, has not yet been done by the mentioned work in chapter 2. Günther and Grosch (2015) has an increase in error when an object contains glossy or glass material, but this is caused by the difference in color and not by looking explicitly at the material property. Using the reflection term from Walter et al. (2007) for the error calculation is promising, because highlights are not reprojected on a glossy material, but other parts of a glossy material are. This is not visualized in the results, because we use a rotation of the camera instead of a movement of the camera towards another position, which is needed for a change in the highlights.

By ignoring glass and high reflective materials, the advantage of reprojection method decreases while the quantity of such materials in a scene increases. Establishing a mapping of reflections between frames, like they do in Xie et al. (2017), could solve this. We recommend using adaptive sampling, to solve this, because mapping reflections introduces more overhead, memory usage and unnecessary complexion when a ray reflects multiple times.

The benefit of the reprojection method increases when we rotate less, which is clearly visible in all the result graphs. It is also an obvious conclusion, because with a smaller rotation more of the previous frame is still visible in the new frame. However, while rotating with 50° the reprojection method is less effective, it still improves the default path tracer in the Sponza scene and Cornell box scene. Just like the amount of rotation the amount of samples per step gives an expected result. If we take more samples per step, we get a better result, because more samples, which are already closer to the correct value, get reprojected. We can also see an increase in result when the amount of samples we reproject is set to four. The reason that it improves, is because the chance that a pixel reprojects at least one value increases, where reprojecting one of the four samples is always better than none. Changing the maximum allowed error between $\frac{1}{256}$ and $\frac{8}{256}$, does not seem to affect the result of the reprojection method. Therefore we advise to use the lower maximum error to come closer to a physical correct image.

The frame time of a path tracer increases more based on the amount of triangles in the scene, but the overhead of reprojection decreases when more triangles are added. This is because the execution time of reprojection scales less on the amount of triangles compared to the path tracing algorithm. The only part of reprojection that is influenced by the amount of triangles is the verification phase, because here a ray

needs to be traced. Overall the reprojection method increases the frame time on every scene, but within the same amount of milliseconds as the default path tracer, it still gives a better result.

Combining the methods is useful for small movements in a real-time application. Noise introduced in new visible parts of a scene, after a movement occurs, are quickly reduced (see figure 34), which is an advantage for filters and the visual aspect of the image. The adaptive sampling method also amplifies the pixels that could not be reprojected by the reprojection method.

The benefit of combining adaptive sampling with reprojection seems to be dependent on the scene complexity. On the Cornell box scene the combination method results are worse compared to the reprojection method. A reason could be that the final rotation in the Cornell box scene does not introduce complex scenery, which makes the adaptive sampling focus first on parts that are not important. For the Sponza and San Miguel scene the combination method gives the best result. The difference with the reprojection method is small, but there still is an improvement in the first 500 milliseconds for the Sponza and San Miguel scene. Using different parameters within the combination method gives the expected result. Just like in the reprojection method, there is an improvement in order of amount of samples per step and amount of samples reprojected. For the setting of min_{spp}, we recommend using four, because in most cases it improves on a $min_{spp} = 2$.

Improvement on the default path tracer is a good step towards real-time applications, but as we can see in the measurements the amount of milliseconds it takes to get one frame is still too much for real-time applications (at least 30 frames per second). Even if the frame time was below 33 milliseconds the graphics quality result would not be satisfying enough, because the results still contain noise what we can see in figures 21, 25, 28, 34.

6 Conclusions and future work

The goal of the thesis is to improve the efficiency of a real-time path tracer using reprojection and/or adaptive sampling, without exceeding a specified maximum error. Within the same amount of time as the default path tracer, which uses variance reduction techniques, the reprojection method improves the efficiency after a movement occurs in the first milliseconds. Here we chose a maximum reprojection error of $\frac{1}{256}$, because it is the smallest integer value difference that can be visualized on most of the standard monitors. We calculate the reprojection error with the reflection term of Walter et al. (2007), which takes the roughness of the materials into account. For glass or highly reflective materials we do not reproject samples, because such materials are very dependent on the view position.

Using adaptive sampling on top of the default path tracer yields a better efficiency, if we compare the variance of the distance to the reference pixels, with the default path tracer. The amount of efficiency depends on the minimum and maximum amount of samples we take, where in between samples are distributed by a decreasing threshold based on the variance of the pixels. The amount of samples a pixel gets between the minimum and maximum amount of samples determines a probability, which is used to distribute new samples. By scaling the variance with the luminance of a pixel, our adaptive sampling method focuses more on dark areas of a scene.

Combining the methods gives adaptive sampling more focus on the pixels that are new or did not get a sample from the reprojection method. Therefore, it quickly reduces the largest amount of noise which is useful for other techniques like filtering. The combination is a small improvement on the reprojection method when used in larger scenes. Different scene properties like: lighting, materials and amount of triangles, influences both the adaptive sampling as the reprojection method. This makes the amount of efficiency that both methods provide scene dependent.

Currently the methods are only available for static scenes. For future work we would like to see an implementation of the methods in scenes with moving objects and lights. For moving objects, an object ID could be stored on the first intersection point. The ID can be compared with the object that is found during the verification phase. Moving lights are more difficult, but should be possible by casting shadow rays to the moving light and verificate if it is still visible. The bidirectional scattering distribution function can also be used to determine a reprojection error for moving lights, by using the incoming light direction.

Future work for the adaptive sampling method is dynamically increasing or decrease the probability of sampling a pixel. This could be done by keeping the variance of the pixels updated, or, if the render time does not matter, taking neighbour pixels into account to estimate an error to the reference pixel like Rousselle et al. (2011) and Li et al. (2012).

We would like to see the method in combination with a real-time filter, which may remove all the noise in the first milliseconds. An implementation of the method into a bidirectional path tracer would also be interesting, because a bidirectional path tracer gives a faster result on complex scenes, especially for glass materials that introduce caustics. We expect that the combination method will enhance the result of a bidirectional path tracer, but a drawback would be the frame time, which increases for a bidirectional path tracer.

7 Acknowledgments

The author wishes to thank 3Dimerce with Huub van Summeren for lending their hardware and workplace to make this thesis possible. The thesis is implemented on top of the default path tracer, which contains variance reduction techniques and other additional requirements for a path tracer. Therefore, the author thanks Jacco Bikker for providing the default path tracer, and his guidance throughout the project. The author also thanks Marc van Kreveld for reviewing and evaluating this thesis. Proofreading is done by Jacco Bikker and Maaike Horst, wherefore the author shows his gratitude.

References

Adelson, S. J. and Hodges, L. F. (1993). Exact ray-traced animation frames generated by reprojection. Technical report, Georgia Institute of Technology.

Corso, A. D., Salvi, M., Kolb, C., Frisvad, J. R., Lefohn, A., and Luebke, D. (2017). Interactive stable ray tracing. In *Proceedings of High Performance Graphics*, page 1. ACM.

Günther, T. and Grosch, T. (2015). Consistent scene editing by progressive difference images. In *Computer Graphics Forum*, volume 34-4, pages 41–51. Wiley Online Library.

Hachisuka, T., Jarosz, W., Weistroffer, R. P., Dale, K., Humphreys, G., Zwicker, M., and Jensen, H. W. (2008). Multidimensional adaptive sampling and reconstruction for ray tracing. In *ACM Transactions on Graphics (TOG)*, volume 27-3, page 33. ACM.

Hachisuka, T. and Jensen, H. W. (2009). Stochastic progressive photon mapping. *ACM Transactions on Graphics (TOG)*, 28-5:141.

Jakob, W. and Marschner, S. (2012). Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (TOG)*, 31-4:58.

Jensen, H. W., Christensen, P. H., Kato, T., and Suykens, F. (2002). A practical guide to global illumination using photon mapping. *SIGGRAPH 2002 Course Notes CD-ROM*.

Kajiya, J. T. (1986). The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20-4, pages 143–150. ACM.

Kirk, D. and Arvo, J. (1991). Unbiased sampling techniques for image synthesis. In *ACM SIGGRAPH Computer Graphics*, volume 25-4, pages 153–156. ACM.

Lafortune, E. P. and Willems, Y. D. (1993). Bi-directional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*.

Li, T.-M., Wu, Y.-T., and Chuang, Y.-Y. (2012). Sure-based optimization for adaptive sampling and reconstruction. *ACM Transactions on Graphics (TOG)*, 31-6:194.

Lochmann, G., Reinert, B., Ritschel, T., Müller, S., and Seidel, H.-P. (2014). Real-time reflective and refractive novel-view synthesis. In *VMV*, pages 9–16.

Mitchell, D. P. (1991). Spectrally optimal sampling for distribution ray tracing. In *ACM SIGGRAPH Computer Graphics*, volume 25-4, pages 157–164. ACM.

Nehab, D., Sander, P. V., Lawrence, J., Tatarchuk, N., and Isidoro, J. R. (2007). Accelerating real-time shading with reverse reprojection caching. In *Graphics hardware*, volume 41, pages 61–62.

Overbeck, R. S., Donner, C., and Ramamoorthi, R. (2009). Adaptive wavelet rendering. *ACM Trans. Graph.*, 28,4:140–1.

Pharr, M., Jakob, W., and Humphreys, G. (2016). *Physically based rendering: From theory to implementation*. Morgan Kaufmann.

Purgathofer, W. (1987). A statistical method for adaptive stochastic sampling. *Computers & Graphics*, 11-2:157–162.

Rousselle, F., Knaus, C., and Zwicker, M. (2011). Adaptive sampling and reconstruction using greedy error minimization. In *ACM Transactions on Graphics (TOG)*, volume 30-6, page 159. ACM.

Tamstorf, R. and Jensen, H. W. (1997). Adaptive sampling and bias estimation in path tracing. In *Rendering Techniques' 97*, pages 285–295. Springer.

Tole, P., Pellacini, F., Walter, B., and Greenberg, D. P. (2002). Interactive global illumination in dynamic scenes. In *ACM Transactions on Graphics (TOG)*, volume 21-3, pages 537–546. ACM.

Veach, E. and Guibas, L. J. (1997). Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76. ACM Press/Addison-Wesley Publishing Co.

Walter, B., Marschner, S. R., Li, H., and Torrance, K. E. (2007). Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 195–206. Eurographics Association.

Welford, B. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4-3:419–420.

Whitted, T. (2005). An improved illumination model for shaded display. In *ACM Siggraph 2005 Courses*, page 4. ACM.

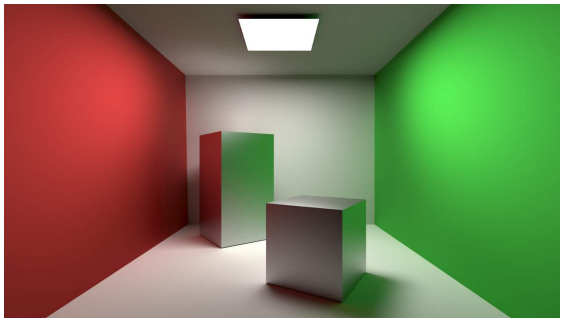
Xie, N., Wang, L., and Dutré, P. (2017). Reflection reprojection using temporal coherence. *The Visual Computer*, pages 1–13.

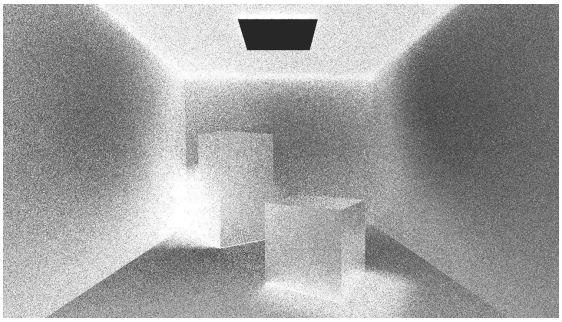
Zimmer, H., Rousselle, F., Jakob, W., Wang, O., Adler, D., Jarosz, W., Sorkine-Hornung, O., and Sorkine-Hornung, A. (2015). Path-space motion estimation and decomposition for robust animation filtering. In *Computer Graphics Forum*, volume 34-4, pages 131–142. Wiley Online Library.

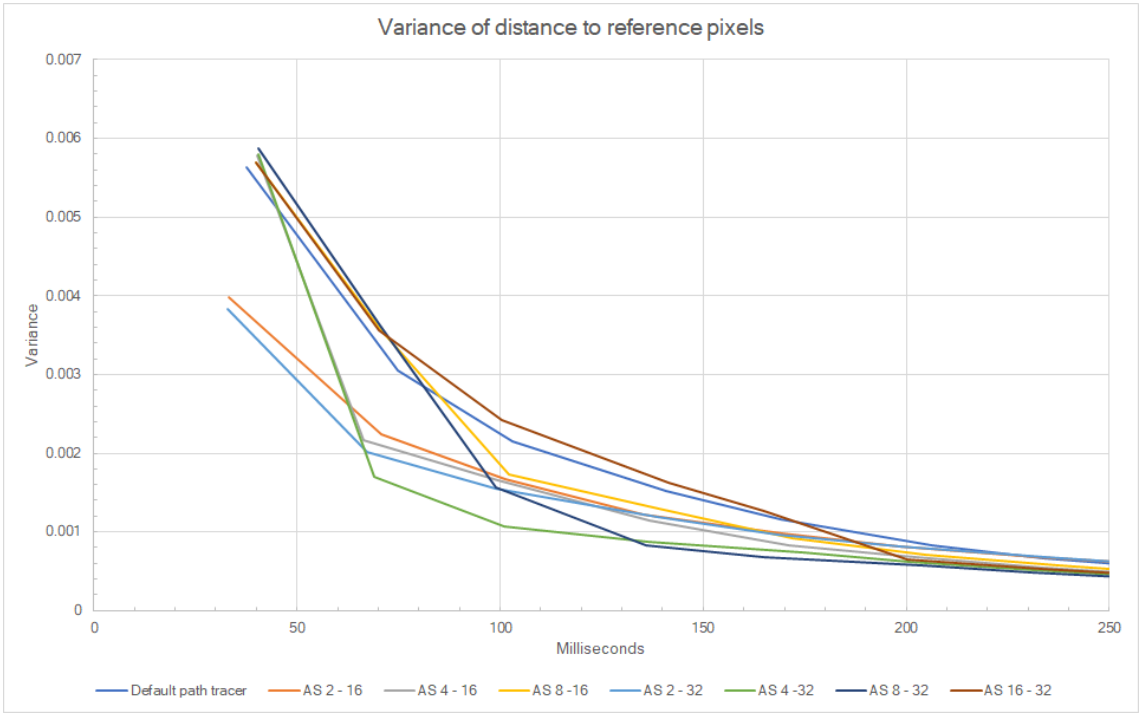
Zwicker, M., Jarosz, W., Lehtinen, J., Moon, B., Ramamoorthi, R., Rousselle, F., Sen, P., Soler, C., and Yoon, S.-E. (2015). Recent advances in adaptive sampling and reconstruction for monte carlo rendering. In *Computer graphics forum*, volume 34-2, pages 667–681. Wiley Online Library.

A Adaptive sampling results

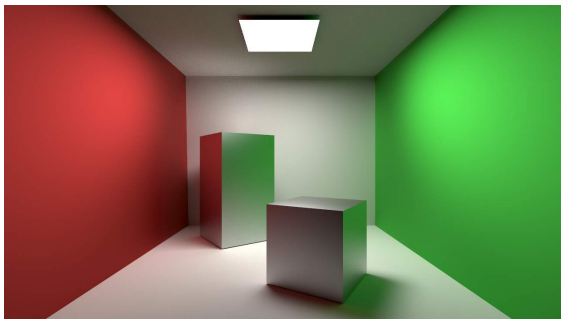
A.1 Cornell box

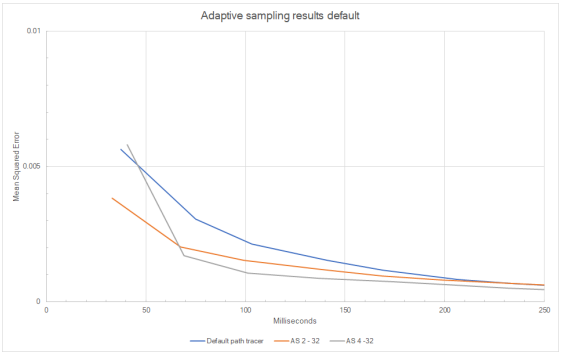


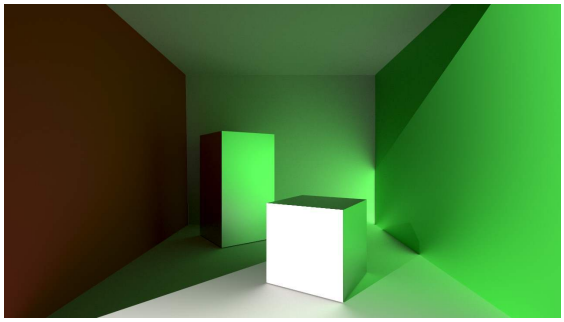


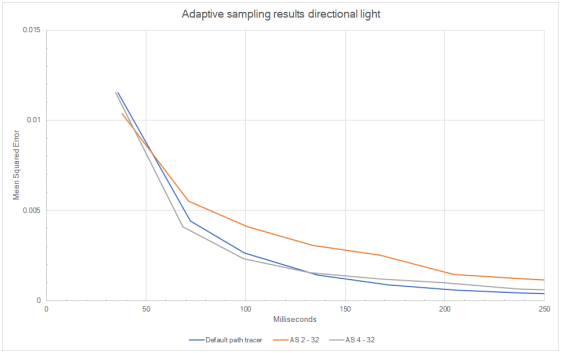


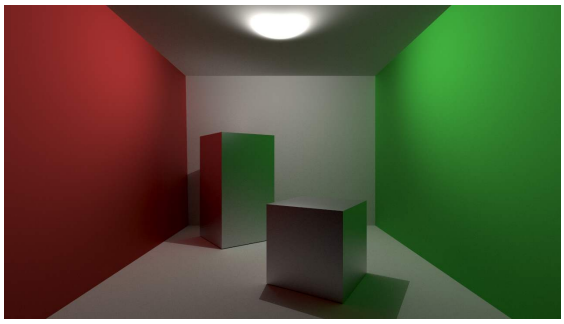
A.1.1 Results for different scene properties

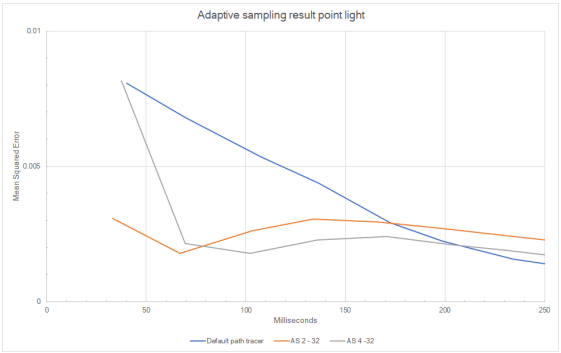


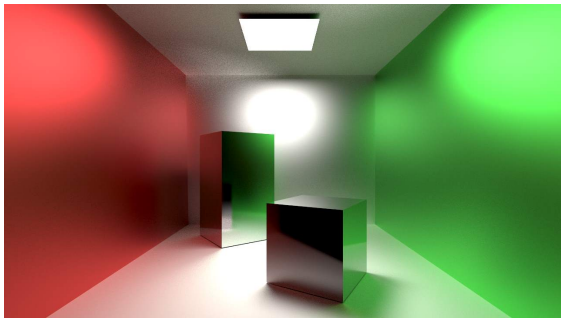


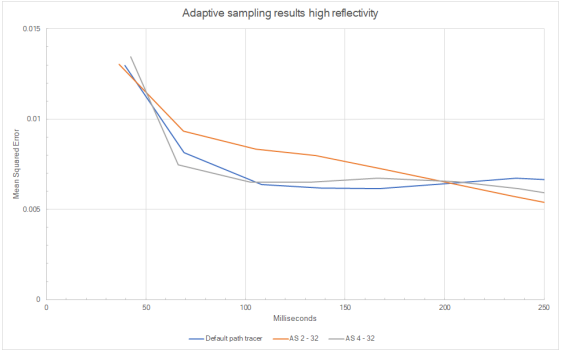


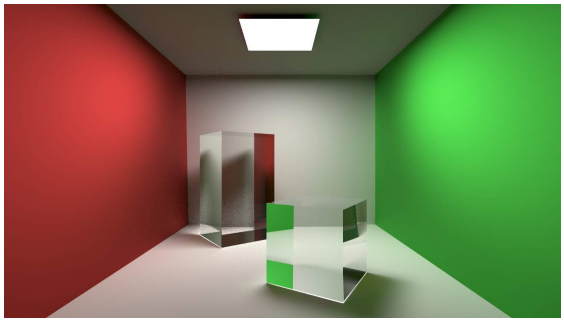

Adaptive sampling results directional light
Mean Squared Error
0.015
0.01
0.005
0
0
50
100
150
200
250
Milliseconds
Default path tracer
AS 2 - 32
AS 4 - 32

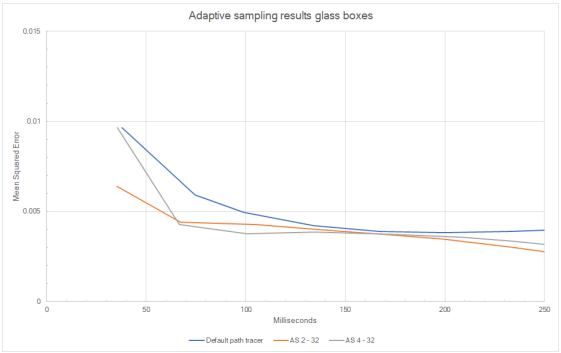


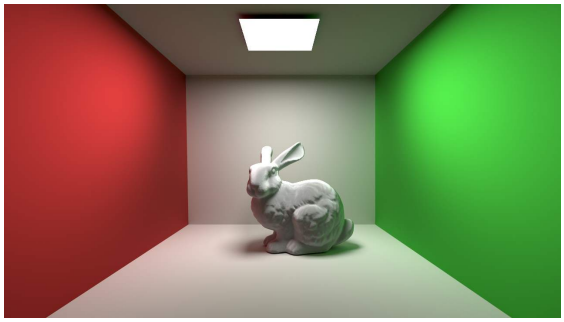

Adaptive sampling result point light
Mean Squared Error
0.01
0.005
0
0
50
100
150
200
250
Milliseconds
Default path tracer
AS 2 - 32
AS 4 -32

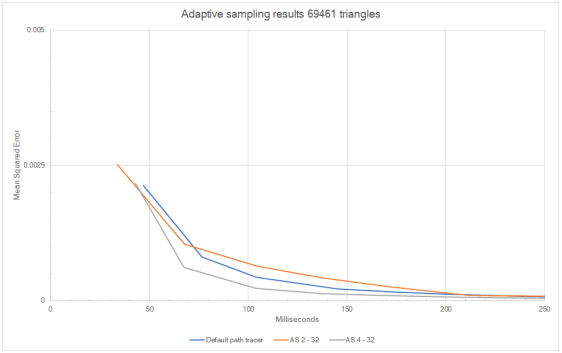


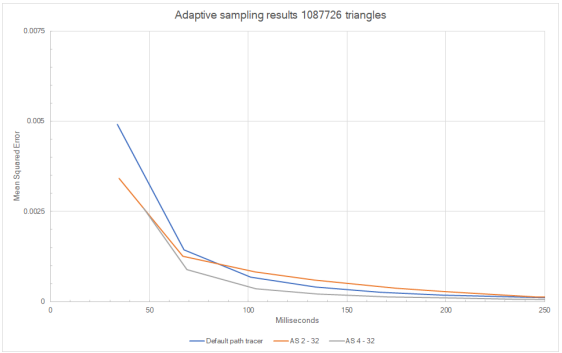

Adaptive sampling results high reflectivity
Mean Squared Error
0.015
0.01
0.005
0
0
50
100
150
200
250
Milliseconds
Default path tracer
AS 2 - 32
AS 4 - 32



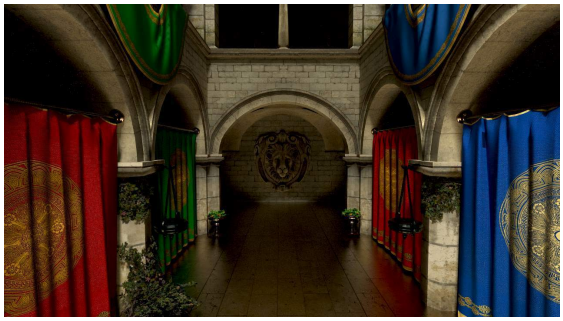

Adaptive sampling results glass boxes
Mean Squared Error
0.015
0.01
0.005
0
0
50
100
150
200
250
Milliseconds
Default path tracer
AS 2 - 32
AS 4 - 32

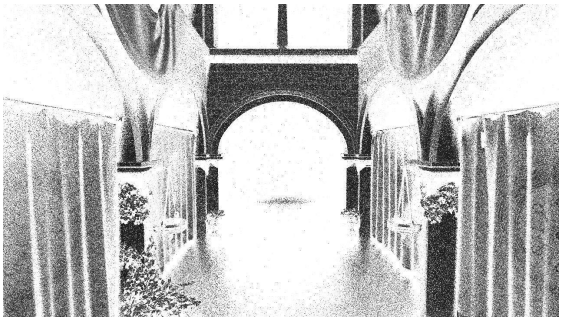


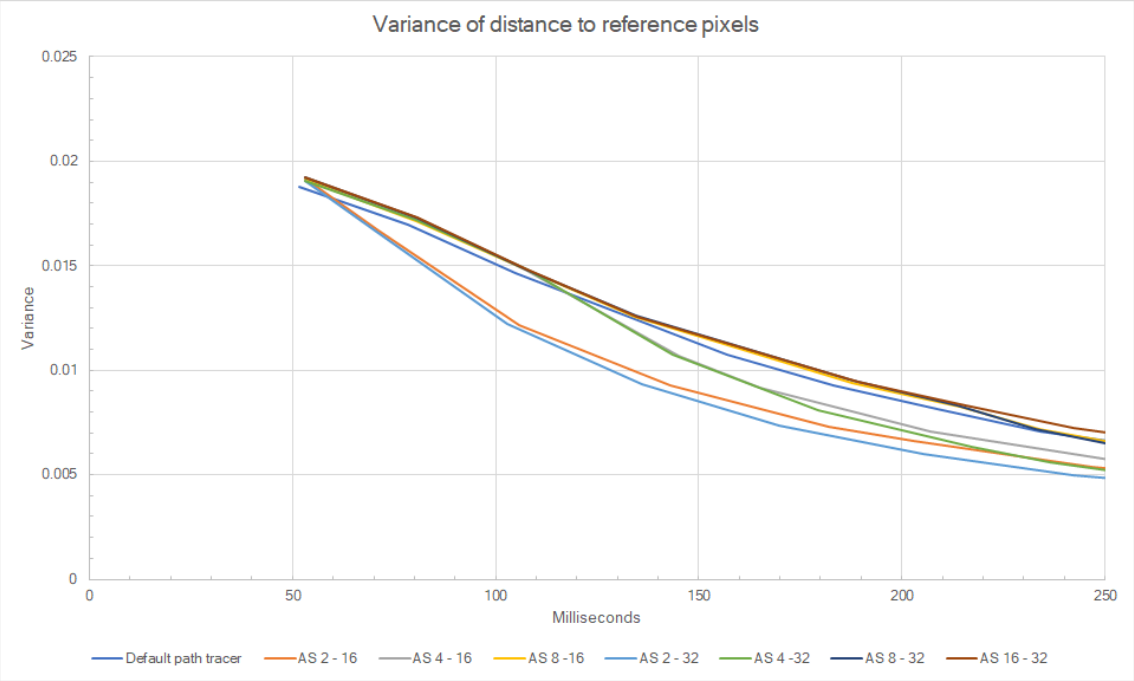

Adaptive sampling results 69461 triangles
Mean Squared Error
0.005
0.0025
0
0
50
100
150
200
250
Milliseconds
Default path tracer
AS 2 - 32
AS 4 - 32


Adaptive sampling results 1087726 triangles
Mean Squared Error
0.0075
0.005
0.0025
0
0
50
100
150
200
250
Milliseconds
Default path tracer
AS 2 - 32
AS 4 - 32

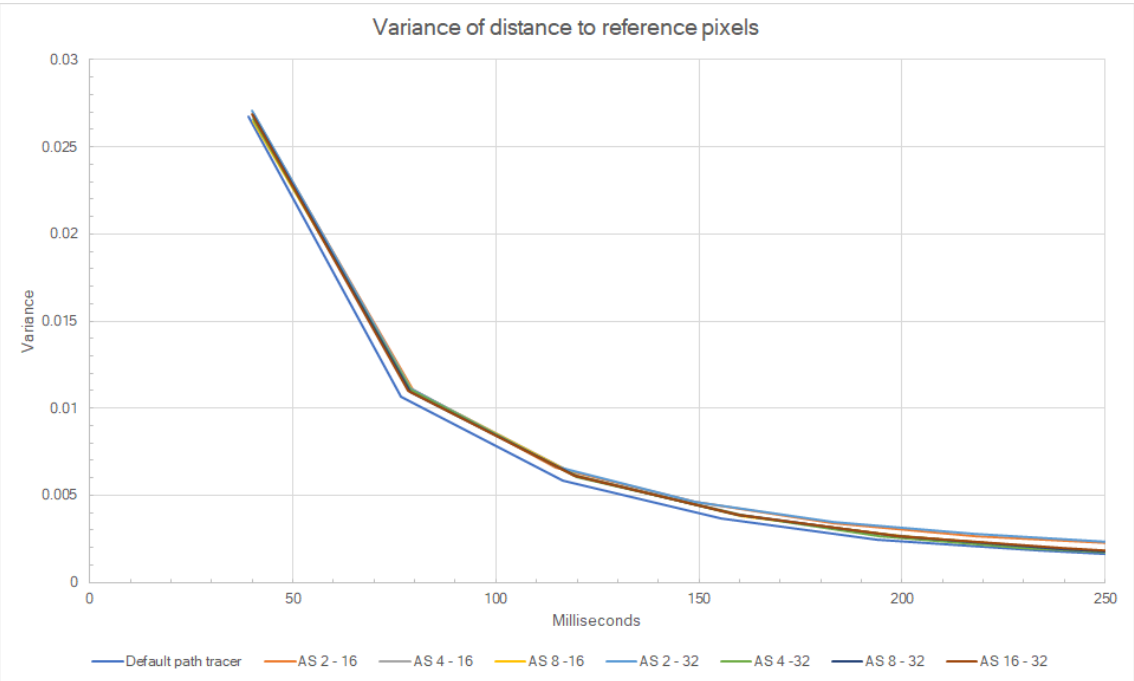
A.2 Sponza





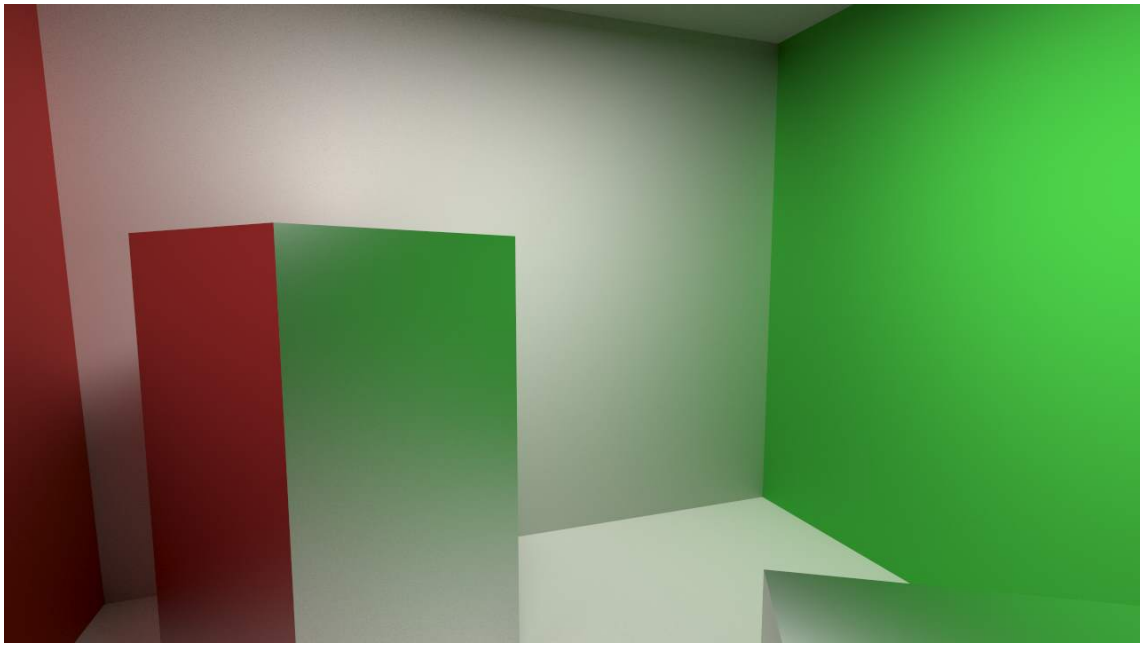


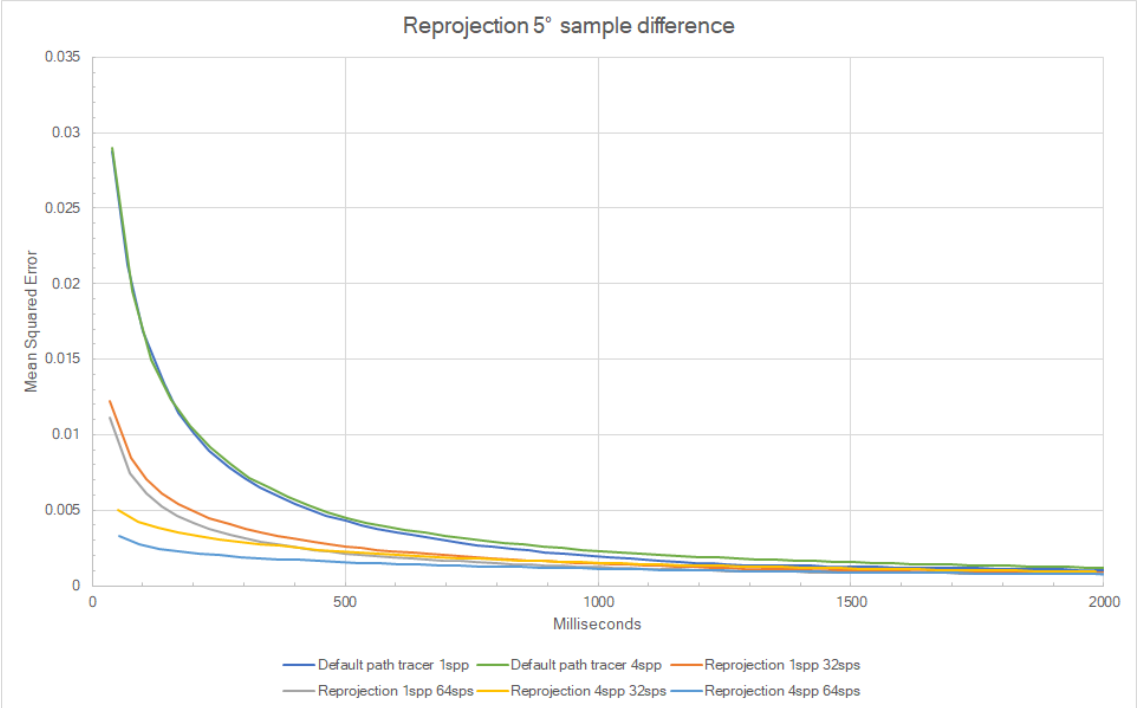
A.3 San Miguel



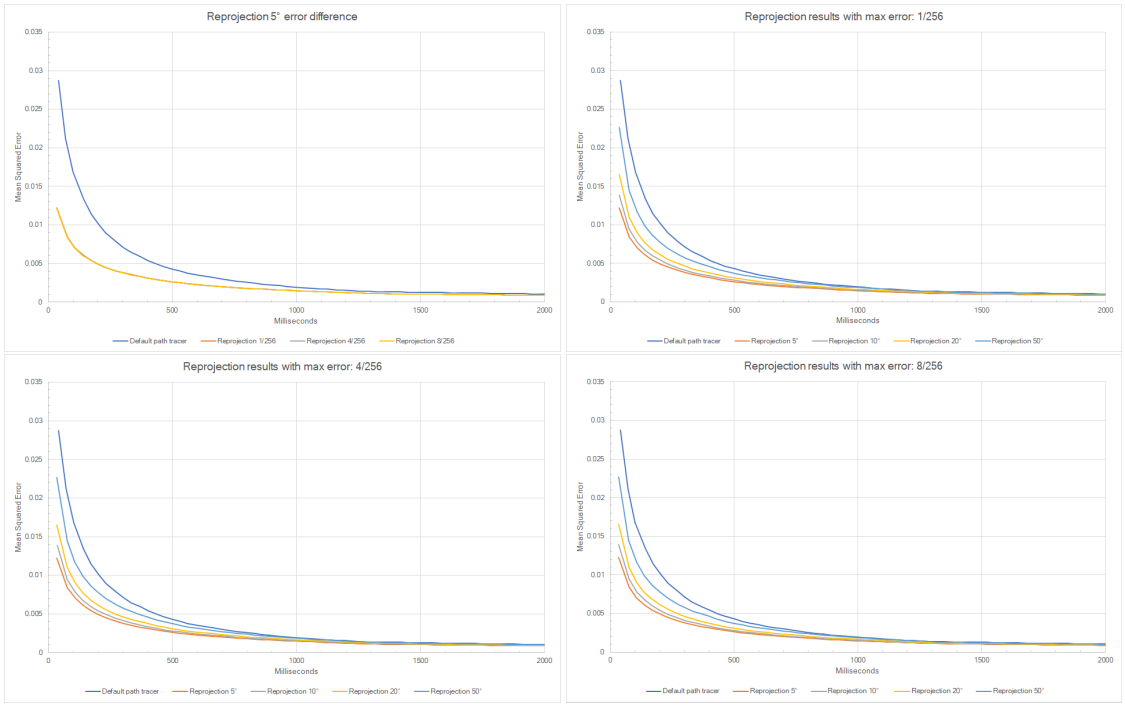
B Reprojection results

B.1 Cornell box

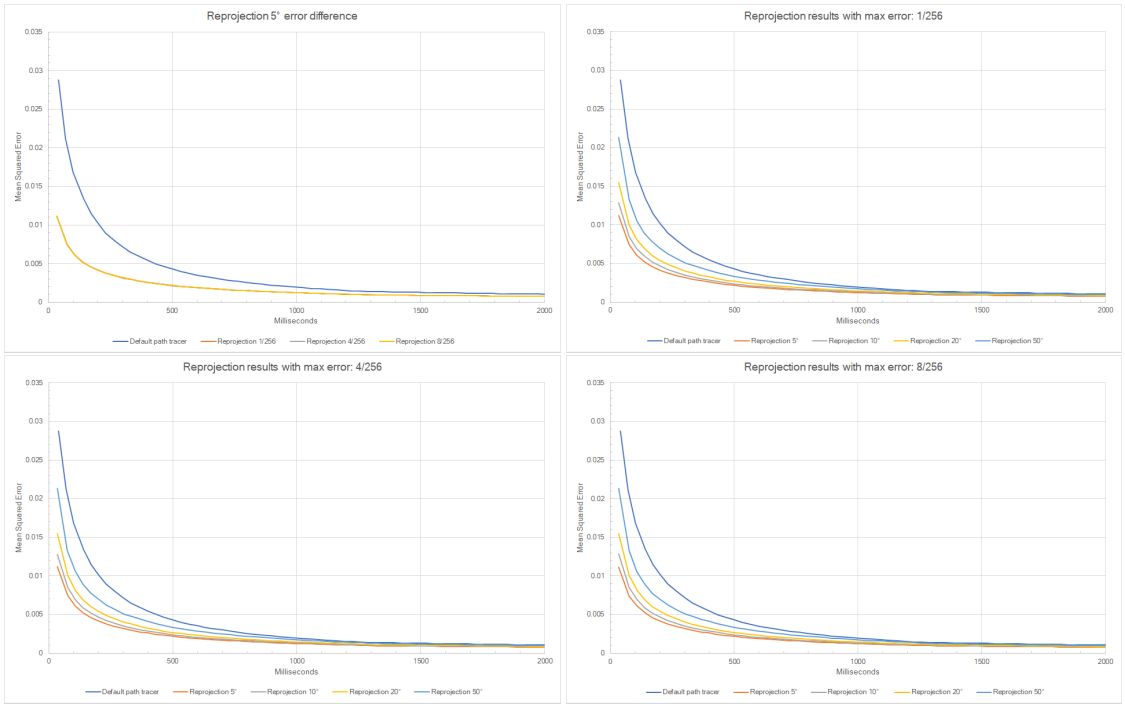




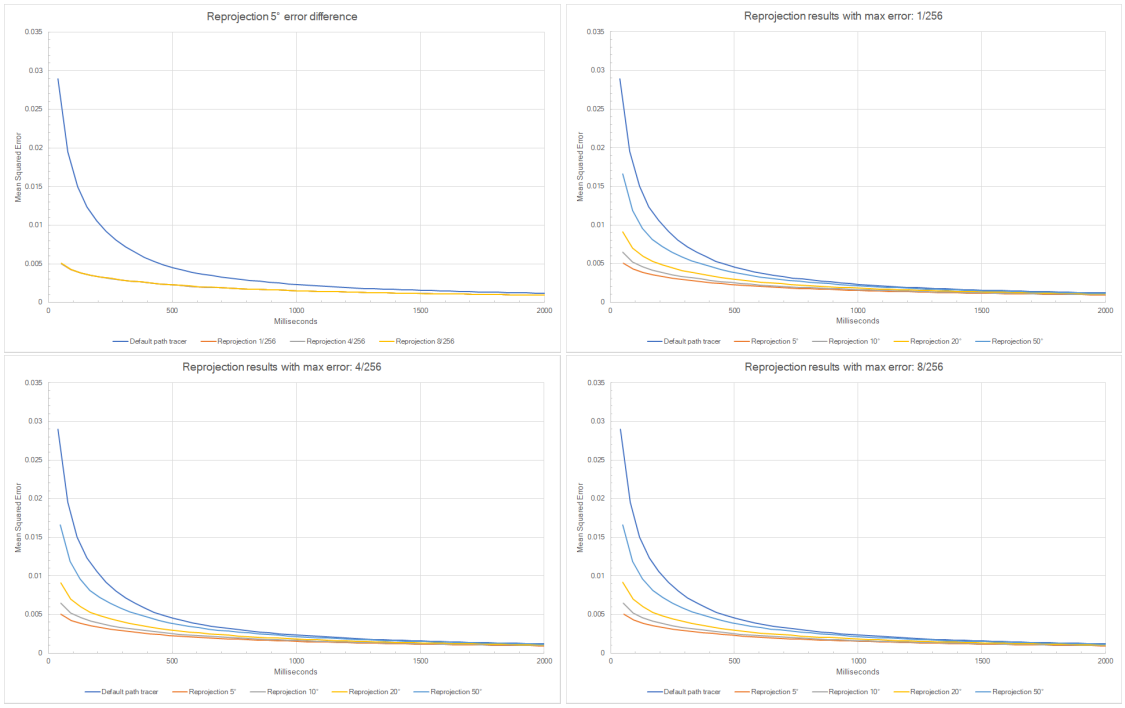
B.1.1 Results for 1spp 32sps



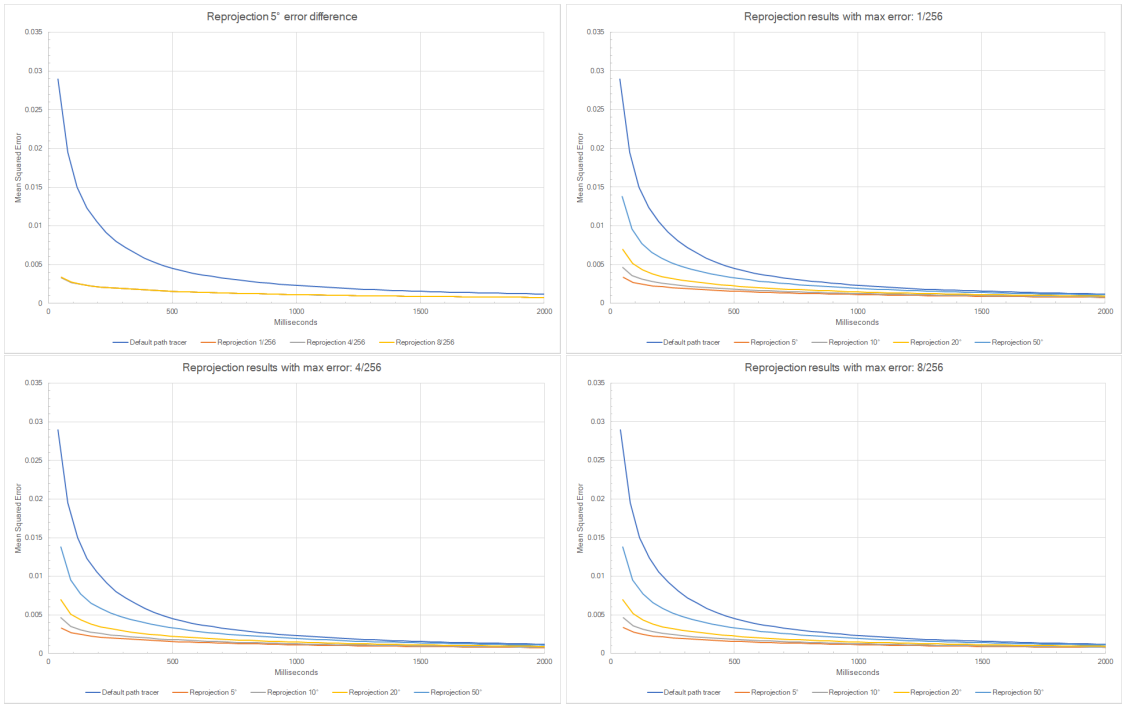
B.1.2 Results for 1spp 64sps



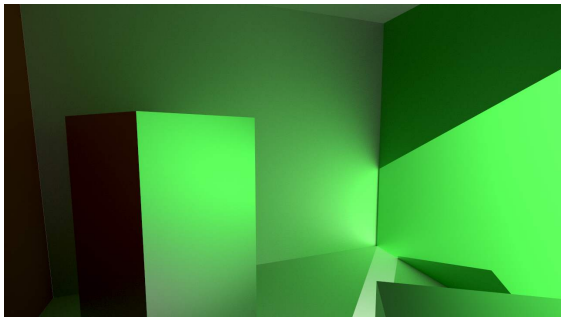
B.1.3 Results for 4spp 32sps

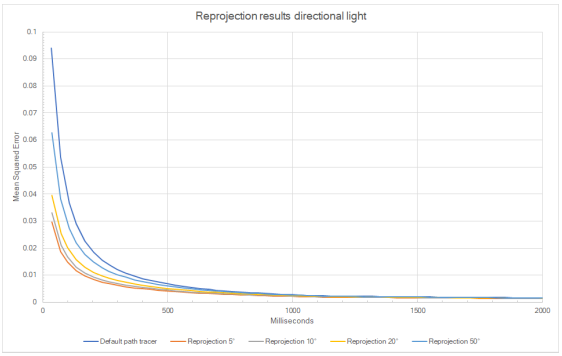


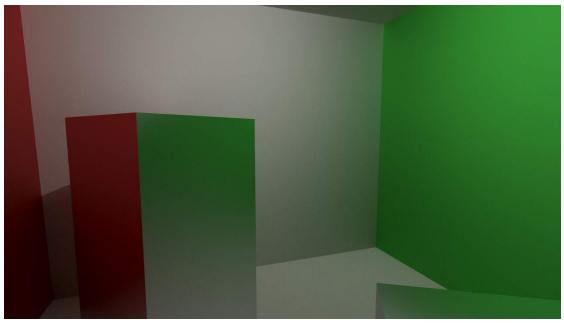
B.1.4 Results for 4spp 64sps

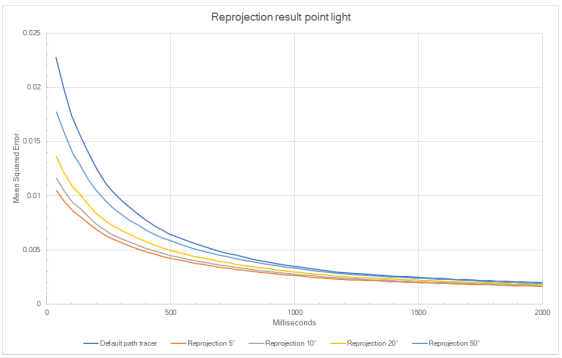


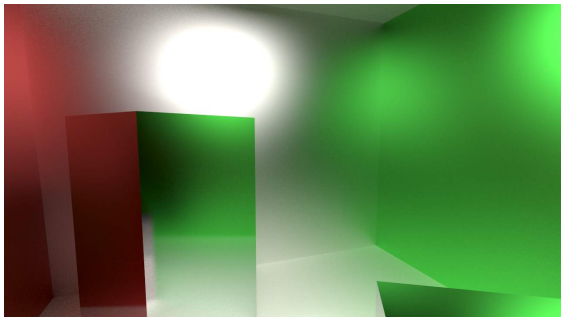
B.1.5 Results for different scene properties

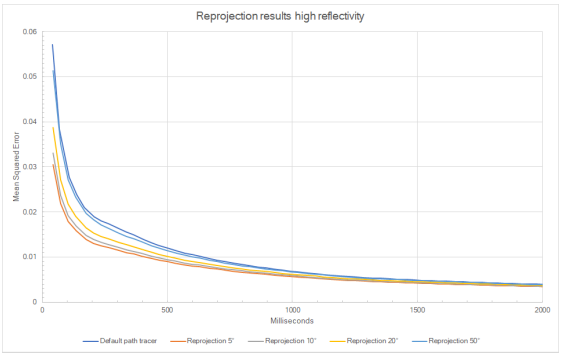


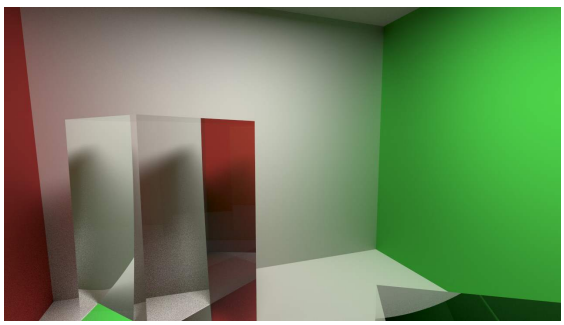


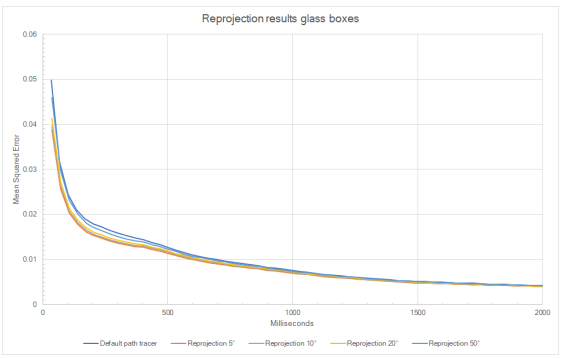


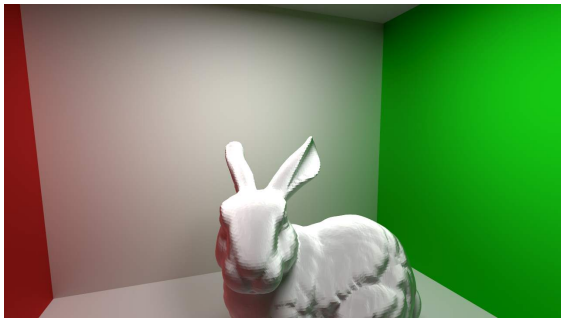


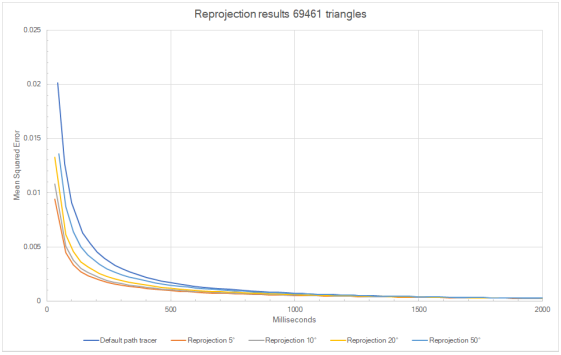


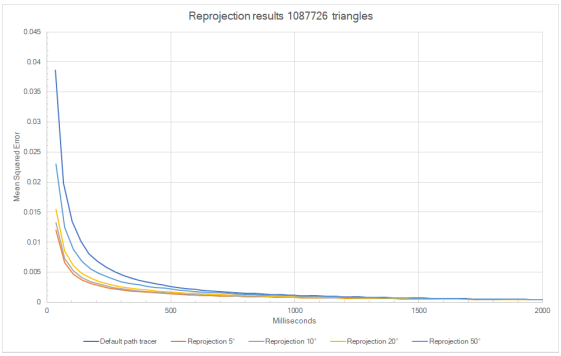




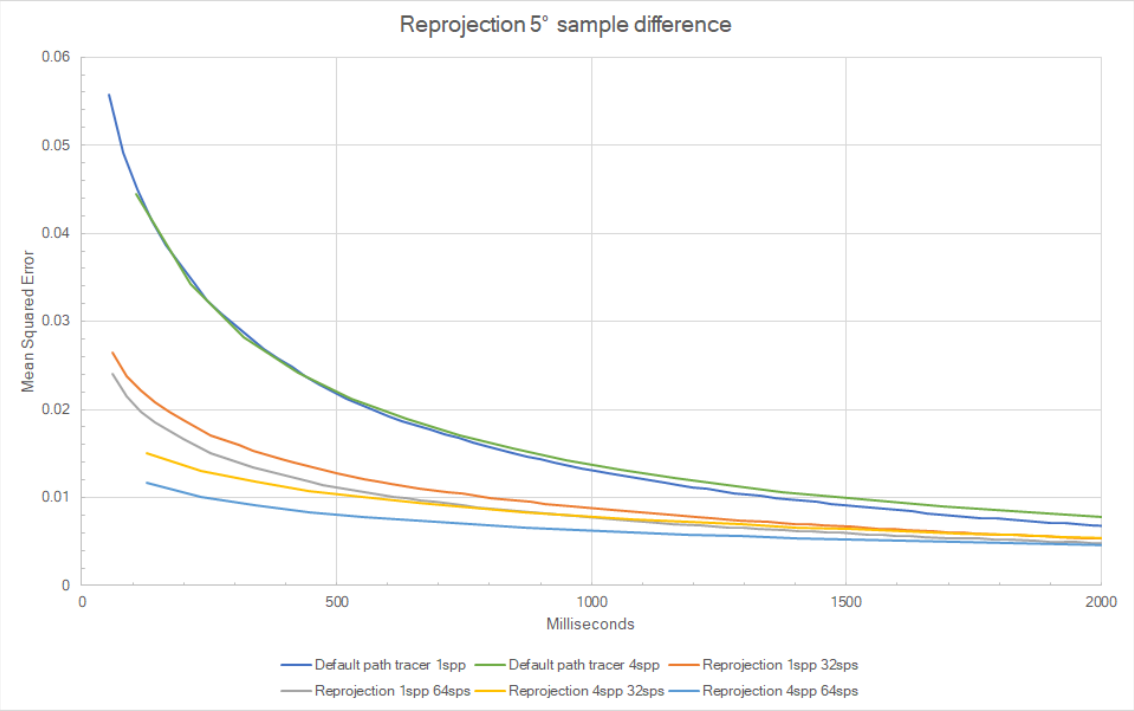




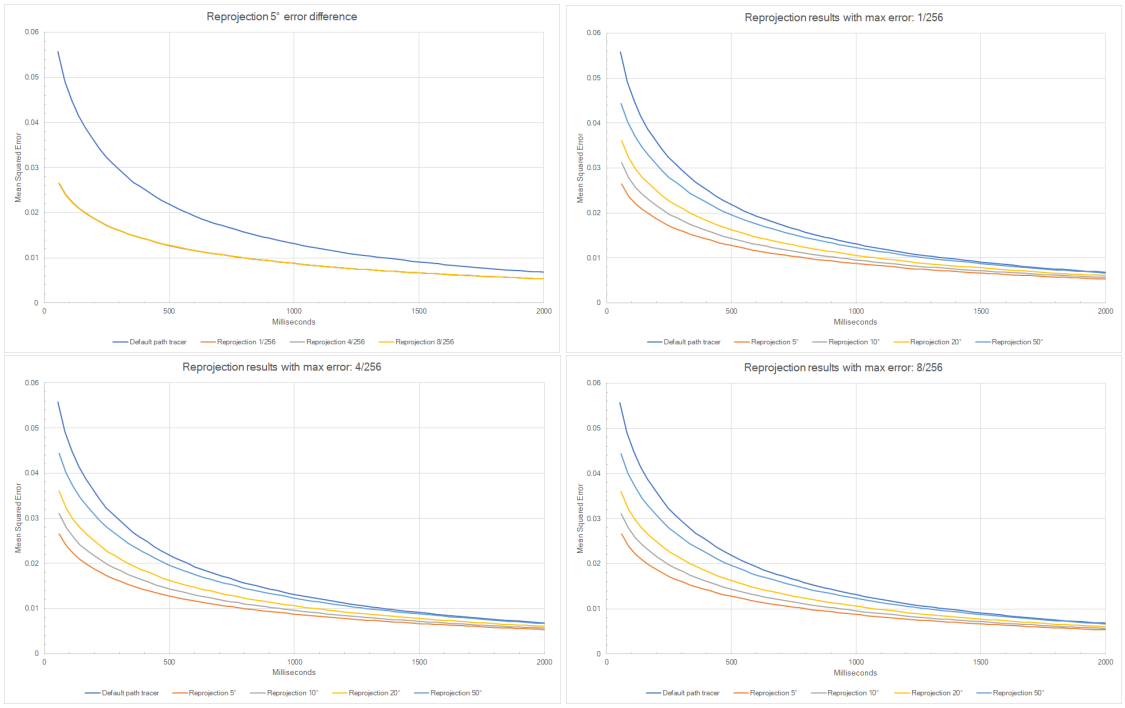

Reprojection results 69461 triangles
Mean Squared Error
0.025
0.02
0.015
0.01
0.005
0
0
500
1000
1500
2000
Milliseconds
Default path tracer
Reprojection 5°
Reprojection 10°
Reprojection 20°
Reprojection 50°


Reprojection results 1087726 triangles
Mean Squared Error
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
500
1000
1500
2000
Milliseconds
Default path tracer
Reprojection 5°
Reprojection 10°
Reprojection 20°
Reprojection 50°

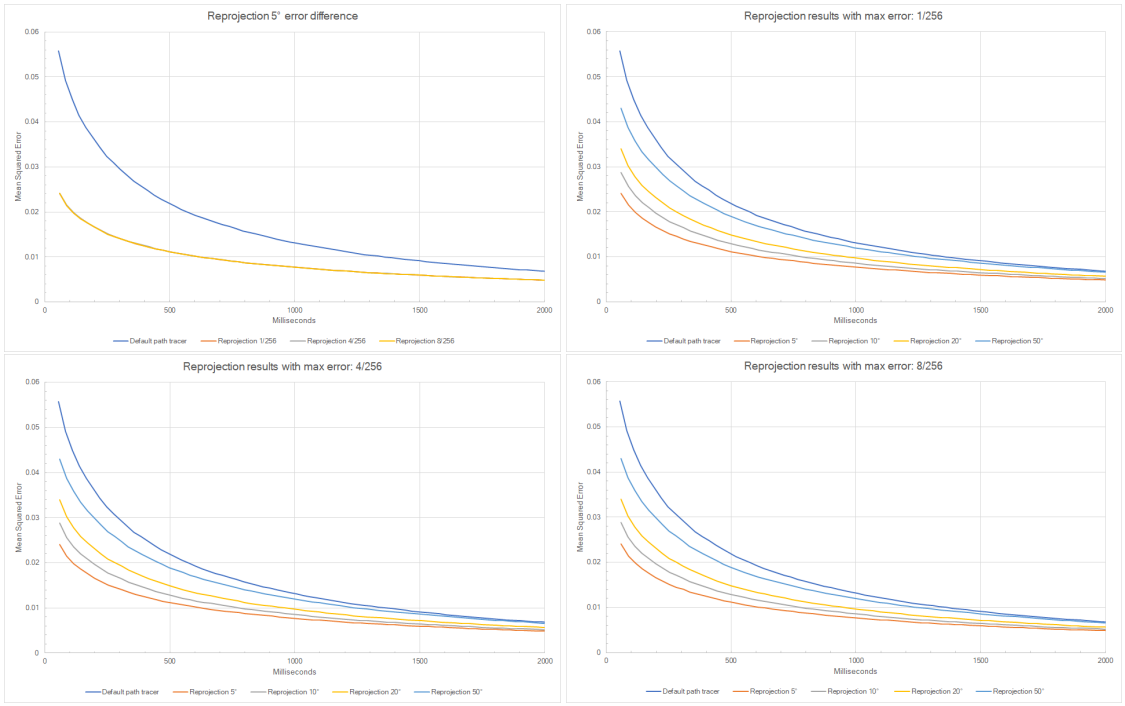
B.2 Sponza



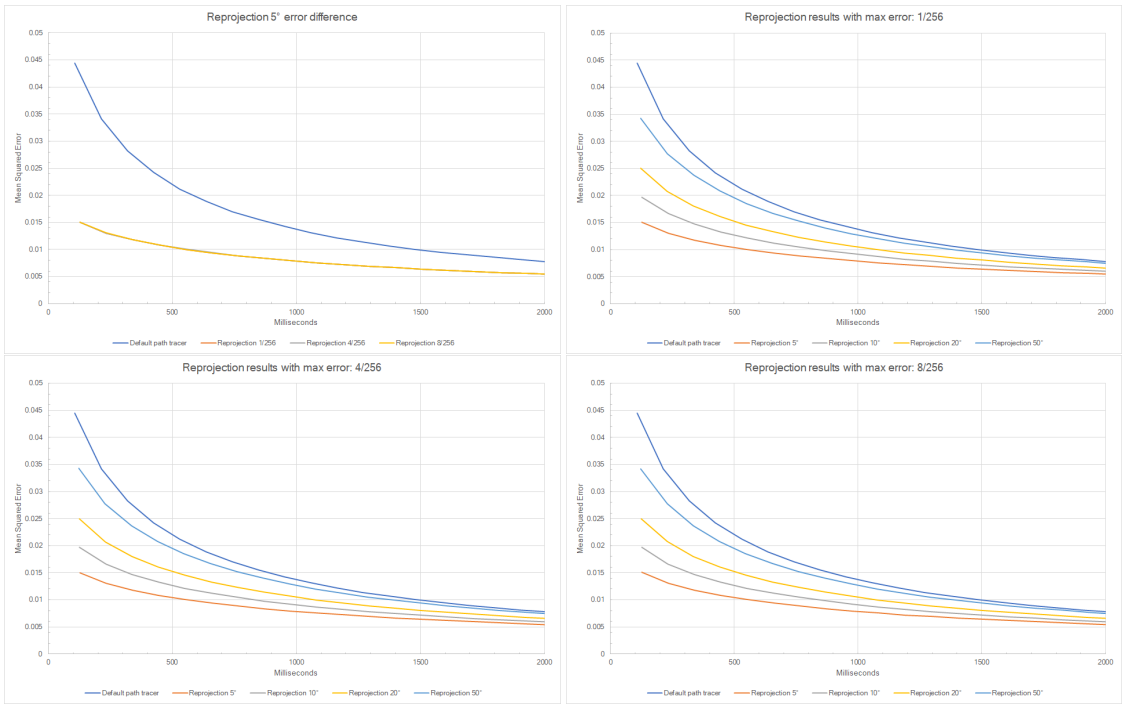
B.2.1 Results for 1spp 32sps



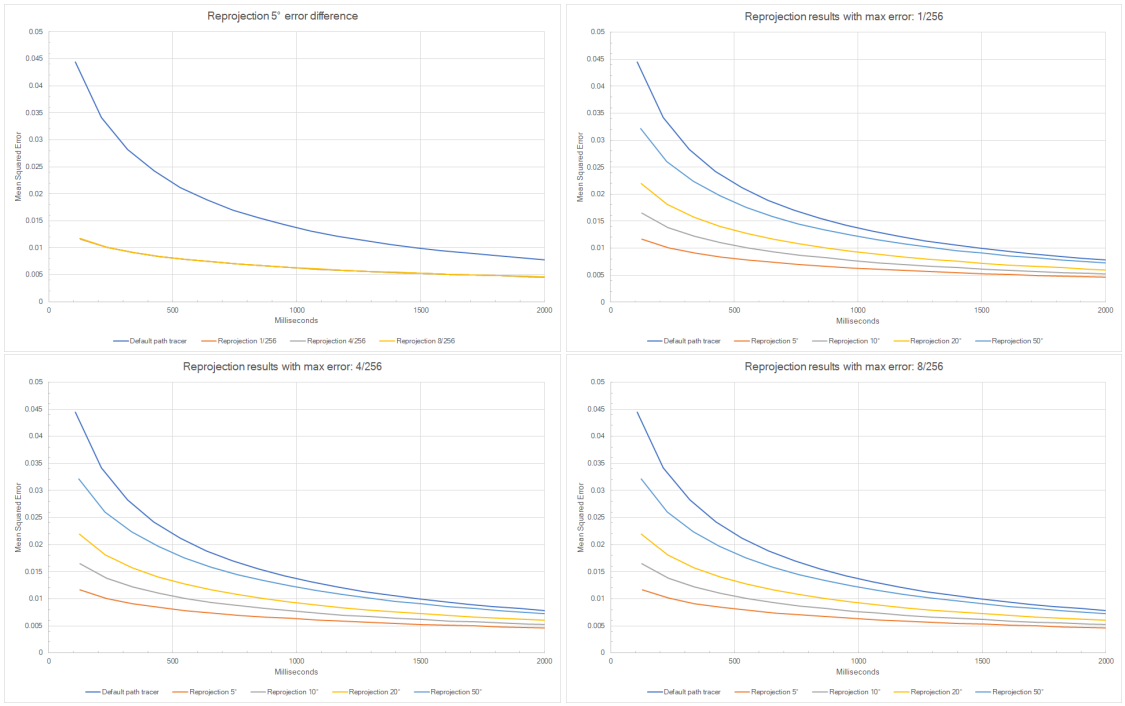
B.2.2 Results for 1spp 64sps



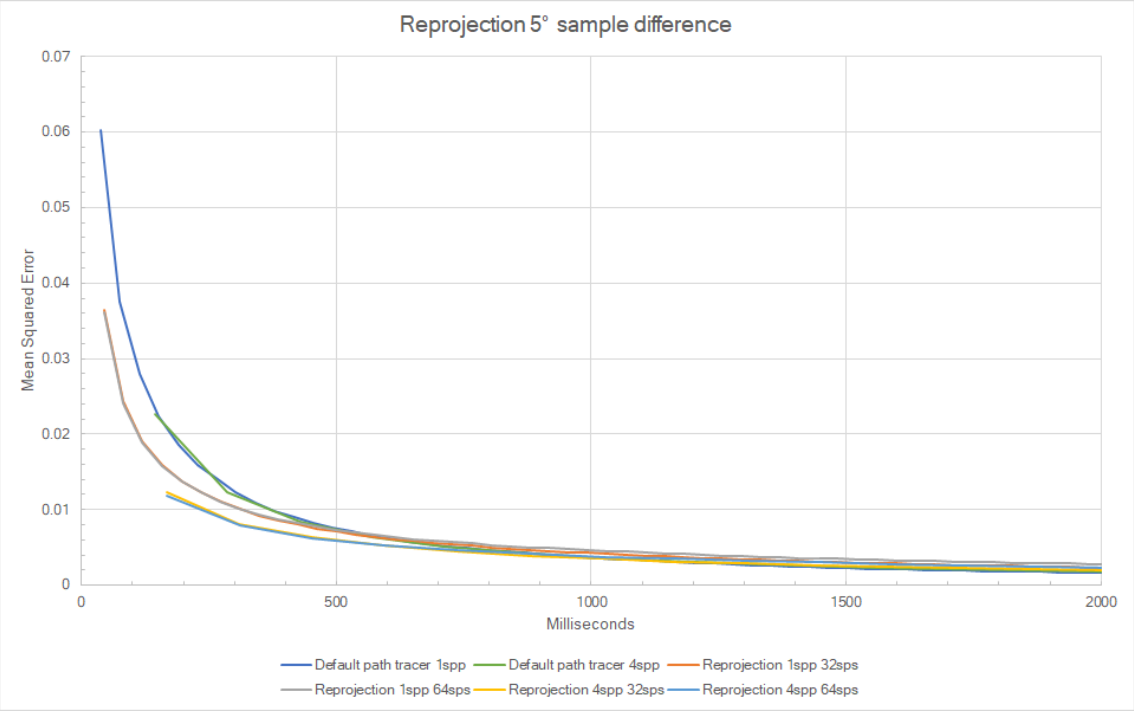
B.2.3 Results for 4spp 32sps



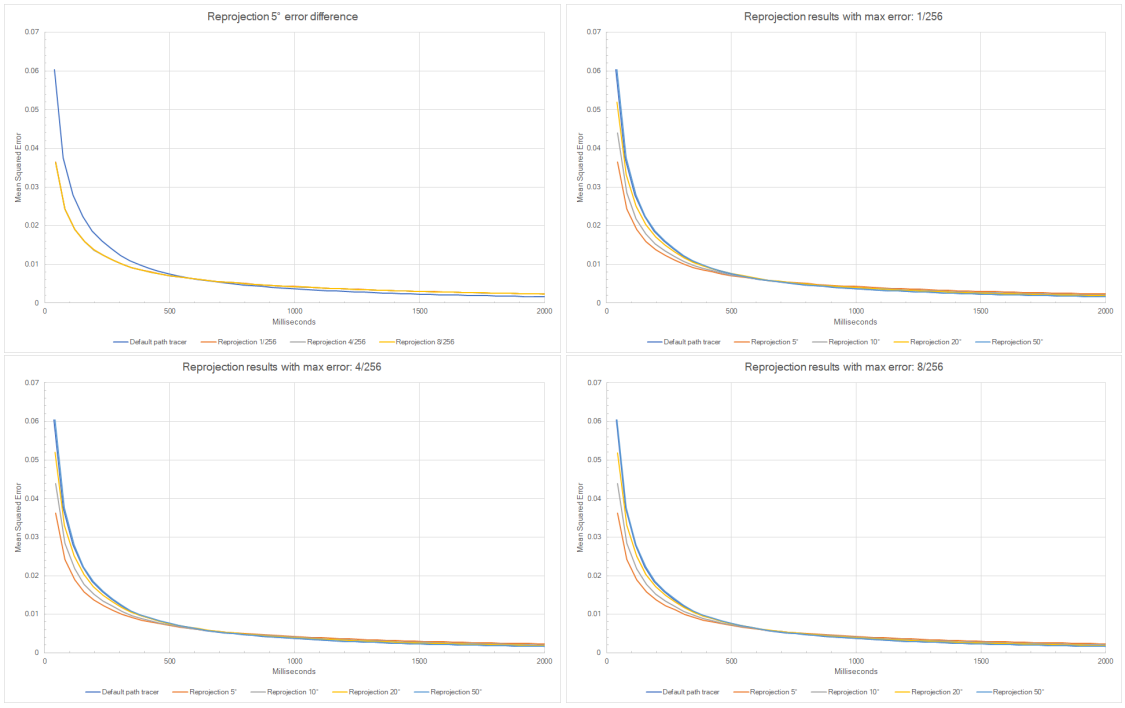
B.2.4 Results for 4spp 64sps



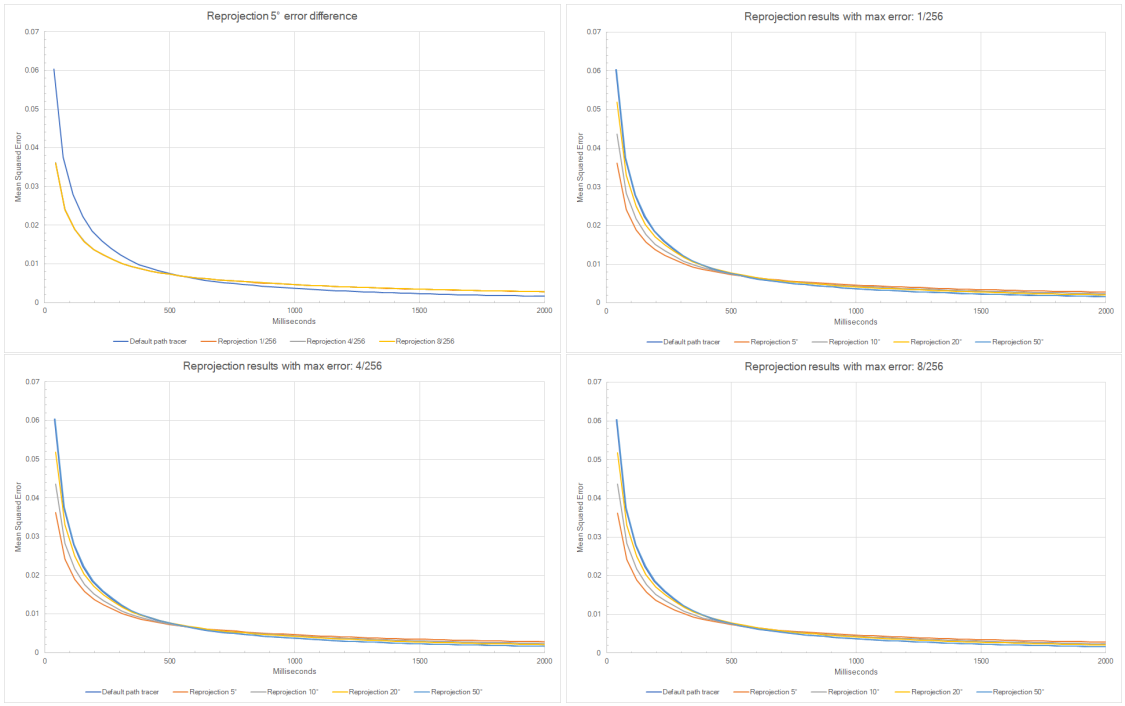
B.3 San Miguel



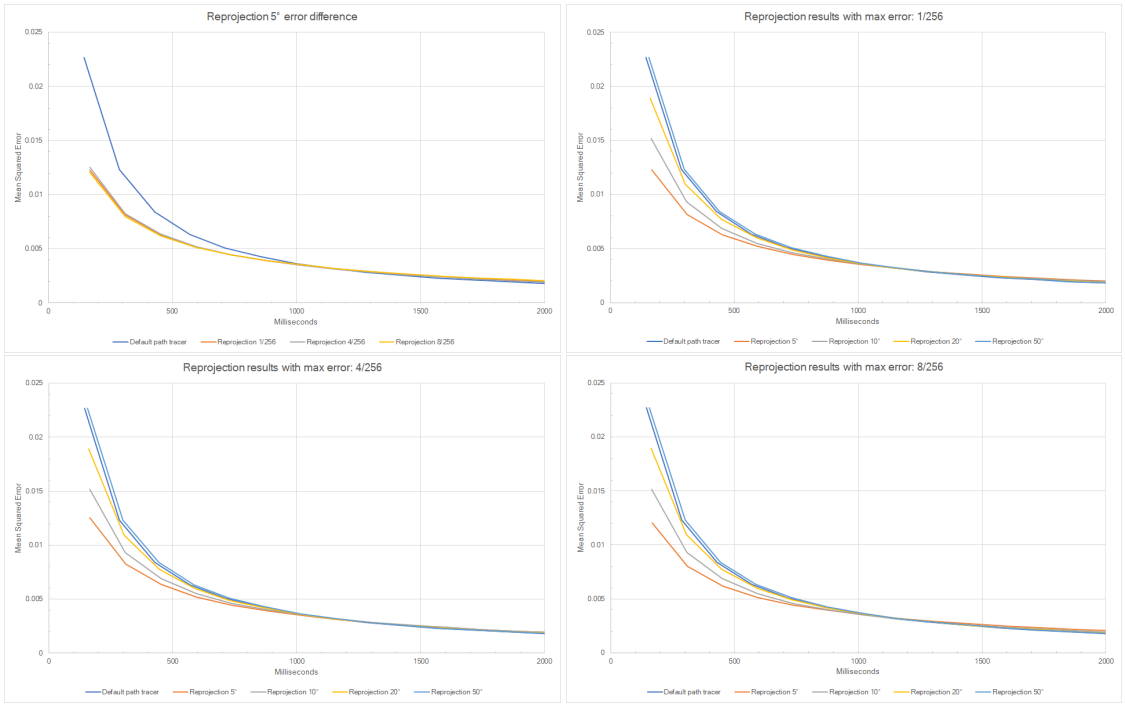
B.3.1 Results for 1spp 32sps



B.3.2 Results for 1spp 64sps



B.3.3 Results for 4spp 32sps



B.3.4 Results for 4spp 64sps