Interactive Physics-Based Rendering with AI-Accelerated Denoiser

Gonçalo Soares[1]
[1] *1Instituto Superior Técnico/Inesc-ID, Universidade de Lisboa, Lisboa, Portugal*
goncalofdsoares@tecnico.ulisboa.pt

Keywords: Nvidia RTX technology, Vulkan, Path Tracing, Bidirectional Path tracing, AI-based Denoiser

Abstract: Path tracing in real time has long been ailed has the holy grail of real time rendering. This technique, commonly used for photorealistic non-real time rendering, provides incredibly realistic graphics by simulating the physical behaviour of light, at the cost of being computationally heavy. With the introduction of Nvidia RTX-enabled GPUs family, light transport simulations started to look like a reality for real time. In this paper we describe the implementation of interactive Monte Carlo renderers, namely Path Tracing and Bidirectional Path Tracing algorithms, by using the low level Vulkan API and its RTX extension (VK_KHR_ray_tracing), recently made available by Khronos Group. Furthermore, in order to accomplish low variance rendered images, we integrated the AI-based denoiser available in the Nvidia´s OptiX framework. With such system, we were capable not only to compare the two types of light transport algorithms when used with and without the denoiser but also to answer to a relevant question: is it worthwhile to keep invest in more complex light transport algorithms, now that an interactive denoised unidirectional path tracing is available?

1 INTRODUCTION

1.1 Motivation

Real-Time Ray tracing has for long been considered the technology of the future, a far to reach dream where rendering photo-realistic images by simulating the physical behavior of light could be calculated fast enough so that we can interact with the scene and perceive the changes in real-time. This future is now possible with the introduction of hardware-accelerated ray tracing (Daniel Koch Tobias Hector and Werness, 2020) simulation provided by the NVidia RTX family of graphics cards.

Typical production renderers render an image by using a vast number of samples per pixel to render the best-looking image possible, however recent work in image reconstruction, also referred to as denoising, has opened the possibility of generating images with far fewer samples per pixel, and therefore less time, with substantial quality.

Ray Tracing is a technique used to simulate the physical interactions of the light in a scene, and over the years, multiple algorithms have been developed to solve this problem, with varying complexity, image quality results, and performance. Now that it is possible to make these simulations at never before seen rates, it is relevant to compare these different ray tracing algorithms that emerged over the years, when implemented on graphics cards, and used in conjunction with reconstruction methods.

1.2 Objectives

The main objective of our research is to create a renderer capable of implementing multiple Light Transport algorithms, with the ability to systematically compare them to each other. Furthermore, with the rise of denoising algorithms, we seek to answer how these behave with more complex algorithms such as Bidirectional Path Tracing (Lafortune and Willems, 1993) and Vertex Connection and Merging (Georgiev, 2013), verifying how a simple algorithm with a denoiser compares to the state of the art methods, and ultimately see if complex algorithms have become unnecessary. In other words, we are trying to make an application that helps find answers to the following questions:

- Can we use Ray Tracing algorithms in real-time?
- How well is a denoiser capable of reconstructing images with low samples?
- Is it worth using complex light transport algorithms over simpler ones with denoising?

2 CONTRIBUTIONS

To answer the questions above, we developed a renderer that leverages the graphics card RTX capabilities; we achieve this by utilizing the Ray Tracing extension for the Vulkan low-level graphics API (Daniel Koch Tobias Hector and Werness, 2020). Our implementation has an architecture that allows changing between different light transport, multiple scenes, and the ability to denoise the generated image either every frame or after reaching a defined number of accumulated samples or total elapsed time. This denoising is computed and accelerated by the graphics card CUDA cores exposed via the Optix Framework (Parker et al., 2010). Optix provides a function that can take as input nosy images generated by light transport algorithms with a small number of samples and output images with much higher quality, resembling images generated with many more samples. This work has a focus on making an application that facilitates the creation of different ray tracing algorithms so that it is easy to add new algorithm implementations into our working pipeline, and compare it with the previously created algorithms. We implemented two light transport algorithms for this thesis's scope: Path tracing and Bidirectional Path Tracing.

We gathered data from multiple scenes rendered with both algorithms, with and without denoising the output so that we can compare the performance and image quality of both algorithms to be able to answer the proposed questions.

3 TESTING

3.1 Evaluation Methodology

To evaluate the different algorithms we use a set of scenes, with different characteristics, e.g. reflection of caustics, glossy materials, or difficult to reach light sources, to benchmark each algorithm's strengths weaknesses, we implemented on a graphics card.

In order to evaluate the success of each algorithm, we will measure renders with varying amounts of accumulated samples with the following metrics.

3.1.1 Frames Per Second

This metric measures the number of frames a given algorithm can produce per second. We will record the average and the standard deviation of the frames per second. With this metric we gain information about the time need to render a single frame and if the algorithm can maintain a stable throughput.

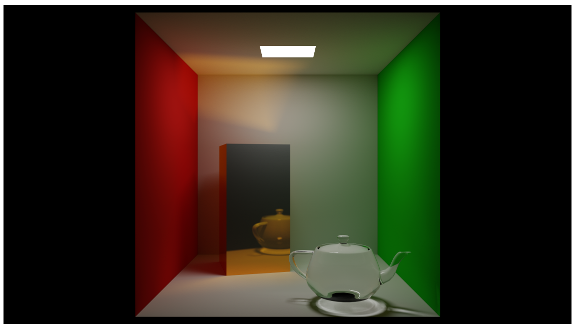
Figure 1: Teapot Cornell Box

3.1.2 Structural Dissimilarity

Structure Dissimilarity Index Metric (DSSIM) (kornelski, 2020) is an image quality metric that gives a positive value where 0 means its an identical image and $DSSIM > 0$ represents the amount of difference between the images. Structural Similarity Index Metric is one other popular metric used when comparing Images and DSSIM can be derived from SSIM (Zhou Wang et al., 2004) where $DSSIM = 1/SSIM - 1$ This metric as seen much use to compare the output of a Monte Carlo renders to its reference image.

3.1.3 Time to Converge

In this scenario we test the algorithms in a static scene and with a stationary camera, this way we can measure how long, and additionally how many iterations are necessary for a certain algorithm to achieve a defined DSSIM between the output and a reference image.

3.2 Test Scenes

For testing purposes, we chose scenes that either have been used to benchmark Ray Tracing application or are able to showcase most features of our application. These scenes are shown in Figures 1, 2, 3 and 4.

We chose this set of scenes to provide a wide range of characteristics and different complexities. The "Teapot Cornell Box" is a simple scene with very few primitives, featuring as the name suggests, the Cornell Box with a teapot, showcasing different materials, such as diffuse, glass, and colored metal. The "The Covered Dragon" utilizes the same Cornell Box, though we added a rectangle covering the light source, this scene serves to test how the implemented light transport algorithms behave when the

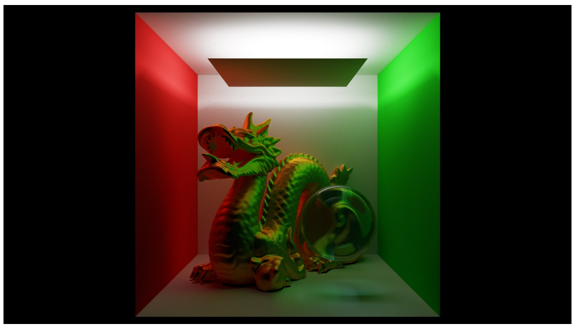
Figure 2: Covered Dragon

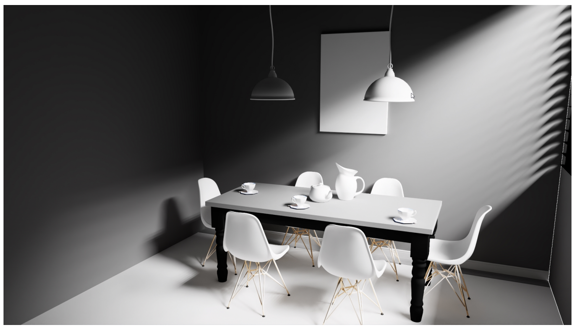
Figure 3: The Breakfast Room (Bitterli, 2016)

light sources are hard to reach. This scene also includes the Standford Dragon, composed of approximately 5,500,000 triangles, and a sphere to showcase that renderer supports custom primitives, besides triangles. "The Breakfast Room" (Bitterli, 2016) features multiple different geometries, the light source in this scene is outside the room and enter it through a window with blinders. Finally, we chose the "Japanese Classroom" (Bitterli, 2016), a commonly used scene to benchmark ray tracing applications and denoisers.

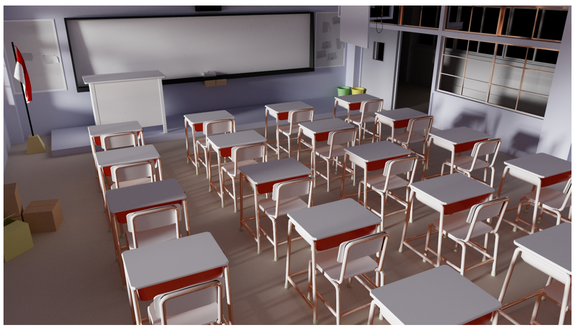
Figure 4: Japanese Classroom (Bitterli, 2016)

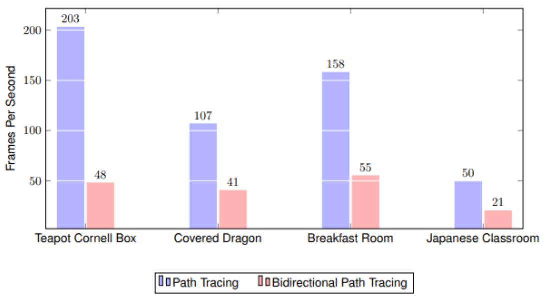

Figure 5: Average Frame Rate on 1920x1080 Resolution

4 RESULTS

This section will present and analyze the results we gathered from rendering our test scenes with both Path Tracing and Bidirectional Path Tracing. We will also see how the recursion step affects performance as well as image quality. Every test was made on the same machine utilizing the NVidia RTX 2070 graphics card, every test we made was GPU bound, so the only component that matters is the graphics card. Every frame, we trace a single path per pixel, meaning that we calculate one sample per pixel per frame. Both algorithms had a maximum path length of 32, which represents the maximum number of bounces a path can make through a scene. Appendix A contains more images rendered by our application, and Appendix B has Tables and Figures with more test results.

4.1 Performance

We start by analyzing the performance of both ray tracing algorithms; Figure 5 shows the average frames per second that the scene was being rendered at (see Table 1 for more details). Here we can see that that at the resolution of 1920x1080 Path tracing can achieve real-time rates in every scene and that Bidirectional Path Tracing is more expensive performance wise but can still achieve iterable frame rates on the scenes we tested.

The table 2 reflects the duration of denoising a single frame. The duration of the denoiser is independent of the scene and ray tracing algorithm, the only factor that affects its duration is the image resolution. For example, a denoised frame of the scene *Breakfast Room*, at the resolution of *1920x1080*, using Path Tracing takes $6.3 + 14.5 = 20.9$ milliseconds, which means that this scene runs at $1/0.0209 = 47.8$ frames per second with the denoiser running on every frame.

	Frame Duration [ms]	
	PT	BDPT
Teapot Cornell Box 1280x720	2.3	9.6
Covered Dragon 1280x720	4.6	11.3
Breakfast Room 1280x720	2.9	8.5
Japanese Classroom 1280x720	9.3	22.2
Teapot Cornell Box 1920x1080	4.8	16.7
Covered Dragon 1920x1080	9.5	24.4
Breakfast Room 1920x1080	6.3	18.6
Japanese Classroom 1920x1080	20.2	52.4

Table 1: Scene Frame Durations

Image Resolution	1280x720	1920x1080
Denoiser Duration [ms]	7.5	14.5

Table 2: Denoiser Durations

4.2 Image Quality Assessment

In this section, we take a look at the images the algorithms can produce and how they improve in quality over time, and how the application of the reconstruction step affects the image quality. The Figure 6 is the result of accumulating eight samples, meaning that every pixel only has a contribution from a maximum of eight paths. We can see that with this low sample count, the image contains a high amount of variance. For comparison Figure 7 shows the result from rendering the same scene for an equal amount of time with the Bidirectional Path Tracer, where we are only able to accumulate 4 samples. Figure 8 is the outcome of denoising Figure 6. And finally figure 9, shows a side by side comparison of these images, where we can see how the denoiser helps reducing variance, from a noisy image.

The charts depicted in figures 10 and 11 exhibit how the rendered image quality progressively increases by accumulating samples. Figure 11 is a zoom in of figure 10 regarding a time interval till 3 seconds These results come from computing the DSSIM value between a generated image and the reference for that scene, utilizing a command line tool(kornelski, 2020). Here we can see that both algorithms converge to the reference image, reflected by the DSSIM decreasing over time until getting close to 0. In this scene, we can see that Bidirectional Path Tracer produces higher

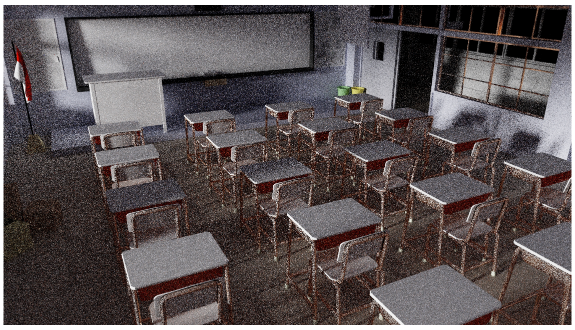
Figure 6: ”Japanese Classroom” - Path Tracing - 8 Total Accumulated Samples - Rendered in 16 milliseconds

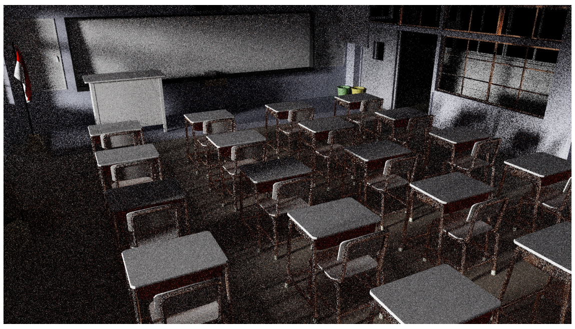
Figure 7: ”Japanese Classroom” - Bidirectional Path Tracing - 4 Total Accumulated Samples - Rendered in 16 milliseconds

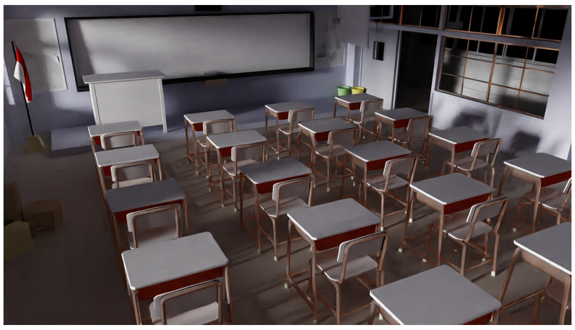
Figure 8: ”Japanese Classroom” - Path Tracing with Denoising - 8 Total Accumulated Samples - Rendered in 16 milliseconds + 14.5 milliseconds of denoising

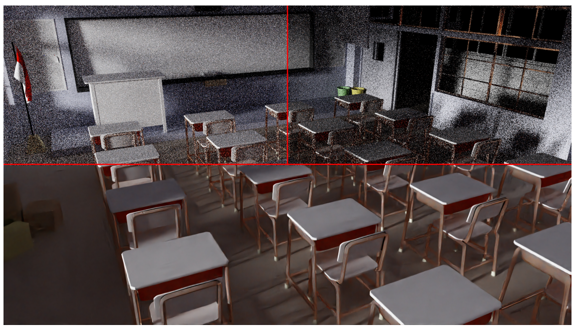
Figure 9: ”Japanese Classroom” - Side to Side comparison rendered in 16 milliseconds. Path Tracing (Top-Left), Bidirectional Path Tracing (Top-Right), Path Tracing Denoised (Bottom)

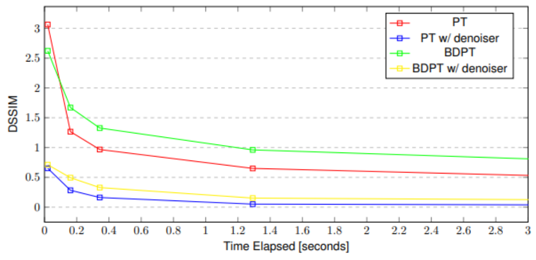

Figure 10: "Japanese Classroom" Structural Dissimilarity over Time

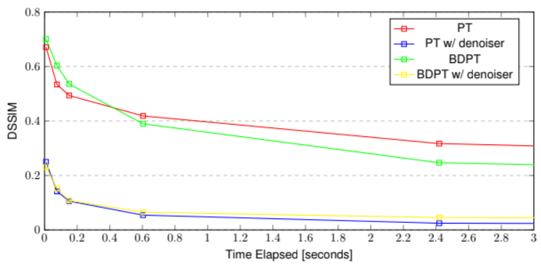

Figure 11: "Japanese Classroom" Structural Dissimilarity over Time, Zoom in of the first 3 seconds

quality images than a Path Tracer. However, a Path Tracer with Denoising converges much faster to the target image, when comparing with methods without the denoising step.

Algorithm	Time [s]	Samples	DSSIM	Denoised SSSIM
PT	0.02	1	3.0636	0.652
PT	0.161	8	1.2662	0.282
PT	0.343	16	0.9665	0.161
PT	1.292	64	0.6499	0.050
PT	5.196	256	0.3810	0.023
PT	20.773	1024	0.2092	0.014
PT	83.245	4096	0.0858	0.009
PT	331.638	16384	0.0270	0.007
BDPT	0.048	1	2.6231	0.711
BDPT	0.161	4	1.6706	0.493
BDPT	0.343	7	1.3271	0.327
BDPT	1.292	25	0.9605	0.152
BDPT	5.196	49	0.6182	0.091
BDPT	20.773	405	0.3751	0.074
BDPT	83.245	1562	0.2204	0.067
BDPT	331.638	6299	0.1245	0.064

Table 3: "Japanese Classroom" Image quality..

5 CONCLUSIONS

For most scenes, we can exceed Real-Time rates using Path Tracing, and Bidirectional Path Tracing can keep up with a Real-Time frame rate at 1920x1080 resolution, provided that though we are only calculating one path per pixel every frame. Overall, results seem to indicate that using Path Tracing in conjunction With a Denoiser is the best approach to obtain the highest image quality in the shortest amount of time. Modern denoising powered by deep learning is an excellent solution that is able to transform low sampling images to images with much higher quality. Even images with high sample counts can benefit from denoising their output, to help clear out some high variance that is very hard to converge. Complex ray tracing algorithms may still be able to outperform the simpler Path Tracer under challenging scenes, such as scenes where light sources are very hard to reach. However, for the vast majority of scenes, light sources are not that hard to intersect, so Path Tracing performs very well in these cases.

5.1 Achievements

Our work tries to pave the way for more research work of light transport algorithms by providing an application to compare implementations of said algorithms leveraging hardware acceleration by utilizing the GPU. Our main goal was to establish a base architecture so that other Monte Carlo methods could be integrated and studied. We managed to successfully implement a ray tracing renderer on the GPU capable of using a Denoiser, supporting multiple scenes and a diverse set of materials. We also implemented two light transport algorithms: Path Tracer and Bidirectional Path Tracer, that we tested extensively so that we could answer the problems we were trying to solve.

5.2 Future Work

During the development of the application, we had to compromise on this work's scope; initially, we planned to implement two more light transport algorithms: Photon Mapping and Vertex Connection and Merging. We feel confident that we built a solid base for these algorithms to be implemented on, and there are more of such algorithms that could be of interest to study when accelerated by the graphics card. The Optix denoiser is not spatiotemporally stable, meaning that if we move the camera while denoising, a flicker artifact is produced; another direction of extending our work would be implementing a spatiotemporally stable denoiser to remove this type of artifact and rendering animation in real-time.

We hope that the growth of GPU accelerated ray tracing keeps evolving so that we can afford even more rays per pixel in the near future to allow for entirely path traced games and faster 3D renderers.

REFERENCES

Bitterli, B. (2016). Rendering resources. https://benedikt-bitterli.me/resources/.

Daniel Koch Tobias Hector, J. B. and Werness, E. (2020). Ray tracing in vulkan. https://www.khronos.org/blog/ray-tracing-in-vulkan.

Georgiev, I. (2013). Implementing vertex connection and merging.

kornelski (2020). Image similarity comparison simulating human perception. https://github.com/kornelski/dssim.

Lafortune, E. P. and Willems, Y. D. (1993). Bi-directional path tracing.

Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., and Stich, M. (2010). Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics*.

Zhou Wang, Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.