Adaptive Frameless Rendering with NVIDIA OptiX

C. Hsiao, B. Watson

North Carolina State University, US

Abstract

We implement adaptive frameless rendering (AFR) on NVIDIA OptiX, a real–time ray tracing API taking advantage of NVIDIA GPUs including their latest RTX functionality. OptiX is a parallel system that sits on top of NVIDIA's better–known CUDA API. AFR has sampling and reconstruction processes that use information distributed across both space and time, aiming to generate low–latency updates. Previous AFR implementations were sequential prototypes. Our parallel prototype is allowing us to confront several unique challenges, including closed loop control of both sampling and reconstruction, and load balancing between CPU and GPU.

CCS Concepts

•***Computing methodologies*** $\rightarrow$ *Ray tracing;*

1. Introduction

As interest in and demand for virtual reality, augmented reality and large displays grows, so does the need for adaptive, low-latency rendering. However, real-time rendering continues to be dominated by non-adaptive double buffering, which avoids temporal image "tears" by buffering images offscreen, and ensuring that every pixel is at least two frames old. Some recent research tries to minimize latency in virtual and augmented reality by sidestepping the double buffering bottleneck [LPH16] [ZFH14].

Frameless rendering [BGE94] eliminates frames completely. It renders random pixels directly to the display rather than any off–screen buffer, using the latest input available. The result is a much more up–to–date but temporally incoherent image, with the incoherence visible during change as "pixel dust" and blurring. To minimize these artifacts and improve rendering quality, adaptive frameless rendering [DAD05] (AFR) concentrates samples and shapes reconstruction according to patterns of spatio–temporal change. Their method improved rendering accuracy significantly. Nevertheless, these projects were ahead of their time: they were only implemented as sequential, non–parallel prototypes; and in the following years, research and development focused on improvements readily available in more traditional framed rendering environments.

With the renewed interest in low–latency rendering, we are building a real–time, parallel version of AFR. We are implementing it using NVIDIA's OptiX ray tracing API [PSA10], a parallel API that sits on top of NVIDIA's CUDA API and takes advantage of its latest RTX ray tracing hardware.

2. Real–Time AFR Challenges

As originally described, AFR uses two parallel sampling and reconstruction subsystems, potentially introducing race conditions. However, the original authors' prototypes were sequential [DAD05], sidestepping these problems. OptiX renders rays in parallel, introducing the potential for additional parallel conflicts. It also supports distribution of computation across both the CPU and the GPU. Below, we describe these problems in more detail, along with our current solutions:

- *CPU/GPU distribution*. In AFR, adaptive sampling is controlled through a dynamically maintained hierarchy defining a tiling of the current space-time sample buffer (*deep buffer*). AFR strives to ensure that all tiles contain a roughly equal amount of color change; each of these tiles is then sampled at the same rate. Large tiles emerge over image regions with fewer spatio–temporal edges and they are sampled at lower rates. Because hierarchies are difficult to maintain on GPUs, we currently maintain the hierarchy on the CPU, and all other AFR components on the GPU. There have been a few hierarchy implementations on the GPU previously; we plan to experiment with these in the future in the effort to improve rendering speed and reduce latency.
- *Parallel conflicts between sampling and reconstruction*. To avoid any race conditions in accessing the deep buffer, our first implementation separated sampling and reconstruction into temporally disjoint phases: sampling continues until a frame must be reconstructed, then halts for reconstruction, then sampling restarts. Our current implementation runs sampling and reconstruction in parallel; early results show improvements in speed and quality.
- *Parallel conflicts in sampling*. Because OptiX samples rays in parallel, conflicts can occur when two threads access the same pixel. We avoid this with a simple locking scheme, threads that

DOI: 10.2312/egp.20191045

Sampling Threads	Sampling	Reconstruction	Total	Total FPS
256x64	6.94	34.44	41.38	24.17
256x85	8.25	35.92	44.17	22.64
256x128	10.6	37.21	47.81	20.92
256x256	20.66	38.38	59.04	16.94
256x512	36.69	38.04	74.73	13.38

Table 1: *Average sampling and reconstruction time in a scene with motion. The number of reconstruction threads is 256x256, time is measured in millisecond (ms), and FPS is 1000 divided by time.*

find a pixel locked simply sample a different pixel. Preliminary results indicate this is effective in maintaining rendering speed.

- *Avoiding parallel conflicts and barriers in reconstruction.* When sampling and reconstruction were parallel, samples may change as a filter's support is being traversed for reconstruction. Our current implementation does not use locking to avoid these inconsistencies. Preliminary results are that while these inconsistencies create minor artifacts, they may be compensated for by reduced latency. We have also begun experimenting with continuous reconstruction, with different image regions updated at different times. Splatting-based reconstruction will require special care, since it records temporally weighted samples for later reconstruction.

3. Preliminary results

Figures 1 and 2 show experiments demonstrating the quality of our output. In both experiments, during moderate view motion, we reconstruct imagery with 256x256 image resolution, using temporally disjoint sampling and reconstruction. Figure 1 shows the effect of increasing sampling threads. Edges sharpen meaningfully until there is one thread for every pixel. Table 1 shows the corresponding effect on rendering speed, which declines as additional threads are allocated to the render, and introduces a tradeoff between visual and temporal quality. These times are measured on a Windows laptop, equipped with an NVIDIA 950M. Figure 2 is a comparison between our prototype and a non-adaptive frameless renderer. We use 256x64, 256x128, 256x256 sampling threads for both renderers. The non-adaptive renderer creates deceptively sharp imagery, and leaves many mismatching older samples in each frame. Our adaptive prototype includes motion blur, and removes older samples, producing coherent and more up-to-date imagery.

4. Conclusion and Future Work

This paper describes an early, real–time implementation of adaptive frameless rendering on Nvidia OptiX, and the design challenges and choices we made in our current implementation. Our future work will focus on parallelizing sampling and reconstruction, and moving more computation to the GPU to avoid bandwidth and communication overhead. Ultimately, we plan to integrate our system into VR and AR for low-latency display, with the aid of novel displays permitting partial updates.

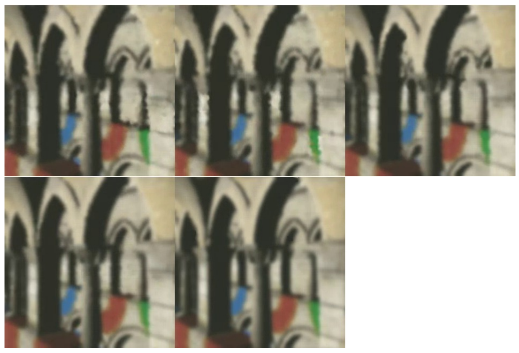

Figure 1: *Varying the number of sampling threads, to render a scene with moderate viewpoint motion. upper left: 256x64; upper mid: 256x85; upper right: 256x128; lower left: 256x256; lower mid: 256x512*

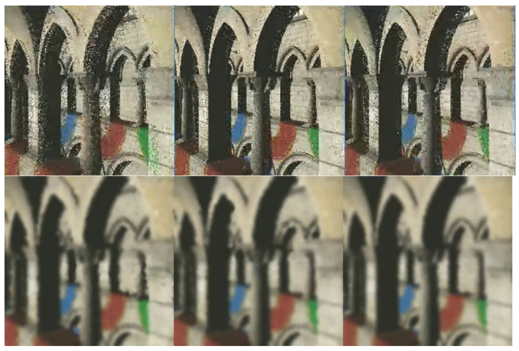

Figure 2: *Comparing our adaptive frameless rendering prototype (bottom) to a non-adaptive frameless renderer (top). The result of using differing numbers of sampling threads are shown from left to right: 256x64, 256x128, 256x256*

References

[BGE94] Bishop G Fuchs H M. L., EJS Z.: Frameless rendering: Double buffering considered harmful. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM, pp. 175–176. 1

[DAD05] Dayal A Woolley C W. B., D L.: Adaptive frameless rendering. In *Proc. Eurographics Conference on Rendering Techniques* (2005), ACM, pp. 265–275. 1

[LPH16] Lincoln P Blate A S. M. W. T. S. A. L. A., H F.: From motion to photons in 80 microseconds: Towards minimal latency for virtual and augmented reality. *IEEE transactions on visualization and computer graphics 22*, 4 (2016), 1367–1376. 1

[PSA10] Parker SG Bigler J D. A. F. H. H. J. L. D. M. D. M. M. M. K., A... R.: Optix: a general purpose ray tracing engine. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 66. 1

[ZFH14] Zheng F Whitted T L. A. L. P. S. A. M. A., H F.: Minimizing latency for augmented reality displays: Frames considered harmful. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on* (2014), IEEE, pp. 195–200. 1