Reducing Computational Complexity of Real-Time Stereoscopic Ray Tracing with Spatiotemporal Sample Reprojection

Markku Mäkitalo, Petrus Kivi, Matias Koskela and Pekka Jääskeläinen
Tampere University, Finland
{*markku.makitalo, petrus.kivi, matias.koskela, pekka.jaaskelainen*}*@tuni.fi*

Keywords: Computer Graphics, Real-Time Rendering, Ray Tracing, Stereoscopic, Temporal Coherence, Reprojection, Quality, Metrics.

Abstract: Sample reprojection is a computationally inexpensive way of increasing the quality of real-time ray tracing, where the number of samples that can be traced per pixel within the time budget is limited often to just one. Stereoscopic rendering further doubles the amount of rays to be traced, although it exhibits significant correlation not only temporally between frames, but also spatially between the viewpoints of the two eyes. We explore various reprojection schemes taking advantage of these correlations, and propose to quantify their contributions on the improved quality in terms of effective sample per pixel counts. We validate that sample reprojection is an effective way of reducing the computational complexity of real-time stereoscopic ray tracing, bringing potential benefits especially to lower-end devices.

1 INTRODUCTION

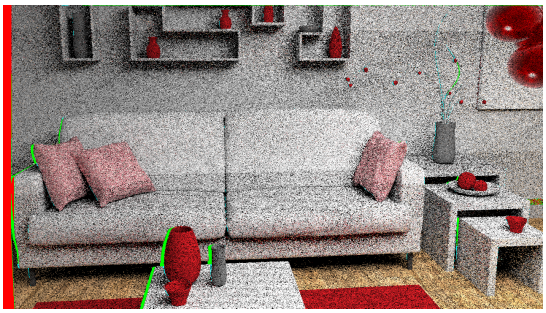
Real-time ray tracing is currently gaining significant traction, due to modern hardware and state-of-the-art software beginning to provide a means to achieve it in practice. However, we are still limited to tracing about one sample per pixel (1 spp), and rely on sophisticated post-processing filters for reconstructing an acceptable quality result. With virtual reality and augmented reality devices also becoming mainstream, enabling stereoscopic real-time ray tracing is an important challenge for the near future. However, stereoscopy also implies that the computational costs are roughly doubled, as we need to render a separate image for both eyes.

Image and video data, whether it is natural or computer generated imagery, usually exhibits correlation between the pixels in both spatial and temporal domains: the scene can contain similar structures in different parts of the same frame (spatial self-similarity), and successive frames often differ only slightly due to a small movement of the camera or the movement of objects in the scene (temporal coherence). In stereoscopic data, spatial correlation also manifests itself between the two eyes, as we are observing the scene from two different viewpoints at the same time instant.

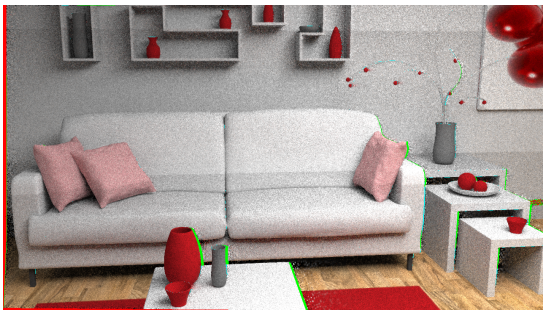
A common way to reduce the computational complexity of ray tracing, as well as rasterization, is to reproject already rendered samples into new locations, based on our knowledge of the movement and other parameters of the virtual camera. Often we can also utilize auxiliary data, such as information about the depth, normals, or world positions of the scene. We can reproject samples both temporally, i.e., using data from previous frame(s), and spatially, e.g., from the viewpoint of one eye to the viewpoint of the other eye.

Some of the challenges of spatiotemporal sample reprojection include occlusions, non-diffuse shading (i.e., a pixel whose colour may change in another viewpoint), and that the reprojection mapping is generally non-bijective (i.e., many pixels may map onto a single pixel in another viewpoint, and some pixels may not have any pixel mapped to them from another viewpoint) (Walter et al., 1999). Note that the term "viewpoint" here may indicate either a temporally different frame, or another viewing direction from a different spatial angle at the same time instant (in our case, the viewpoint of the other eye in a stereoscopic scene).

We explore both spatial and temporal reprojection, in an effort to validate their relevance and usefulness in increasing the quality of stereoscopic real-time ray tracing. We show that combining reprojection and sparse ray tracing is an effective way of reducing the computational complexity, as reprojecting a sample is a very inexpensive operation compared to tracing a ray for that sample (typically an order of magnitude



(a) Stereo



(b) Temporal

Figure 1: Pixels discarded by the reprojection algorithm in (a) stereoscopically reprojected and (b) temporally blended ray traced images: green for world position and cyan for shading normal discards. Pixels reprojected outside of the frame can be seen as red around the images.

cheaper). Moreover, we advocate a way of quantifying the obtained improvements not only by the percentage of samples where ray tracing can be omitted, but also more interestingly by the effective spp of the result: how many samples per pixel would it have taken to obtain the same quality with merely ray tracing all samples. We also argue that this evaluation method would be beneficial in bringing a more complete understanding of the speed/quality tradeoffs associated with various parts of a real-time ray tracing framework in general; it is in no way restricted to reprojection.

We specifically target the real-time 1 spp case, as it is the scenario where sample reprojection arguably provides the most tangible gains. In particular, reprojection accelerates the rendering by providing approximate fast results when the balance of speed and quality is highly important; the quality could be further enhanced by various adaptive sparse sampling schemes (Viitanen et al., 2018). In contrast, when the ray tracing is not done in real time, the gains attained through reprojection are less important: typically one would then prefer to ray trace additional high quality samples at the cost of making an already slow rendering process slower.

2 SAMPLE REPROJECTION

2.1 Temporal Reprojection

The concept of temporal coherence, that the contents of successive frames do not usually change significantly, can be traced back to at least the early 1970s, when it was discussed in conjunction with visibility determination (Sutherland et al., 1974). The following decade saw several breakthroughs in the field of ray tracing, with seminal works about recursive ray tracing (Whitted, 1979), distributed effects such as motion blur and depth of field (Cook et al., 1984), and Monte Carlo style stochastic ray tracing (*path tracing*) that computes global illumination through the rendering equation (Kajiya, 1986). Soon after, an algorithm for using temporal coherence to accelerate ray tracing for motion picture production was introduced in (Badt, 1988). Specifically, it gathered object space information from the previous frame and then estimated where the objects would be in the current frame, thus performing *forward* sample reprojection. This resulted in having to trace less than 40% of the pixels in the current frame. The forward reprojection approach was generalized in (Adelson and Hodges, 1995), where it was reported to yield up to 92% savings in rendering time.

In (Walter et al., 1999), a render cache was introduced, which enabled interactive ray tracing for low resolutions (about 8 frames per second for a 320×320 resolution). It builds an acceleration structure by caching previously rendered samples, storing their colours and also the 3D data and shading information. That allowed for various heuristics to be used when reprojecting the samples onto the new frame, such as comparing the depth data and colour contrast for detecting holes, disocclusions, and other artifacts (Wald and Slusallek, 2001). While the original render cache still used forward reprojection, its concept of storing the earlier data paved way for a *backward reprojection* cache (also known as reverse reprojection), introduced independently in (Nehab et al., 2007) and (Scherzer et al., 2007).

Backward reprojection works by starting from the current frame to be rendered instead of an earlier frame, and for each pixel in the current frame determining its location in the earlier frame. If the pixel was visible in the earlier frame and thus was stored in the cache, it can potentially be reprojected onto the current frame. Various heuristics can also be used here in deciding whether to ultimately reproject the found pixel or not, for instance based on the depth values or surface normals. While this backward mapping simplifies the reprojection and mitigates the issue of

non-bijectivity, it does bring additional requirements in terms of storing and handling past data in memory.

For further details and history about temporal coherence and reprojection, as well as for reviewing a multitude of other ways of accelerating ray tracing to an interactive/real-time level, we refer the reader to (Wald and Slusallek, 2001) and (Scherzer et al., 2012).

Nowadays reprojection is used by modern real-time denoisers for stochastic sampling (Koskela et al., 2019; Schied et al., 2018; Schied et al., 2017; Mara et al., 2017), in order to generate a better quality input before the actual denoising filter. Moreover, reprojection is used extensively in rasterized game graphics. One of the most commonly used reprojection methods is called Temporal Anti-Aliasing (TAA) (Karis, 2014), which generates an anti-aliased image without extra spatial samples. Instead, the temporal samples are reprojected and used for smoothing the edges in the image. With TAA, the camera is typically sub-pixel jittered with a Halton sequence for achieving the same smoothing effect even with a static camera.

2.2 Stereo Reprojection

In stereo reprojection, we reproject samples from one spatial viewpoint to another at the same time instant. In the standard stereoscopic case, the eyes can be thought of as two distinct cameras that are separated by approximately 6.5 centimeters (the eye separation of an average human), and we reproject samples from one camera's viewpoint to the other.

There are two well-established methods for setting up stereoscopic cameras: a *parallel* or *sensor-shift* camera setup, where the two virtual cameras are translated only horizontally, and a *converged* or *toe-in* camera setup, which additionally introduces a slight inward rotation for convergence. Converged camera setups have been shown to produce visual distortions such as *keystoning* (Woods et al., 1993), and thus parallel stereoscopic cameras are preferred for viewing comfort (Allison, 2007).

Stereo reprojection can be thought of as a special case of temporal reprojection (Adelson and Hodges, 1993). Specifically, if there is only camera movement and the scene remains otherwise static, there is no fundamental difference whether the camera movement is interpreted as a change in viewpoint spatially or temporally.

An early example of stereo reprojection (Ezell and Hodges, 1990) built on the work of (Badt, 1988) and combined it with a calculation of stereo disparity information in order to do the reprojection. With their method, between 20% and 50% of the pixels in the target frame needed to be ray traced after reprojection. A more optimized method (Adelson and Hodges, 1993) made a simplifying observation that they could only reproject the x-coordinates of the samples, based on an assumption that the observer's eyes are level (i.e., their head is not tilted). They obtain an estimate of 93% reduction in the amount of rays that need to be traced for the second eye, albeit they do not address all problems related to the disparity between the two views (Ip et al., 1997).

A more general approach that does not assume horizontally level eyes, introduced in (Ip et al., 1997), leveraged coherence in the epipolar geometry of a stereo image pair, subdividing the space with epipolar planes; their algorithm ran in 30–50% of the time of their comparison algorithm.

Reprojection based methods can also support animated rigid objects (Rosado, 2007). If there is a way to compute the screen space motion vector for an animated object, it can be both forward and backward reprojected. The motion vectors can be computed for common rigid body animations such as translation, rotation and scaling.

With the modern rise in popularity of 3D content, stereoscopic ray tracing and reprojection algorithms have also evolved to cover more general multi-view synthesis; see, e.g., (Andersson et al., 2011) and the references therein. However, as even monoscopic real-time ray tracing is only now becoming tractable, enabling stereoscopic real-time ray tracing remains a challenge for the near future. The recent NVIDIA Turing architecture offers hardware acceleration for rendering up to four views in a single render pass, allowing discrepancy between the eyes also in the y-coordinate, whereas the acceleration introduced in their earlier Pascal architecture only supported two views (implying a narrower field of view) and discrepancy only in the x-coordinate, assuming horizontally level eyes (Bhonde and Shanmugam, 2018). However, its potential in combination with their ray tracing hardware is not detailed. Nevertheless, as we highlight in this paper, sample reprojection provides one practical way of reducing the computational complexity of stereoscopic real-time ray tracing.

3 QUALITY EVALUATION

As seen in Section 2, the advantage of sample reprojection is often expressed in terms of the percentage of saved rendering time, or especially in ray tracing, as the percentage of reduction in the amount of traced rays. While such statistics are useful and descriptive, they do not convey information about the quality of

the rendered result as such. We feel that the quality aspect is generally overlooked whenever such figures are reported, which can be due to several reasons.

First, when ray tracing was still exclusively a method for non-real-time rendering, sample reprojection was applied simply to make the computation time tolerable in the first place, but the long rendering process was still continued until the resulting frame was of high enough quality. Second, when sample reprojection is used within the rasterization pipeline, we do not face a similar issue as in the path tracing approach, where individual samples are noisy and we need to accumulate samples in order to decrease the amount of noise.

However, even in the recent literature concerning real-time ray tracing, we have not seen a rigorous evaluation of how much sample reprojection actually contributes to the quality of the final ray traced result, despite temporal reprojection being a relatively widely used building block of real-time ray tracing frameworks. In particular, we think that an intuitive metric for measuring the quality improvement gained through reprojection is to estimate what the effective spp count of the reprojected result is. The effective spp gives us simultaneously an indication of both the quality and the reduction in computational complexity.

To outline our approach, we reproject as many samples onto the target frame as we can, and ray trace only the missing samples where reprojection was deemed unsuccessful. The target frame can have existing data, on top of which we accumulate reprojected data and blend them together with a certain ratio (e.g., for stable temporal accumulation, but stereo accumulation can also be done), or the target frame can be empty (e.g., reprojecting from one eye to another). Then, through comparisons to a pristine reference that is ray traced with thousands of samples per pixel, we can compute the spp of a purely ray traced image that would effectively yield the same error as the reprojected image. The error can be evaluated by common error metrics, such as root mean square error (RMSE), or the structural similarity index (SSIM) (Wang et al., 2004), or more sophisticated metrics.

While it can be argued that RMSE and SSIM do not adequately correspond to the complex subjective evaluation done in the human visual system, they persist as widely used metrics due to their relative simplicity. Moreover, we think that the choice of the metric is not critical in our case, as we aim to measure the relative differences between different quality images instead of concentrating on absolute quantities indicated by the metrics.

4 EXPERIMENTS

We evaluate the effects of spatiotemporal sample reprojection for seven stereoscopic test scenes, each having 60 frames with 720p resolution for each eye, and a moving observer (i.e., moving camera). As a baseline, we have three diffuse scenes (Sponza, Living Room, and Classroom) with static lighting. In order to consider more varied conditions, we also investigate the Sponza and Living Room scenes with glossy materials, and with a moving light source. Figure 2 shows an example 4096 spp reference frame of each static light scene; the moving light versions of Sponza and Living Room are done on top the scenes in Figure 2a and Figure 2b, respectively.

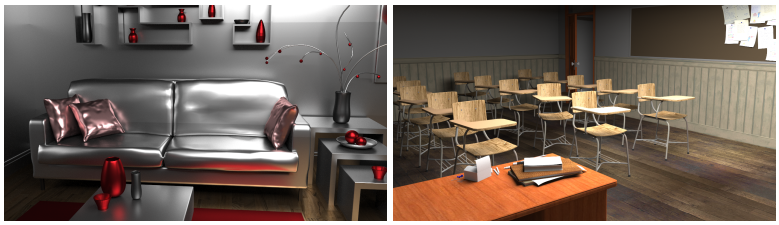
4.1 Implementation

In the stereo reprojection for each individual frame, we first path trace the complete right-eye image with 1 spp (using one primary ray, one secondary ray, and two shadow rays). At this point, we do not have any path traced data for the left-eye image. However, we do assume that we have access to the depth and shading normal buffers of both eyes via the renderer, so that we can choose which pixels to discard in the reprojection step in order to avoid ghosting artifacts. Specifically, we sample the depths and shading normals of both current and previous frame, and also compute the 3D world space positions from these depths, which can be done since we know the camera parameters of both frames. Then the differences of the two world positions and shading normals are compared. Higher differences than scene-specific limit values mean that there has been an occlusion and we need to discard the history data; the current 1 spp path traced input is then directly used as the output of those pixels. The scene-specific limits used in our measurements were each decided based on the scale of the particular scene.

Then, we want to reproject as much of the right-eye data as possible into the thus far empty left-eye image, so we apply backward reprojection for each pixel in the target left-eye image, hence filling $n\%$ of the left-eye image with spatially reprojected 1 spp data. The missing $(100-n)\%$ of the pixels in the left-eye image, where the reprojected sample was discarded, are finally path traced with 1 spp. Note that the difference in reprojecting from right eye to left eye or vice versa is trivial; our choice of reprojecting from right eye to left eye is arbitrary. Mixing both reprojection directions could also be considered for a more advanced implementation.

For temporal reprojection, we accumulate data

(a) Sponza (b) Living Room (c) Sponza (Glossy)



(d) Living Room (Glossy) (e) Classroom

Figure 2: 4096 spp references of the scenes used in the experiments. Scenes (a) and (b) are also used as a base for Sponza (Moving Light) and Living Room (Moving Light), respectively.

from earlier frames with a 20% blending ratio, i.e., weighing the existing accumulated data with a factor of 0.8, and combining it with the newest reprojected frame with a weight factor of 0.2. We found this ratio to provide a good balance between keeping the frame sequence temporally stable and incorporating newly acquired data, while also keeping the implementation simple. However, a content-aware heuristic for deciding the ratio per scene could also be considered, as well as one dependent on the camera movement.

We measure the effective spp counts as detailed in Section 3 in terms of RMSE; for the RMSE value obtained for each frame, we find the n and $n+1$ spp datasets, between whose corresponding RMSE values the obtained RMSE lies, and interpolate the effective spp based on that. The measurements are done for stereo reprojection and temporal reprojection separately, and also with both reprojection methods combined; in the combined case, we first apply temporal reprojection to the right eye, after which the stereo reprojection is done from right eye to left eye, and the resulting left-eye quality is evaluated. Note that in this case we also need to temporally accumulate left-eye data in order to fill the discarded pixels with accumulated data.

While backward reprojecting data from the right eye to the left eye, we use *bilinear interpolation* to decide how the four closest pixels in the right eye contribute to the reprojected pixels in the left eye, as the pixels don't align perfectly after the reprojection. We discard invalid samples based on depth and shading normal values and give weights to the remaining samples according to their linear distance from the reprojected left eye pixel's center. We use the same sampling scheme for temporal reprojection, where the data is now backward reprojected from the current right eye viewpoint to a future right eye viewpoint. We found bilinear interpolation to give visually and metrically more pleasing results compared to *nearest neighbour* sampling, where only the closest right eye pixel is sampled. This is due to the fact that bilinear interpolation acts as a simple denoiser against the stochastic noise introduced by path tracing.

4.2 Results

Table 2 presents the effective spp counts for all datasets, and for all three reprojection options (stereo, temporal, and stereo + temporal). The average spp is taken over all 60 frames, and the framewise minimum and maximum spp counts are also reported. For stereo reprojection, we obtain approximately 1.6 spp for all datasets, with minor variance between frames.

For temporal reprojection, the results are more dependent on the scene contents and the particular camera movements: the average spp counts range from approximately 7 spp to 12.5 spp, which is a significant improvement in quality even in the worst case. In general, we see that the Sponza scenes are more difficult, which is likely due to them having more complex geometry in the new data entering the frame as the camera moves. The minimum effective spp is 1, as there is no previous data to reproject on the first frame. It takes about 5–10 frames for the temporal reprojection to initially reach a stable spp, as seen also in the framewise spp graph for Sponza in Figure 3; the convergence speed is dependent on the temporal blending ratio. The combined stereo+temporal reprojection

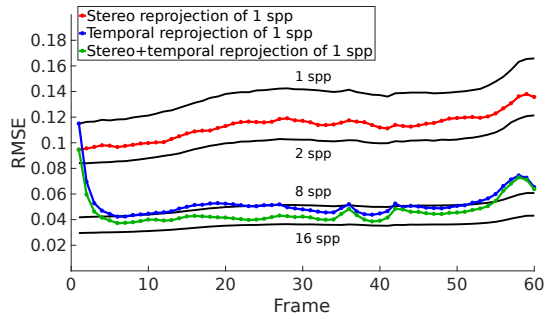


Figure 3: Sponza dataset: Framewise RMSE values of the reprojected frames, and of comparison data having 1 spp, 2 spp, 8 spp and 16 spp.

expectedly yields the best spp counts for all datasets, with averages ranging from about 9 spp to 18 spp. These clearly demonstrate that reprojection is a powerful method despite its simplicity. However, more research is needed for evaluating the performance in more dynamic conditions, such as in a virtual reality environment with the user constantly looking around in the scene; in our datasets, the camera movement is either smoothly curved panning (Living Room and Classroom), or smoothly curved backward movement (Sponza).

Moreover, Table 1 shows how many percent of the samples were discarded on average; those samples are the ones that need to be path traced in the target frame after the reprojection. Thus, we see that with stereo reprojection, only 2–4% of the samples have to be path traced for the left eye on average, while the resulting quality is still 60% higher at 1.6 spp than what normal 1 spp path tracing would yield. For temporal reprojection, we observe discard rates of 1.5–4%; again, we notice that more samples are discarded in the Sponza sets due to the complex geometry. Figure 1 visualizes the discarded pixels for an example frame of Living Room.

Figure 4 shows an example frame comparison of Classroom with the 1 spp original data, 4096 spp reference, 16 spp reference, and the reprojected result with an effective 16.38 spp. We see that the quality of the reprojection matches the 16 spp reference quite well also visually, although it exhibits some artifacts on the rightmost part where new data has just entered the frame and temporal accumulation has not yet had enough data to smooth it out.

5 CONCLUSIONS

We explored stereoscopic and temporal sample reprojection in the context of real-time ray tracing. We investigated the quality improvements and the reduction in computational complexity associated with reprojection, and proposed to use the effective spp count as a descriptive metric for them. To substantiate our standpoint, we provided experimental results quantifying the relative contributions of reprojection in terms of effective spp. The metric and the concept of effective spp is not limited to reprojection, but could also be used to more rigorously assess the relative contributions of other components in a real-time ray tracing framework. We think the proposed approach would be beneficial in evaluating these various components and understanding the compromises between speed and quality associated with them, and thus being able to make more informed choices when designing and optimizing the frameworks.

In practical real-time applications, reprojection is typically combined with more sophisticated denoising filters for high quality results. The rendering could be further accelerated with adaptive sparse sampling schemes. Such topics remain a subject of future investigation, as well as considering adaptive blending ratios, especially for fast-paced scenarios like games.

Table 1: Percentage of discarded pixels after temporal and stereoscopic reprojection with depth, world position and shading normal checking. The average, minimum and maximum for stereo (N=60 frames) and temporal (N=59 frames) discard percentages are shown.

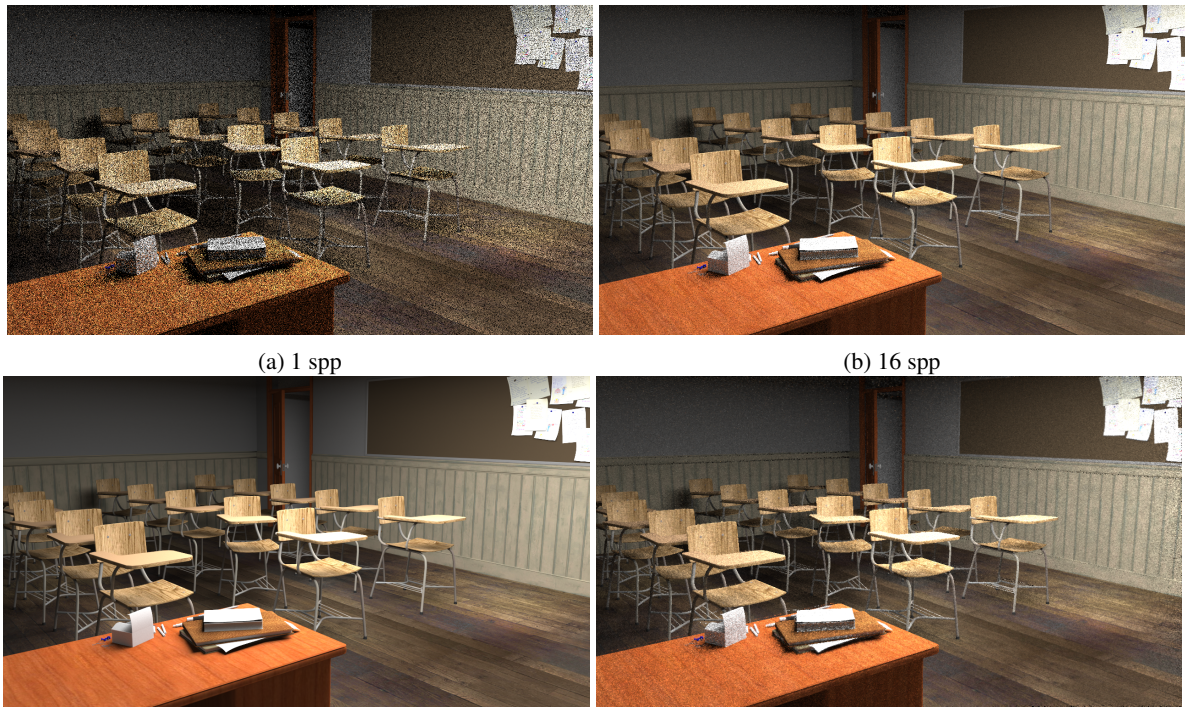
	Stereo		
	Avg	Min	Max
Sponza (All scenes)	**2.01**	1.24	2.96
Living Room (All scenes)	**2.50**	2.31	2.72
Classroom	**4.36**	3.58	5.91
	Temporal		
	Avg	Min	Max
Sponza (All scenes)	**3.96**	3.57	4.46
Living Room (All scenes)	**1.48**	0.70	2.27
Classroom	**2.19**	1.52	2.97

ACKNOWLEDGEMENTS

This work is supported by Business Finland (funding decision 40142/14, FiDiPro-StreamPro), Academy of Finland (funding decisions 297548, 310411), ECSEL JU project FitOptiVis (project number 783162), and the Tampere University of Technology Graduate School.

REFERENCES

Adelson, S. J. and Hodges, L. F. (1993). Stereoscopic ray-tracing. *The Visual Computer*, 10(3):127–144.



(a) 1 spp (b) 16 spp

(c) 4096 spp (d) Reprojected

Figure 4: Different spp counts (a,b,c) in the Classroom scene compared to the stereoscopically and temporally reprojected and blended 1 spp (d), whose effective spp is 16.38 spp.

Table 2: Effective spp after three different reprojection schemes, as measured by RMSE. The average spp is taken over all 60 frames, with the respective framewise minimum and maximum spp counts also shown.

	Stereo			**Temporal**			**Stereo + temporal**		
	Avg	Min	Max	**Avg**	Min	Max	**Avg**	Min	Max
Sponza	**1.63**	1.55	1.71	**7.69**	1.00	10.44	**10.07**	1.66	13.48
Living Room	**1.61**	1.55	1.64	**12.76**	1.00	14.36	**18.22**	1.62	20.94
Sponza (Glossy)	**1.58**	1.33	1.82	**7.97**	1.00	11.25	**10.09**	1.68	15.24
Living Room (Glossy)	**1.63**	1.48	1.69	**11.27**	1.00	12.93	**16.63**	1.61	19.80
Sponza (Moving Light)	**1.59**	1.48	1.65	**6.92**	1.00	8.82	**8.65**	1.57	11.59
Living Room (Moving Light)	**1.61**	1.55	1.65	**11.40**	1.00	12.83	**15.45**	1.62	17.98
Classroom	**1.60**	1.54	1.64	**10.83**	1.00	14.30	**16.50**	1.58	19.55

Adelson, S. J. and Hodges, L. F. (1995). Generating exact ray-traced animation frames by reprojection. *IEEE Computer Graphics and Applications*, 15(3):43–52.

Allison, R. (2007). Analysis of the influence of vertical disparities arising in toed-in stereoscopic cameras. *Journal of Imaging Science and Technology*, 51(4):317–327.

Andersson, M., Johnsson, B., Munkberg, J., Clarberg, P., Hasselgren, J., and Akenine-Möller, T. (2011). Efficient multi-view ray tracing using edge detection and shader reuse. *The Visual Computer*, 27(6-8):665.

Badt, S. (1988). Two algorithms for taking advantage of temporal coherence in ray tracing. *The Visual Computer*, 4(3):123–132.

Bhonde, S. and Shanmugam, M. (2018). Turing Multi-View Rendering in VRWorks. https://devblogs.nvidia.com/turing-multi-view-rendering-vrworks Accessed 16th of January 2019.

Cook, R. L., Porter, T., and Carpenter, L. (1984). Distributed ray tracing. In *ACM SIGGRAPH computer graphics*, volume 18, pages 137–145. ACM.

Ezell, J. D. and Hodges, L. F. (1990). Some preliminary results on using spatial locality to speed up ray tracing of stereoscopic images. In *Stereoscopic Displays and Applications*, volume 1256, pages 298–307. International Society for Optics and Photonics.

Ip, H. H., Law, K. C., and Fung, G. K. (1997). Epipolar plane space subdivision method in stereoscopic ray tracing. *The Visual Computer*, 13(6):247–264.

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

Karis, B. (2014). High-quality temporal supersampling. Advances in Real-Time Rendering in Games, SIGGRAPH Courses.

Koskela, M., Immonen, K., Mäkitalo, M., Foi, A., Viitanen, T., Jääskeläinen, P., Kultala, H., and Takala, J. (2019). Blockwise multi-order feature regression for real-time path tracing reconstruction. *Provisionally accepted to ACM Transactions on Graphics (TOG)*.

Mara, M., McGuire, M., Bitterli, B., and Jarosz, W. (2017). An efficient denoising algorithm for global illumination. In *Proceedings of High Performace Graphics*. 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.

Rosado, G. (2007). *GPU Gems 3*, chapter 27. Motion Blur as a Post-Processing Effect. Addison-Wesley Professional.

Scherzer, D., Jeschke, S., and Wimmer, M. (2007). Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 45–50. Eurographics Association.

Scherzer, D., Yang, L., Mattausch, O., Nehab, D., Sander, P. V., Wimmer, M., and Eisemann, E. (2012). Temporal coherence methods in real-time rendering. In *Computer Graphics Forum*, volume 31, pages 2378–2408. Wiley Online Library.

Schied, C., Kaplanyan, A., Wyman, C., Patney, A., Chaitanya, C. R. A., Burgess, J., Liu, S., Dachsbacher, C., Lefohn, A., and Salvi, M. (2017). Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*, page 2. ACM.

Schied, C., Peters, C., and Dachsbacher, C. (2018). Gradient estimation for real-time adaptive temporal filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(2):24.

Sutherland, I. E., Sproull, R. F., and Schumacker, R. A. (1974). A characterization of ten hidden-surface algorithms. *ACM Computing Surveys (CSUR)*, 6(1):1–55.

Viitanen, T., Koskela, M., Immonen, K., Mäkitalo, M., Jääskeläinen, P., and Takala, J. (2018). Sparse sampling for real-time ray tracing. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 1, pages 295–302.

Wald, I. and Slusallek, P. (2001). State of the art in interactive ray tracing. *State of the Art Reports, EUROGRAPHICS*, 2001:21–42.

Walter, B., Drettakis, G., and Parker, S. (1999). Interactive Rendering using the Render Cache. In Lischinski, D. and Larson, G., editors, *Rendering techniques '99 (Proceedings of the 10th Eurographics Workshop on Rendering)*, volume 10, pages 235–246, Granada, Spain. Springer-Verlag/Wien.

Wang, Z., 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.

Whitted, T. (1979). An improved illumination model for shaded display. In *ACM SIGGRAPH Computer Graphics*, volume 13, page 14. ACM.

Woods, A. J., Docherty, T., and Koch, R. (1993). Image distortions in stereoscopic video systems. In *Stereoscopic displays and applications IV*, volume 1915, pages 36–49. International Society for Optics and Photonics.