Subpixel Reconstruction Antialiasing for Ray Tracing

Chiu, Y.-F
National Tsing Hua University, Taiwan
yfchiu@ibr.cs.nthu.edu.tw

Chen, Y.-C
National Tsing Hua University, Taiwan
louis@ibr.cs.nthu.edu.tw

Chang, C.-F
National Taiwan Normal University, Taiwan
chunfa@ntnu.edu.tw

Lee, R.-R
National Tsing Hua University, Taiwan
rrlee@cs.nthu.edu.tw

ABSTRACT

We introduce a practical antialiasing approach for interactive ray tracing and path tracing. Our method is inspired by the Subpixel Reconstruction Antialiasing (SRAA) method which separates the shading from visibility and geometry sampling to produce antialiased images at reduced cost. While SRAA is designed for GPU-based deferred shading renderer, we extend the concept to ray-tracing based applications. We take a hybrid rendering approach in which we add a GPU rasterization step to produce the depth and normal buffers with subpixel resolution. By utilizing those extra buffers, we are able to produce antialiased ray traced images without incurring performance penalty of tracing additional primary rays. Furthermore, we go beyond the primary rays and achieve antialiasing for shadow rays and reflective rays as well.

Keywords: antialiasing, ray tracing, path tracing.

1 INTRODUCTION

With the abundance of computation power and parallelism in multicore microprocessors (CPU) and graphics processors (GPU), achieving interactive photorealistic rendering on personal computers is no longer a fantasy. Recently, we have seen the demonstration of real-time ray tracing [6, 17] and the emergence of real-time path tracing with sophisticated global illumination [2, 20]. Though real-time path tracing can produce rendering of photorealistic quality that include complex lighting effects such as indirect lighting and soft shadow, the illusion of a photograph-like image breaks down quickly when jaggy edges are visible (Figure 1 shows an example).

Jaggy edges are one of the typical aliasing artifacts in computer generated images. A straightforward antialiasing technique is to increase the sampling rate by taking multiple samples uniformly at various subpixel positions. However this approach induces significant performance penalty that makes it an afterthought in real-time ray tracing. A more practical approach is to increase subpixel samples adaptively for image pixels where discontinuity is detected. Although adaptive sampling approach avoids the huge performance hit of the multisampling approach, it still requires additional subpixel samples and introduces large variation to the estimation of rendering time.

In this work, we introduce an antialiasing approach that works well for real-time ray tracing and path tracing. We take a hybrid rendering approach in which we add a GPU rasterization step to produce the depth and normal buffers with subpixel resolution. By utilizing those extra buffers, we are able to produce antialiased ray traced images without incurring performance penalty of tracing additional primary rays. Our method is inspired by the Subpixel Reconstruction Antialiasing (SRAA) [3] which combines per-pixel shading with subpixel visibility to produce antialiased images. While SRAA is designed for GPU-based deferred shading renderer, we extend the concept to ray-tracing based applications. Furthermore, we apply our antialiasing approach to shadow and reflection which SRAA cannot resolve with its subpixel buffers.

Our main contributions in this work are:

- We propose an efficient antialiasing technique which improves the perception of photorealism in interactive or real-time ray tracing without sacrificing its performance.
- Unlike adaptive sampling or subpixel sampling, our approach does not penalize the performance of a CPU ray tracer because no additional primary ray needs to be traced. Our hybrid rendering approach obtains the necessary subpixel geometric information by leveraging the GPU rasterization pipeline.
- While SRAA works well for improving the sampling on image plane, we extend its application beyond the primary rays and achieve antialiasing for shadow rays and reflective rays as well.

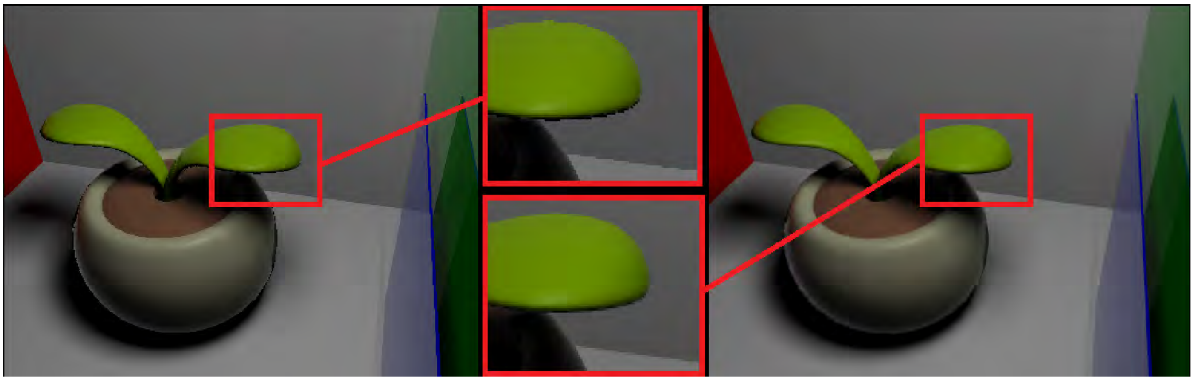

Figure 1: Antialiasing can improve the rendering quality in interactive ray tracing. The left image is rendered without applying any antialiasing method. The right image is rendered with our method using relatively inexpensive geometric information to improve expensive shading results.

2 RELATED WORK

2.1 Real-time Ray Tracing

With the rapid improvement of computation power and parallelism in multicore microprocessors (CPU) and graphics processors (GPU), various works have been published on speeding up the ray tracing, either on CPUs [1, 19], GPUs [5], or special-purpose platforms [21]. Recently, we have seen the demonstration of real-time ray tracing with Whitted-style reflection and refraction (a.k.a. specular rays) [6, 17] and the emergence of real-time path tracing with sophisticated global illumination [2, 20]. The NVIDIA OptiX acceleration engine [12] elevates rendering applications to a new level of interactive realism by greatly increasing ray tracing speeds with GPU solutions. While real-time ray tracing is now feasible, most renderers still rely on Monte Carlo path tracing to obtain more sophisticated global illumination effects such as soft shadow and indirect lighting. Noisy preview images are usually produced first at interactive rates and then gradually converge to high quality images. Therefore, antialiasing often becomes an afterthought as it further slows down the rendering.

2.2 Adaptive Sampling

The ray tracing algorithm is basically a loop over all screen pixels to find the nearest visible object in the scene. We can consider ray tracing as a point sampling based rendering method in signal processing view. However, point sampling makes an all-or-nothing choice in each pixel and thus leads to jaggies. Antialiasing of ray-traced images could be achieved by supersampling the image. However the supersampling approach demands significantly larger amount of computation resource. Therefore antialiasing by supersampling is rarely adopted by software renderers. Adaptive sampling [10, 11] reduces the overhead by casting additional rays only if significant color variation across image samples is detected. Variations of the adaptive sampling techniques have also been proposed in [8].

2.3 Post Filter Antialiasing

A small disadvantage of adaptive sampling is that some image pixels still need additional subpixel samples to be fully shaded or traced. It would be desirable if expensive shading could be avoided at additional subpixel locations. Reshetov [16] proposes an image filtering approach, Morphological antialiasing (MLAA) to recover edges from input image with per-pixel color information. However, this sort of color-only information could fail to identify some geometry edges, especially those edges without high contrast. Geometric Post-process Anti-Aliasing (GPAA) [13] and Geometry Buffer Anti-Aliasing (GBAA) [14] extend the MLAA ideas and use extra edge information explicitly to eliminate the jaggy edges. Normal Filter Anti-Aliasing (NFAA) [18] reduces aliasing by searching for contrasting luminosity changes in the final rendering image. It builds a normal displacement map to apply a per-pixel blur filter in highly contrast aliased areas. However, it softens the image due to the filtering of textures. More filter-based approaches are discussed in [7].

2.4 Shading Reconstruction Filter

Decoupled sampling [9, 15] presents an approach to generate shading and visibility samples at different rates in GPU pipelines to speed up the rendering in applications with stochastic supersampling, depth of field, and motion blur. Yang et al. [22] present a geometry-aware framebuffer level of detail (LOD) approach for controlling the pixel workload by rendering a subsampled image and using edge-preserving upsampling to the final resolution. Subpixel Reconstruction Antialiasing (SRAA) [3] takes a similar decoupled sampling approach and applies a cross-bilateral filter (as in the geometry-aware framebuffer LOD method) to

upscale shading information using subpixel geometric information that is obtained from the GPU rasterization pipeline. It is based on the assumption that the subpixel geometric information could be obtained much more easily without fully going through the expensive shading stage. SRAA can produce good edge antialiasing but it cannot resolve shading edges in texture, shadow, reflection and refraction. Our work follows the same assumption by avoiding emitting subpixel samples for the primary rays. This maintains the advantage over adaptive sampling because no subpixel ray needs to be traced.

3 ANTIALIASING

SRAA [3] relies on the fact that shading often changes more slowly than geometry in screen space and generates shading and visibility at different rates. SRAA performs high-quality antialiasing in a deferred rendering framework by sampling geometry at higher resolution than the shaded pixels. It makes three modifications to a standard rendering pipeline. First, it must produce normal and depth information at subpixel resolution. Second, it needs to reconstruct the shading values of sampled geometric subpixel from neighboring shaded samples with bilateral filter using the subpixel geometric (normal and depth) information. Finally, the subpixel shading values are filtered into an antialiased screen-resolution image.

SRAA detects the geometric edges with geometric information to resolve aliasing problem. However, the edges of shadow and reflection/refraction could not be detected by the subpixel geometric information generated from the eye position. For example, the shadow edges mostly fall on other continuous surfaces that have slowly changing subpixel depths and normals. To extend the SRAA concept to ray-tracing based applications, we perform antialiasing separately for primary rays, shadow rays and secondary rays to resolve this issue. The following subsections offer the detail.

3.1 Primary Ray

Like SRAA, our goal is to avoid the performance penalty of shading subpixel samples. In Figure 2, geometric information and shading are generated at different rates. Each pixel has 4 geometric samples on a 4×4 grid and one of those geometric samples is also a shaded sample. The shading value at each geometric sample is reconstructed by interpolating all shaded neighbors in a fixed radius using the bilateral weights. We take both depth and normal change into account when compute the bilateral weight. A neighboring sample with significantly different geometry is probably across a geometric edge and hence receives a low weight.

$$w_{ij} = G(\sigma_z(z_j - z_i))G(\sigma_n(1 - sat(n_j \cdot n_i))) \quad (1)$$

In Equation 1, $G(x)$ is the Gaussian function of the form $exp(-x^2)$. z_i and n_i are the depth and normal of the i^{th} subpixel sample. σ_z and σ_n are the scaling factors for controlling how quickly the weights fall off and allowing us to increase the importance of the bilateral filter. We set σ_z to 10 and σ_n to 0.25 in all our testing. The $sat(x)$ function is implemented as $max(0, min(1, x))$. The result w_{ij} is the weight associated with the j^{th} subpixel sample while performing shading reconstruction for the i^{th} subpixel sample.

For tracing the primary rays that are emitted from the eye position, we use a hybrid rendering approach that utilizes the GPU to generate the subpixel geometric information including position, normal and depth. We create 3 auxiliary geometric buffers to store position, normal and depth by GPU rasterization with the same resolution as the shaded buffer. Each geometric buffer is rendered with a subpixel offset applied to the projection matrix. The subpixel offset is applied not only to form a $4\times$ rotated-grid but also to do pixel alignment between rasterization and ray tracing rendering. Since the GPU rasterization pipeline produces the subpixel geometric information very efficiently, this overhead is insignificant when compared to the ray tracing stage.

3.2 Shadow Ray

As mentioned above in Section 3, the shadow edges cannot be detected by the geometric information that is generated from the eye position alone. What we need is subpixel information that is more meaningful to the shadow edges. The naive solution for shadow antialiasing is through a shadow map drawn at a higher resolution. However, this approach is inefficient because the increased resolution of the shadow map (from the light's view) does not contribute directly to the subpixels at the screen space. Therefore, we generate subpixel shadow information by ray casting and combine this shadow value with the bilateral filter weighting equation as shown in Equation 2. The subpixel shadow rays are generated by utilizing the position information in the geometric buffer as mentioned in Section 3.1.

Figure 2 shows our algorithm reconstructs the color value of a geometric sample in a non-shadowed area not only by taking the Euclidean distance and the normal change between the source and the target samples but also under the influence of shadow boundaries to exclude the neighboring samples in shadowed area. This is the reason why the original SRAA adds excessive blur to the shadow boundaries, yet our method achieves a better quality that is comparable to $16\times$ supersampling.

$$w_{ij} = \begin{cases} G(\sigma_z(z_j - z_i))G(\sigma_n(1 - sat(n_j \cdot n_i))), & \text{if } s_i = s_j \\ 0, & \text{if } s_i \neq s_j \end{cases} \quad (2)$$

In Equation 2, s_i is the shadow value of the i^{th} subpixel sample and is equal to 1 if it is in shadowed area. Otherwise it is equal to 0. If s_j is different from s_i, then the j^{th} subpixel falls on the other side of a shadow edge. Therefore we set the weight w_{ij} associated with the j^{th} subpixel to 0 to exclude it from the shading reconstruction for the i^{th} subpixel sample.

3.3 Secondary Ray

In the original SRAA framework, it uses geometric information to detect geometric edges in the subpixel reconstruction process. However, the edge of secondary shading (such as those from the reflection) cannot be detected by this geometric information generated from the eye position. Take the reflection rays for an example as shown in Figure 5 (c), if we perform subpixel shading reconstruction as shown in Equation 1 with the geometric information generated from the eye position, it will not be able to detect the edges of the reflected objects, and in consequence add excessive blur to the reflected colors.

Therefore we must take geometric information that is generated from the hit points of primary rays to perform subpixel-level bilateral filer when computing the shading value of the secondary rays that originate from the primary hit point. The subpixel secondary rays for hit points are generated by utilizing the position and normal information in the geometric buffer. Our method which performs subpixel reconstruction separately for primary and secondary shading achieves better quality than the original SRAA approach. Please see our results in Section 4 and Figure 5.

4 RESULT

Our algorithm is implemented using NVIDIA CUDA 4.0, the raytracer is built with OptiX 2.0 and rasterization with OpenGL. All results shown in this paper were obtained using an Intel Xeon E5504 processor and an NVIDIA Geforce GTX 570 GPU.

4.1 Quality

Figure 4 shows the quality comparison between our method and other antialiasing techniques in a Cornell box scene. The original SRAA adds excessive blur to the shadow, yet our method achieves similar quality to 16× supersampling.

Figure 5 highlights some interesting cases for primary shading, shadow and secondary shading in the Sponza scene. For the shading from primary rays, both our method and SRAA use the geometric information to improve image quality and the results are almost identical between ours and SRAA. For the shadow, our method uses both the geometry and the shadow edge information to perform subpixel reconstruction, thus produces better shadow line in the highlighted area than SRAA. For secondary shading, we perform subpixel reconstruction separately for primary and secondary shading, while SRAA uses only the final color of each sampled subpixel for this purpose. This results in over blurring for secondary shading in SRAA.

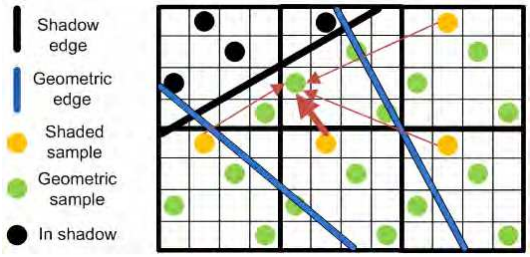

Figure 2: Here we add shadow edge into consideration to perform subpixel shading reconstruction. Each pixel has 4 geometric samples on a 4×4 grid. One of those geometric samples is also a shaded sample. Shading value for each geometric sample in non-shadowed area is reconstructed from nearby shaded samples except the shaded samples in shadowed area and weighted by their distance and the normal change between the source and the target sample.

To summarize, we observe that antialiasing with geometric information from primary rays could be problematic in some difficult cases and our method offers a solution to the highlighted cases in Figure 5.

Our method does have a limitation in handling material variation or textured surfaces. Figure 6 shows such an example where the floor contains patches of different colors. Since the extra subpixel depth and normal information does not help us detect the edges between patches of different colors, jagged edges could still appear on the floor.

4.2 Performance

There are two rendering passes in our current implementation. The first pass is the geometric information generation step and the second pass is the antialiasing process. Table 1 shows that the geometric information generation step with raytracer solution takes about 70 percent of the total processing time for rendering the Sponza scene [4] in Figure 5. This overhead to generate geometric information for primary rays can be reduced with a GPU hybrid solution. Figure 3 shows that our method maintains the interactive rate while rendering the Sponza scene in Figure 5 with a GPU hybrid solu-

	Rendering Pass		
Resolution	1^{st}	2^{nd}	Total
256x256	18	5	23
512x512	35	14	49
768x768	69	28	97
1024x1024	116	49	165

unit: millisecond

Table 1: Time measurement of our method for rendering the Sponza scene in Figure 5.The first pass is geometric information generation and the second pass is antialiasing process. Note that the time shown in first pass is measured with raytracer solution.

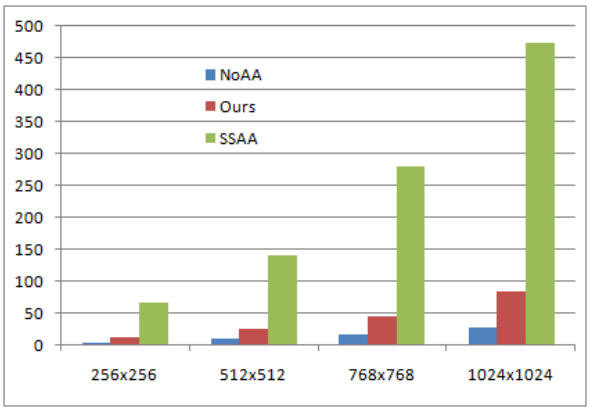

Figure 3: Performance comparison between NoAA (no antialiasing applied), our method with GPU hybrid approach, and SSAA (16× supersampling antialiasing) for rendering the Sponza scene under various output resolutions. The vertical axis is the rendering time in millisecond. The overall rendering performance of our method with a GPU hybrid approach is about 6× speedup in average compared to the 16× supersampling approach.

tion and achieves about 6× speedup in average compared to the 16× supersampling approach.

5 CONCLUSION

We introduce the concept in SRAA to path-tracing based rendering methods for antialiasing. Our method extends the subpixel geometric sampling concept beyond the primary rays and achieves antialiasing for shadow rays and reflective rays as well. By adopting a hybrid approach, our method improves the image quality without incurring performance penalty of tracing additional primary rays. We hope our method encourages the adoption of antialiasing even for the computationally constrained real-time ray tracing or path tracing.

6 ACKNOWLEDGEMENTS

This work is supported in part under the "Embedded software and living service platform and technology development project" of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs (Taiwan), and by National Science Council (Taiwan) under grant NSC 100-2219-E-003-002.

7 REFERENCES

[1] Carsten Benthin. *Realtime Ray Tracing on Current CPU Architectures*. PhD thesis, Saarland University, 2006.

[2] Jacco Bikker. Arauna real-time ray tracer and Brigade real-time path tracer.

[3] Matthäus G. Chajdas, Morgan McGuire, and David Luebke. Subpixel reconstruction antialiasing for deferred shading. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 15–22, 2011.

[4] Marko Dabrovic. Sponza atrium, http://hdri.cgtechniques.com/ sponza/files/, 2002.

[5] Johannes Gunther, Stefan Popov, Hans-Peter Seidel, and Philipp Slusallek. Realtime ray tracing on gpu with bvh-based packet traversal. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing*, pages 113–118, 2007.

[6] Daniel Reiter Horn, Jeremy Sugerman, Mike Houston, and Pat Hanrahan. Interactive k-d tree gpu raytracing. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 167–174, 2007.

[7] Jorge Jimenez, Diego Gutierrez, Jason Yang, Alexander Reshetov, Pete Demoreuille, Tobias Berghoff, Cedric Perthuis, Henry Yu, Morgan McGuire, Timothy Lottes, Hugh Malan, Emil Persson, Dmitry Andreev, and Tiago Sousa. Filtering approaches for real-time anti-aliasing. In *ACM SIGGRAPH Courses*, 2011.

[8] Bongjun Jin, Insung Ihm, Byungjoon Chang, Chanmin Park, Wonjong Lee, and Seokyoon Jung. Selective and adaptive supersampling for real-time ray tracing. In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, pages 117–125, 2009.

[9] Gábor Liktor and Carsten Dachsbacher. Decoupled deferred shading for hardware rasterization. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, pages 143–150, New York, NY, USA, 2012. ACM.

[10] Don P. Mitchell. Generating antialiased images at low sampling densities. In *Proceedings of the 14th annual conference on Computer graph-*

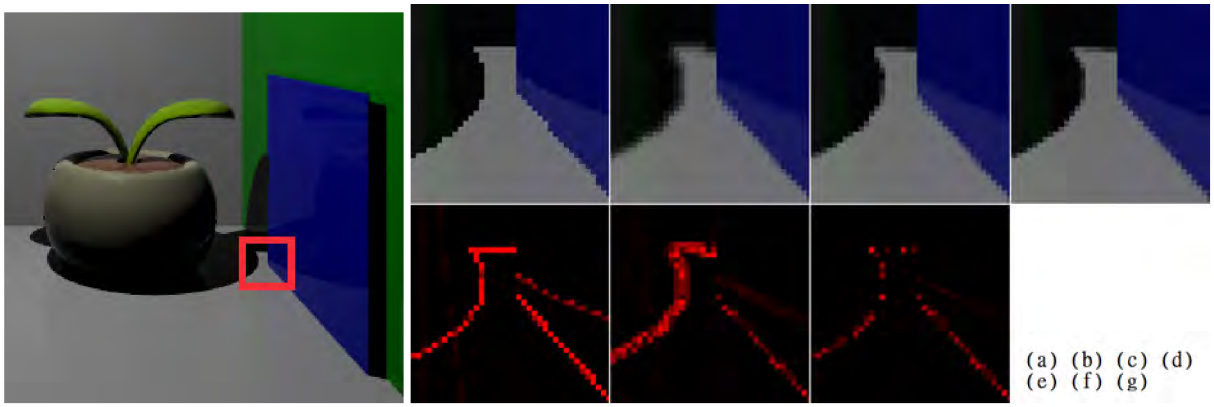

Figure 4: The leftmost image shows the Cornell box generated with our method. The smaller images to its right show the 8× zoom-in of the marked region under various antialiasing techniques. (a) is the result without any antialiasing. (b) is from SRAA. (c) is from our method. (d) is the reference image (16× supersampling antialiasing). (e)(f)(g) show the difference between (a)(b)(c) and the reference image (d) respectively. The original SRAA often adds excessive blur to the shadow and secondary shading, yet our method achieves similar quality to 16× supersampling.

ics and interactive techniques, SIGGRAPH '87, pages 65–72, 1987.

[11] J. Painter and K. Sloan. Antialiased ray tracing by adaptive progressive refinement. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '89, pages 281–288, 1989.

[12] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. Optix: a general purpose ray tracing engine. *ACM Trans. Graph.*, 29:66:1–66:13, July 2010.

[13] Emil "Humus" Persson. Geometric post-process anti-aliasing (GPAA), march 2011, http://www.humus.name/index.php?page=3d&id=86.

[14] Emil "Humus" Persson. Geometry buffer anti-aliasing (GBAA), july 2011, http://www.humus.name/index.php?page=3d&id=87.

[15] Jonathan Ragan-Kelley, Jaakko Lehtinen, Jiawen Chen, Michael Doggett, and Frédo Durand. Decoupled sampling for graphics pipelines. *ACM Trans. Graph.*, 30(3):17:1–17:17, May 2011.

[16] Alexander Reshetov. Morphological antialiasing. In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, pages 109–116, 2009.

[17] Min Shih, Yung-Feng Chiu, Ying-Chieh Chen, and Chun-Fa Chang. Real-time ray tracing with CUDA. In *Proceedings of the 9th International Conference on Algorithms and Architectures for Parallel Processing*, ICA3PP '09, pages 327–337, 2009.

[18] Styves. Normal filter anti-aliasing, http://www.gamedev.net/topic/580517-nfaa—a-post-process-anti-aliasing-filter-results-implementation-details, 2010.

[19] Ingo Wald. *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Saarland University, 2004.

[20] Sven Woop and Manfred Ernst. Embree - photo-realistic ray tracing kernels, http://software.intel.com/en-us/articles/embree-highly-optimized-visibility-algorithms-for-monte-carlo-ray-tracing/, June 2011.

[21] Sven Woop, Jörg Schmittler, and Philipp Slusallek. RPU: a programmable ray processing unit for realtime ray tracing. *ACM Trans. Graph.*, 24:434–444, July 2005.

[22] Lei Yang, Pedro V. Sander, and Jason Lawrence. Geometry-aware framebuffer level of detail. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering 2008)*, 27(4):1183–1188, 2008.

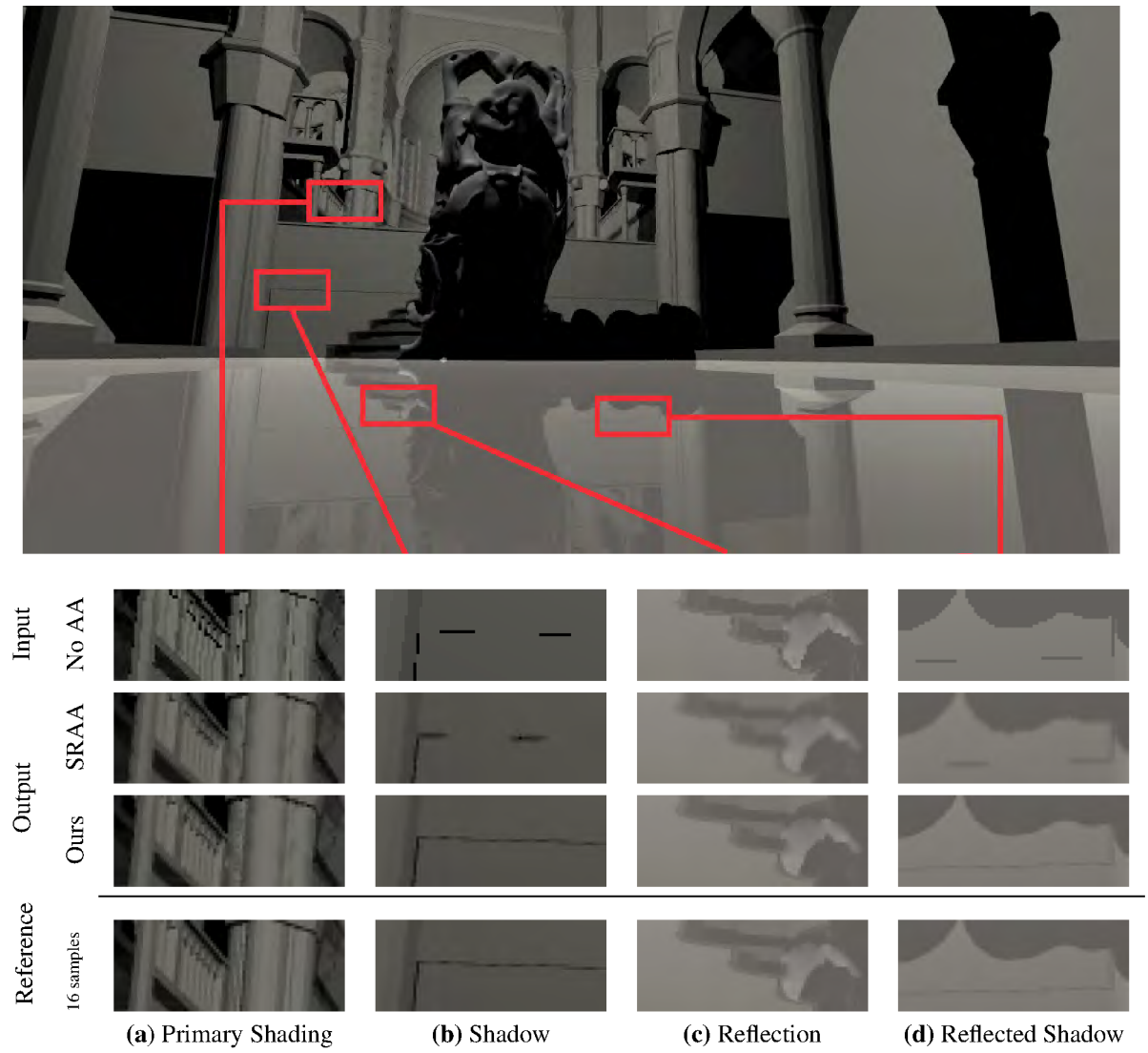

Figure 5: Quality comparison between our method and the other antialiasing techniques in highlighted areas of primary shading, shadow, reflection, and reflected shadow. (Row 1) No antialiasing, (Row 2) SRAA: one subpixel with shading value and 4 subpixels with primary geometric information, (Row 3) Ours: one subpixel with shading value and 4 subpixels with geometric information for primary, shadow and secondary rays, (Row 4) Reference image: 16× supersampling.

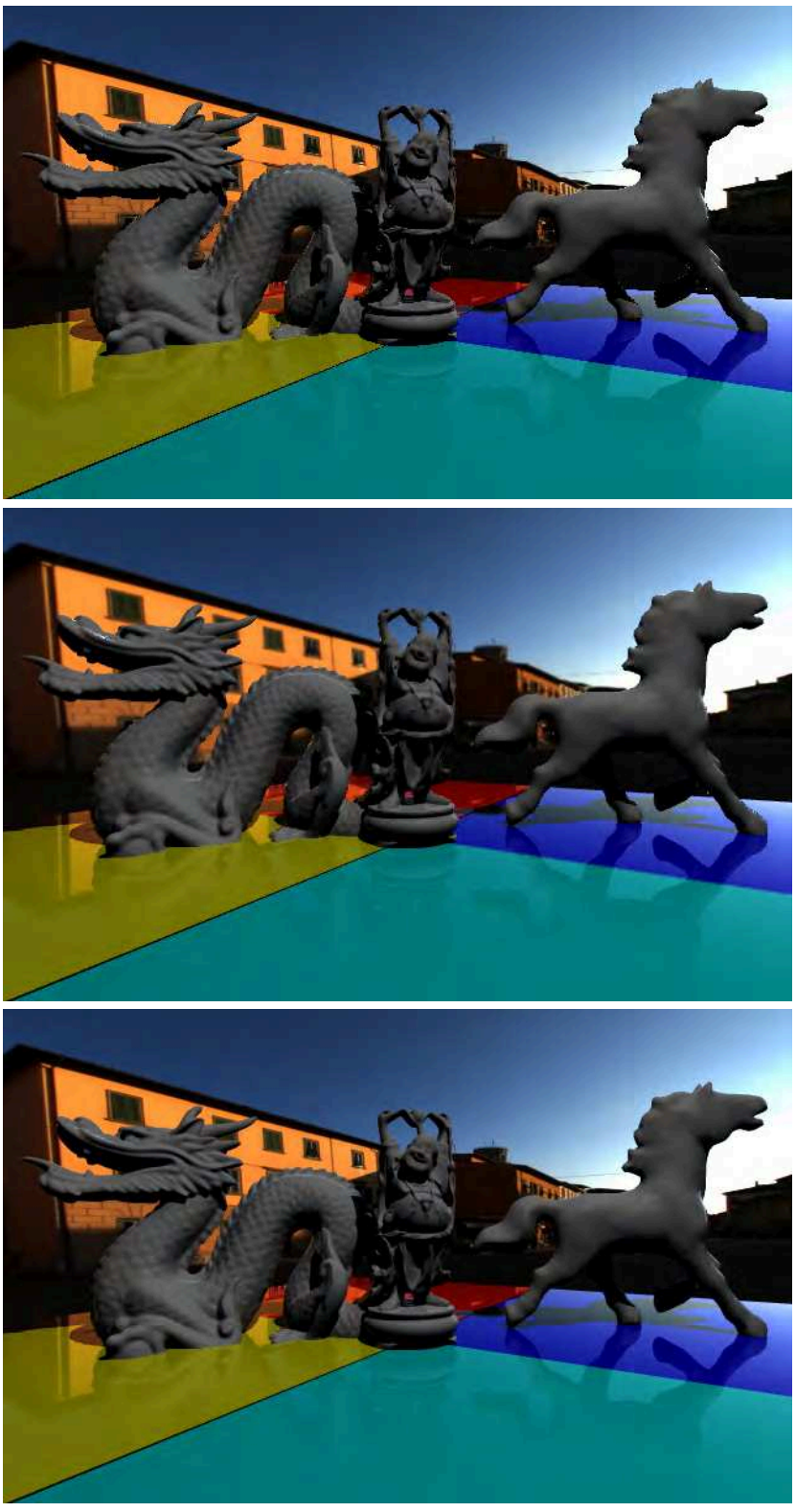

Figure 6: The scene in this figure shows a limitation of our method in handling material variation or textured surfaces. The floor contains patches of different colors. Since the extra subpixel depth and normal information does not help us detect the edges between patches of different colors, jagged edges still appear on the floor in the middle image that is rendered by our antialiasing method. For comparison, the top image shows the result without antialiasing and the bottom image is produced with 16× supersampling.