Low Cost Adaptive Anti-Aliasing for Real-Time Ray-Tracing

Maxim Shevtsov, Mikhail Letavin and Alexey Rukhlinskiy
Intel Corporation
Nizhniy Novgorod, Russia
{maxim.y.shevtsov, mikhail.letavin, alexey.v.rukhlinskiy}@intel.com

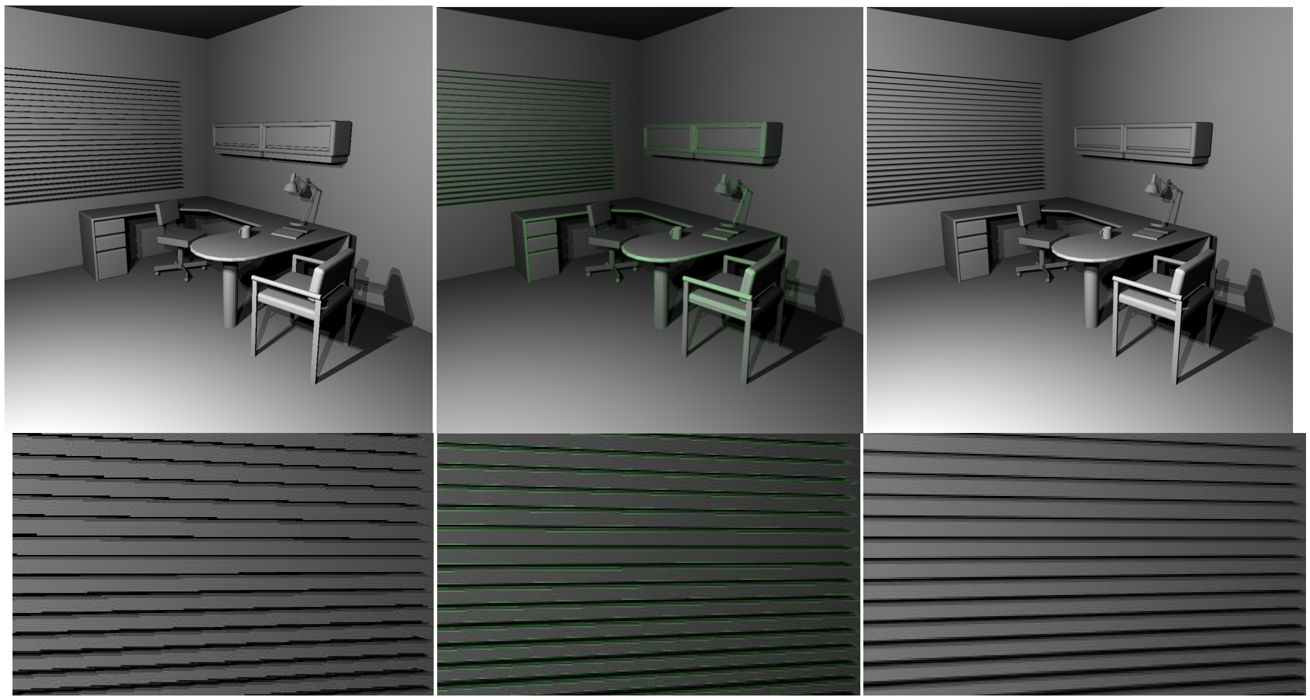


Figure 1: Evaluation of the proposed adaptive anti-aliasing scheme in the context of simple packet ray-tracing pipeline. Left is no-AA image (exactly one eye/shadow ray per pixel). In the middle additional samples are shown (notice how shadow boundary is also detected for additional sampling). Right image is anti-aliased result. By computing additional samples only where image-space color gradient is high, we significantly save computations. The resulting quality is identical to the 8X super-sampling, with only doubling rendering time in compare to no AA version.

Abstract

This paper describes simple, practical scheme for adaptive anti-aliasing, particularly suitable for packet style ray tracing. The sampling patterns are organized in a SIMD-friendly fashion. The technique explores image–space attributes to compute the gradient. Only where value of the gradient is high, additional samples are used. The final result is anti-aliasing with 8X super-sampling quality, for just 2X rendering time increase (on average).

Keywords: *Rendering, adaptive anti-aliasing, ray-tracing, SIMD, MSAA-patterns.*

1. INTRODUCTION

Raster displays use a finite number of pixels to display the scene: visible artefacts appear where pixels cannot adequately represent high-frequency data.

Rasterization and ray tracing are combating aliasing in a different manner. The two most popular ways within the rasterization context are super-sampling (SSAA) and multi-sampling anti-aliasing (MSAA). Super-sampling is brute-force approach performed by rendering the scene at a higher resolution and then down-sampling to the target resolution. Super-sampling is expensive in terms of both performance and memory bandwidth. Today's GPUs use MSAA is an approximation to super-sampling, avoiding unnecessary shader invocations, and CSAA [1] further improves speed by decoupling coverage samples from color/z/etc samples. Hardware CSAA/MSAA modes are characterized by the pattern of the sampling grid, refer to the nice overview [2].

At the same time, both MSAA and CSAA are still brute-force methods. Performing equally for the whole render target, they lack adaptivity. The power is wasted on smooth areas, whereas some problematic pieces of the image might still lack the samples. For relatively low frequency effects an interesting alternative to HW AA is mixed resolution rendering [3] which is more adaptive.

Finally, rasterization antialising considers quality of edges primarily. For geometry aliasing (e.g. aliasing of shadows/reflections) the existing GPU algorithms rely either on filtering or increasing overall resolution, for shadow or environment maps respectively. In contrast, for RT the true shadow/reflection rays can be traced to get the additional data at any specific frequency.

At the same time direct super-sampling methods are too expensive for real-time ray tracing. Also computing pixel coverage that is required for MSAA/CSAA would essentially mean shooting additional rays anyway, thus falling back to super-sampling.

The iterative nature of ray tracing allows for adaptive schemas, when additional rays are spawned only where it is necessary, so high-quality results are obtained without significantly increasing the storage resources and rendering time.
The major question for any adaptive scheme is the way how troublesome pixels that would need additional samples are identified. A recent scheme in [6] uses edge-detection filter that again works well for edge smoothing, but doesn't consider geometry aliasing.
For RT there is an option to find discontinuities on the per-packet basis, but smoother results are obtained, when some global (e.g. frame-wide) information is used. In the paper we focused on simple three-pass scheme. Initial pass is sparse stratified sampling, sharing as many samples as possible with adjacent pixels. Second pass is discontinuity detection. In the final pass additional samples (i.e. rays) are traced.
Researchers proposed algorithms for tracing coherent ray packets instead of single rays ([8]) using SIMD instructions. Thus a sample pattern should also be packet-friendly. The paper describes efficient grouping of the pattern rays for SIMD-aware ray tracing algorithms. We consider the 4-way SIMD (e.g. SSE) primarily.
We argue that to stay real-time one would need to consider inter-pixel pattern design. We introduce 2x2 pixels pattern. At the same time the proposed primary pattern still allows for sub-pixel accuracy.

2. RELATED WORK

Whitted was first to suggest adaptive super-sampling with recursively subdividing the pixels [9] for ray tracing. Mitchell presented effective non-uniform sampling patterns and applied contrast measure thresholds [10]. Painter and Sloan [11]] presented hierarchical adaptive stochastic sampling for ray tracing that worked in progressive manner.
Cone tracing [12] is an example of the instant ray-filtering approach that overcomes the aliasing problems resulting from the point sampling approach of ray tracing. The space is probed with a finite-width cone instead of a ray. The intent is to prefilter by computing the integral of the image function within a circle on the image plane. Similar goals were pursued by different researchers through the introduction of polygonal beams [13] and finally frustum tracing with MLRTA [14]. MLRTA provides a natural measure of the geometric complexity (i.e. aliasing probability) of specific image regions. But no applications of the MLRTA for the anti-aliasing are described by the best of our knowledge, particularly for shadows/reflections, that are less advantageous for frustum tracing.
Also, the geometry complexity is not the only mechanism that contributes to unwanted high frequencies in ray-traced image: shadow edges, specular highlights, mapped textures, reflected and refracted details, etc. The only contribution that can be pre-filtered in advance is the texture aliasing (combined with ray differentials [15]). The rest require increasing the sampling rate.
We follow the previous adaptive sampling techniques in the approach of detecting problematic regions via frame-buffer *color* comparisons. This in fact, the very property that leads to the most general discontinuity detection, while also efficiently accounting for the aliasing mechanisms altogether. The only exception is texture anti-aliasing that is done locally on a surface [15], rather than in the image plane.
There are recent approaches [6] where geometry attributes (like normal) and shadow existence contribute separately to the multi-valued threshold vector. While this approach produces better quality it does increase the resource and computational pressure. Also it can be too conservative (and expensive) for areas where some additional attributes might appear aliased thus causing additional sampling, that would be avoided if final shading and blending were performed first.
There are plenty of sampling patterns [2]. However most of them are concerned with *intra*-pixel sampling strategies. It does make sense for distribution ray tracing [4]. We consider *inter*-pixel design by sharing samples within 2 x 2 block of pixels. Since more sparse sampling may under-utilizes SIMD units, due to lowered rays coherency, we pack the pattern rays in 3 coherent groups.

3. SOLUTION

Adaptive super-sampling is a smart way of refining the rendering of the scene at those exact places where it will deliver the greatest benefit. In the paper we focused on simple three-pass scheme, Figure 2, left. We consider each pass in details below, leaving the analysis for the next section.
Initial pass is sparse pre-sampling, sharing as many samples as possible with adjacent pixels (Figure 2 right). We use FLIPTRI pattern [5] for inexpensive sampling during initial pass. FLIPTRI costs only 1.25 samples per pixel on average. It is also exhibits reasonable stratification for horizontal, vertical and diagonal strata. FLIPTRI is the most efficient filter, in terms of quality/cost for most cases [2].
We also tried scheme based on FLIPQUAD as a primary sampling pattern, refer to analysis in section 4. Both FLIPTRI and FLIPQUAD are determined for one pixel first, and the rest of the sets are then obtained by mirroring along the axis of translation. This implicitly results in interleaved sampling [16]:

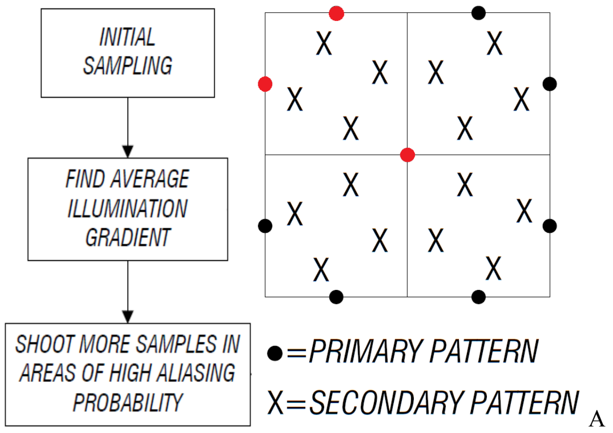

INITIAL SAMPLING
FIND AVERAGE ILLUMINATION GRADIENT
SHOOT MORE SAMPLES IN AREAS OF HIGH ALIASING PROBABILITY
X
●=PRIMARY PATTERN
X=SECONDARY PATTERN
A

Figure 2: Simple three-pass scheme for adaptive anti-aliasing, left. An example how primary and secondary pattern rays are shared for a 2x2 pixel block. Single FLIPTRI pattern is marked with read, right.

To utilize available CPU's SIMD units completely for primary pattern, we group initial samples in SSE packets: 2 edge packets and one corner package. These three packets are efficiently shared for the 2x2 pixel block, Figure 3. Similarly are shared the samples within four (3 edge/1 corner) packets of FLIPQUAD.

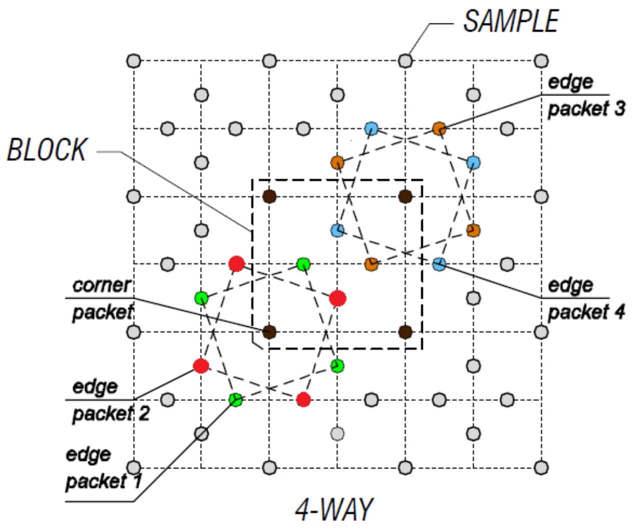


Figure 3: Packet-grouping of the rays within FLIPTRI-based primary pattern. Two edge packets are shared within 2x2 block of pixels, e.g. (edge) packets 1 with 2, or packets 3 with 4. The corner (black dots) packet is also shared, but within another 2x2 block. This irregularity breaks up symmetry somewhat, which increases the quality.

Second pass is discontinuity detection. It involves computation of gradients, performed pair-wise between samples in a primary pattern, Figure 4. The gradients can be computed either for luminance (i.e. brightness) value or separately per-color component. Finally the average gradient magnitude is computed for the frame. This value serves as a threshold in the final pass, where additional samples (i.e. rays) are generated. If super-sampling threshold value from the previous frame is used, then storing all pre-sampling results for finding average gradient can be avoided. Then, no dedicated pass is required, instead, the decision to super-sample or not can be immediately applied, once the values of initial sampling pattern are determined. This way the original three-pass scheme (Figure 2, left) can be boiled down to a single pass, while average gradient estimation (for the next frame) can be coupled with post-processing routine like tone-mapping. This approach improves cache utilization, resulting in overall performance improvements of ~5%.

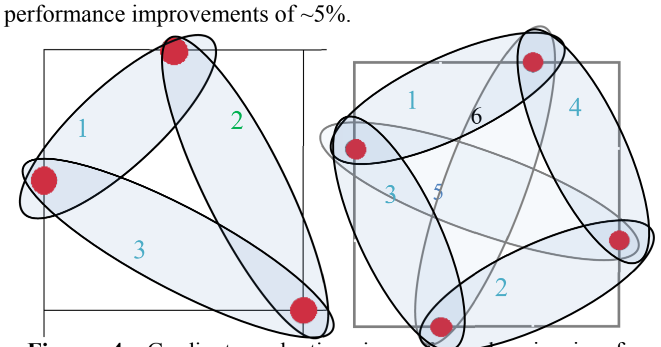


Figure 4: Gradient evaluation is performed pair-wise for samples of the primary pattern. Three gradients are computed for FLIPTRI (left), and six for FLIPQUAD (right).

For additional samples for FLIPTRI scheme we use conventional instantiation of N-rooks sampling [17], known as rotated grid super-sampling (RGSS), refer to Figure 2, right. The final color value for pixel is computed via simple averaging with equal weights for all samples (e.g. box-filter).

4. Results and analysis

Below is a performance for the scenes with the anti-aliasing approach described in the previous section:

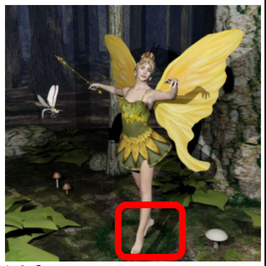
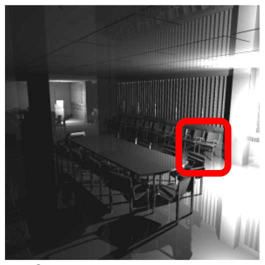
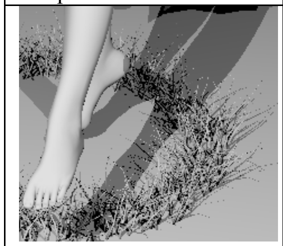
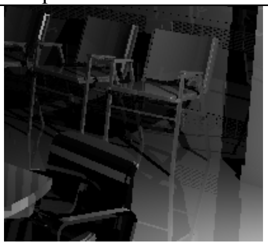
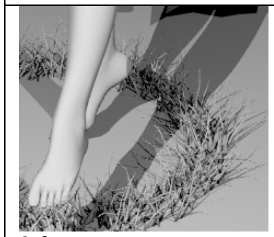
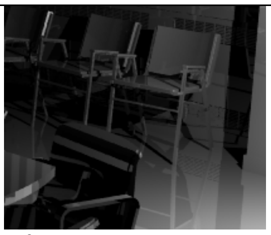
Fairy (178K triangles) 1 point light, no reflections	Conference(282K tris) 1 point light, 1 reflection bounce	scene lights refl.
 16 fps	 12 fps	
 16 fps	 12 fps	no AA
 8 fps	 7 fps	FLIP TRI

Table 1: Performance/quality results for the proposed adaptive anti-aliasing scheme vs conventional no-AA rendering. FLIPTRI is used for primary pattern. Models are ray traced at 1024x1024 on a Intel ®Core™ i7 @3.33 GHz machine with 4 Gb RAM. For the Fairy scene close-up the material colors are turned off to better demonstrate the effect.

The baseline time to frame is only 1.25X of the no-AA version (due to inexpensive primary pattern). The rest is contributed by additional sampling that costs from 20% to 30% of the frame time, depending on the scene. The final result is anti-aliasing with 8X super-sampling quality, for just 2X rendering time (on average).

The FLIPQUAD primary pattern is more expensive while produces better quality, refer to Figure 5. In compare to other sampling patterns costing 2 samples per pixel (e.g. Quincunx), it's behaviour is clearly preferable [2], [18]. It is also able to find and fix more discontinuities than FLIPTRI. Coupled with more additional samples, the FLIPQUAD might be recommended as a good higher-quality preset:

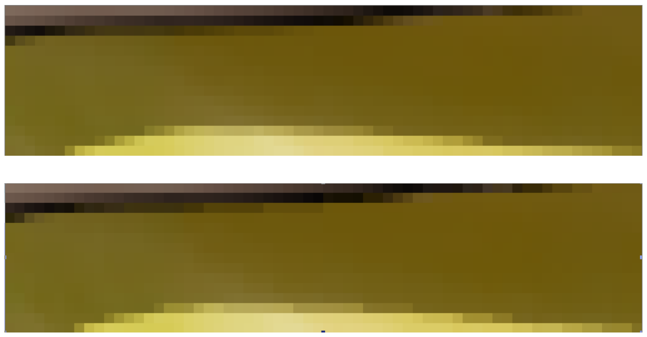


Figure 5: Quality comparison for 2 anti-aliasing presets: FLIPQUAD with *16 additional per-pixel* samples (upper image) and FLIPTRI with *4 samples shared for 2x2 pixels block* (lower image). The performance difference is not very high (just 2X) due to better ray coherence of the FLIPQUAD-based preset.

Note on texture anti-aliasing. For scenes with textures exhibiting large variations, discontinuity detection in the image space might generate unnecessary sampling. Since texture anti-aliasing is performed locally on a surface, it makes sense to estimate gradients (section 3) separately from textures, in spirit of [7]. However we found that using local average texture intensity for gradient estimation works fine, while avoiding many false positives, see Figure 6.

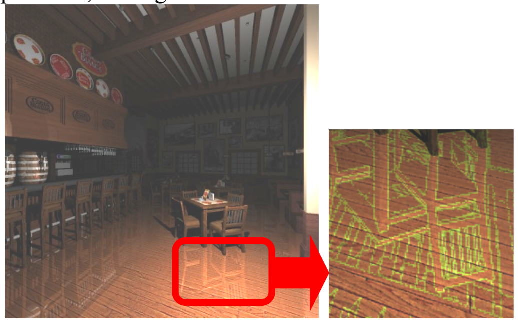


Figure 6: When the texture exhibits high variation, only its local average value is considered for gradient estimation. This helps to avoid unnecessary super-sampling. Still other sources of aliasing are detected correctly (e.g. reflection boundaries: at the right).

5. FUTURE WORK

We consider using MLRTA as a topic for future research. This would allow quickly skipping areas that don’t exhibit geometry aliasing. While this is obvious for primary rays, the research is required for secondary rays.

Currently we use box-shaped reconstruction filter and equal weights for all samples. Increasing the size of the reconstruction filter from 1×1 to 2×2 (or 3x3, which is advantageously symmetric) pixels enables the sampling pattern to more accurately approximate a wider reference filter [17]. This neither increase resource consumption, nor complexity of the filtering algorithm.

6. REFERENCES

[1] NVIDIA Corporation Rendering. “*CSAA (Coverage Sampling Antialiasing)*” (2007). http://developer.nvidia.com/object/coverage-sampled-aa.html

[2] Hasselgren J., Akenine-Moller T., Laine S.: “*A Family of Inexpensive Sampling Schemes*”, Computer Graphics Forum, v. 24 (2005).

[3] Shopf J. “*Mixed Resolution Rendering*”, GDC 2009 slides.

[4] Boulos S., Edwards D., Lacewell J D., Kniss J., Kautz J., Shirley P., and Wald I. “*Interactive Distribution Ray Tracing”* Tech. rep., University of Utah, 2006

[5] Akenine-Moller T.: “*An Extremely Inexpensive Multisampling Scheme*”. Tech. rep., Chalmers University of Technology, 2003.

[6] Iourcha K., Yangy J. C., Pomianowskiz A. “*A Directionally Adaptive Edge Anti-Aliasing Filter”* , Proceedings of High Performance Graphics, 2009.

[7] Bongjun Jin, Insung Ihm and Byungjoon Chang. “*Selective and Adaptive Super-sampling for Real-Time Ray Tracing* ”, Proceedings of High Performance Graphics, 2009.

[8] Wald I., Benthin C., Wagner M., and Slusallek P., “*Interactive Rendering with Coherent Ray Tracing*”. Proceedings of Eurographics 2001.

[9] Whitted T., "*An Improved Illumination Model for Shaded Display*", Comm. ACM, 23(6), 1980.

[10] Mitchell D.P., "*Generating Antialiased Images at Low Sampling Densities*", Computer Graphics, 21(4), 1987.

[11] Painter J., and Kenneth S., "*Antialiased Ray Tracing by Adaptive Progressive Refinement*", Computer Graphics, 23(3), 1989.

[12] Amanatides, J. "*Ray Tracing with Cones*", Computer Graphics, 18(3), 1984.

[13] Heckbert P., Pat Hanrahan P., "*Beam Tracing Polygonal Objects*", Computer Graphics, 18(3), 1984.

[14] Reshetov A., Soupikov A.. and Hurley J. ”*Multi-level ray tracing algorithm*”. Proceedings of ACM SIGGRAPH (2005).

[15] Keller A., Heidrich W. “*Interleaved sampling*”. Proceedings of Eurographics Workshop on Rendering (2001).

[16] Igehy H. “*Tracing ray differentials*”. Proceedings of SIGGRAPH(1999).

[17] Shirley P. “*Discrepancy as a quality measure for sampling distributions*”. Proceedings of Eurographics 1991.

[18] Laine S., Aila, T. “*A Weighted Error Metric and Optimization Method for Antialiasing Patterns*”, Computer Graphics Forum, v.25 (1), 2006.

About the authors

Maxim Shevtsov is a Research Scientist in Nizhniy Novgorod Laboratory of Intel Corporation. He received MS degree in CS from Novosibirsk State University in 2003. His contact email is maxim.y.shevtsov@intel.com

Mikhail Letavin is a Software Engineer in Nizhniy Novgorod Laboratory of Intel Corporation. He received MS degree from Nizhniy Novgorod State Technical University in 1998. His contact email is mikhail.letavin@intel.com

Alexey Rukhlinskiy is a Graphics Software Engineer in Nizhniy Novgorod Laboratory of Intel Corporation. He received MS degree in CS from Novosibirsk State University in 2002. His contact email is alexey.v.rukhlinskiy@intel.com