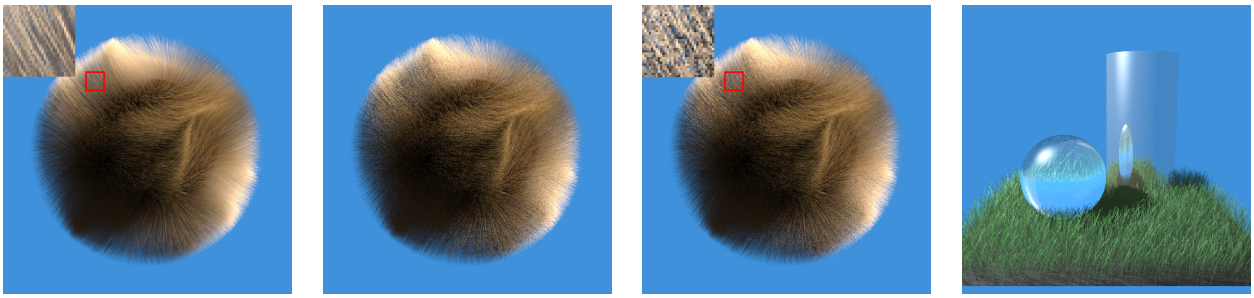
Distance Aware Ray Tracing for Curves

Koji Nakamaru, Toru Matsuoka, Masahiro Fujita

Light Transport Entertainment Research*



(a) (b) (c) (d)

Figure 1: *(a) Our approach with one sample per pixel, 5.83 min. (b) 8×8 samples per pixel, 11.35 min. (c) 16×16 samples per pixel, 47.42 min. Our approach achieves a smooth result while others are noisy regardless of many samples. (d) Reflection/refraction is also supported.*

1 Introduction

The "curves" primitive provides a way to define ribbon-like objects such as fur and hair. Ray tracing thin primitives is a difficult problem. It is possible to compute ray/curve intersections without tessellation [Nakamaru and Ohno 2002], but it requires many samples to reduce aliasing. On the other hand, GPU/rasterization based methods incorporate the concept of an A-buffer for correctly accumulating fragments' colors according to their coverages [Yu et al. 2012].

In this work, we propose distance-aware ray tracing for curves, and combine it with the concept of an A-buffer for smooth rendering with low samples. "Distance-aware" means that the method can support not only actual intersections but also distances to curves. We use ray differentials [Igehy 1999] and distances to approximate curves' coverages and accumulate their colors and opacities in the correct order.

2 Our Approach

The original ray-curves intersection algorithm utilizes a ribbon width as a margin for culling sub-curves generated in subdivision. We instead use a largest ray differential that can be easily determined by computing ray differentials at near/far z positions of the curve. After the curve is fully subdivided, we compute the distance to the survived sub-curve and corresponding ray differentials. In our current implementation, the coverage is then approximated as follows:

```
// uw: half the width of the ribbon.
//  d: distance to the ribbon's central axis.
//  r: ray extent's radius based on ray differentials.
d0 = d - uw; d1 = d + uw;
if (d0 >= 0)  // outside
  coverage = (min(d1 / r, 1.0) - min(d0 / r, 1.0)) * 0.5;
else  // inside
  coverage = (min(d1 / r, 1.0) + min(-d0 / r, 1.0)) * 0.5;
```

where coverages larger than zero are returned with other information and they are added to the (pseudo) intersection list.

For rendering many curves, we have to take into account ray differentials in traversing an acceleration structure. We currently utilize a uniform grid and traverse it in a manner similar to the slice-based packet traversal [Wald et al. 2006]. By incrementally computing ray differentials (projected onto the plane perpendicular to the major traversal axis), we determine the current ray's "extent" and corresponding cells.

The grid traversal allows for nearly front-to-back checking order, however it doesn't guarantee that the intersection list is sorted. Sorting the list for every traversal step is too expensive, so we sort it once every five traversal steps. The final contribution of each intersection is then determined while accumulating opacity as well. We terminate the traversal and avoid subsequent intersection tests if opacity falls below some small threshold. Similarly, we also avoid tracing reflection/refraction/shadow rays if the final contribution is sufficiently small. These adaptive methods greatly reduce traversal steps and intersection tests.

Figure 1 shows several results. We achieve a smooth rendering even with one sample per pixel. Although the cost of each sample is large, our approach outperforms traditional methods when the scene contains many fine details.

References

IGEHY, H. 1999. Tracing ray differentials. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '99, 179–186.

NAKAMARU, K., AND OHNO, Y. 2002. Ray tracing for curves primitive. In *Journal of WSCG (WSCG '2002 Proceedings)*, V. Skala, Ed., vol. 10, 311–316. ISSN 1213-6980.

WALD, I., IZE, T., KENSLER, A., KNOLL, A., AND PARKER, S. G. 2006. Ray tracing animated scenes using coherent grid traversal. In *ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA, SIGGRAPH '06, 485–493.

YU, X., YANG, J. C., HENSLEY, J., HARADA, T., AND YU, J. 2012. A framework for rendering complex scattering effects on hair. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '12, 111–118.

*e-mail:{nakamaru, toru, syoyo}@lighttransport.com

SIGGRAPH 2012, Los Angeles, California, August 5 – 9, 2012.
ISBN 978-1-4503-1435-0/12/0008