Real-time Global Illumination for Point set using GPUs

Heajung Min Young J. Kim

hjmin@ewhain.net kimy@ewha.ac.kr

Ewha Womans University, Seoul Korea

Introduction

In the process of visualizing a point set representing a smooth manifold surface, global illumination techniques can be used to render a realistic scene with various effects of lighting. Thanks to the continuous demand for ray tracing and the development of graphics hardware, dedicated GPUs and programmable pipeline for ray tracing have been introduced in recent years.

In this paper, real-time global illumination rendering is studied for a point-set model using ray-tracing GPUs. We apply the moving least-squares (MLS) method to approximate the point set to a smooth implicit surface and render it using global illumination by performing massive ray-intersection tests with the surface and generating shading effects at the intersection point. As a result, a complicated point-set scene consisting of more than 0.5M points can be generated in real-time.

Global Illumination for Point Set Process

Offline

- ✓Compute k-nearest neighbors
- ✓Define AABBs
- ✓Generate local coordinate system
- ✓Compute local polynomial approximation

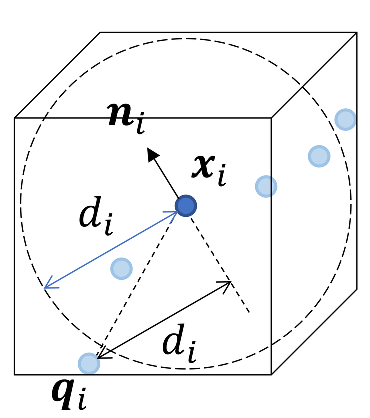
→

Online

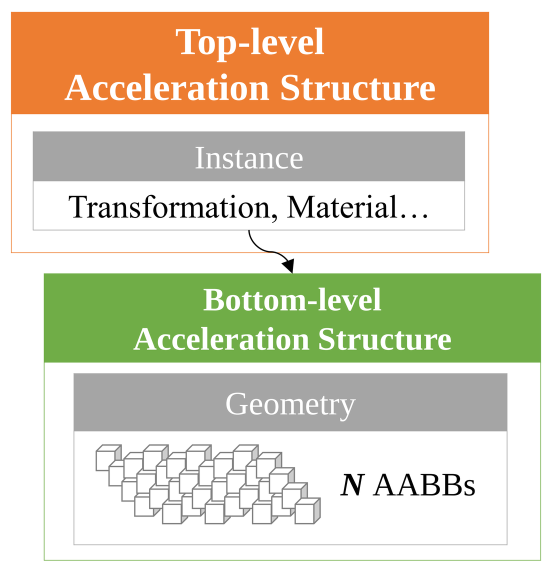
- ✓Ray-surface intersection test
- ✓Compute hit point
- ✓Compute normal
- ✓Real-time global illumination

RTX-based Implicit Surface for Point Set

We define an AABB which is sized by the distance to the k-nearest neighbors around each point. It will be built into a hierarchical structure of RTX to accelerate BVH traversal.

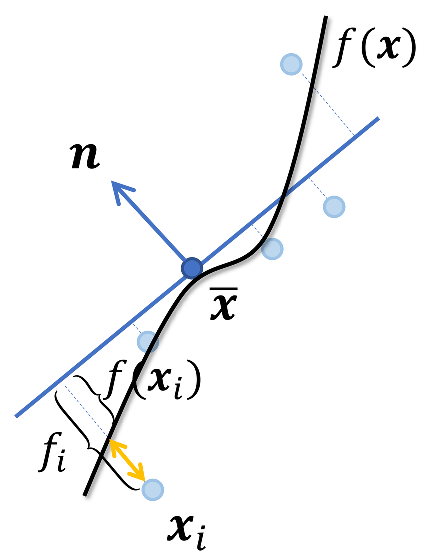


Defining AABBs



RTX AABBs Acceleration Structure

A reference plane is defined by the position and normal of each point, and the k-nearest neighbors are projected to the plane to construct a tangent local coordinate system. By applying MLS per point, the coefficients of a polynomial function are calculated for the surface.



$$f(\boldsymbol{x}) = b(\boldsymbol{x})^T c \begin{cases} b(\boldsymbol{x}) = [1, x, y, x^2, xy, y^2]^T \\ c = [c_1, c_2, c_3, c_4, c_5, c_6]^T \end{cases}$$

$$min \sum_i \theta(\| \overline{\boldsymbol{x}} - \boldsymbol{x}_i \|) \, \| f(\boldsymbol{x}_i) - f_i \|^2$$

Wendland function

$$\theta(r) = \begin{cases} (1-r)^4(4r+1) & if\ r < 1 \\ 0 & if\ r \geq 1 \end{cases}$$

Defining Implicit Surface

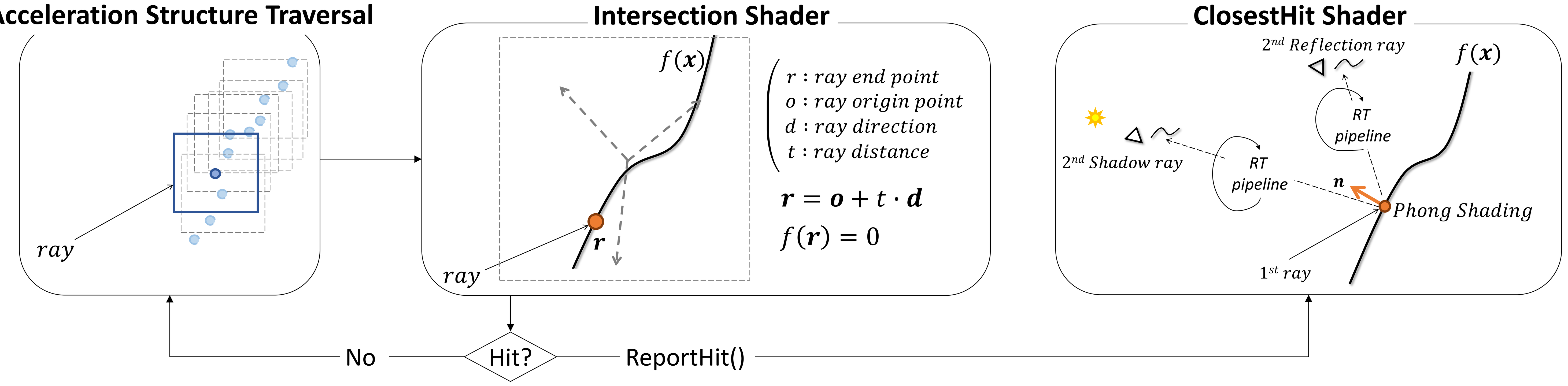
Exploiting the RTX Rendering Pipeline

As a result of the accelerated BVH traversal with RTX, the closest AABB for the ray direction is found.

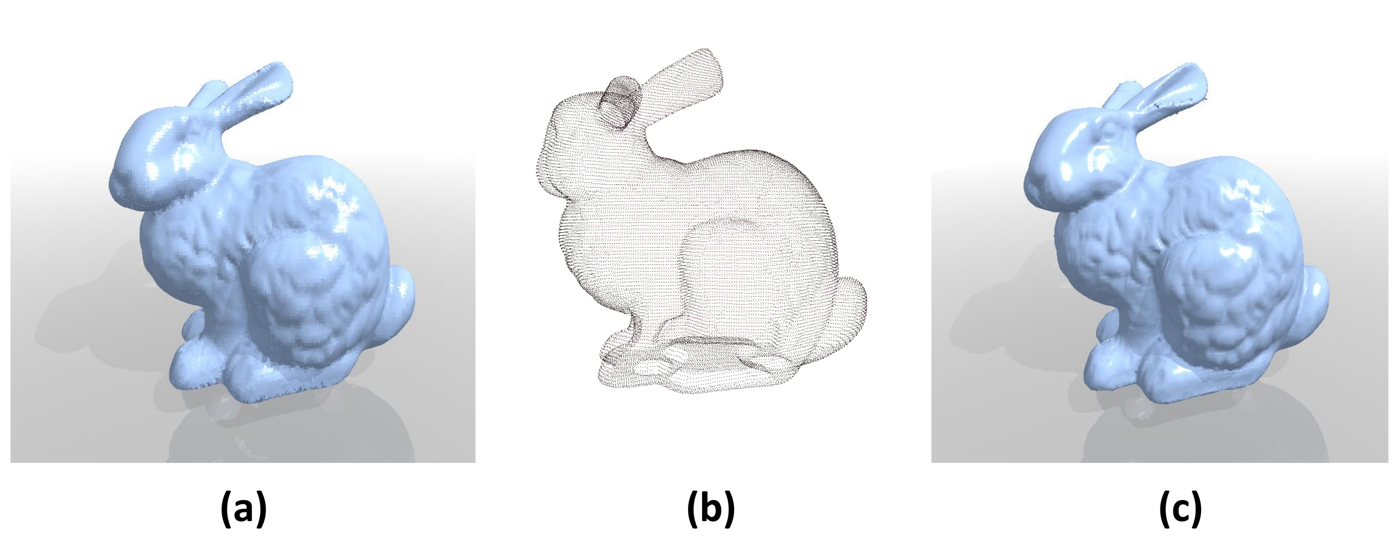
After transforming the ray into the tangent space, we compute the intersection of the ray with the surface and define an intersection point.

The gradient vector of the implicit surface at the intersection point corresponds to a normal. The shading effect of shadows and reflections is computed in the closest shader.

Acceleration Structure Traversal **Intersection Shader** **ClosestHit Shader**



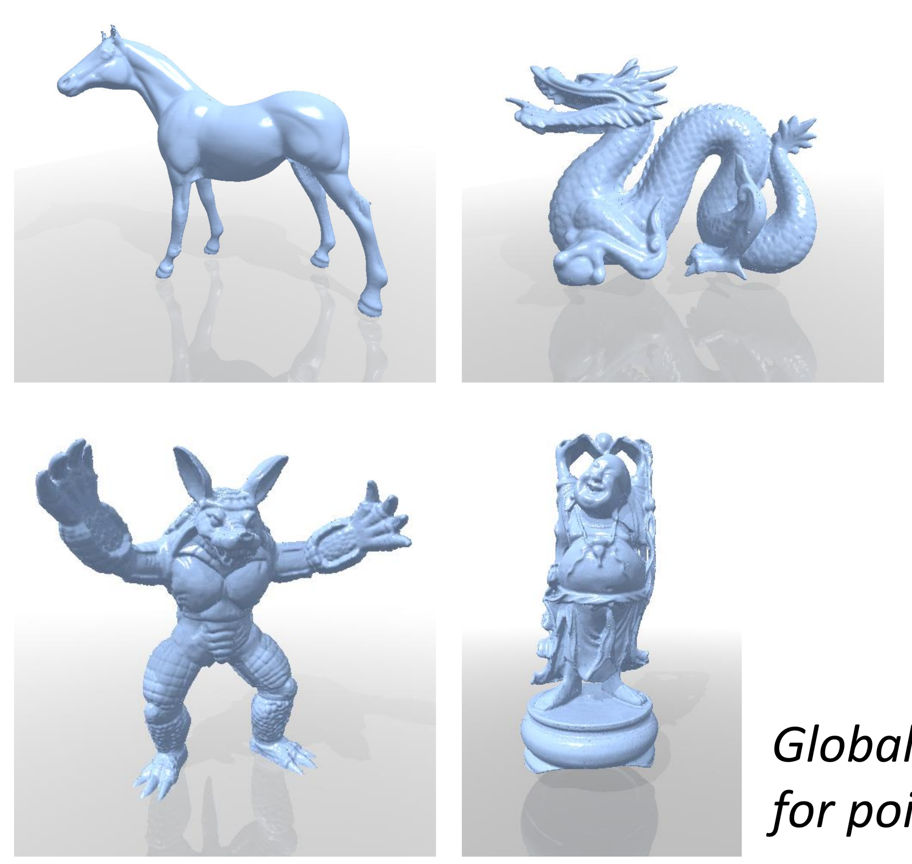
Results



(a) A global illumination rendering result of a triangle model
(b) 36K point-set model
(c) A global illumination rendering result of a point-set model with an MLS surface approximation

- AMD Ryzen 7 3700X CPU, NVIDIA RTX2080 GPU, 16GB RAM
- Windows10, VS2017, DirectX DXR
- 2 Ray Bounces
- Phong shading, Shadow, Reflection Effect

Model	*# of points*	*FPS*	
		640x640 pixels	*1858x1057 pixels*
bunny	35,947	1,000	676
Horse	48,485	927	650
Armadillo	172,974	133	113
Dragon	437,645	84	65
Buddha	543,652	62	46



Global illumination for point set

Conclusion

We propose a rendering method using global illumination in real-time for a point set based on the RTX GPU equipped with a ray tracing acceleration engine. As a result, we can create realistic images with shadows and reflections for scenes containing non-polygonal point-set geometry.

In the future, we will study to improve our method by using out-of-core memory management and data structure optimization to perform global illumination rendering for a large point set.