Path Guiding for Participate Media

20193169 김현수

20193512 임주원

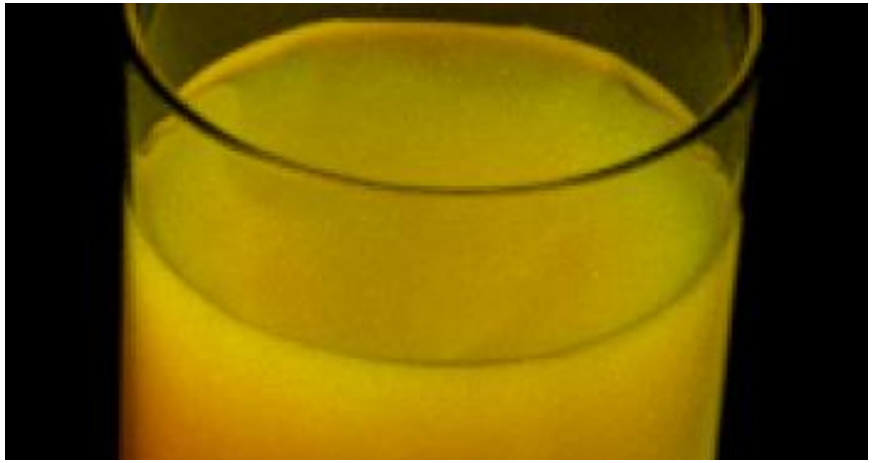
Content

- Introduction
- Related work
- Problem
- Our goal
- Role
- What we have done
- Summary
- Future work

Introduction

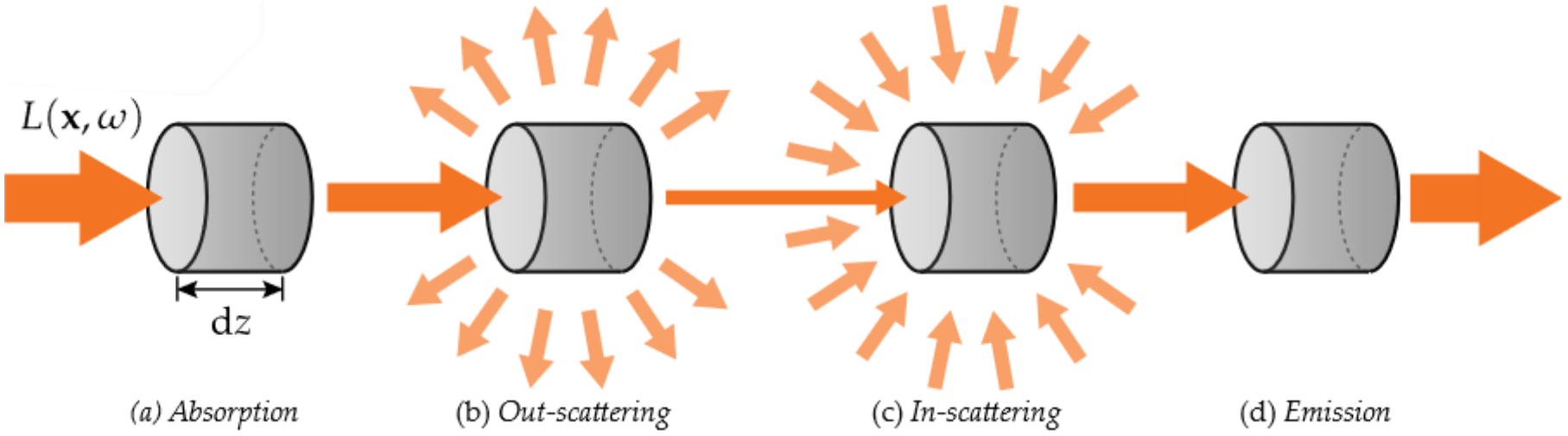
Introduction

- It is quest to produce virtual scene with high detail and physical accuracy.
- Nearly every scene contains participate media.
- Accessing volume data dominates the rendering time.
- Acceleration data structure → Better performance!



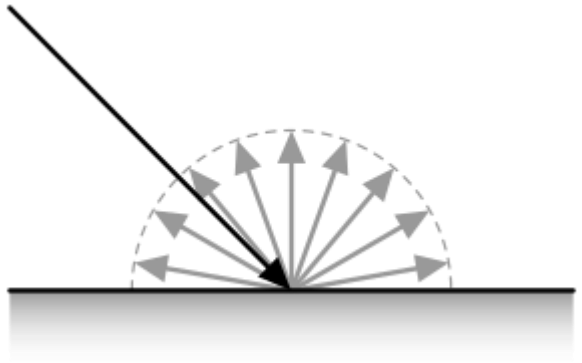
Introduction

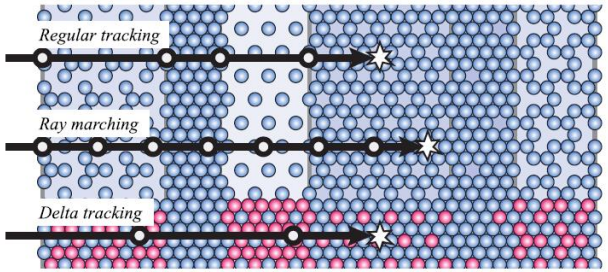
- Participate media
  - Includes cloud, smoke, liquid etc.

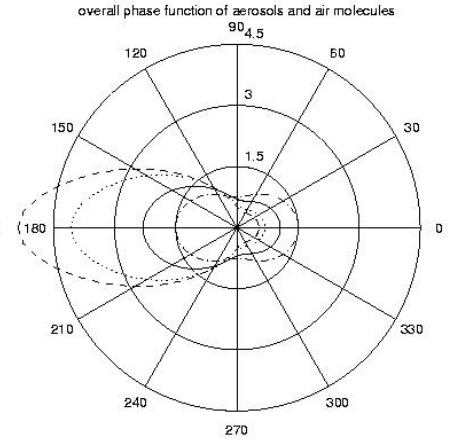


Introduction

Path tracing vs Volumetric path tracing







Light change its direction when it hits the surface

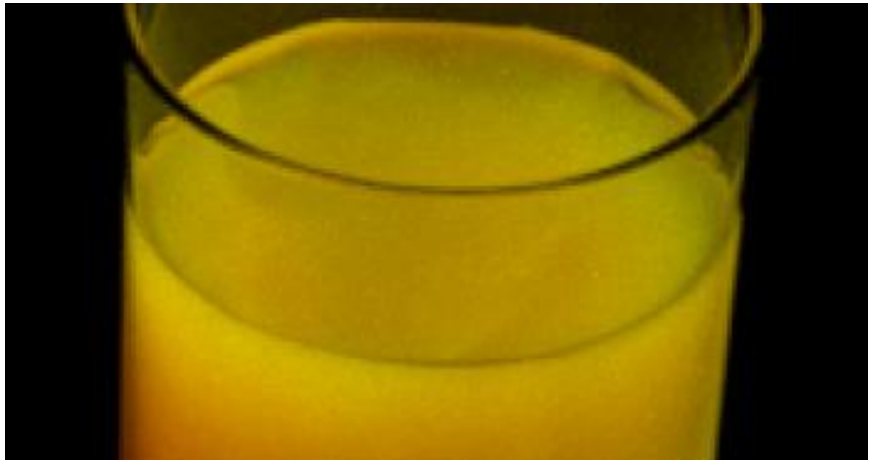
New direction is calculated with BRDF

Light change its direction using distance sampling

New direction is calculated with phase function

Introduction

- It is quest to produce virtual scene with high detail and physical accuracy.
- Nearly every scene contains participate media.
- Accessing volume data dominates the rendering time and space.
- Acceleration data structure → Better performance!



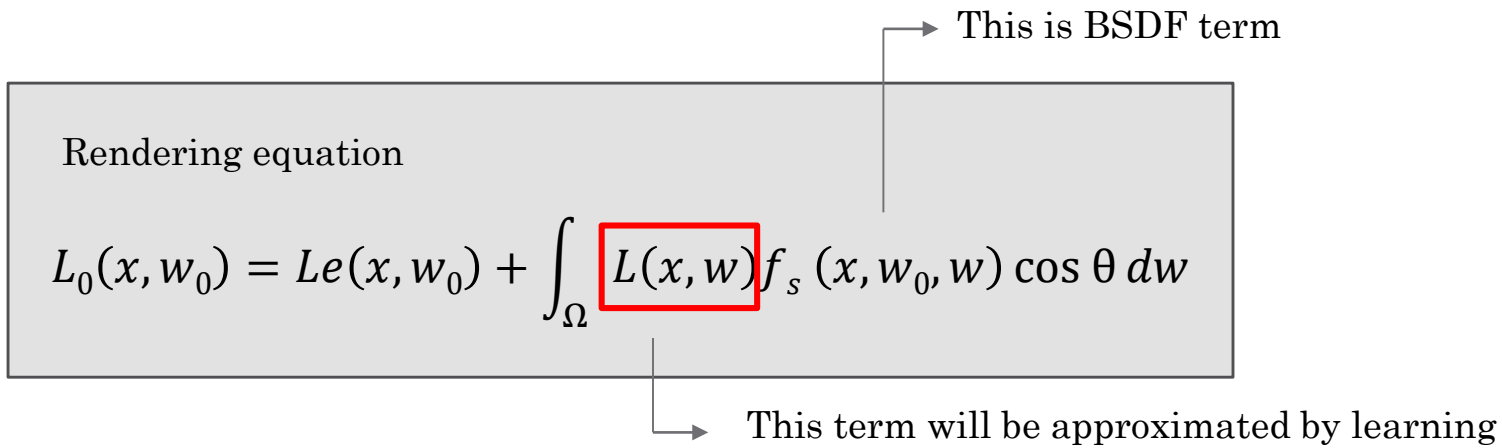
Introduction - motivation

- There is machine learning technique to store and learn distance and directional light samples using special data structure.
- Accelerate the **rendering speed** and save **space** for surface rendering.
- Lets apply this rendering technique for participate media!

Related work

Related work

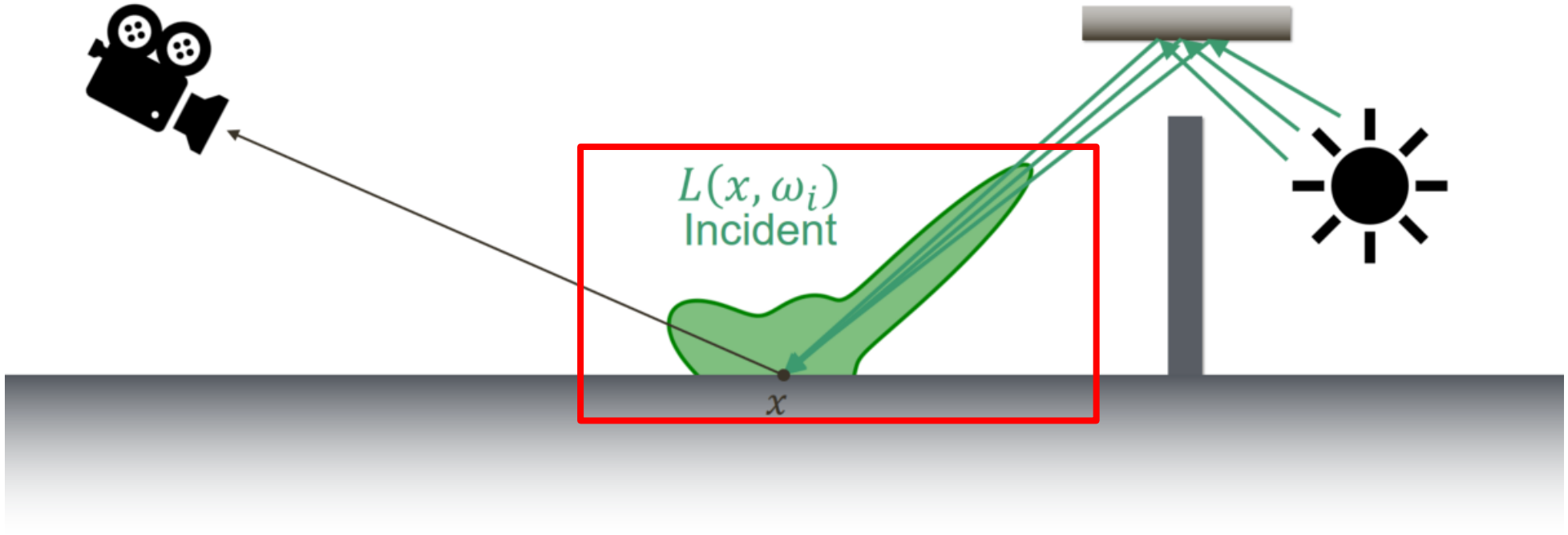
- Approximate the representation of scene's incident radiance field from learning.
- Proposed adaptive spatio-directional hybrid data structure, SD-tree.



Muller, T., Gross, M., Novak, J.: Practical path guiding for efficient light-transport simulation. ´ In: Proceedings of the Eurographics Symposium on Rendering (2017)

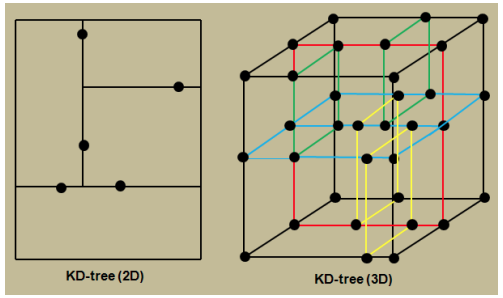
How to Obtain the Incident Radiance for Guiding?

How to compute L?



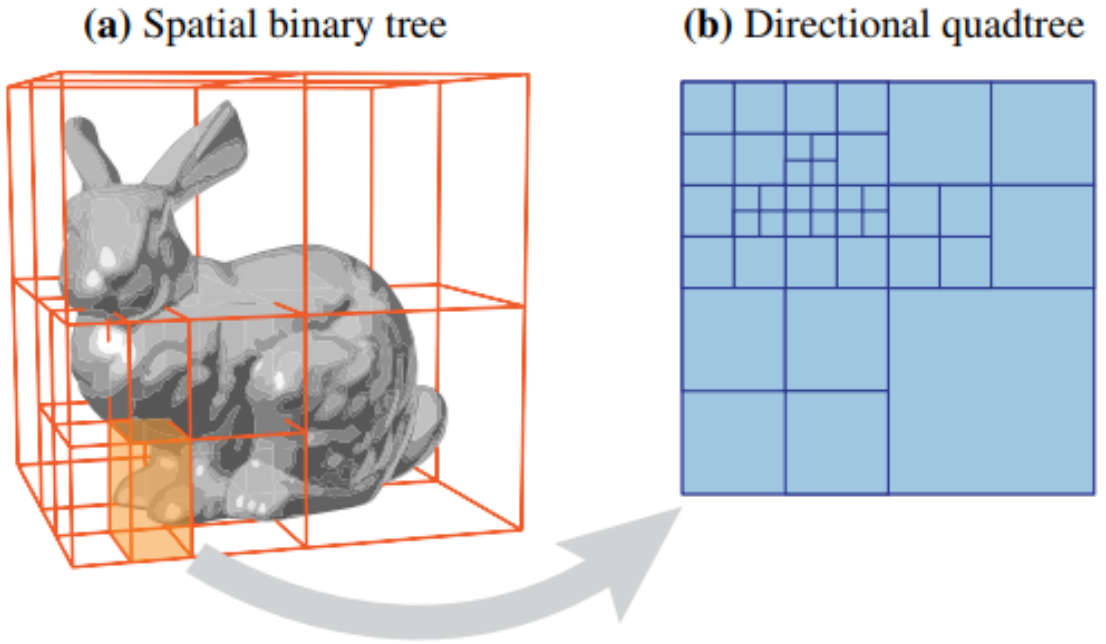
PPT slide Jan Novak's homepage :
http://drz.disneyresearch.com/~jnovak/publications/PathGuide/index.html

Related work



Kd tree is a space-partitioning data structure for organizing points in a k-dimensional space.

- Spatio-directional tree (SD-tree) consist of two level tree.
  - Kd tree that partitions the **3D space domain** of the light field
  - Quadtree that partition the **2D directional domain** of the light field



Muller, T., Gross, M., Novak, J.: Practical path guiding for efficient light-transport simulation. ´ In: Proceedings of the Eurographics Symposium on Rendering (2017)

Related work

- Incident radiance field L will be improved iteratively:
  - Accuracy increases as iteration proceeds, L_1, L_2, …, L_k
- We double the number of samples across iteration and construct spatial/directional binary tree adaptively.
- Spatial tree is split when a node has more than threshold path vertices.
- Directional tree is split so that each leaf contains similar amount of flux.
- L_k is estimated by combining L_{k-1} and BSDF via multiple importance sampling.

Rendering equation

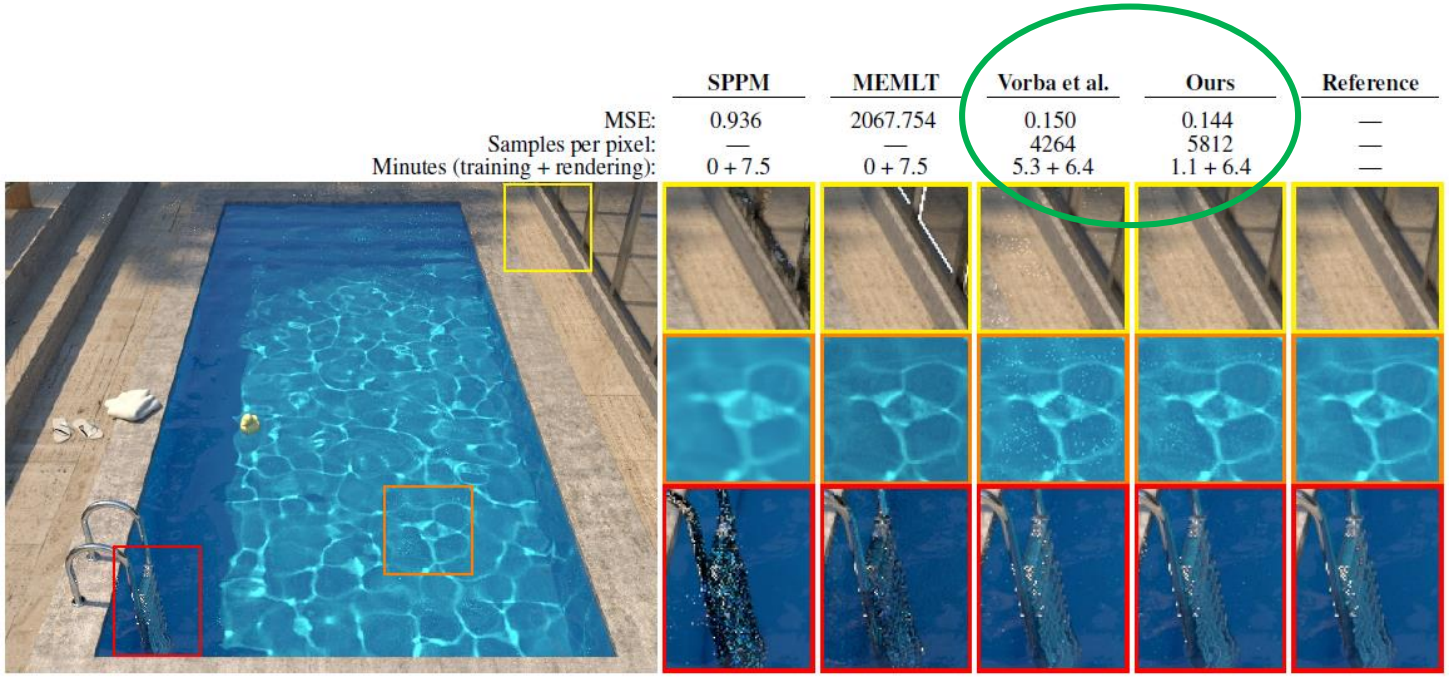
$$L_0(x, w_0) = Le(x, w_0) + \int_\Omega L(x, w) f_s(x, w_0, w) \cos\theta \, dw$$

Muller, T., Gross, M., Novak, J.: Practical path guiding for efficient light-transport simulation. ´ In: Proceedings of the Eurographics Symposium on Rendering (2017)

Related work

Result

- 1.36 times more sample per pixel than Vorba et al. (ACM TOG 2014)'s result.

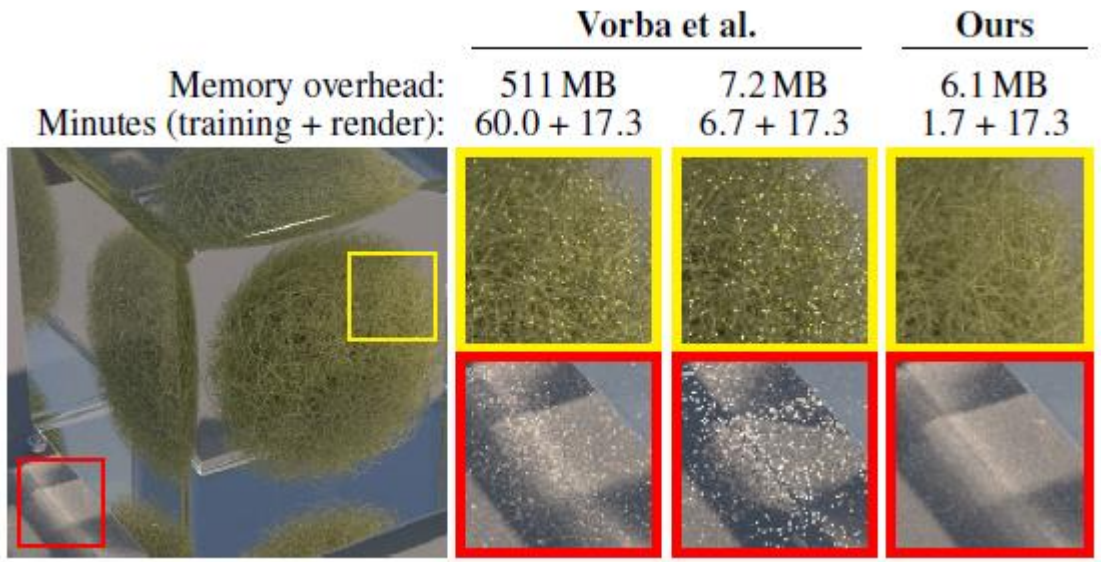


Muller, T., Gross, M., Novak, J.: Practical path guiding for efficient light-transport simulation. ´ In: Proceedings of the Eurographics Symposium on Rendering (2017)

Related work

Result

- 83 times less memory consumption than Vorba et al.'s result.
- Better image quality with less training time.



Muller, T., Gross, M., Novak, J.: Practical path guiding for efficient light-transport simulation. ´ In: Proceedings of the Eurographics Symposium on Rendering (2017)

About our project

Problem

Insight

- The authors choose to build radiance field on spherical directional domain, other than hemispherical one
  - The method can be possibly generalized into volumetric path tracing

Problem

- However, in the authors' implementation, the feature was opted out

> **No Support for Participating Media**
>
> The guided path tracer in this repository was not designed to handle participating media, although it could potentially be extended with little effort. In its current state, scenes containing participating media might converge slowly or not to the correct result at all.

Our goal

1. Extend the current implementation to support participate media

   (Hopefully with little effort, as the original authors mentioned)

2. Figure out what causes the poor performance for volumetric path tracing

3. Implement our solution into current implementation

Role

Joowon Lim

- Theory analysis

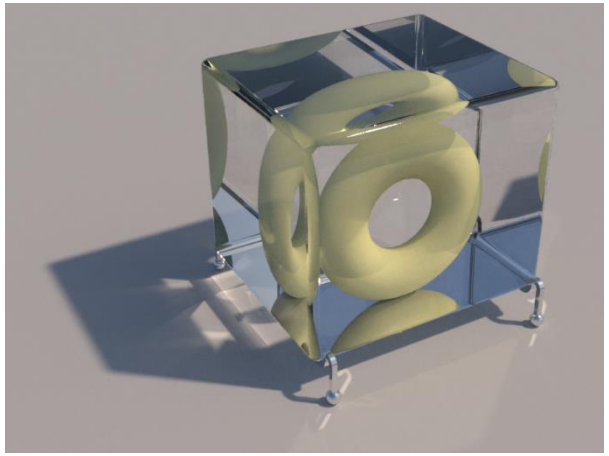
Hyunsoo Kim

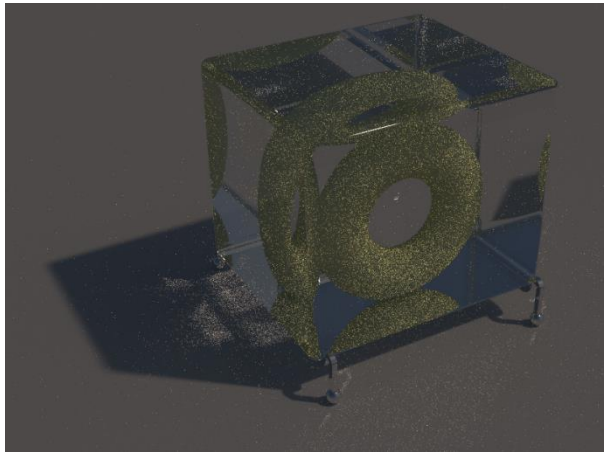
- Programming

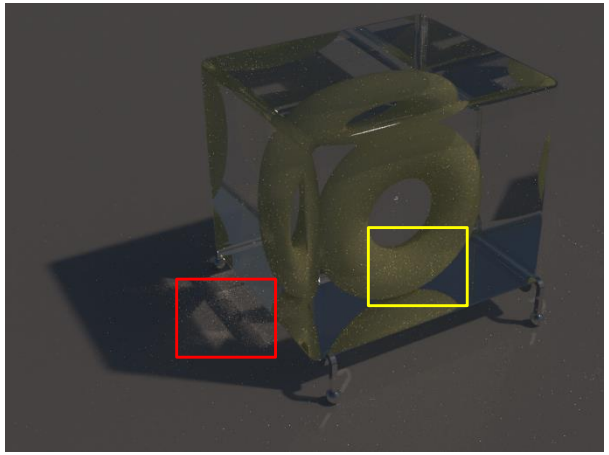
What we have done

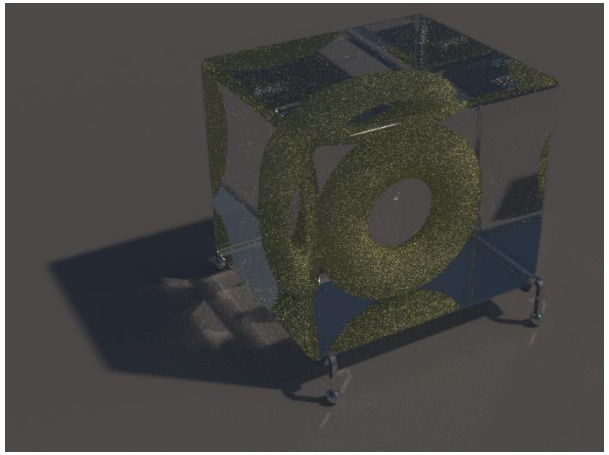
Path guiding without participating media

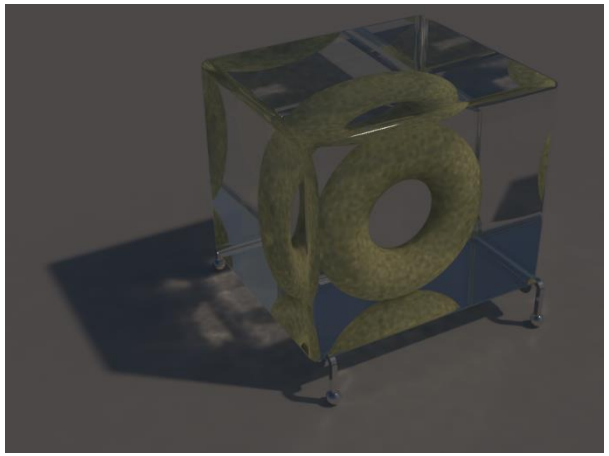
- Test scene – Torus within a glass cube
  - Contains challenging features
    - Caustics
    - Specular-diffuse-specular reflection
- Rendered the test scene without a participating media
  - With rendering methods including guided path tracing method
  - Equal spp (equal time if not applicable) result for each method











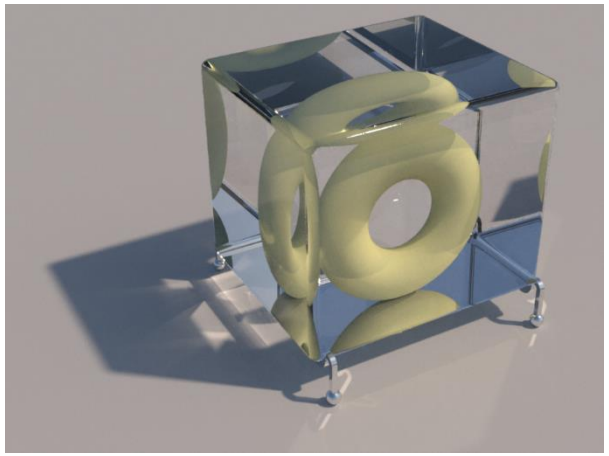
Path tracing

Guided path tracing

Bidirectional path tracing

Photon mapping

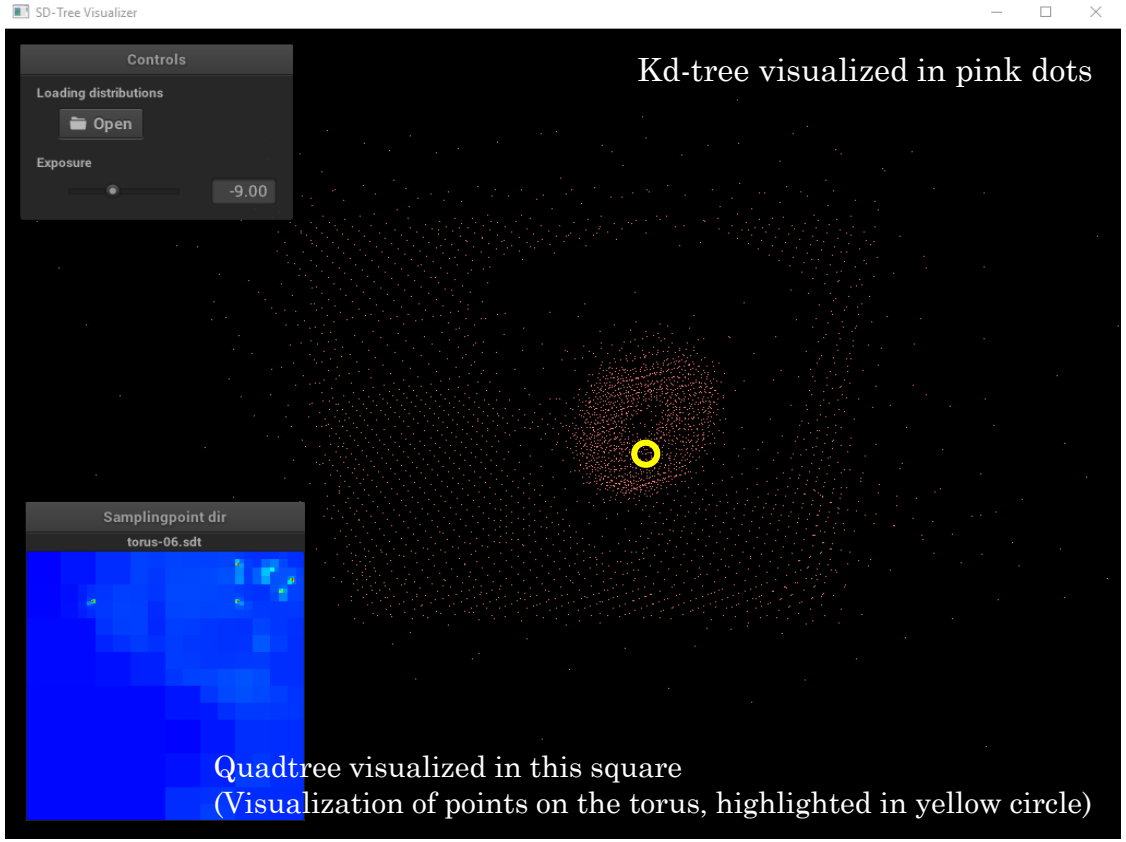
Ground truth



Rendered results seems darker because we messed up with HDR→LDR conversion ☹

Path guiding without participating media

- With visualizer included in the source code, SD-tree can be inspected



Path guiding within participating media

- Supporting a participating medium
  - Mitsuba renderer supports scenes with media
  - Guided path tracing integrator can also render with participating media
    - However, it does not use path guiding; instead it works just same as path tracing

Path tracing within participating media

- Path tracing

```
For each pixel
    (Initialize a ray)
    (Initialize radiance to 0)
    Do
        (Find intersections)
        (Sample direction from BSDF)
        (Generate a new ray with new direction)
    Until (Russian roulette triggers)
    (Update pixel with resulting radiance)
```

- Volumetric path tracing

```
For each pixel
    (Initialize a ray)
    (Initialize radiance to 0)
    Do
        If the ray is in a medium
            (Sample travel distance in the medium)
            (Sample direction from phase function)
            (Generate a new ray with new direction)
        Else
            (Find intersections)
            (Sample direction from BSDF)
            (Generate a new ray with new direction)
    Until (Russian roulette triggers)
    (Update pixel with resulting radiance)
```

⚠Caution: Pseudocodes are (over-)simplified to give a sense of a big picture. Details may not be included

Path guiding within participating media

- Guided path tracing

```
(Initialize SD-tree)
For each iteration
    For each pixel
        (Initialize a ray)
        (Initialize radiance to 0)
        Do
            If the ray is in a medium
                (Sample travel distance in the medium)
                (Sample direction from phase function)
                (Generate a new ray with new direction)
            Else
                (Find intersections)
                If rand() < 0.5 then
                    (Sample direction from BSDF)
                Else
                    (Sample direction from SD-tree)
                (Generate a new ray with new direction)
        Until (Russian roulette triggers)
        (Update pixel with resulting radiance)
    (Update SD-tree for each path with resulting radiance)
```

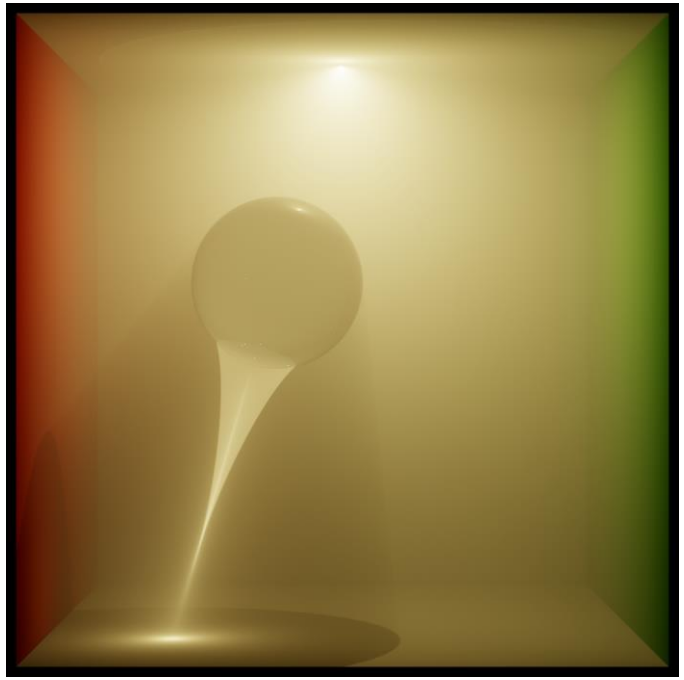
Path guiding within participating media

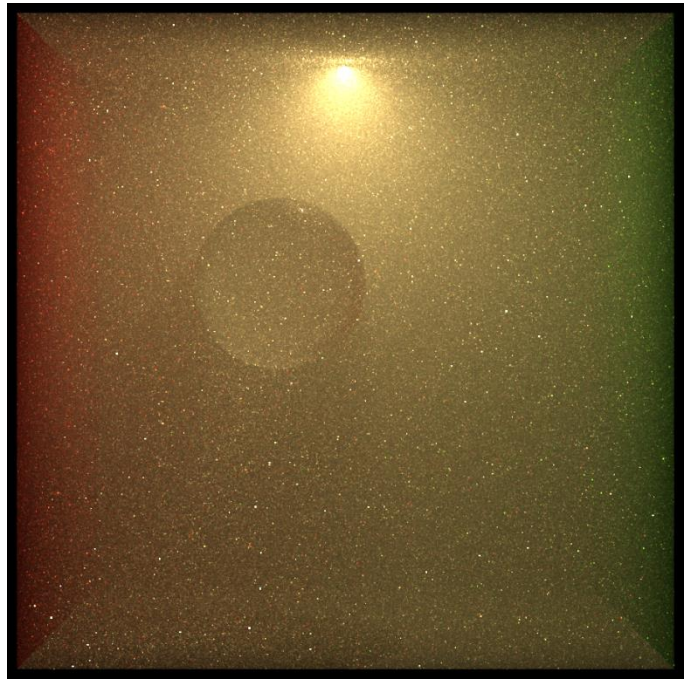
- Volumetric guided path tracing (proposed)

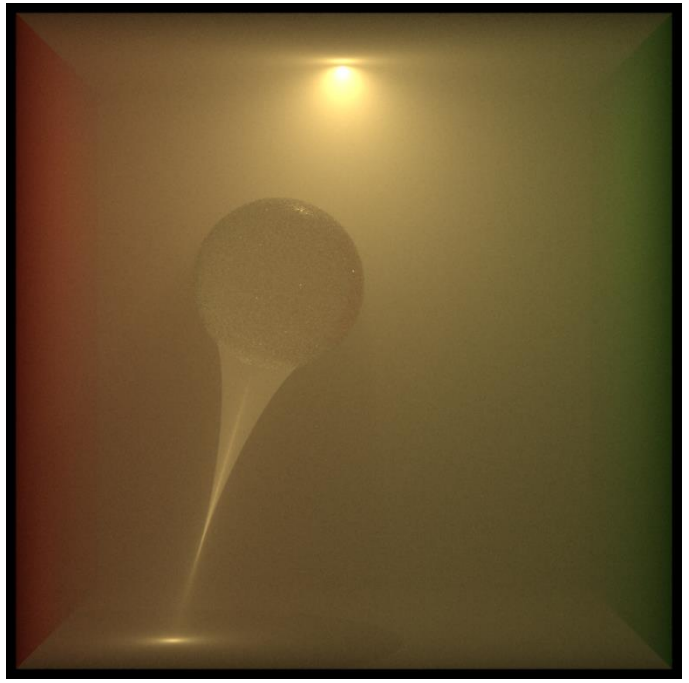
```
(Initialize SD-tree)
For each iteration
    For each pixel
        (Initialize a ray)
        (Initialize radiance to 0)
        Do
            If the ray is in a medium
                (Sample travel distance in the medium)
                If rand() < 0.5 then
                    (Sample direction from phase function)
                Else
                    (Sample direction from SD-tree)
                (Generate a new ray with new direction)
            Else
                (Find intersections)
                If rand() < 0.5 then
                    (Sample direction from BSDF)
                Else
                    (Sample direction from SD-tree)
                (Generate a new ray with new direction)
        Until (Russian roulette triggers)
        (Update pixel with resulting radiance)
    (Update SD-tree for each path with resulting radiance)
```

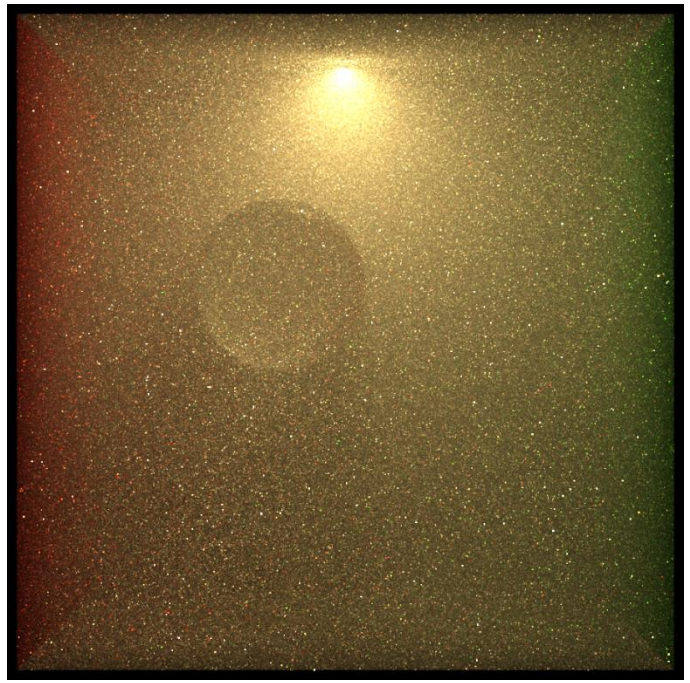
Volumetric rendering test scene

- Volumetric caustics scene
  - Cornell box filled with homogeneous medium
  - Glass ball focusing lights into a point
  - Contains challenging features
    - Scatterings
    - Volumetric caustics
- Rendered the test scene by
  - Volumetric path tracing
  - Bidirectional path tracing
  - Volumetric guided path tracing (Ours)







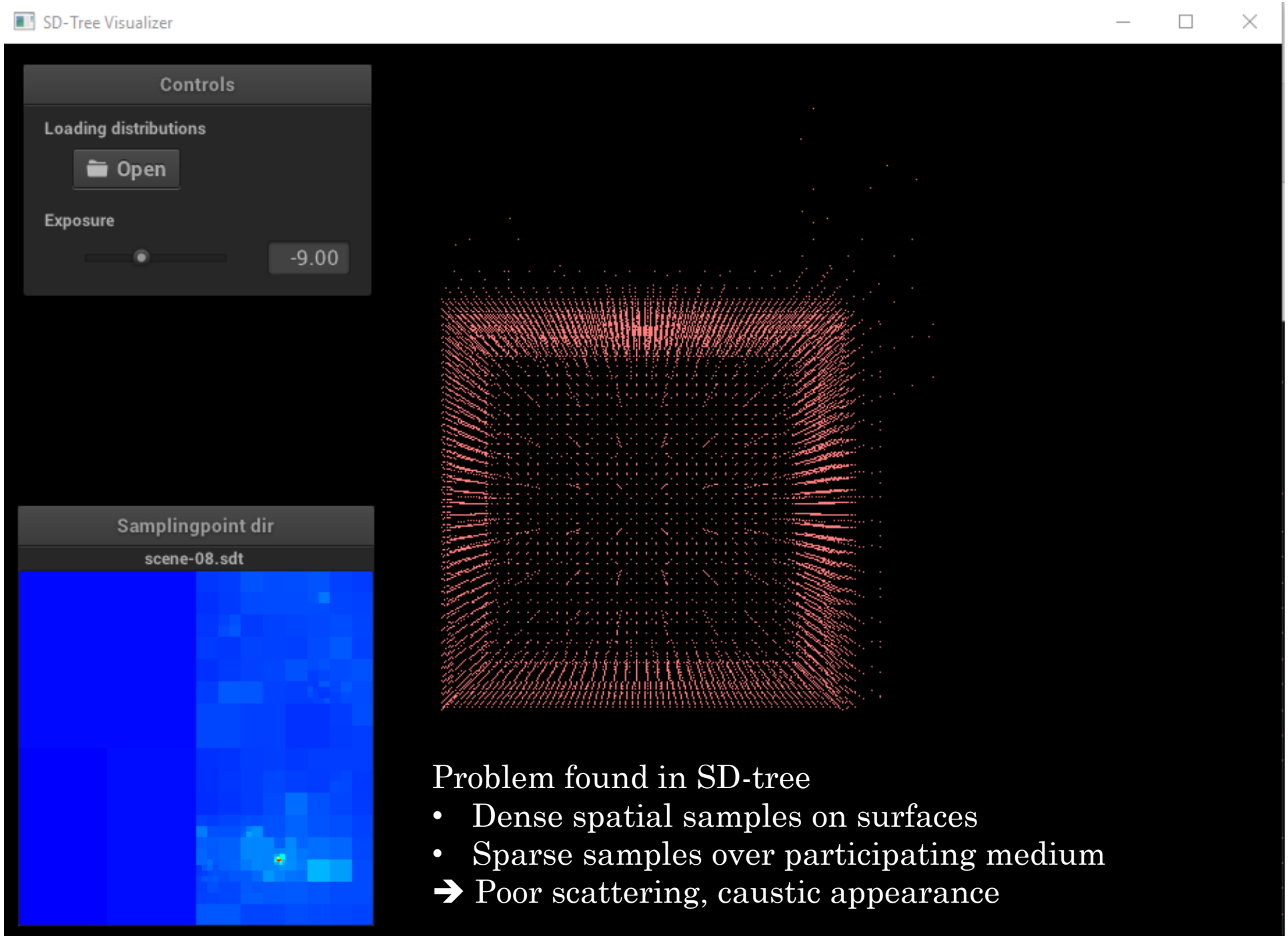


64-spp results

- Top left: Volumetric path tracing
- Top right: Bidirectional path tracing
- Bottom left: Ours
  (Volumetric guided path tracing)

Problems & Discussions

- Does not improve results of scattering
- Does not create visible caustics


SD-Tree Visualizer
Controls
Loading distributions
Open
Exposure
-9.00
Samplingpoint dir
scene-08.sdt
Problem found in SD-tree
• Dense spatial samples on surfaces
• Sparse samples over participating medium
➔ Poor scattering, caustic appearance

Problems & Discussions

- Does not improve results of scattering
- Does not create visible caustics
- Problem found in SD-tree
  - Dense spatial samples on surfaces
  - Sparse samples over participating media

Maybe…

Less collision with a medium than a surface

→ Less spatial samples over media

→ Poor performance

Turns out…

```
* Guided path tracer :
  -  Average path length : 6.52 (437.68 M / 67.11 M)
  -  Total medium hit : 182.343 M
  -  Total surface hit : 102.456 M
```

- More collision with a medium than a surface
  - ~80% more

~~Less~~ More collision with a medium than a surface

→ ?? → Less spatial samples over media

→Poor performance

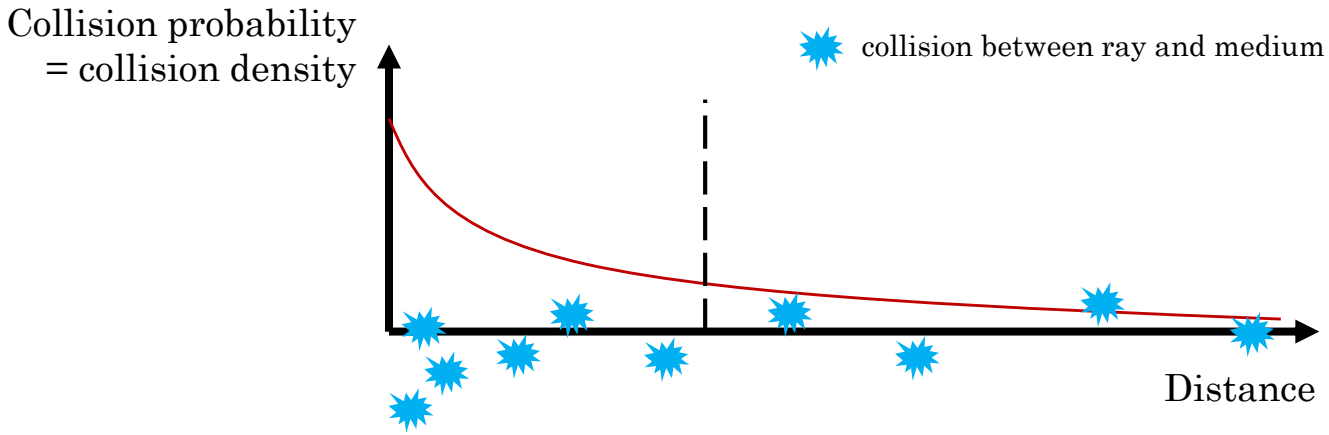
Maybe…

- Collisions are more often in a medium than a surface
  - But, collision density may be lower in a medium
  - Intuitively,
    (#Points to fill a unit cube to certain density) >> (#Points to fill a unit square to certain density)

- Therefore,
  More collision with a medium than a surface
  →But much less collision density (suspected)
  →Less spatial samples over media
  →Poor quality of SD-tree
  →Poor performance

Idea for formal prove

- Collisions in homogeneous medium and kd-tree (1D case)
  - Homogeneous medium
    - → Exponentially decreasing collision probability
  - If there is a surface in some distance,
    - All collisions behind the surface is considered to happen on the surface
  - Applying kd-tree initialization, collision density on the surface is higher
    - Even though collisions are more common in a volume



Finding way to quality SD-tree

- Lower collision density is due to the nature of higher dimension
  - We cannot artificially increase collisions
    - Because it is strictly bounded by the property of a medium

→ We need to gather valuable information only with small samples

Volumetric guided path tracing algorithm (Review)

```
(Initialize SD-tree)
For each iteration
    For each pixel
        …
        [Computing radiance]
        …
    (Update SD-tree for each path with resulting radiance)
```

- In first iteration, with no radiance information in SD-tree, the result is same as plain path tracing
  - The result may be missing high-frequency information such as caustics
- In later iterations, SD-tree is built under the result missing such features
  - Little chance of correction

Improved volumetric guided path tracing (WIP)

```
(Run forward path tracing with a few SPP)
(Initialize SD-tree upon data from previous run)
For each iteration
    For each pixel
        (Initialize a ray)
        (Initialize radiance to 0)
        Do
            If the ray is in a medium
                (Sample travel distance in the medium)
                If rand() < 0.5 then
                    (Sample direction from phase function)
                Else
                    (Sample direction from SD-tree)
                (Generate a new ray with new direction)
            Else
                (Find intersections)
                If rand() < 0.5 then
                    (Sample direction from BSDF)
                Else
                    (Sample direction from SD-tree)
                (Generate a new ray with new direction)
        Until (Russian roulette triggers)
        (Update pixel with resulting radiance)
    (Update SD-tree for each path with resulting radiance)
```

Idea

- Forward path tracing can easily catch high-frequency features like caustics

→ Use it for initialization of SD-tree

Improved volumetric guided path tracing (WIP)

- Potential benefits
  - Starting with SD-tree containing richer information
    - →Containing high-frequency features like caustics
    - →Better SD-tree quality
    - →Better result
  - Initial forward path tracing need a few SPP
    - SD-tree is improved over iterations
      - → Faster than bidirectional path tracing

Improved volumetric guided path tracing (WIP)

- Disadvantages
  - High difficulty of implementation
    - Also the reason why we couldn't implement our proposed algorithm yet

Summary

Summary

- We have implemented path guiding adaptation for volumetric path tracing
- However, naïve adaptation failed to capture challenging features, such as caustics
- After some inspection, we concluded that this is due to inherently lower sampling density within volumes, thus missing high frequency features
- We come up with a hypothesis that initializing SD-tree with forward path tracing will help SD-tree catch those missing information

Future works

- Complete a formal proof that collision density is lower in volumes than on surfaces
- Implement proposed ‘improved volumetric guided path tracing’ method
- Analyze the proposed method with test scenes
  - Bias/variance analysis, ...

Thank you ☺