Real-Time Rendering of Heterogeneous Translucent Materials with Dynamic Programming

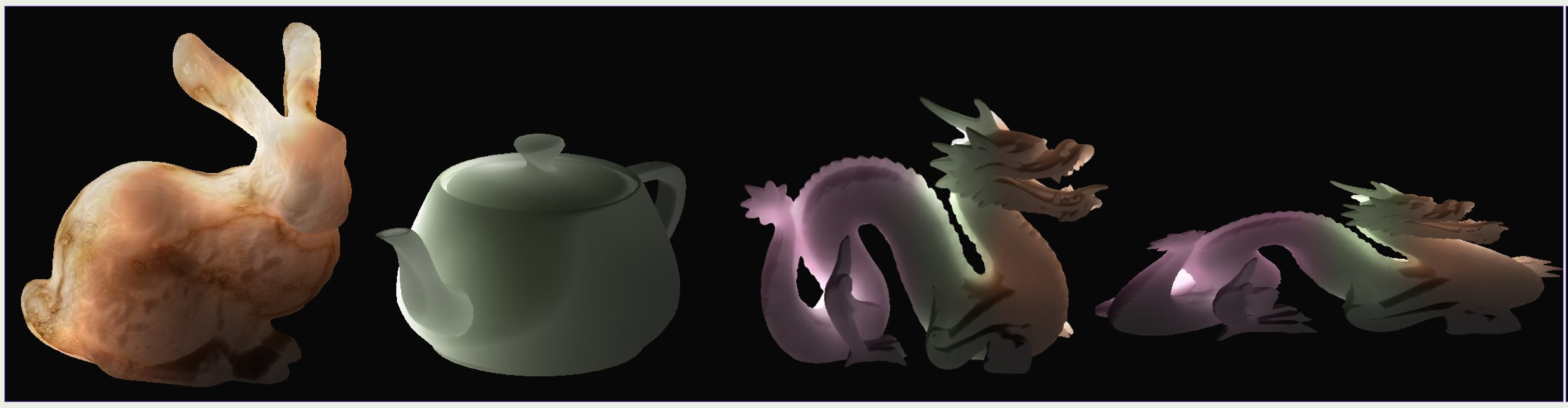
Tadahiro Ozawa[1] Midori Okamoto[1] Hiroyuki Kubo[2] Shigeo Morishima[3]

[1]Waseda University [2]Nara Institute of Science and Technology [3]Waseda Research Institute for Science and Engineering

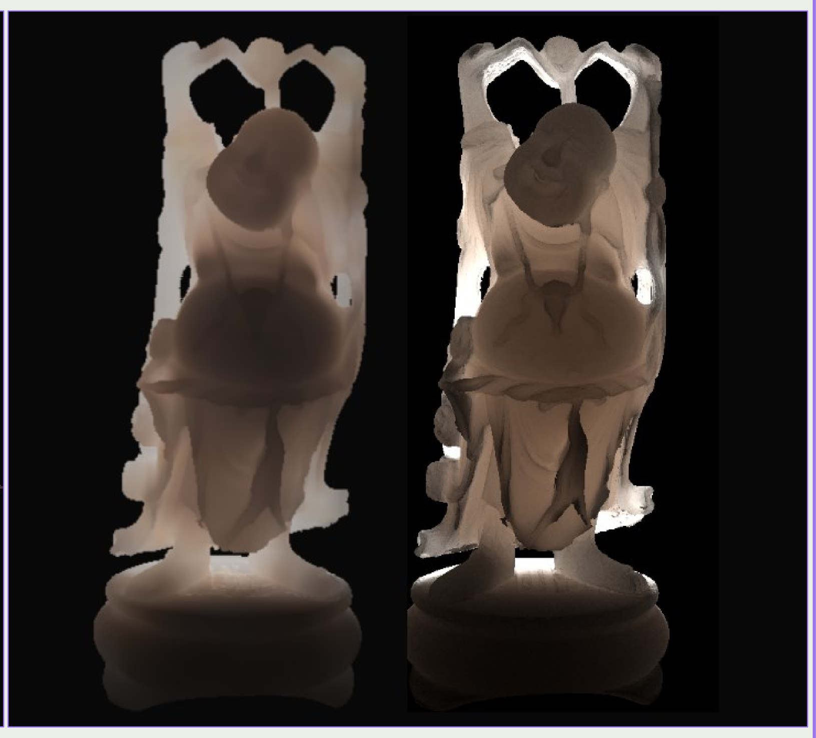
Eurographics

WASEDA University

Rendering of many materials



Bunny (143fps) Teapot (165fps) Deformable Dragon (167fps)

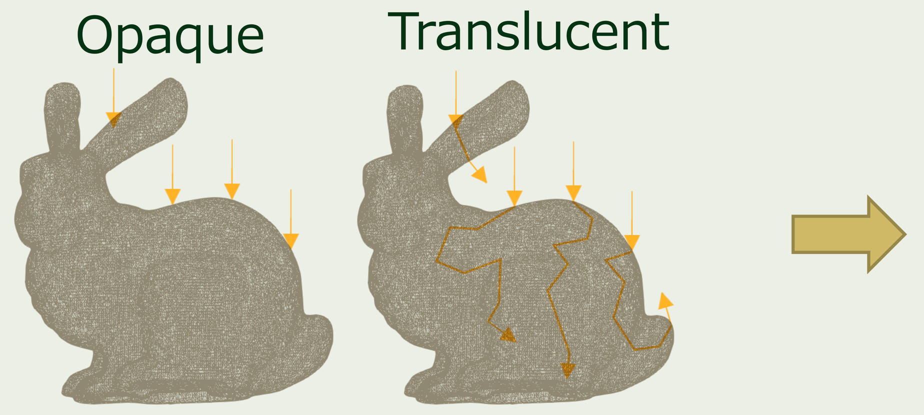


Our result Ground truth

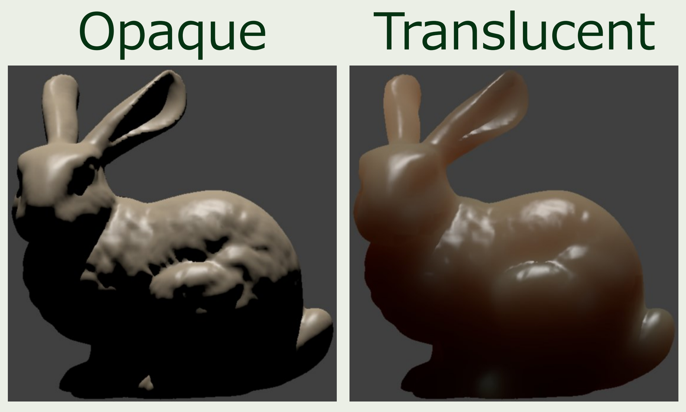
Motivation

Rendering translucent materials in real-time is challenging task.

Subsurface scattering (SSS)



Difference of the behavior of light transport



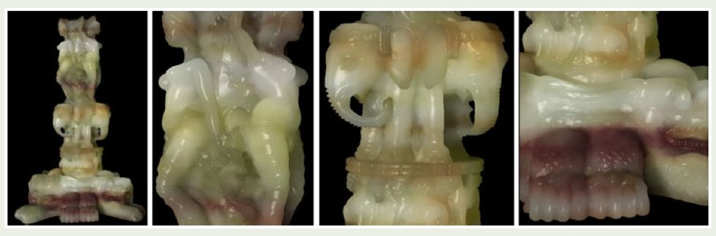
Difference of the appearance

Considering only **"The most important optical path"**.

Related work

Rendering heterogeneous translucent objects at high speed

Volumetric approach

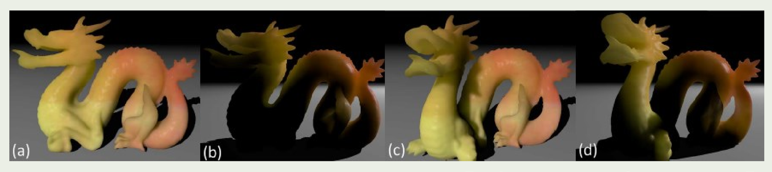


[Wang et al. ,2010]

Solve the diffusion equation in tetrahedron units.

- Find it difficult to handle the deformation with change of the topology like destruction

Real-time rendering



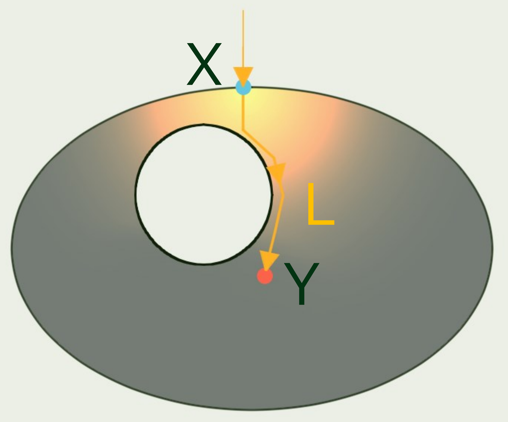
[Chen et al. ,2012]

Multiresolution splatting using several splatting buffer.

- It is necessary to sample many points to synthesize high- quality images.

Our method

The most important optical path

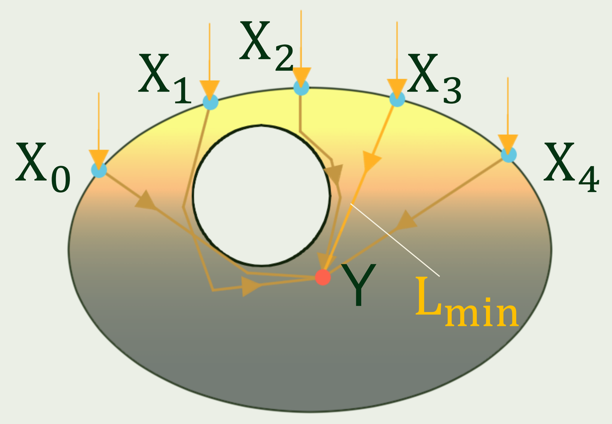


Translucent object illuminated by only single spotlight which irradiation area is microscopic (Narrow beam).

Radiance at X is determined by the distance between incident point X and Y.

Radiance is determined the shortest optical path L.

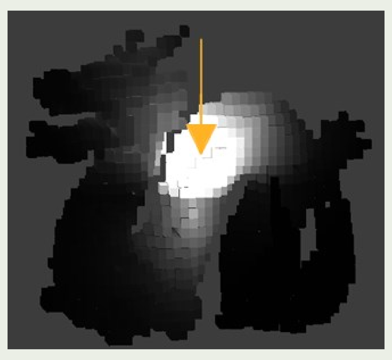
⇒ Radiance can be calculated using Dijkstra Algorithm.

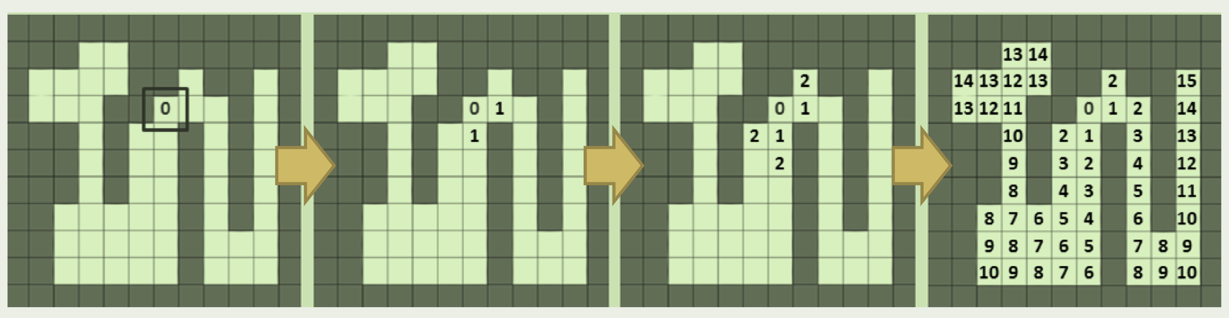


Parallel light source is regarded as a correction of single spotlights.

Radiance is determined the distance between X_3 to Y, the shortest path to the incident points L_{min}.

Dijkstra Algorithm





Create the graph updating value repeatedly.

Radiance Calculation

$$dL(x,\boldsymbol{\omega}) = -(\sigma_a(x) + \sigma_s(x))L(x,\boldsymbol{\omega}) + \sigma_s(x)\int_\Omega (p(\boldsymbol{\omega},\boldsymbol{\omega}',x)L(x,\boldsymbol{\omega})d\boldsymbol{\omega}')\ ds$$

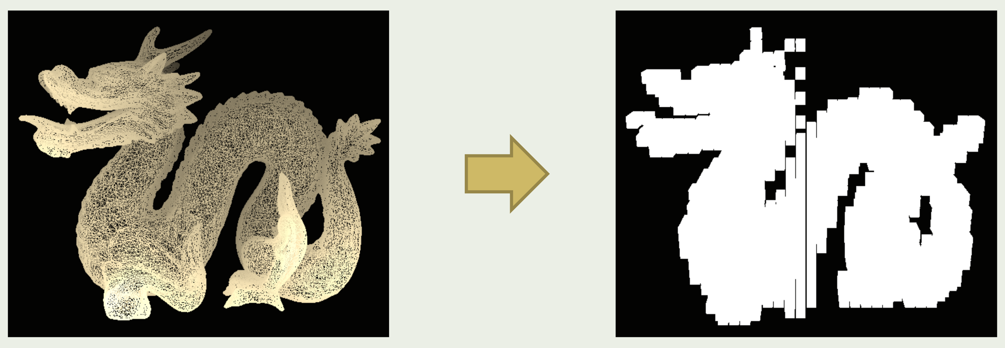
- ✓ Focusing on out-scattering and absorption.
- ✓ Dealing isotropic scattering.

$$\boldsymbol{dL(x) = -(\sigma_a(x) + \sigma_s(x))L(x)\ ds}$$

$L(x,\boldsymbol{\omega})$	:	Radiance
$\sigma_a(x)$	:	Absorption coherent
$\sigma_s(x)$	:	Scattering coherent
$p(\boldsymbol{\omega},\boldsymbol{\omega}',x)$	:	Phase function
$\boldsymbol{\omega}$	:	Incoming light direction
$\boldsymbol{\omega}'$	:	Outgoing light direction
x	:	Position

Implementation

Creation and Initialization of the graph



Graph is created using solid voxelization. Voxels represent nodes and edge is created relative to the adjacent 26 voxels.

A pixel on RSM represents a single small light source.

We apply "*Reflective Shadow Map*" (RSM) to decide the amount of light each voxel receives.

Interpolation of the radiance

We use 3D texture as the same resolution of voxels to interpolate the radiance.