Real-time Rendering of Translucent Materials with Directional Subsurface Scattering

Alessandro Dal Corso

DTU Compute

7th November 2014

Introduction

Problem statement

Our goal is to represent SS phenomena in a synthetically generated image:

- Using a analytical BSSRDF model [Frisvad et al., 2014]
- Visually close as possible to a offline-rendered solution
- That does not need uv-mapping (only position data)
- In real-time or at least at interactive frame rates using a GPUs

The rendering equation

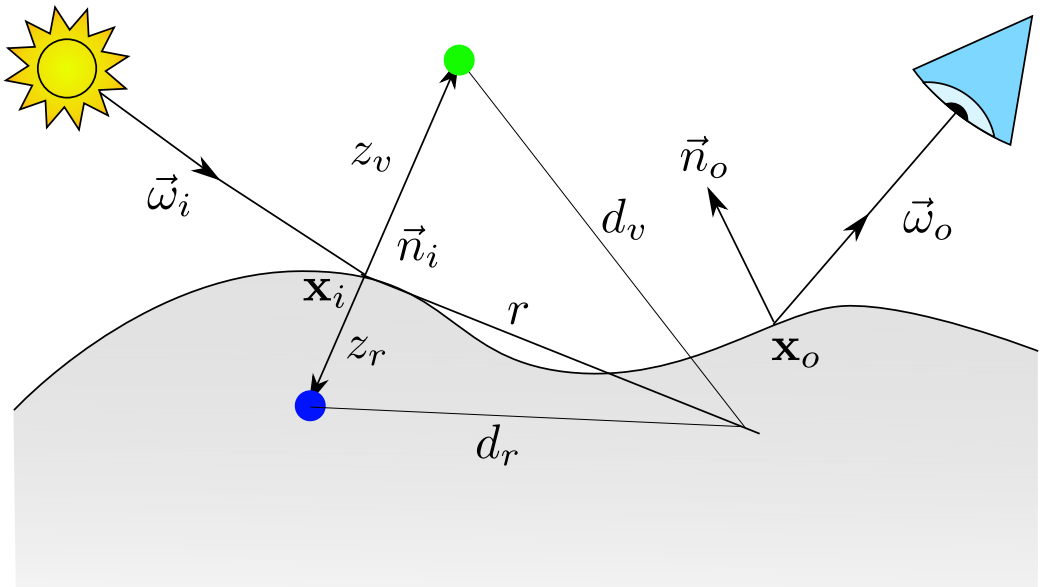
- We use the *area formulation* of the *rendering equation* [Jensen et al., 2001]:

$$L_o(\mathbf{x}_o, \vec{\omega}_o) = L_e(\mathbf{x}_o, \vec{\omega}_o) + \int_A \int_{2\pi} S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_i, \vec{\omega}_i)(\vec{n} \cdot \vec{\omega}_i) d\vec{\omega}_i dA_i$$

- Many BSSRDF functions have been proposed in literature [Jensen et al., 2001; D'Eon and Irving, 2011; Frisvad et al., 2014]

Standard dipole

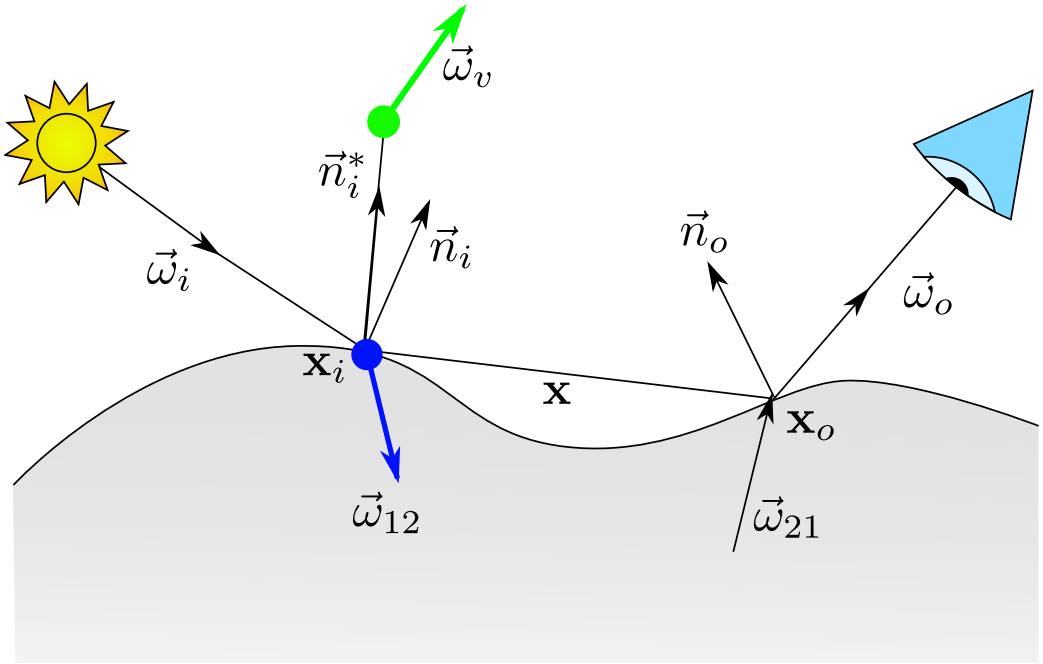
- The first BSSRDF model to be proposed for subsurface scattering (depends only on r)
- It models the interaction as two small light point sources, a virtual and a real source



$$S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_i, \vec{\omega}_o) = \frac{\alpha'}{4\pi^2}\left[\frac{z_r(1+\sigma_{tr}d_r)\,e^{-\sigma_{tr}d_r}}{d_r^3} + \frac{z_v(1+\sigma_{tr}d_v)\,e^{-\sigma_{tr}d_v}}{d_v^3}\right]$$

Directional dipole

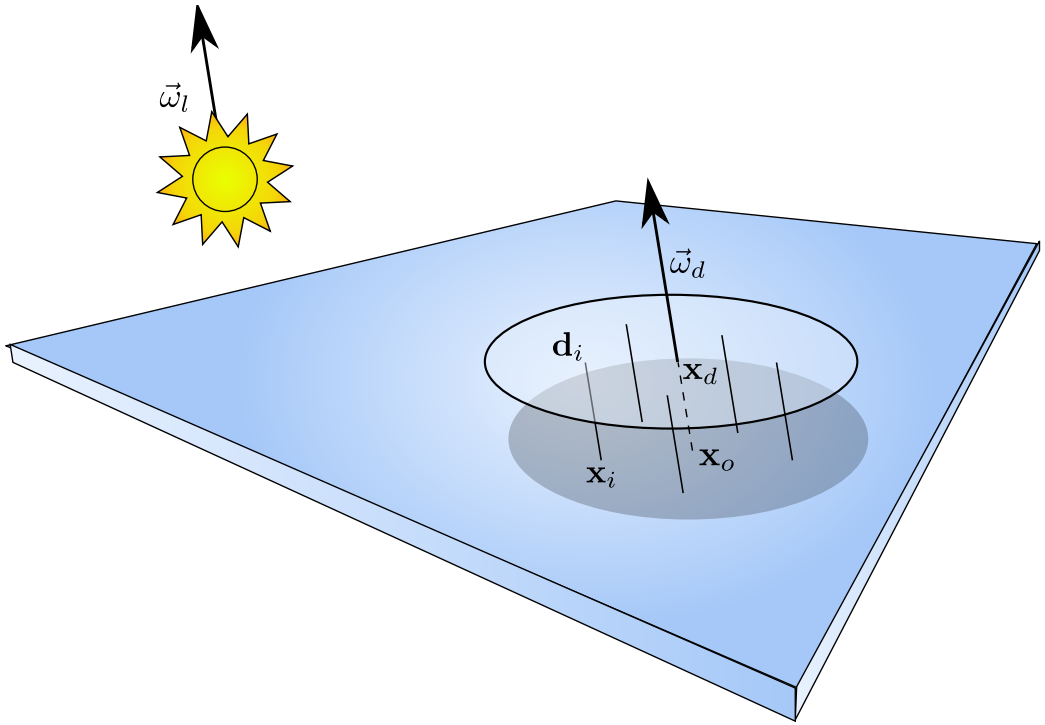
- [Frisvad et al., 2014] defined a new BSSRDF function that keeps the directionality of the incoming light into account
- Depends on the direction of the incoming light $\vec{\omega}_i$, the two points $\mathbf{x}_o$ and $\mathbf{x}_i$ and the two normals



$$S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) = S'_d(\mathbf{x}_o - \mathbf{x}_i, \vec{\omega}_{12}, d_r) - S'_d(\mathbf{x}_o - \mathbf{x}_v, \vec{\omega}_v, d_v)$$

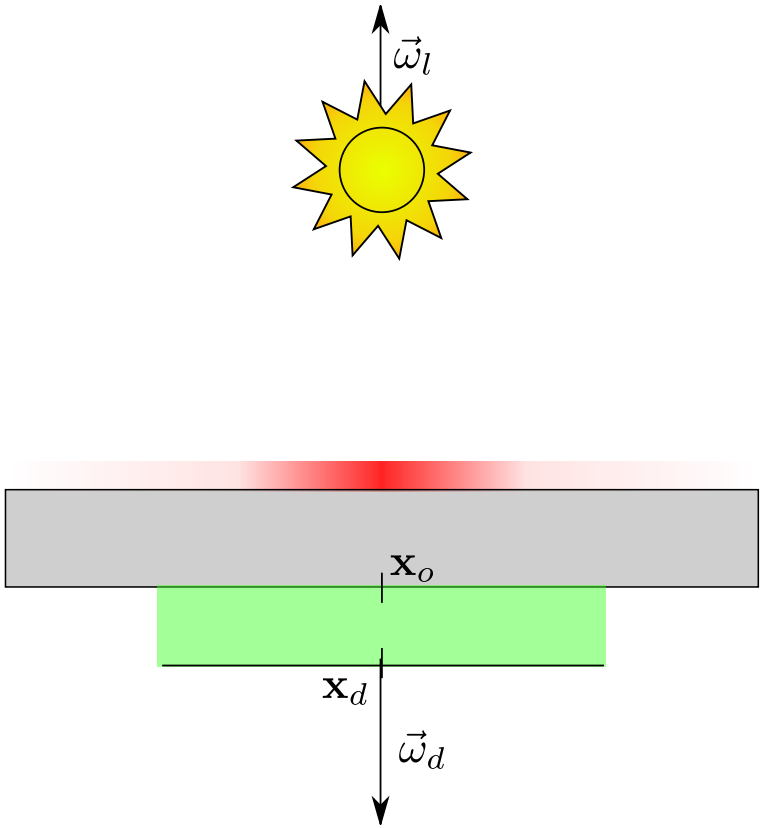
Method

- Scattering effects have a limited range, that depends on the scattering properties of the material (especially on $1/\sigma_{tr}$)
- We can then consider contributions from a disk on the surface
- The disk has a center point $\mathbf{x}_d$ and a direction $\vec{\omega}_d$

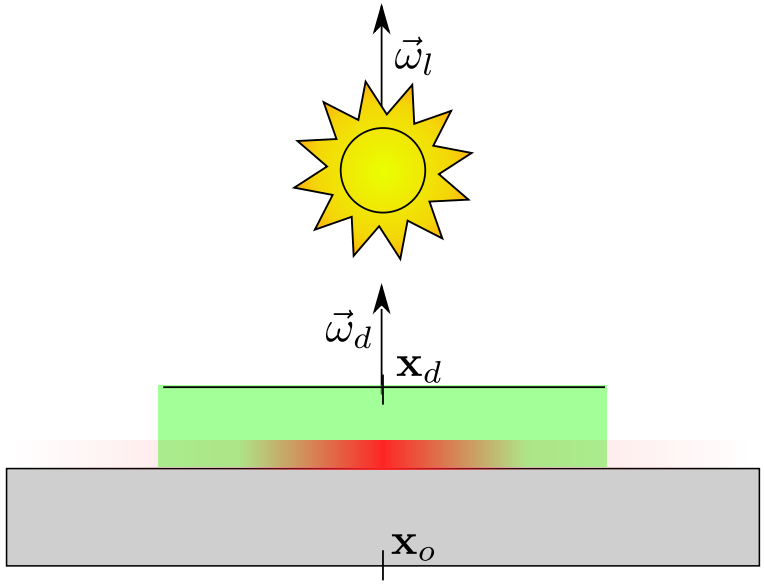


Method

- We place the disk oriented towards the light ($\vec{\omega}_d = \vec{\omega}_l$) and close enough to the light in order to cover the surface



Naïve disk placement



Our disk placement

Method

- Given these assumptions on the disk, we discretize the rendering equation (using an uniform sampling):

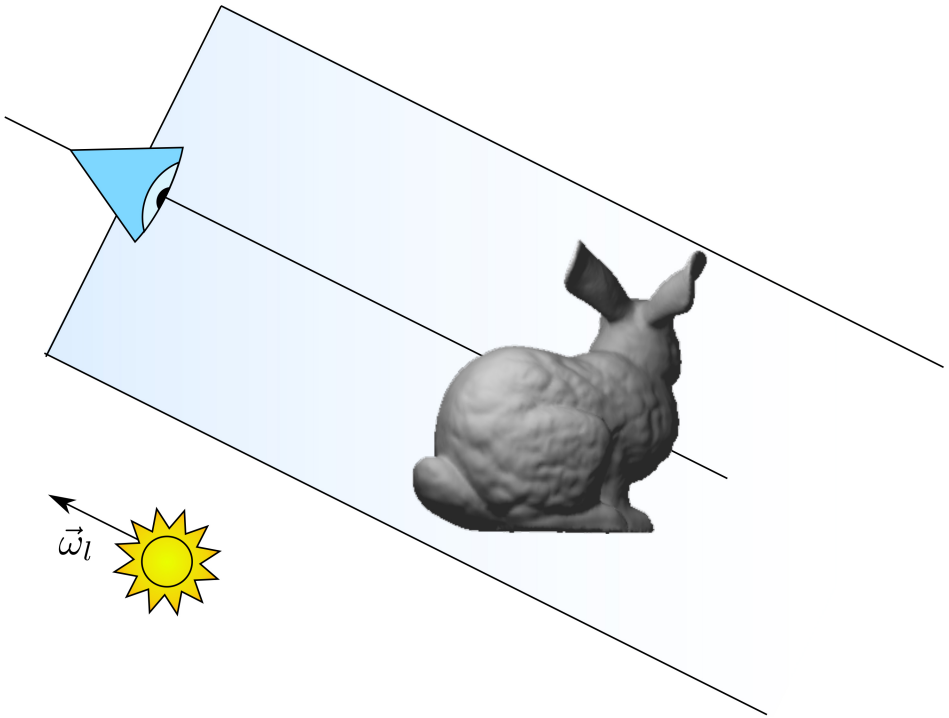
$$L_o(\mathbf{x}_o, \vec{\omega}_o) = L_e(\mathbf{x}_o, \vec{\omega}_o) + \int_A \int_{2\pi} S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_i, \vec{\omega}_i)(\vec{n} \cdot \vec{\omega}_i) d\vec{\omega}_i dA_i$$

$$L_o(\mathbf{x}_o, \vec{\omega}_o) = L_d \frac{A_c}{N} \sum_{i=1}^{N} S(\mathbf{x}_i, \vec{\omega}_d, \mathbf{x}_o, \vec{\omega}_o)(\vec{\omega}_d \cdot \vec{n}_i)$$

- A uniform sampling is not enough, so we need to employ a particular sampling scheme

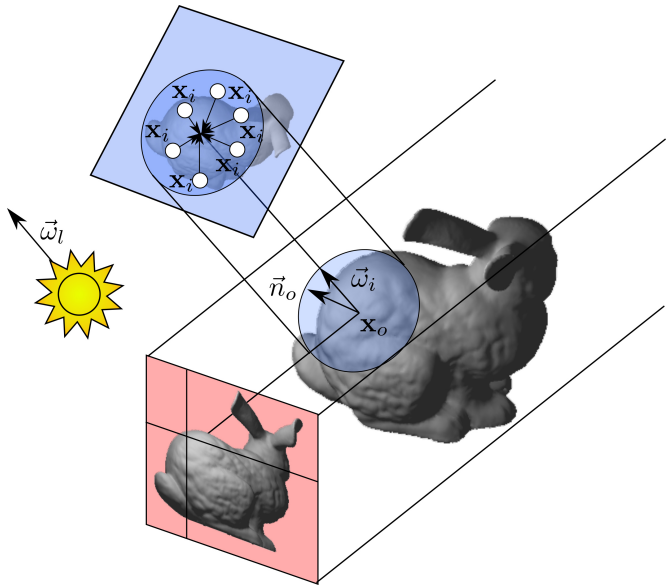
Implementation (step 1)

- We render the scene from the light point of view (as in *shadow mapping*)
- We store positions and normals in a texture, the *light map*
- We get the closest points to light



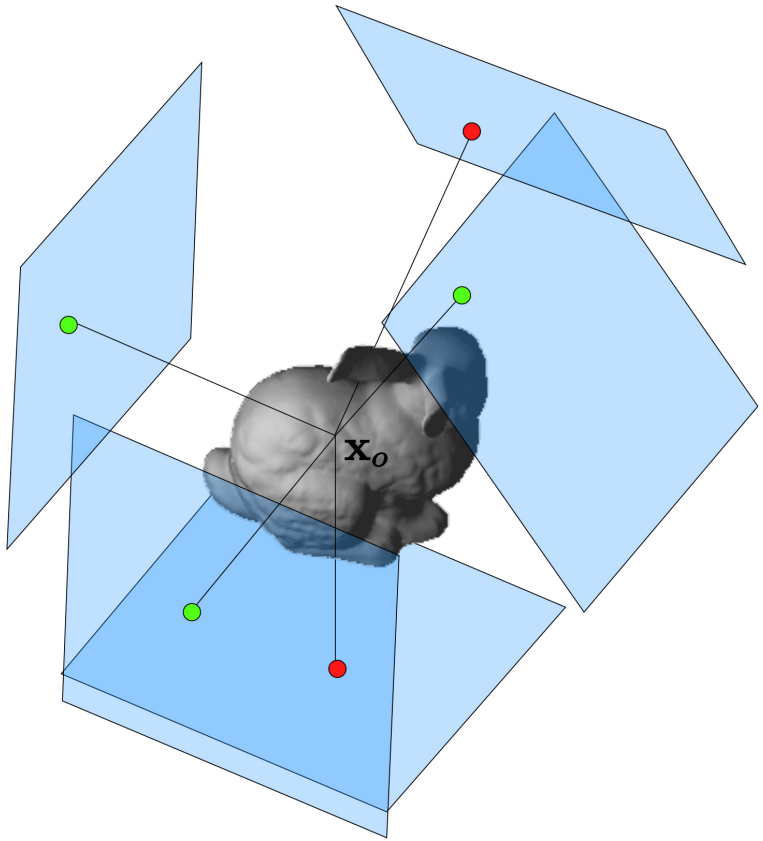
Implementation (step 2)

- We render the object from a certain number of directions in the *radiance map*
- The directions are chosen randomly
- For each pixel we sample the points from the light map and sum up the BSSRDF contribution
- The result is accumulated over multiple frames to reduce noise



Implementation (step 3)

- We finally sample the radiance map to get the single contribution for a point on the surface
- The result is averaged over the directions from which the surface point is visible



Implementation

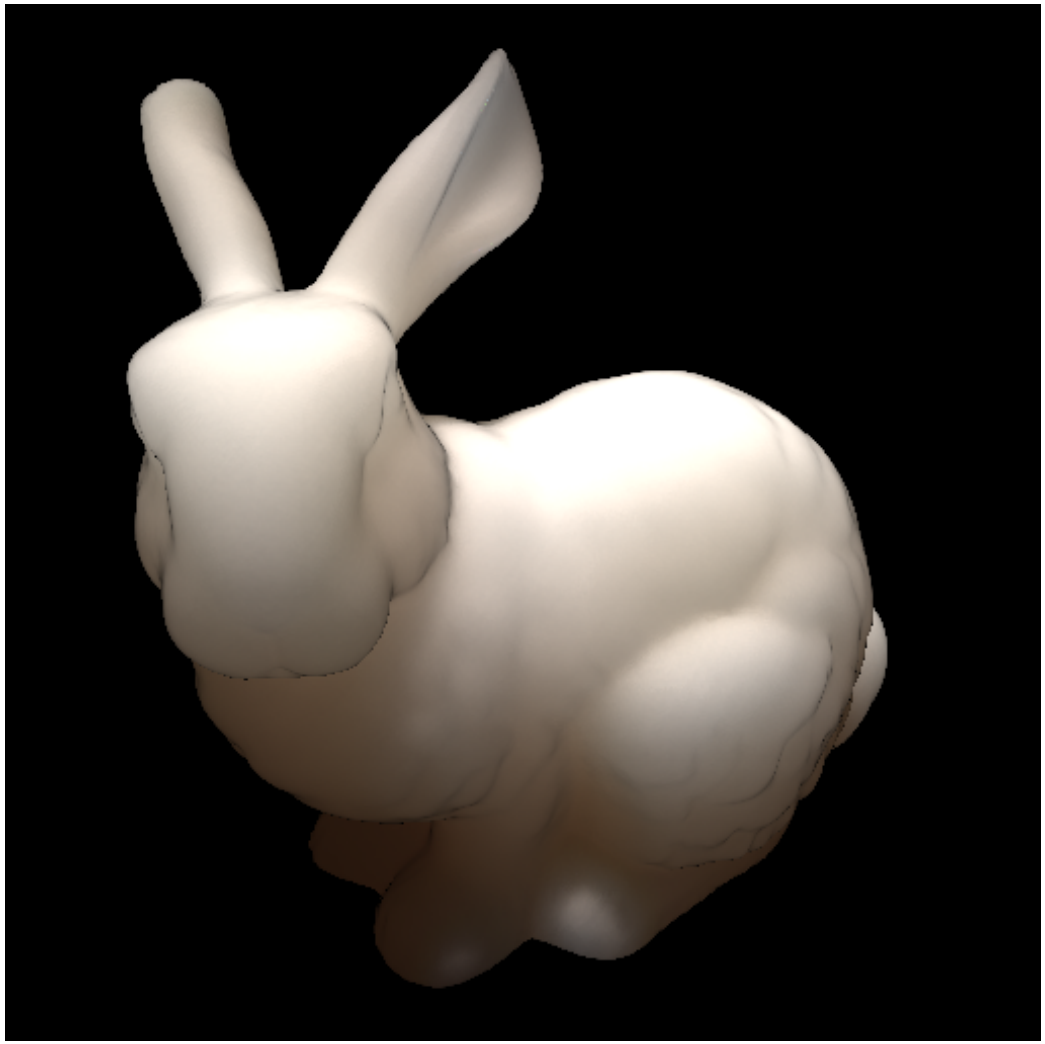


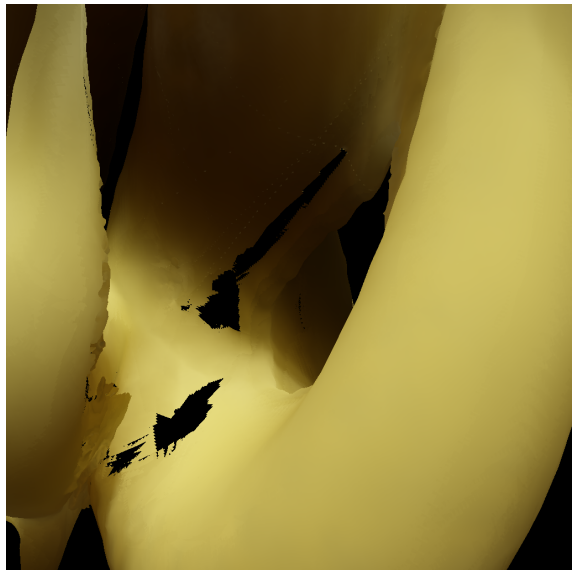
Figure: Stanford Bunny, potato material. Note the self shadowing automatically generated by the algorithm.

Advantages

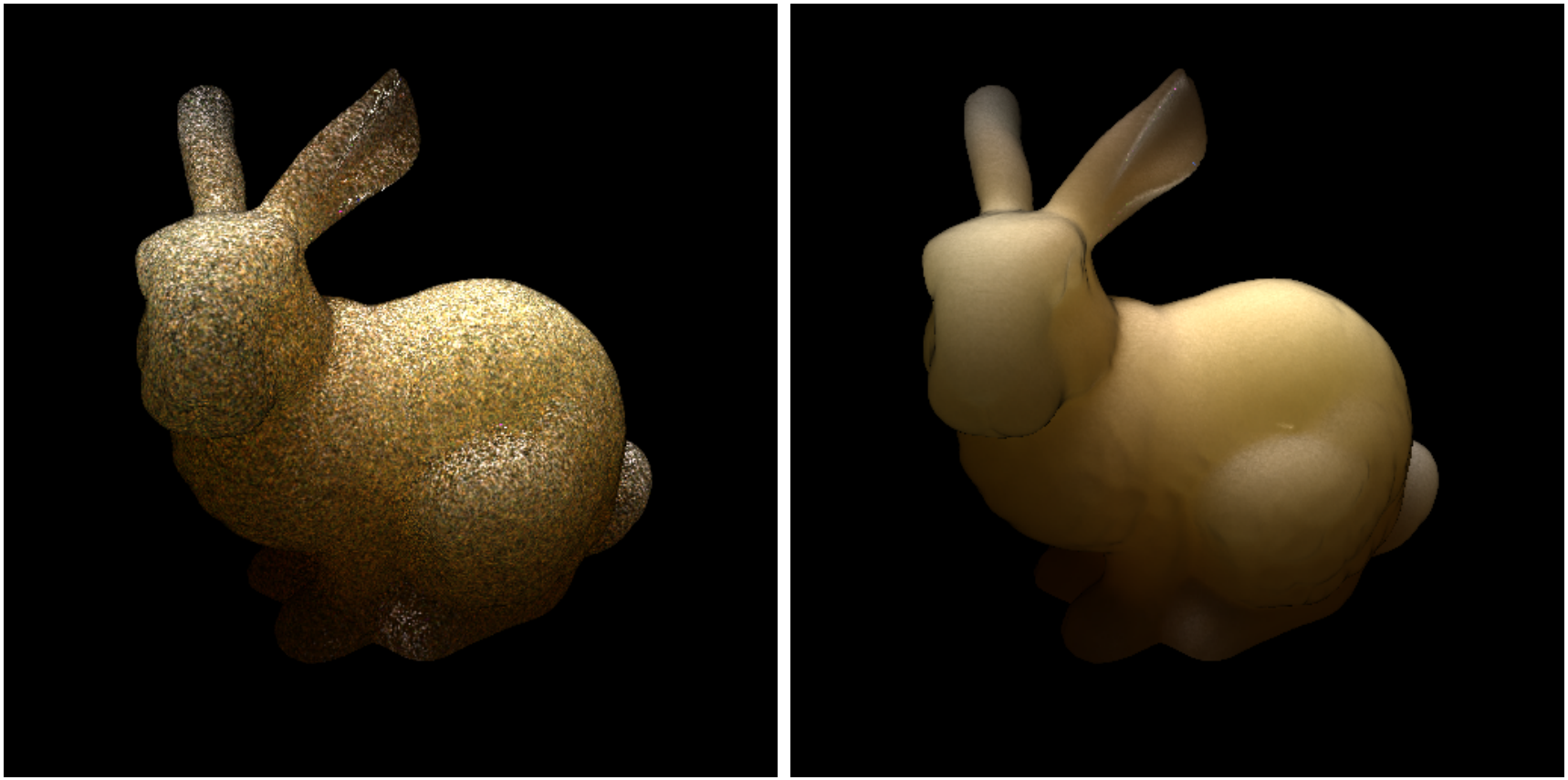
- Accounts for self occlusion
- Accounts for occlusion between different objects
- Directly coupled with an existing shadow mapping pipeline
- Low memory requirements compared to a full voxelization
- No UV mapping is necessary
- The final step can be adapted to forward and deferred shading pipelines

Disadvantages

- Noise, partially overcome by sampling
- Cameras that cover the surface need to be placed manually (to avoid tearing)
- Inherited problems from shadow mapping (constant shadow bias)



Results (quality)



(a)1 frame (5 samples) (b) ca. 100 frames

Figure: Result comparison for the Stanford Bunny model at different times. Parameters for potato.

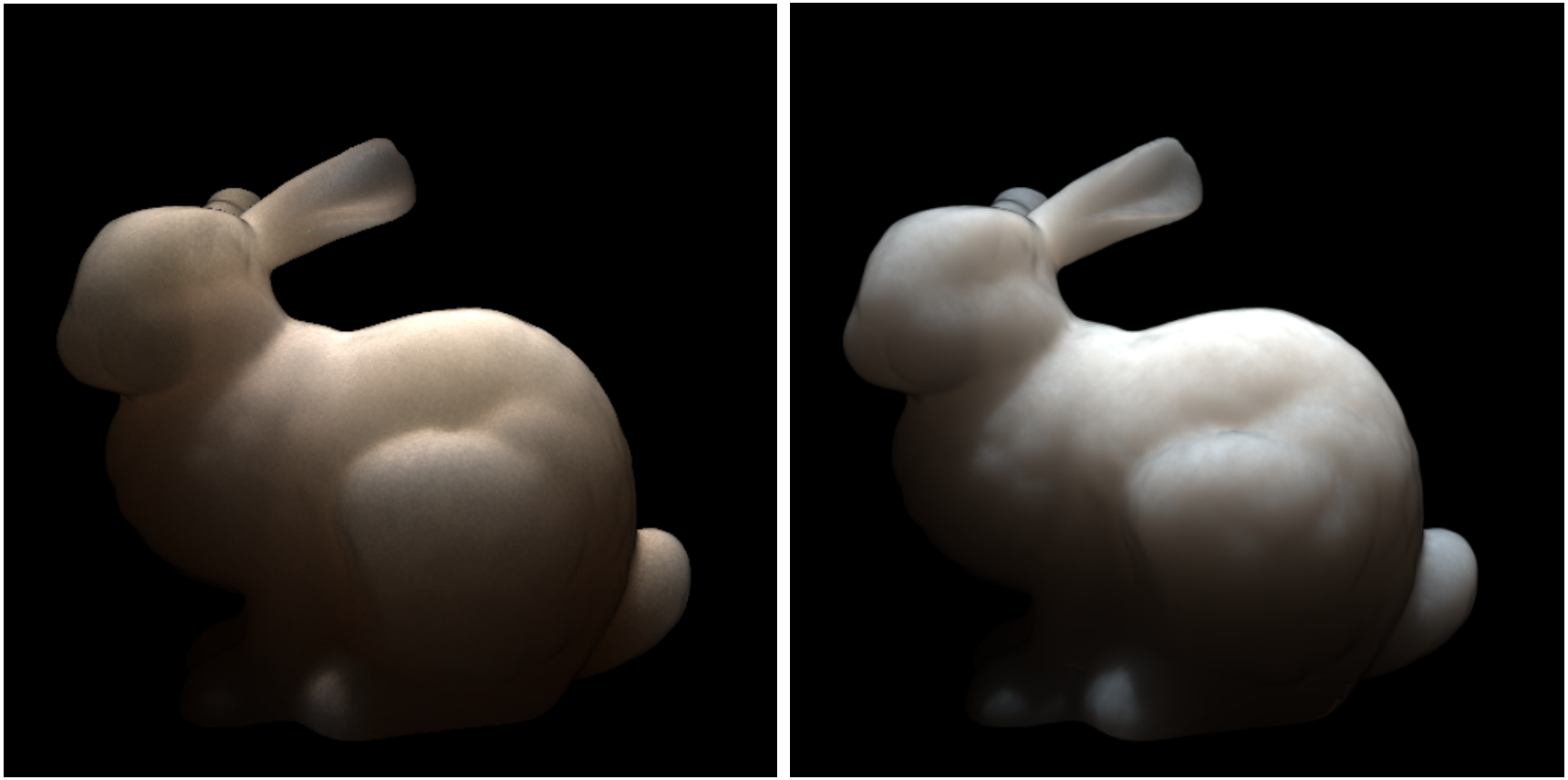
Extensions

- `Multiple lights`
  We sum the contribution of multiple lights in the shader
- `Point lights`
  We scale the incoming intensity by a inverse square distance term

Environment lighting

- We simulate it using 16 directional lights
- We choose "random" points on a environment map
- The distribution chosen accounts to make the points fall in areas with high luminosity

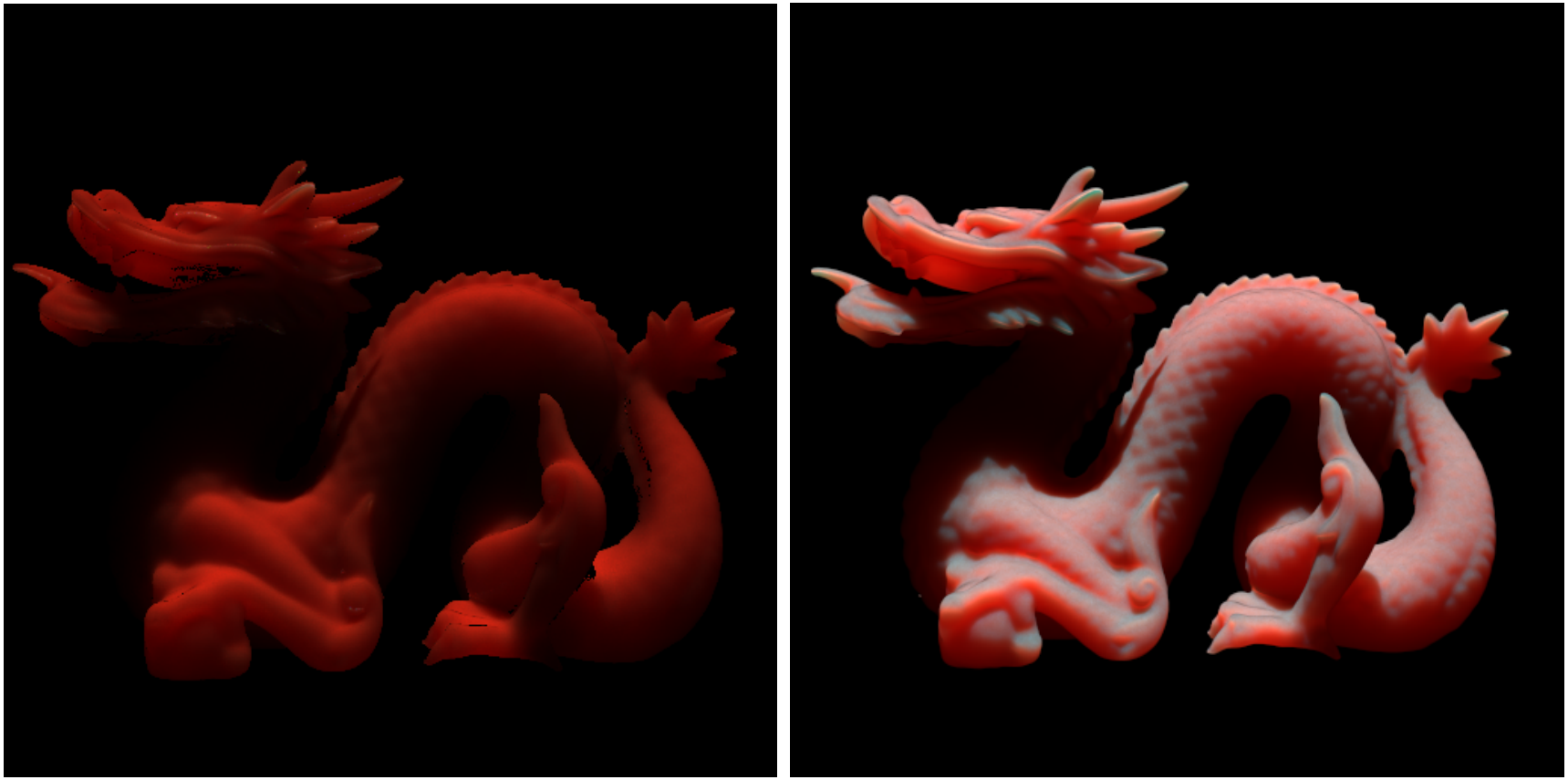
Results (quality)



(a)Our method (b)Reference

Figure: Result comparison for the Stanford Bunny model. Parameters for marble.

Results (quality)

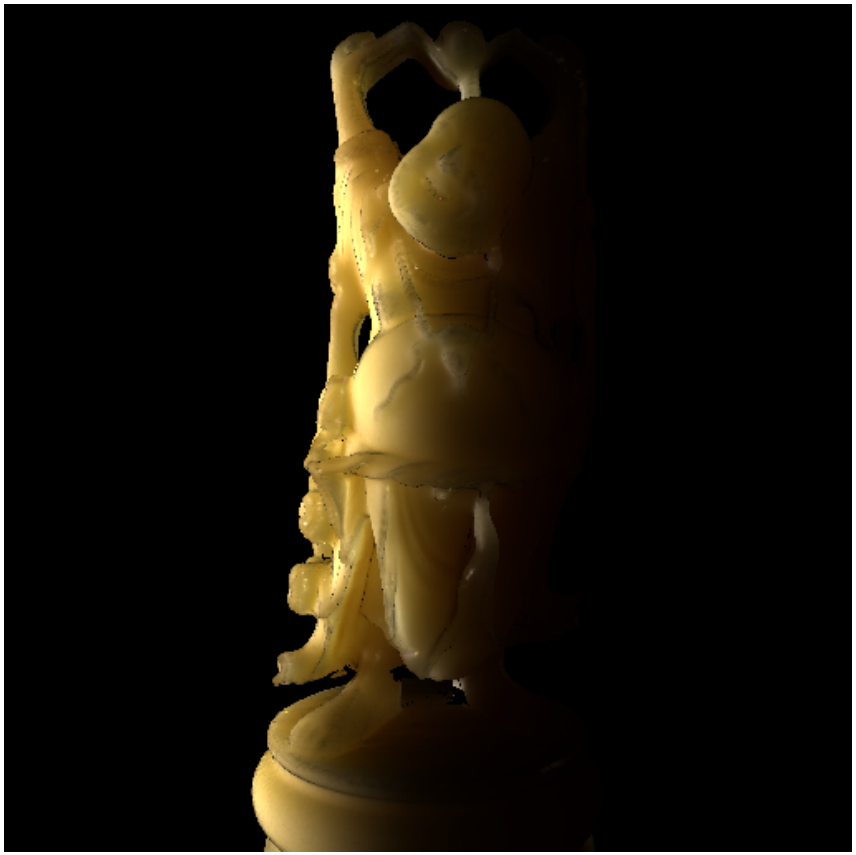


(a)Our method

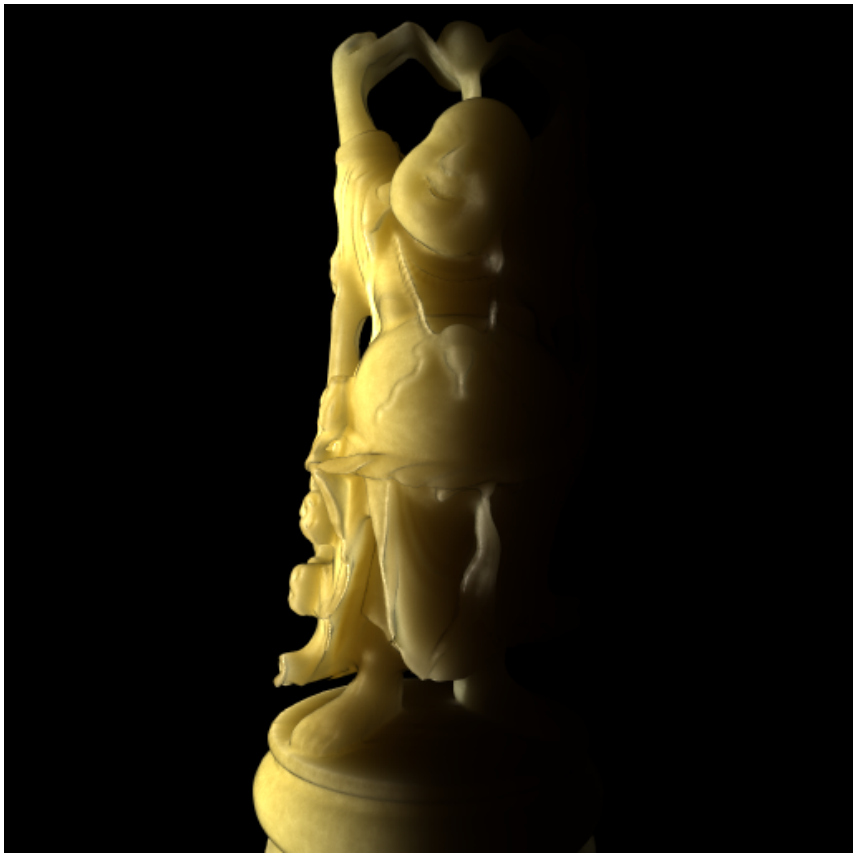
(b) Reference (6 hours, 16 millions samples)

Figure: Result comparison for the Stanford Dragon model. Parameters for ketchup.

Results (quality)



(a)Our result



(b) Reference (30 minutes, 10^6 samples)

Figure: Result comparison for the Happy Buddha model. Parameters for potato.

Results (performance)

Model	#Δ	Number of samples (*N*)			
		1	10	50	100
Bunny	10^4	2.1	5.3	19.8	38.2
Dragon	10^5	12.5	35.2	140.6	275.3
Buddha	10^6	96.7	97.7	128.0	216.0

Table: Timings in milliseconds of our method for different models and number of samples *N* (potato material properties). One directional light, 16 directions for rendering and reconstructing.

Conclusions

What I did so far:

- Improved sampling (various corrections)
- Moved to a GPU random number generation (vs. texture-based)
- Introduced spectrally based computation

What are the next steps:

- Remove undesired color shifts
- Implement fast denoising for the final of the algorithm
- Comparison in performance with Optix ray-traced solution (special course)

Conclusions

Thank you!

References

Eugene D'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, pages 56:1--56:14, New York, NY, USA, 2011. ACM.

J. R. Frisvad, T. Hachisuka, and T. K. Kjeldsen. Directional dipole for subsurface scattering in translucent materials. *ACM Transactions on Graphics, 2014*, -:--, 2014. To appear.

Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 511--518, New York, NY, USA, 2001. ACM.