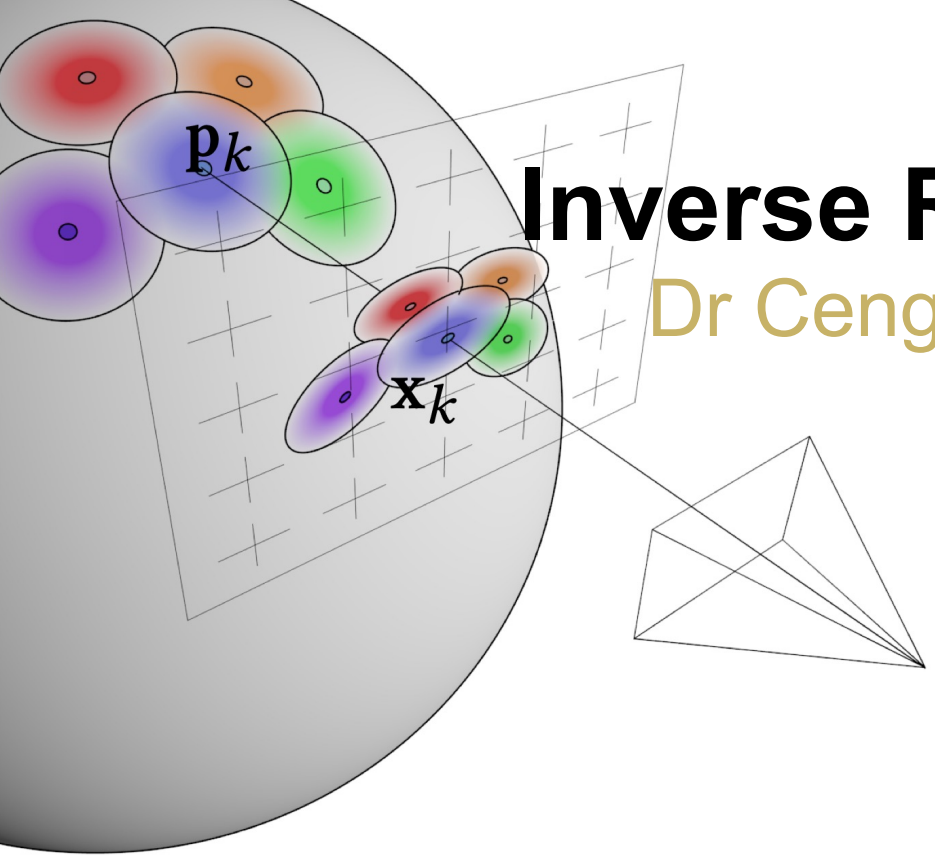
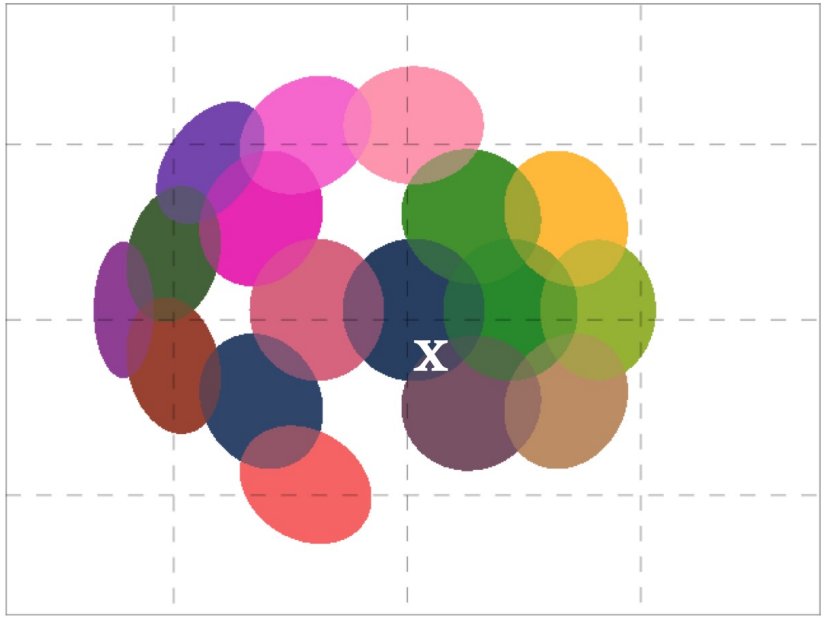


Inverse Rendering

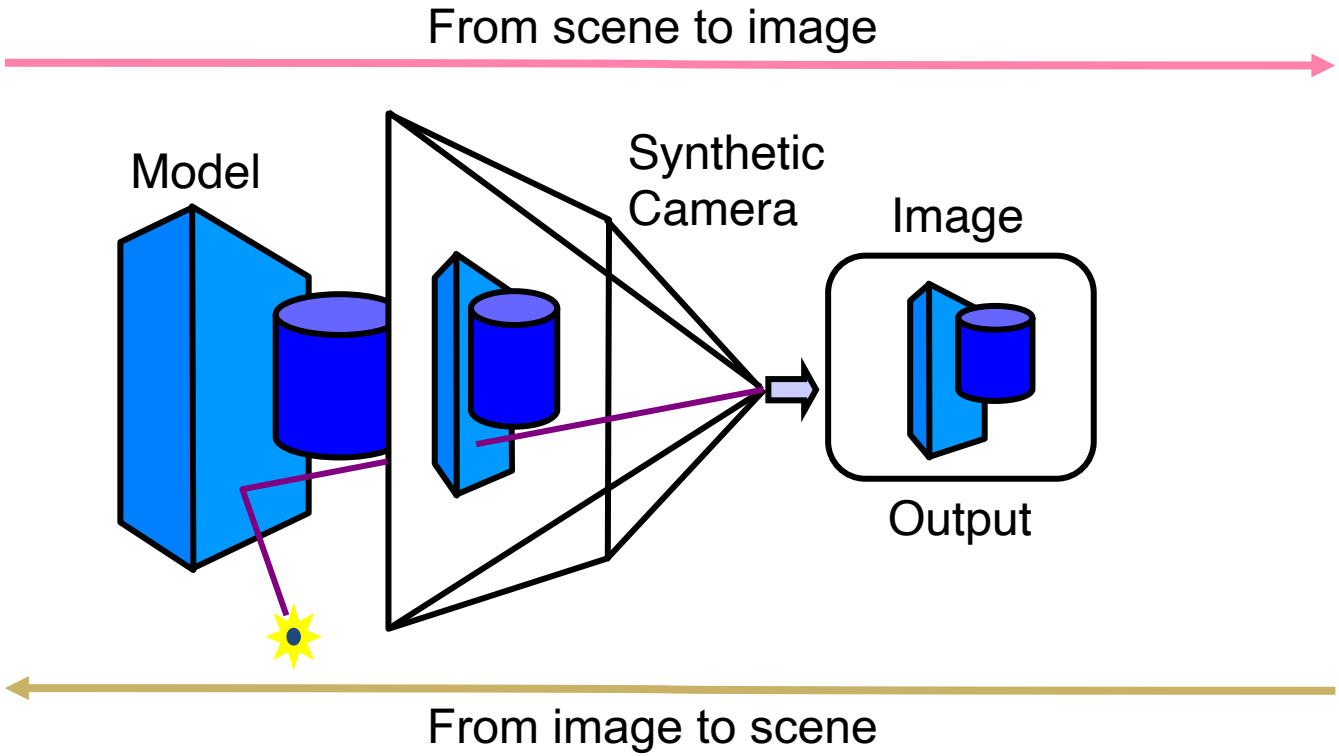
Dr Cengiz Öztireli



UNIVERSITY OF CAMBRIDGE

Forward/ inverse rendering

Graphics is traditionally about forward rendering. That is what we have seen so far: scenes and their renderings. A recent problem in graphics is the inverse of this, i.e. going back from images to scenes via inverse rendering.



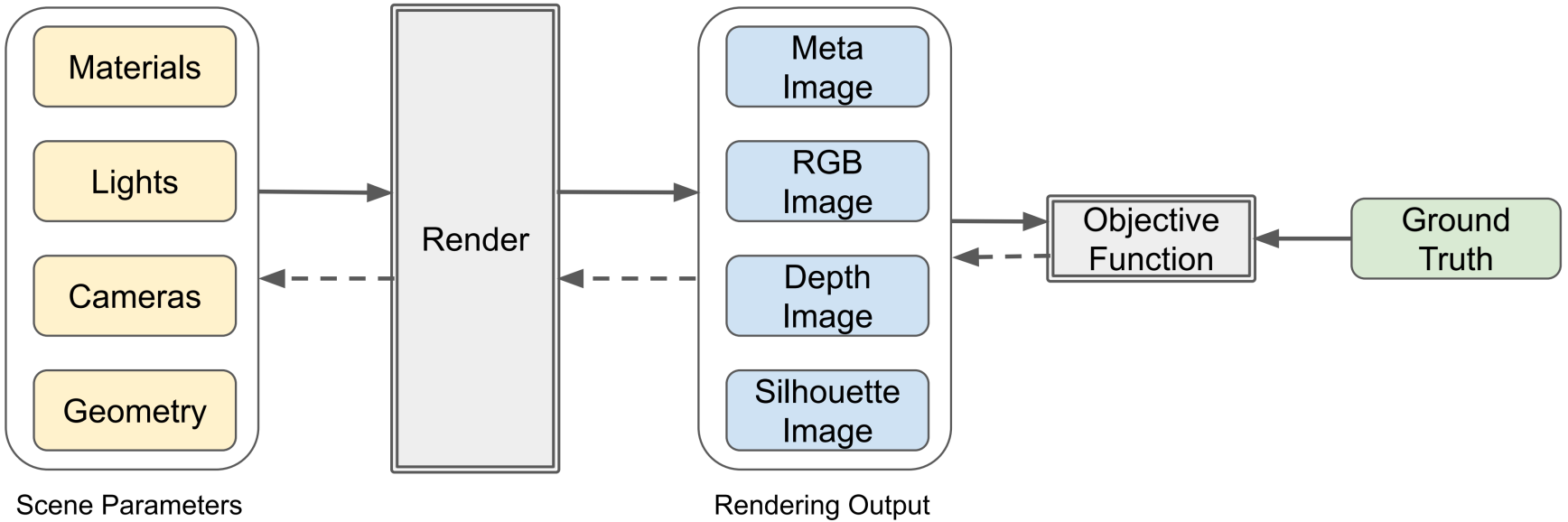
Forward / inverse rendering

Models for
Geometry
Motion
Appearance
Lighting
Camera
…

Forward/ inverse rendering

Forward rendering means taking a scene definition with geometry, materials, cameras, and lights, and generating an image.

Inverse rendering starts from the rendered image, compares it to some form of ground truth image, and updates the parameters of the scene elements with the loss that compares the images.

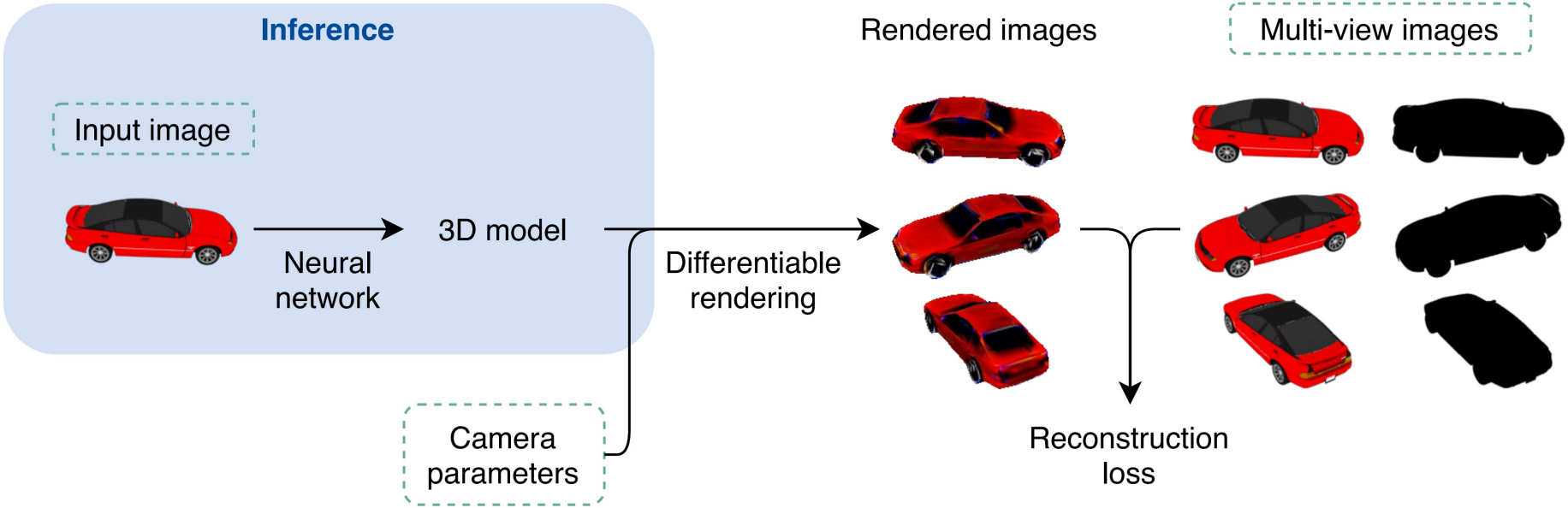


Forward/ inverse rendering

An application is geometry reconstruction from a single image.

At testing (inference) time, we can input an image into a neural network and get the corresponding 3D model. This network is optimized via rendering the 3D model generated from an image and comparing the rendered images with the ground truth images.

Example: 3D geometry reconstruction

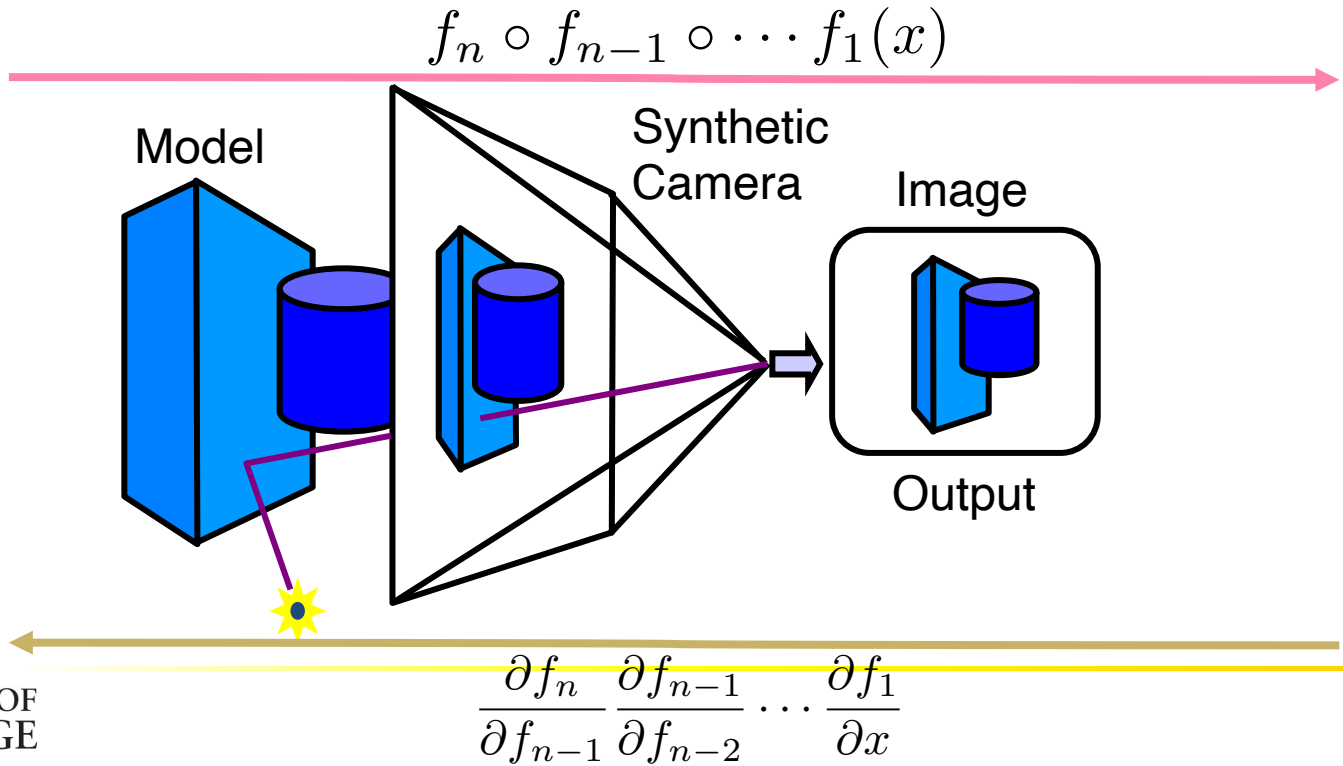


Inverse rendering has many applications.

Inverse rendering

Inverse rendering requires a non-linear optimization: the function from the scene parameters to rendered images is highly non-linear. Deep learning frameworks help us with this optimization, e.g. via stochastic gradient descent. There is one essential requirement for this to work: *we need differentiable rendering*.

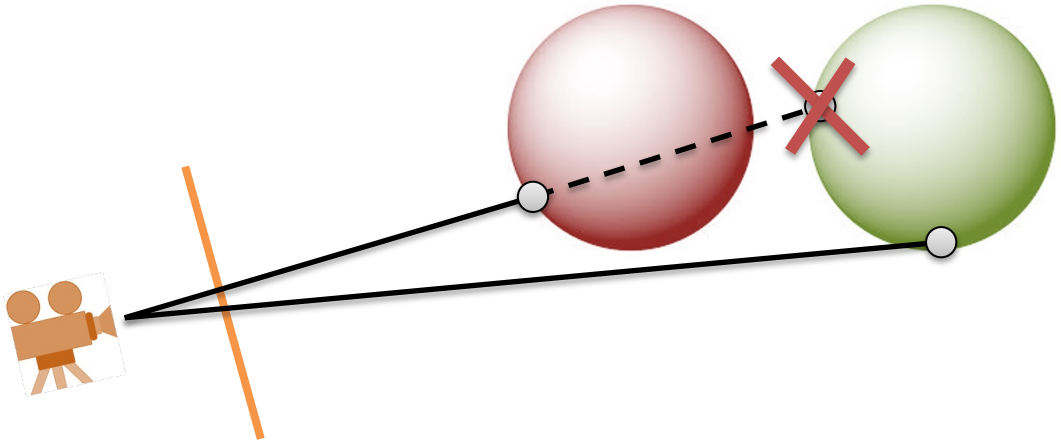
- Deep learning to help: make everything differentiable



Inverse rendering

If you think about the rendering equation, everything is differentiable except one term: visibility.
Visibility is a binary function that can jump abruptly from 0 to 1 and 1 to 0.
This happens when e.g. an object occludes another one.

- Problem: Visibility is not differentiable



Surface Area Form

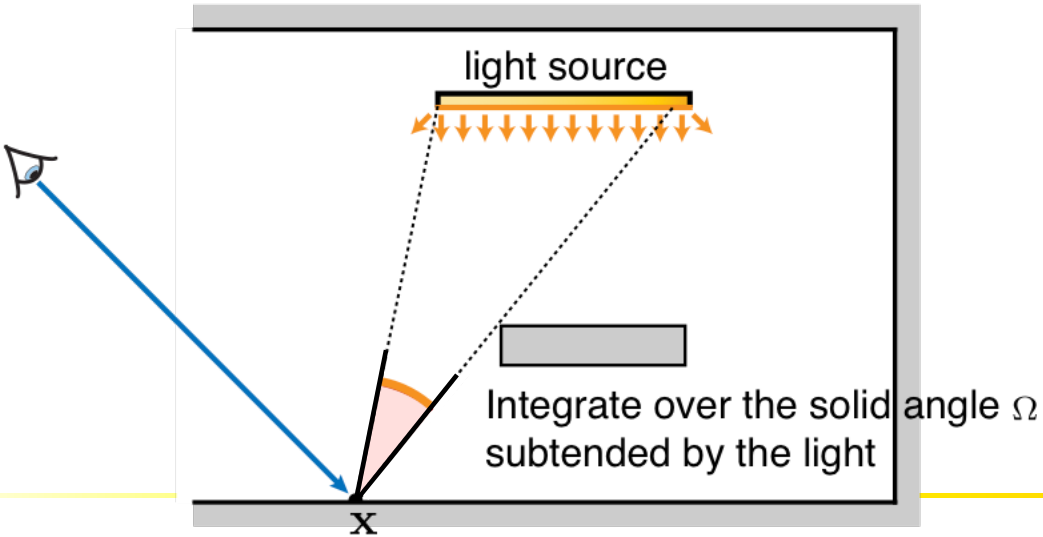
So far, visibility has been hidden in the lighting function in the rendering equation.

Let's make visibility explicit.

We are considering local illumination for the rest of the derivations, i.e. light only comes from light sources.

Recall that $L_e(\mathbf{r}(\mathbf{x}, \vec{\omega}_i), -\vec{\omega}_i)$ is the light leaving a light source at the point r in the direction $-\vec{\omega}_i$, which is equal to the light received at the point x in the direction $\vec{\omega}_i$. We will define what θ_i is shortly.

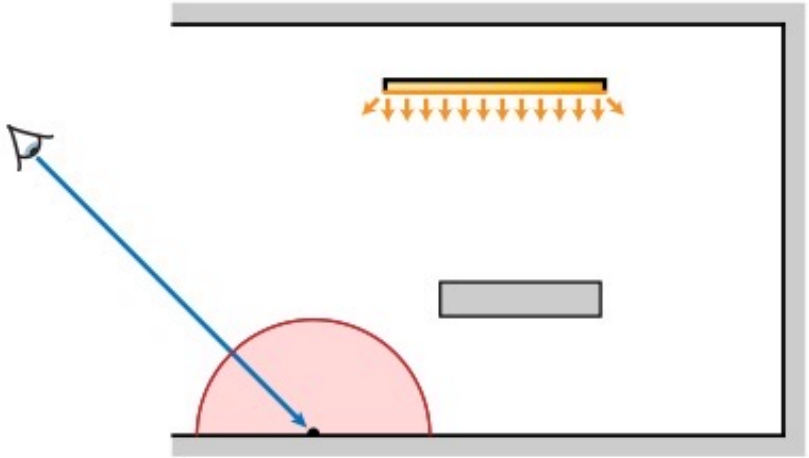
$$L_r(\mathbf{x}, \vec{\omega}_r) = \int_{\Omega} f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_r) L_e(\mathbf{r}(\mathbf{x}, \vec{\omega}_i), -\vec{\omega}_i) |\cos\theta_i| d\vec{\omega}_i$$



Surface Area Form

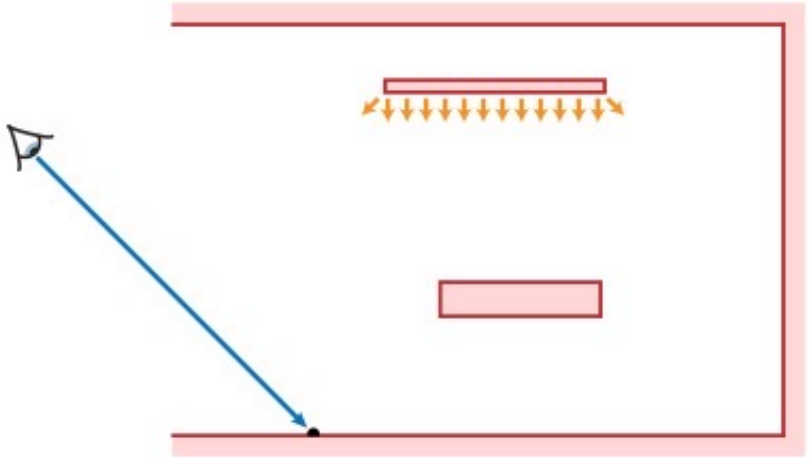
To make visibility and dependence on light sources explicit, we write the rendering equation in a different form. Instead of integrating over the hemisphere of directions at $\mathbf{x}$, we are integrating over the areas of light sources. Note that we are assuming area light sources. Directional lights are a special case (area light with a small area).

Hemispherical integration



$$L_r(\mathbf{x}, \vec{\omega}_r) = \int_{H^2} f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_r) L_i(\mathbf{x}, \vec{\omega}_i) \cos\theta_i \, d\vec{\omega}_i$$

Surface area integration



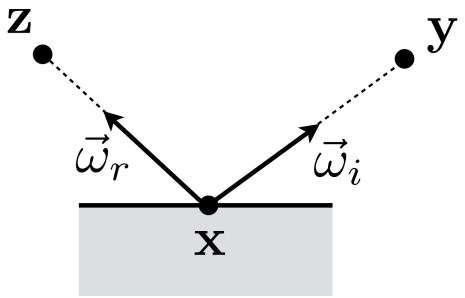
$$L_r(\mathbf{x}, \mathbf{z}) = \int_A f_r(\mathbf{x}, \mathbf{y}, \mathbf{z}) L_i(\mathbf{x}, \mathbf{y}) G(\mathbf{x}, \mathbf{y}) \, dA(\mathbf{y})$$

Surface Area Form

x: the point at which we are calculating the integral, **y**: a point on a light source, **z**: the sensor point
With these variables, we can now derive the solid angle differential (without proof).
The dA is the differential area on the light source we are considering, and θ_0 is the angle between the surface normal of the light source and the incoming light direction.

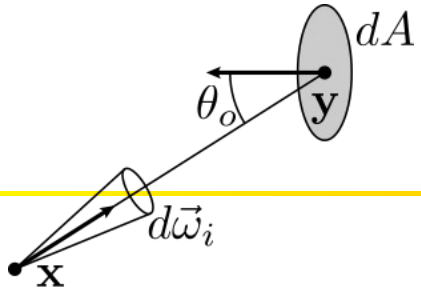
- Change in notation

$$L_i(\mathbf{x}, \vec{\omega}_i) = L_i(\mathbf{x}, \mathbf{y})$$
$$L_r(\mathbf{x}, \vec{\omega}_r) = L_r(\mathbf{x}, \mathbf{z})$$
$$f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_r) = f_r(\mathbf{x}, \mathbf{y}, \mathbf{z})$$



- Transform integral over directions to over surface area

Jacobian of the transform: $$d\vec{\omega}_i = \frac{|\cos\theta_o|}{\|\mathbf{x} - \mathbf{y}\|^2} dA$$

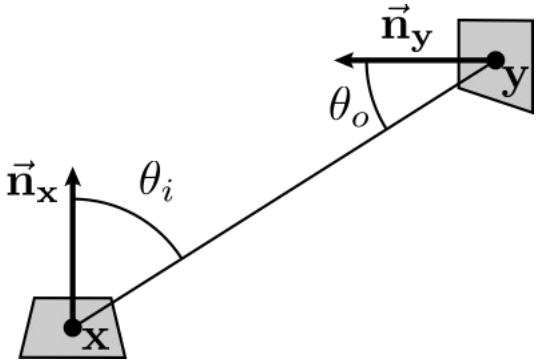


With this change of variables, we can write the new form of the integral with a geometry term.
The geometry term comes with an explicit visibility function V, the angle θ_i between the incoming light direction and the surface normal at $\mathbf{x}$, the angle θ_0 between the incoming light direction and the surface normal of the area light source, and the distance between $\mathbf{x}$ and $\mathbf{y}$ (a point on the light source).

- Surface area form

$$L_r(\mathbf{x},\mathbf{z}) = \int_A f_r(\mathbf{x},\mathbf{y},\mathbf{z}) L_i(\mathbf{x},\mathbf{y}) G(\mathbf{x},\mathbf{y})\, dA(\mathbf{y})$$

Geometry term



$$G(\mathbf{x},\mathbf{y}) = V(\mathbf{x},\mathbf{y}) \frac{|\cos\theta_i||\cos\theta_o|}{\|\mathbf{x}-\mathbf{y}\|^2}$$

Original foreshortening term

Visibility term

Jacobian determinant of the transform

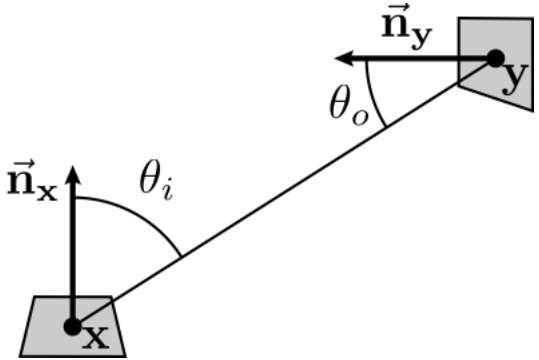
Surface Area Form

- Interpreting the new cosine term

$$G(\mathbf{x}, \mathbf{y}) = V(\mathbf{x}, \mathbf{y}) \frac{|\cos\theta_i| \, |\cos\theta_o|}{\|\mathbf{x} - \mathbf{y}\|^2}$$

• The chance that a photon emitted from a differential patch will hit another differential patch decreases as:

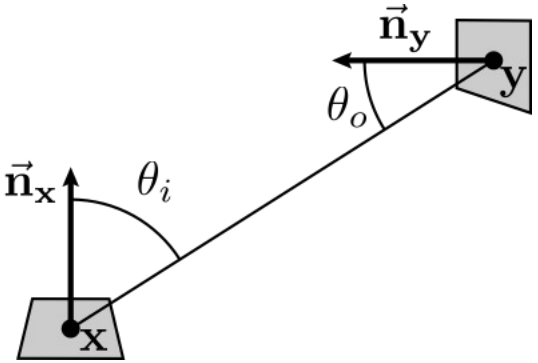
  - the patches **face away** from each other (numerator)
  - the patches **move away** from each other (denominator)

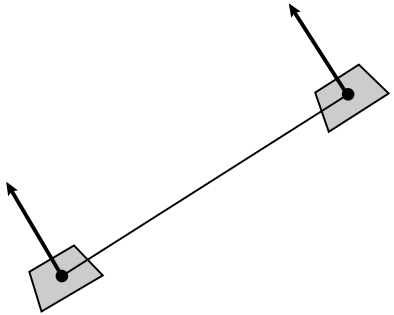


Surface Area Form

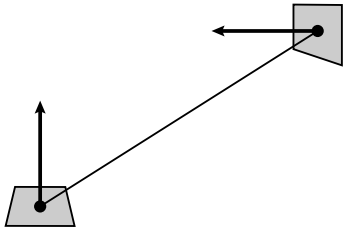
- Interpreting the new cosine term

$$G(\mathbf{x},\mathbf{y}) = V(\mathbf{x},\mathbf{y})\ \frac{|\cos\theta_i|\,|\cos\theta_o|}{\|\mathbf{x}-\mathbf{y}\|^2}$$

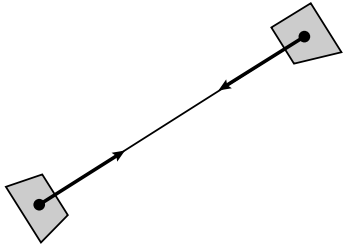




numerator = 0



0 < numerator < 1



numerator = 1

Visibility Term

- Surface area form

$$L_r(\mathbf{x}, \mathbf{z}) = \int_A f_r(\mathbf{x}, \mathbf{y}, \mathbf{z}) L_i(\mathbf{x}, \mathbf{y}) G(\mathbf{x}, \mathbf{y}) \, dA(\mathbf{y})$$

Geometry term

$$G(\mathbf{x}, \mathbf{y}) = V(\mathbf{x}, \mathbf{y}) \frac{|\cos\theta_i| \, |\cos\theta_o|}{\|\mathbf{x} - \mathbf{y}\|^2}$$

Visibility term

$$V(\mathbf{x}, \mathbf{y}) = \begin{cases} 1: & \text{visible} \\ 0: & \text{not visible} \end{cases}$$

Discontinuous!

Visibility Term

Since visibility is discontinuous, the function we integrate is discontinuous.
Hence, we cannot exchange the order of derivatives with the integral.
This is, however, what we need for inverse rendering via optimizing scene parameters with gradient descent.

- Problem with discontinuities and differentiability

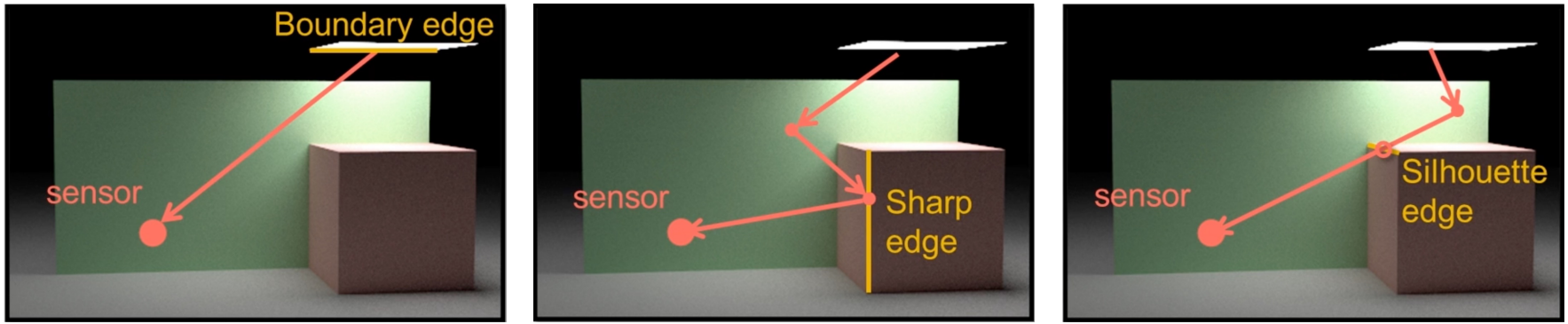
$$\frac{\partial}{\partial \pi} \int_A f(\mathbf{x}) d\mathbf{x} \neq \int_A \frac{\partial}{\partial \pi} f(\mathbf{x}) d\mathbf{x}$$

Visibility Term

Example discontinuities of visibility occurs for:

- at the boundary of a light source, where light suddenly stops reaching the sensor point,
- at a sharp edge of an object, where light suddenly stops bouncing,
- at a silhouette edge, where an object is occluding a light source when viewed from the sensor.

- Sources of discontinuities due to visibility



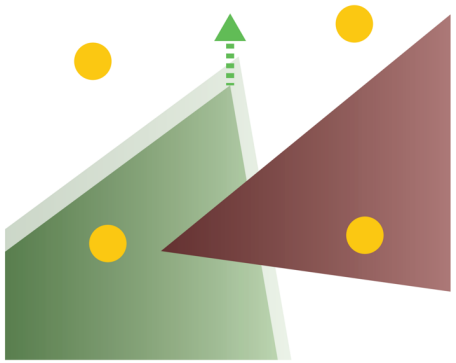
Zhang et al., Path-Space Differentiable Rendering, 2020.

Differentiable Visibility

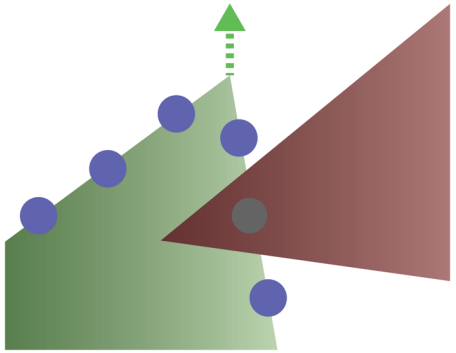
No need to know the details of differentiable visibility techniques.

There are a few techniques to provide a continuous approximation of visibility.
One of them is densely sampling polygon edges in the meshes in a scene, as for meshes those are where discontinuities appear.

- Edge-sampling for accurate gradients



Standard sampling

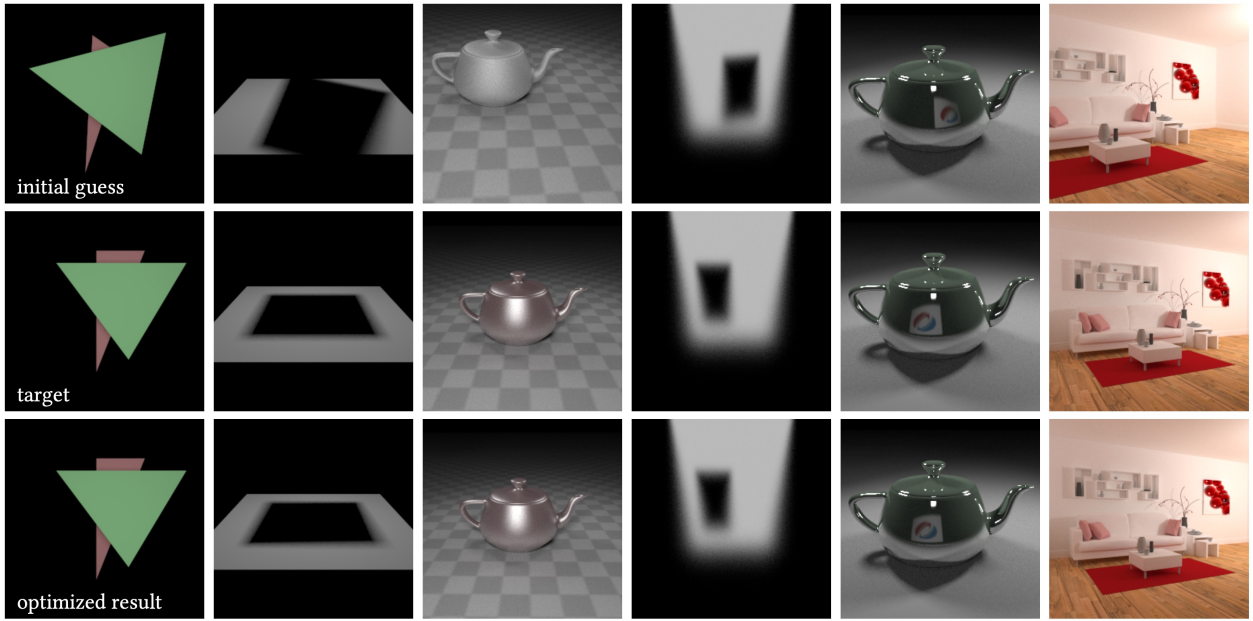


Edge sampling

Li et al., Differentiable Monte Carlo Ray Tracing through Edge Sampling, 2018.

Differentiable Visibility

- Edge-sampling for accurate gradients

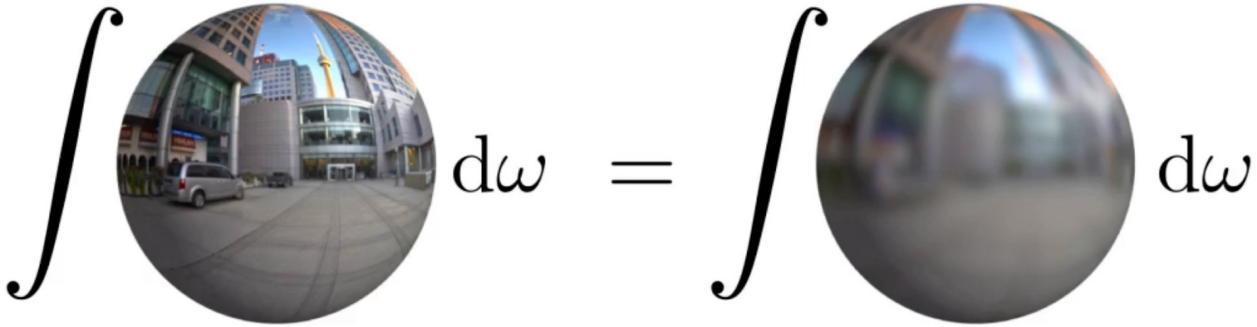


Li et al., Differentiable Monte Carlo Ray Tracing through Edge Sampling, 2018.

Differentiable Visibility

Another idea is convolving the function we are integrating to smooth it to get rid of discontinuities.

- Re-parametrizing discontinuities



integrand convolved integrand

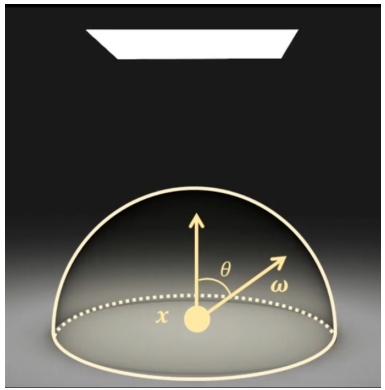
Loubet et al., Reparameterizing discontinuous integrands for differentiable rendering, 2019.

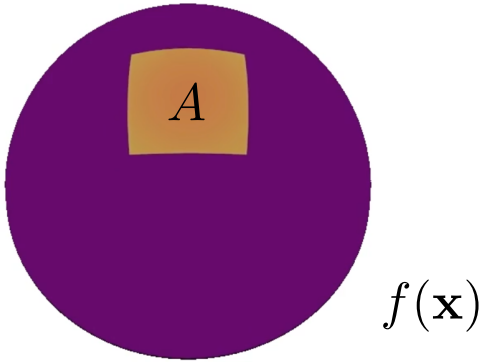
Differentiable Visibility

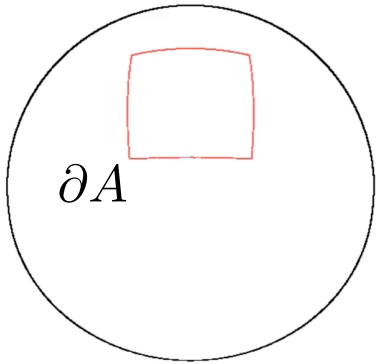
Yet another idea is utilising a general theorem that provides the integral of a discontinuous function.
This theorem introduces an additional integral (last term below) over the boundary of the area we are originally integrating over.

- Reynolds transport theorem

$$\frac{\partial}{\partial \pi} \int_A f(\mathbf{x}) d\mathbf{x} = \int_A \frac{\partial}{\partial \pi} f(\mathbf{x}) d\mathbf{x} + \int_{\partial A} g(\mathbf{l}) d\mathbf{l}$$







Zhang et al., Path-Space Differentiable Rendering, 2020.

Frameworks

- Automatic differentiation
- Fast & parallel computation
- Active development

TensorFlow Graphics

Feedback & Actions

We will have sessions for **going through examples** for the ones who could not get enough supervision.

- These replace office hours.
- If you cannot make it, do let me know.

No need to know more than what is covered in the lecture notes (annotated slides).

Make sure you study with the **lecture notes = annotated slides with details**

Ticking

Ticking session: 13:00 - 14:00 today

- Please check for your group & time slot
- The slots are approximate, please be ready
- If you cannot make it, we will reschedule

Ticking format

- 5 minutes via Zoom or similar
- Please be prepared to explain & run the code
- Please have an ID