Lecture 12

Inverse Graphics

CS328 - Numerical Methods for Visual Computing and Machine Learning

Prof. Wenzel Jakob

EPFL RGL Realistic Graphics Lab

Graduated

Current

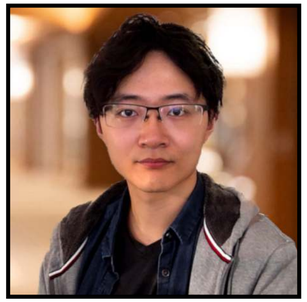
Ph. D. students


Tizian Zeltner


Merlin Nimier-David


Delio Vicini


Baptiste Nicolet


Ziyi Zhang


Lovro Nuic

Postdocs


Ekrem Yilmazer


Mandy Xia


Guillaume Loubet

Research engineers & administrative assistant


Sébastien Speierer

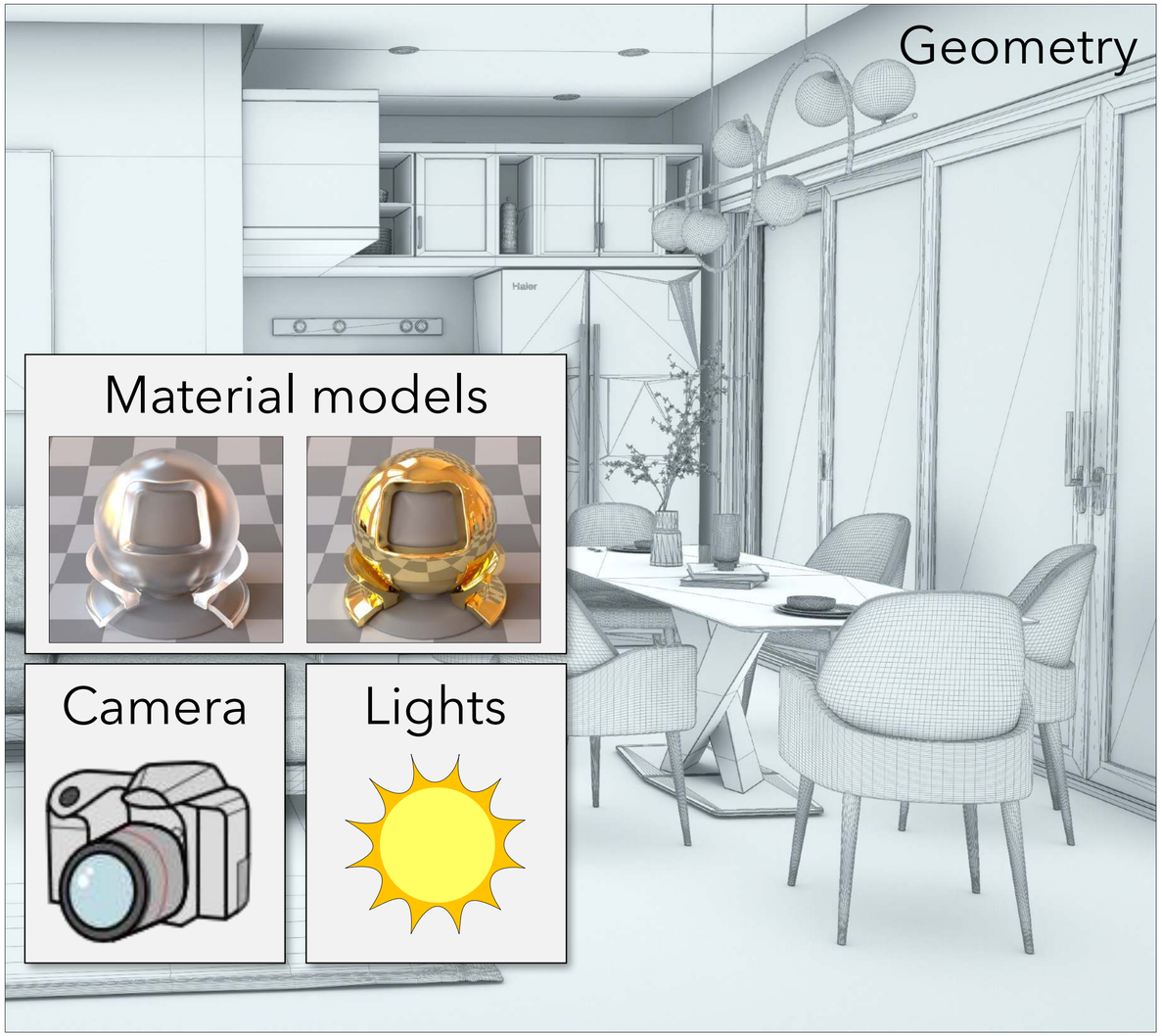

Rami Tabbara


Nicolas Roussel


Pauline Raffestin

Rendering

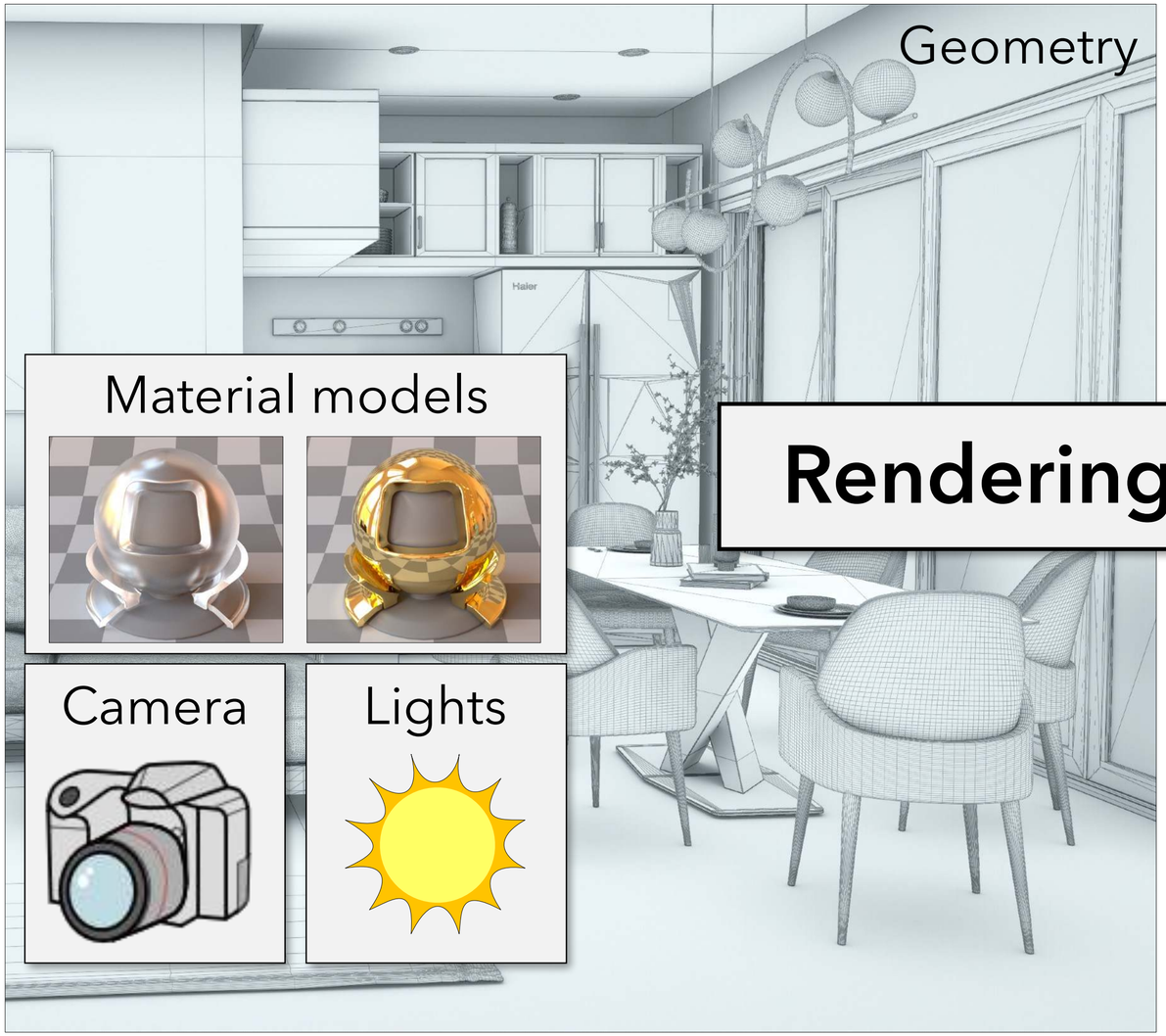
Input scene



Rendered image

Rendering

Input scene



Rendering algorithm

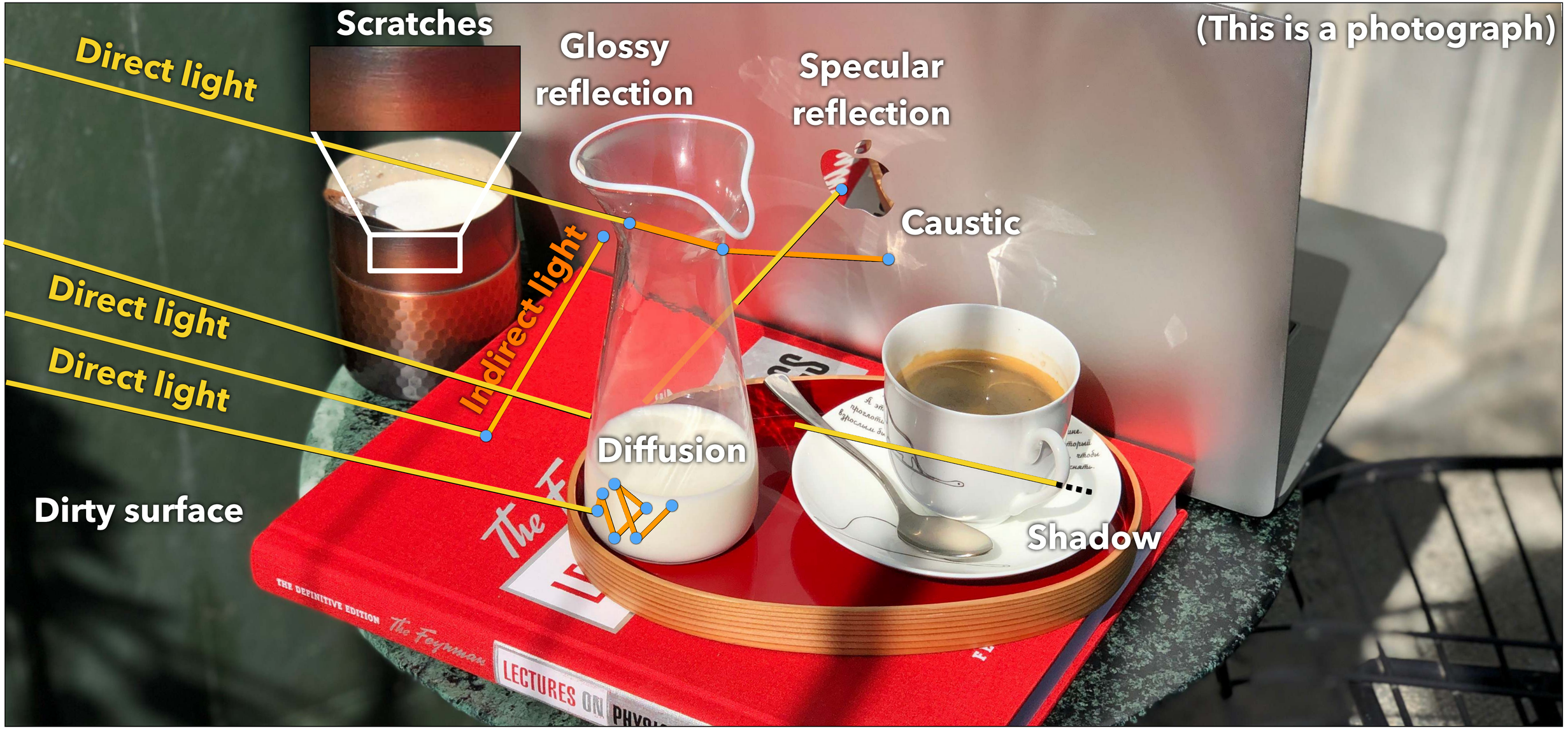
Rendered image


VANJA
Dish towel, assorted patterns white/black
$4.99 / 2 pack
PANNÅ
Place mat, turquoise
$1.99
RÅSKOG
Utility cart
$29.99
LAPPLJUNG RUTA
Rug, low pile, white, black
$79.99
MITOS


[Weta Digital - War for the Planet of the Apes | VFX Breakdown]

Light transport and scattering

Light transport and scattering



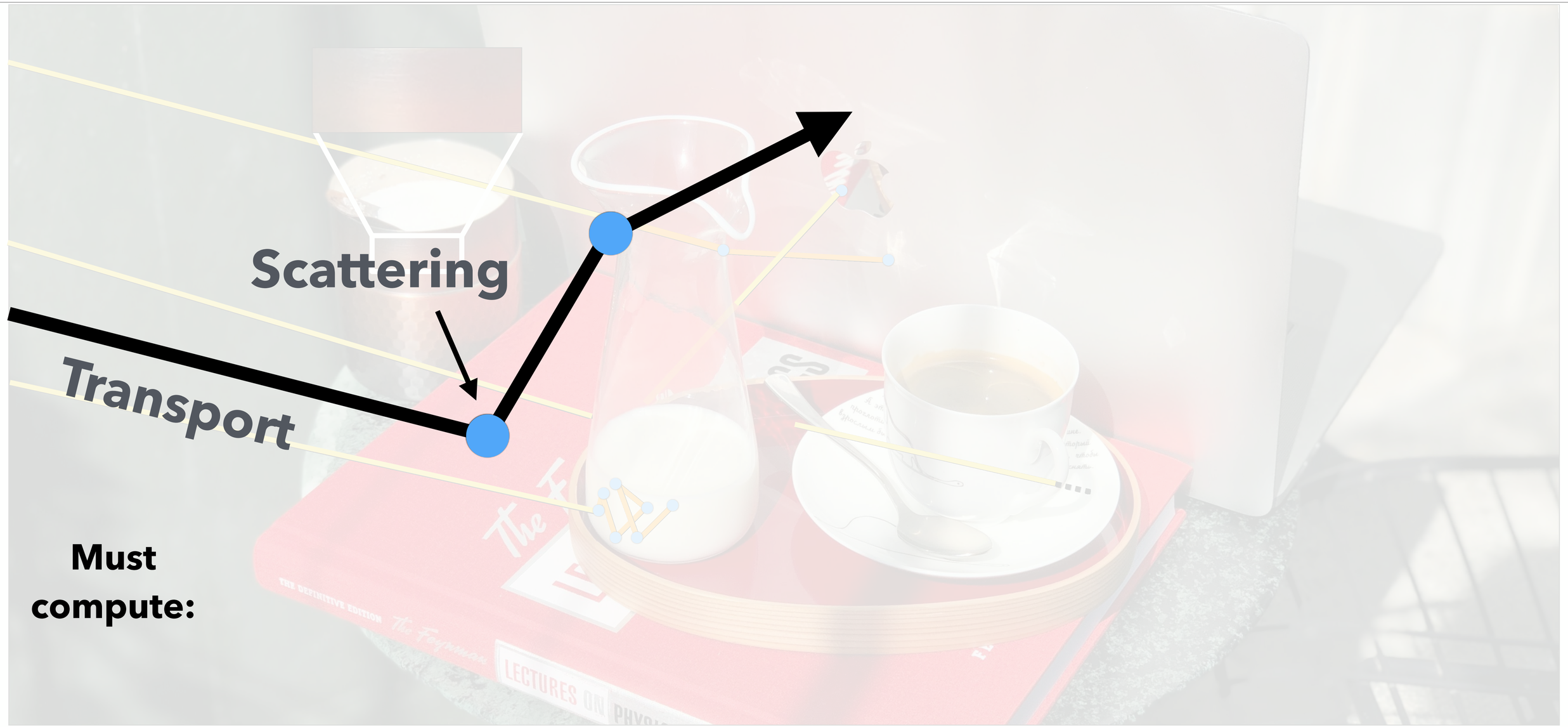
Light transport and scattering

Must compute:

Light transport and scattering

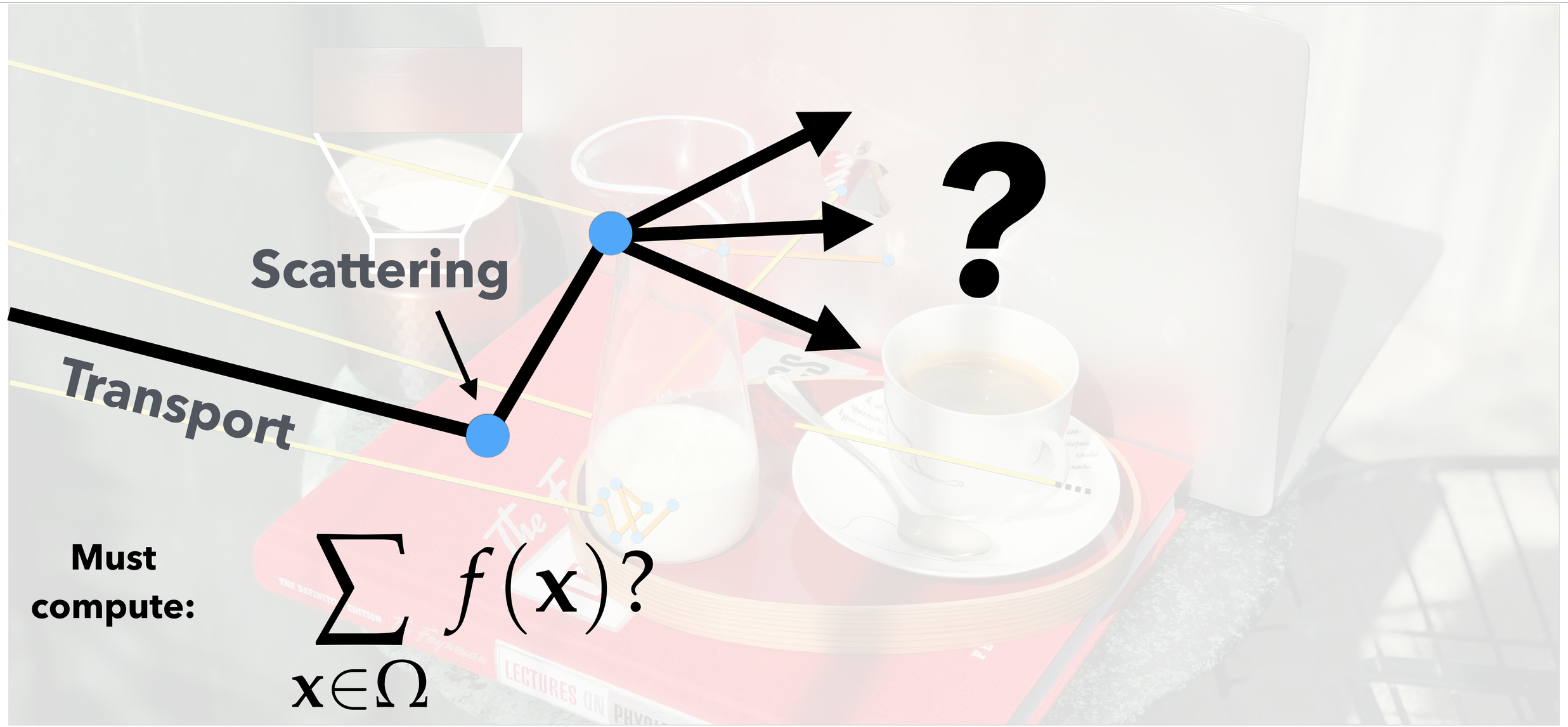
Must compute:

Light transport and scattering

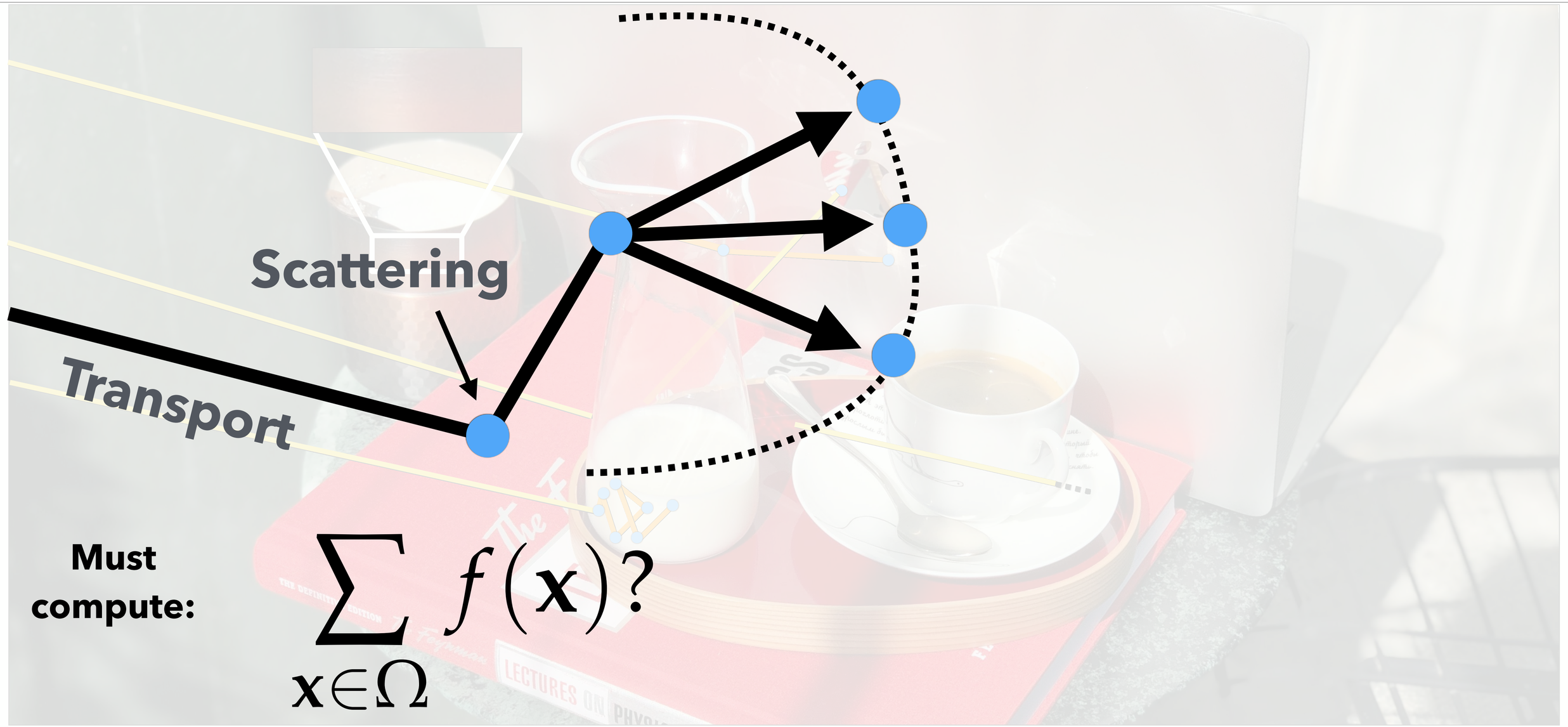


Must compute:

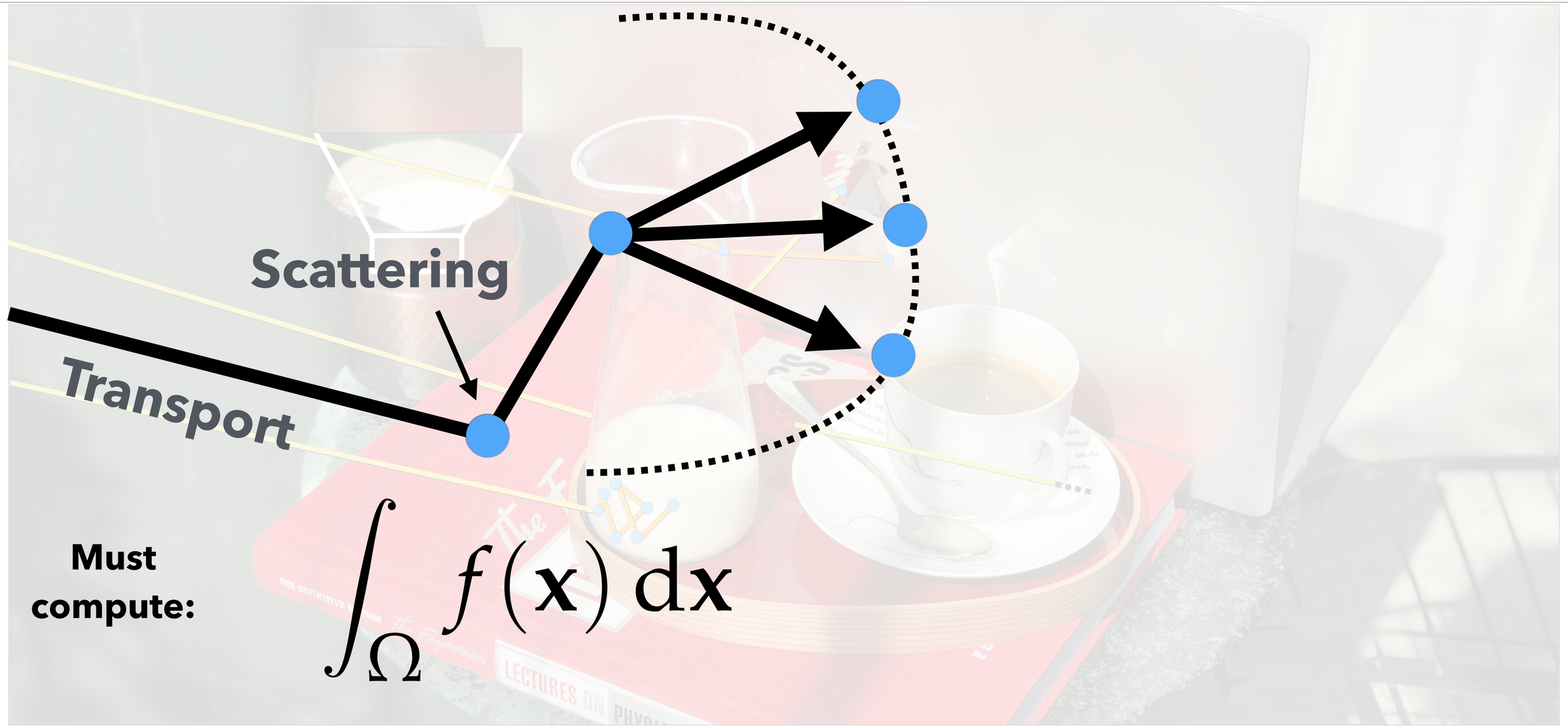
Light transport and scattering



Light transport and scattering



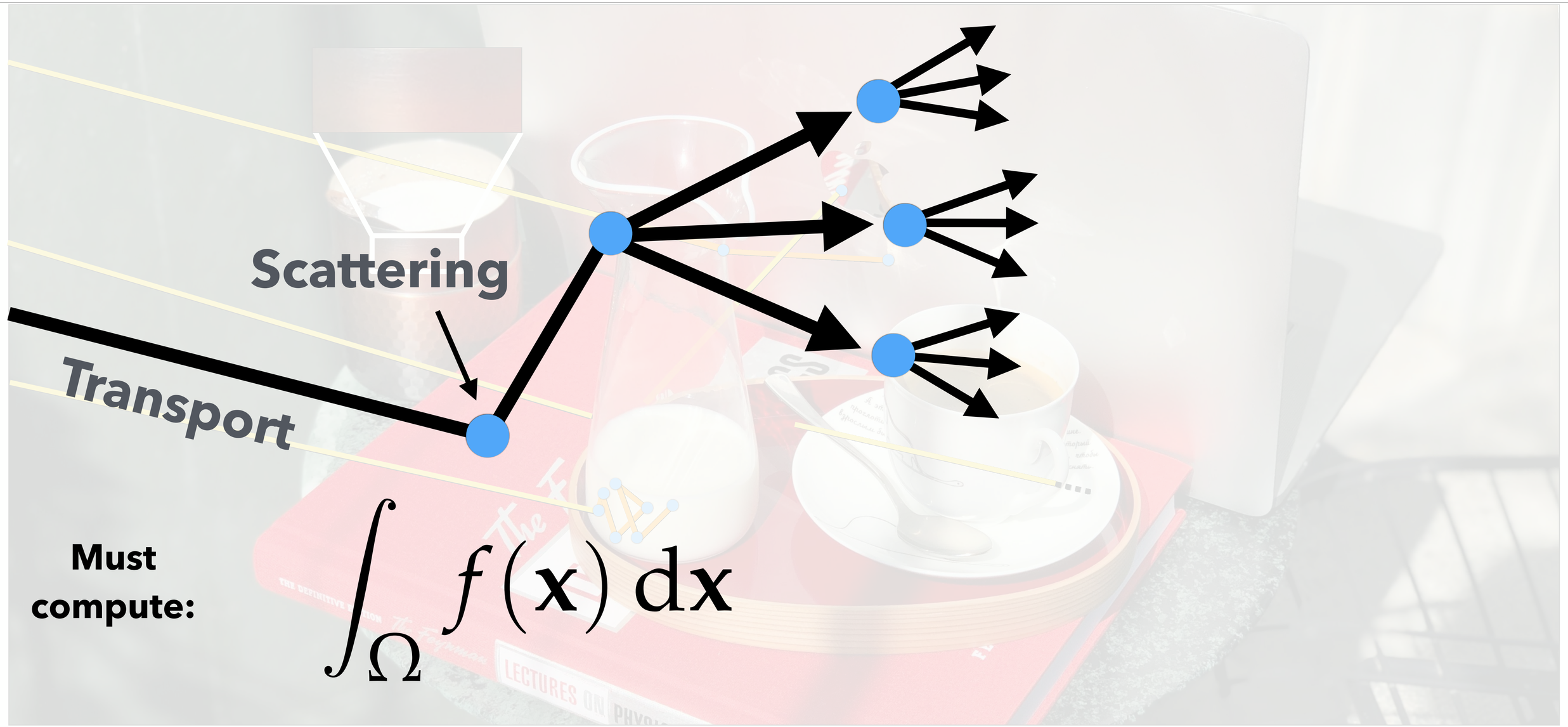
Light transport and scattering



Must compute:

$$\int_{\Omega} f(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

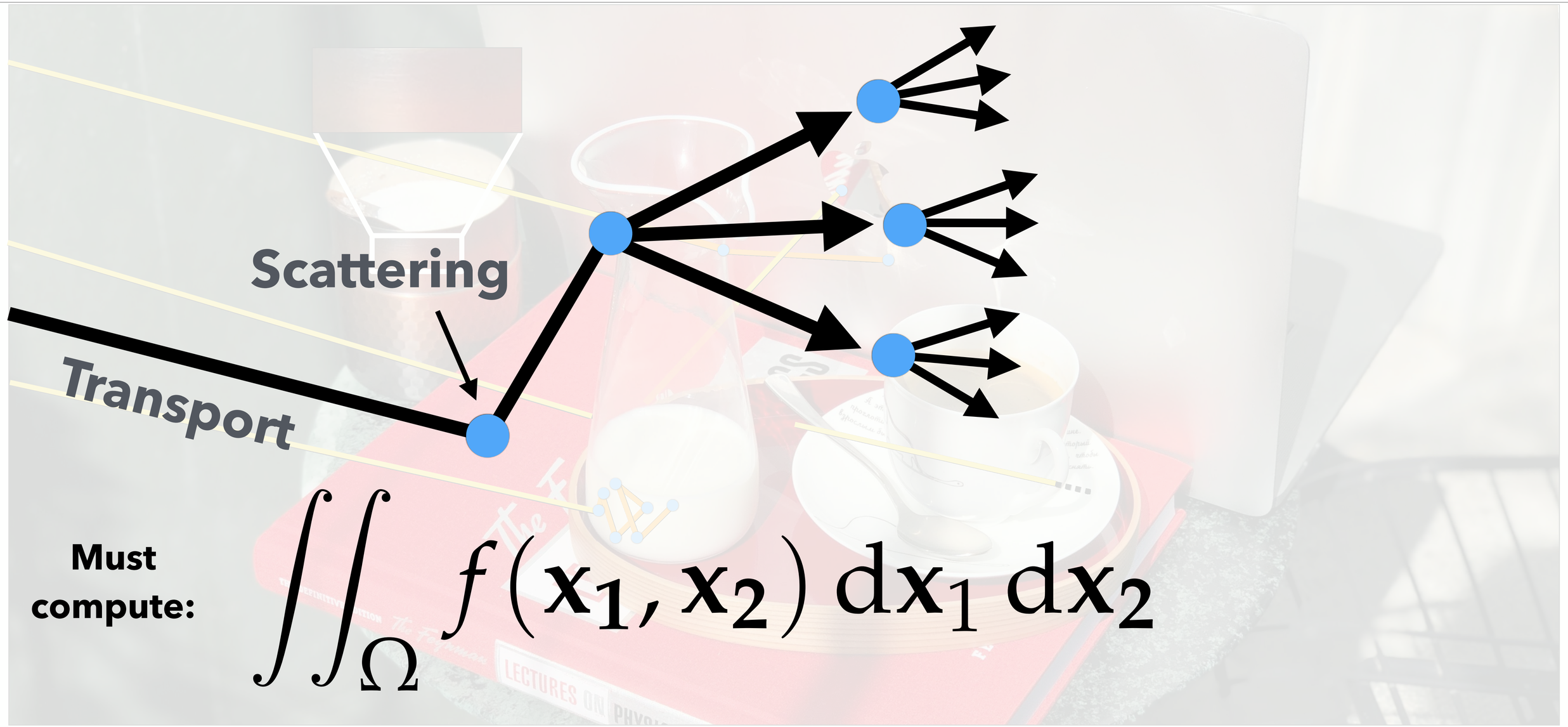
Light transport and scattering



Must compute:

$$\int_{\Omega} f(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

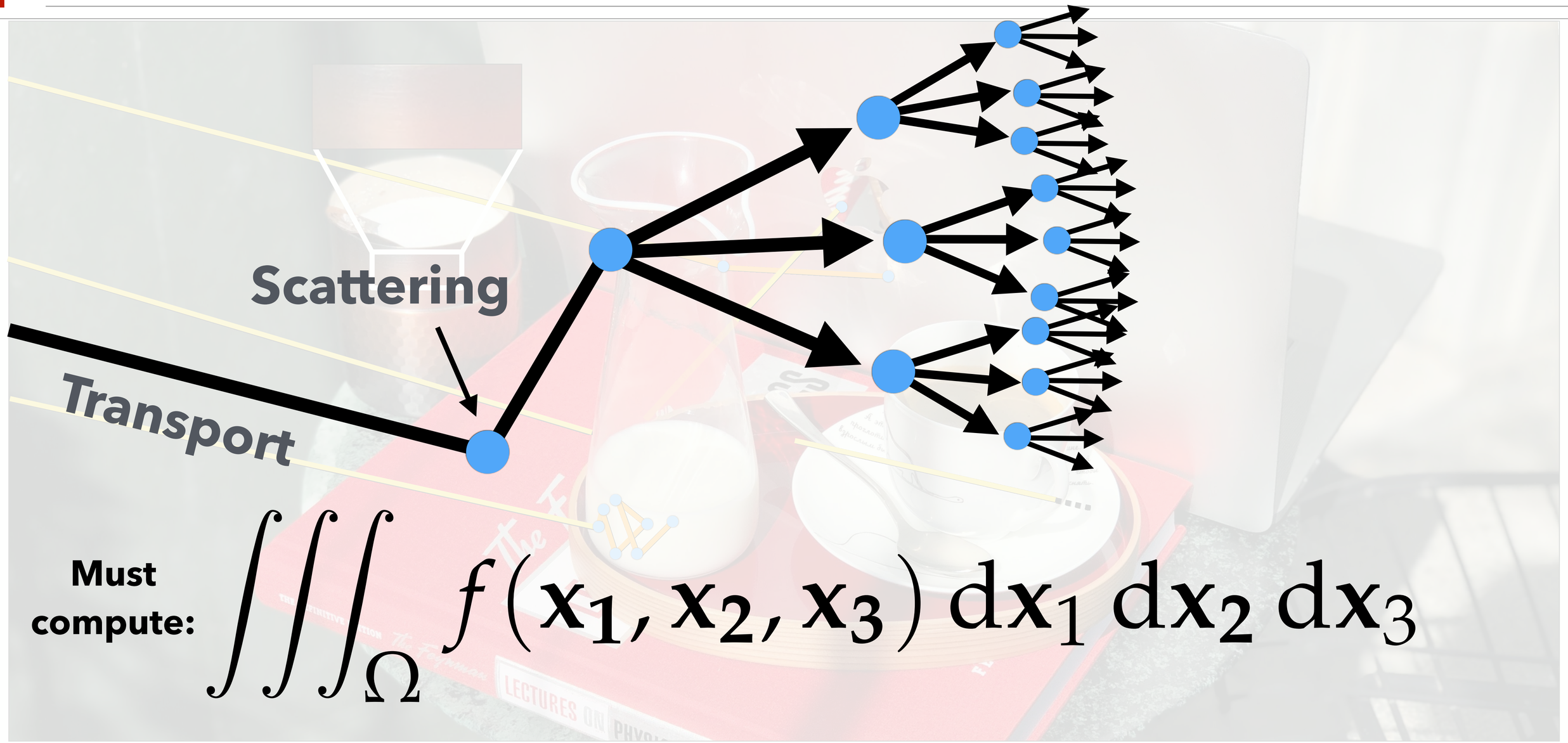
Light transport and scattering



Must compute:

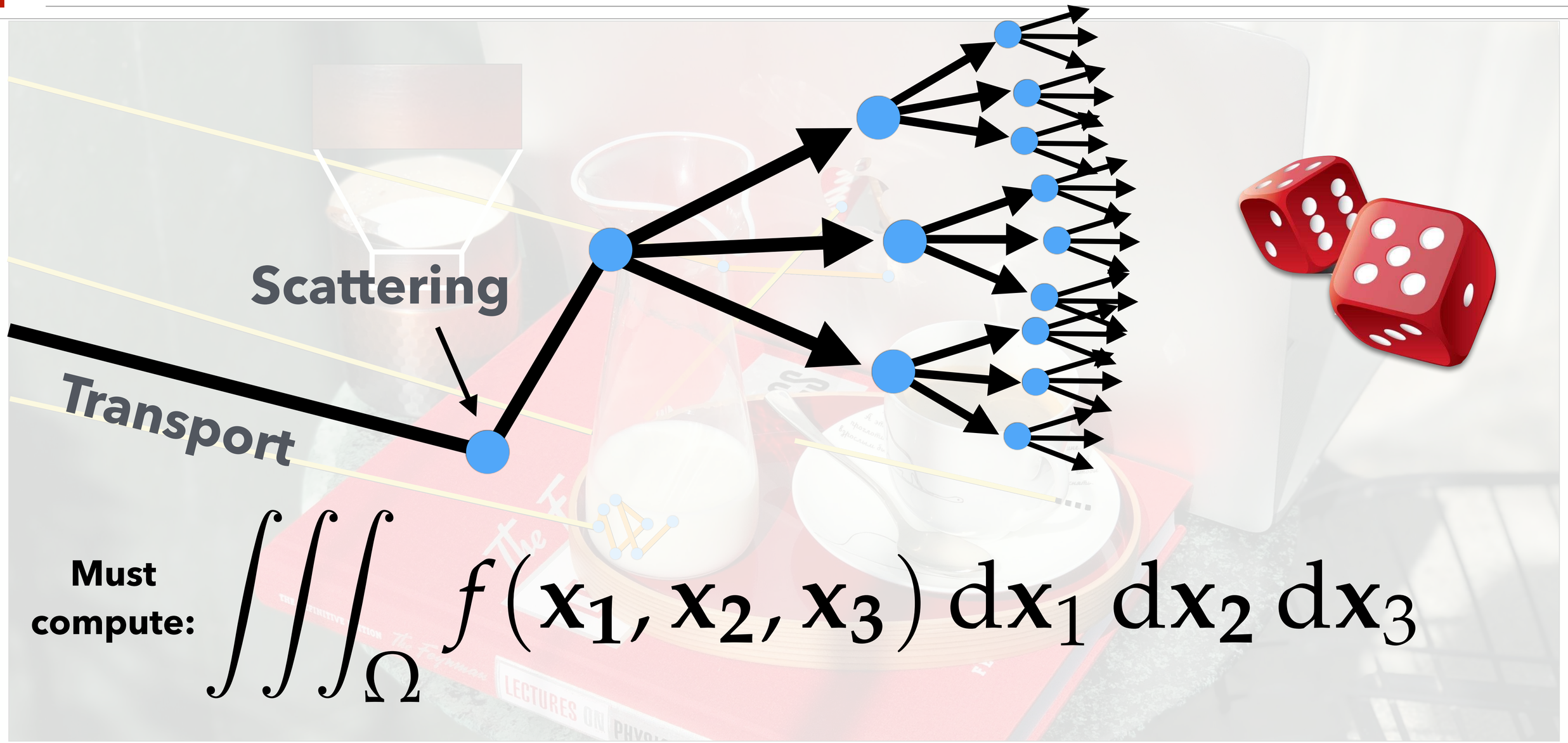
$$\iint_{\Omega} f(\mathbf{x}_1, \mathbf{x}_2)\, \mathrm{d}\mathbf{x}_1\, \mathrm{d}\mathbf{x}_2$$

Light transport and scattering



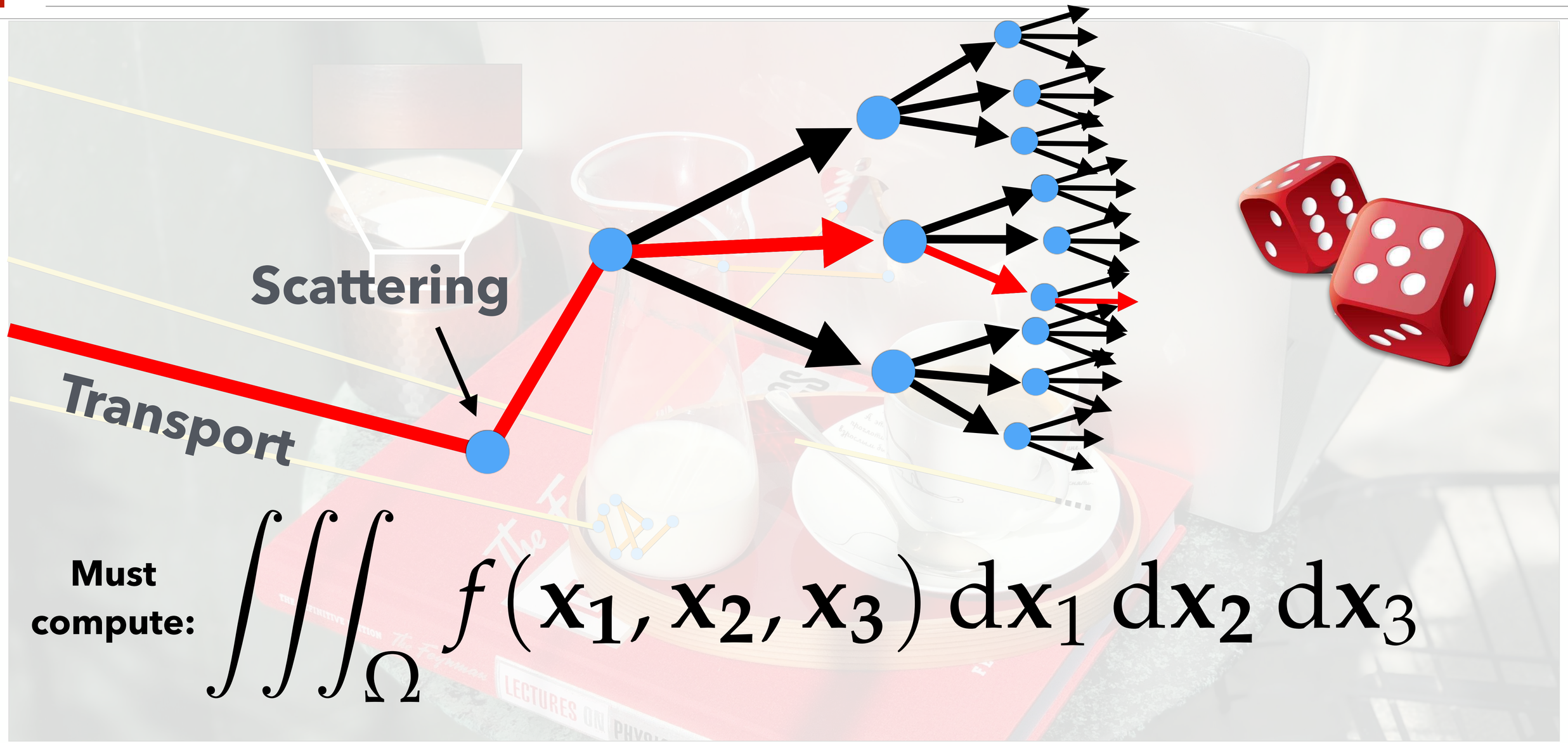
Must compute: $$\iiint_{\Omega} f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)\, \mathrm{d}\mathbf{x}_1\, \mathrm{d}\mathbf{x}_2\, \mathrm{d}\mathbf{x}_3$$

Light transport and scattering



Must compute: $$\iiint_{\Omega} f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)\, \mathrm{d}\mathbf{x}_1\, \mathrm{d}\mathbf{x}_2\, \mathrm{d}\mathbf{x}_3$$

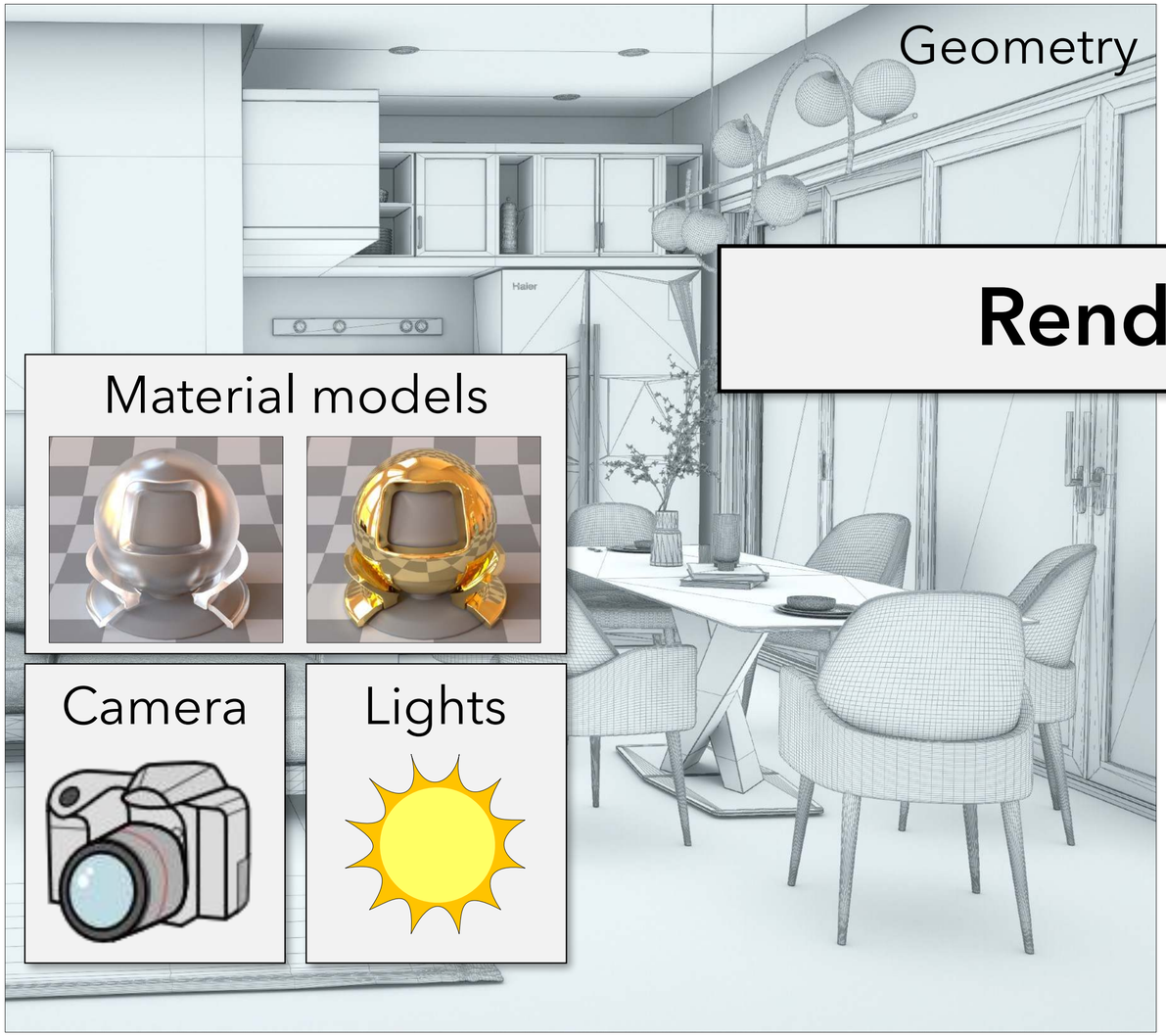
Light transport and scattering



Must compute: $$\iiint_{\Omega} f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)\, \mathrm{d}\mathbf{x}_1\, \mathrm{d}\mathbf{x}_2\, \mathrm{d}\mathbf{x}_3$$

Inverse rendering

Input scene

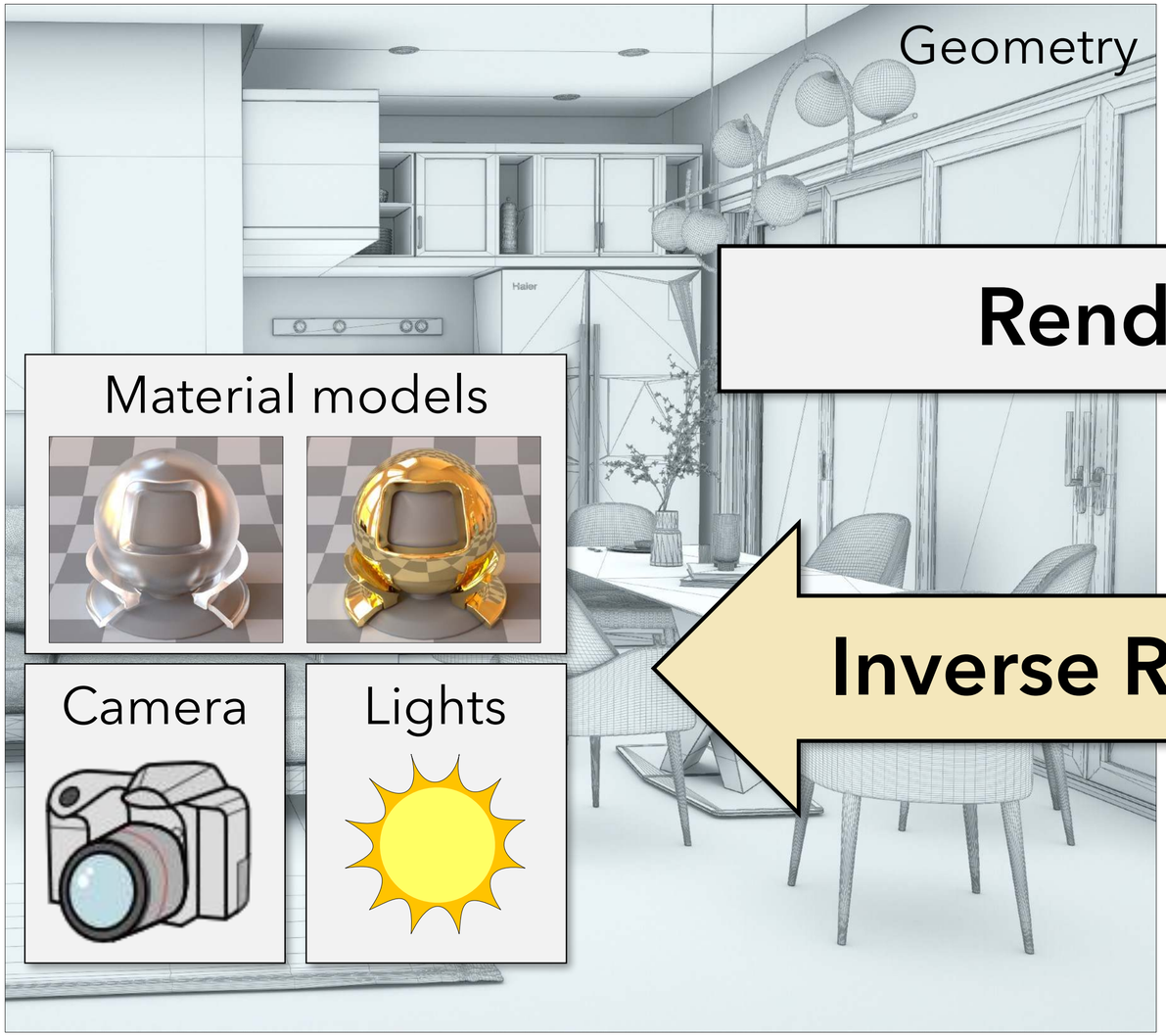


Rendering

Rendered image

Inverse rendering

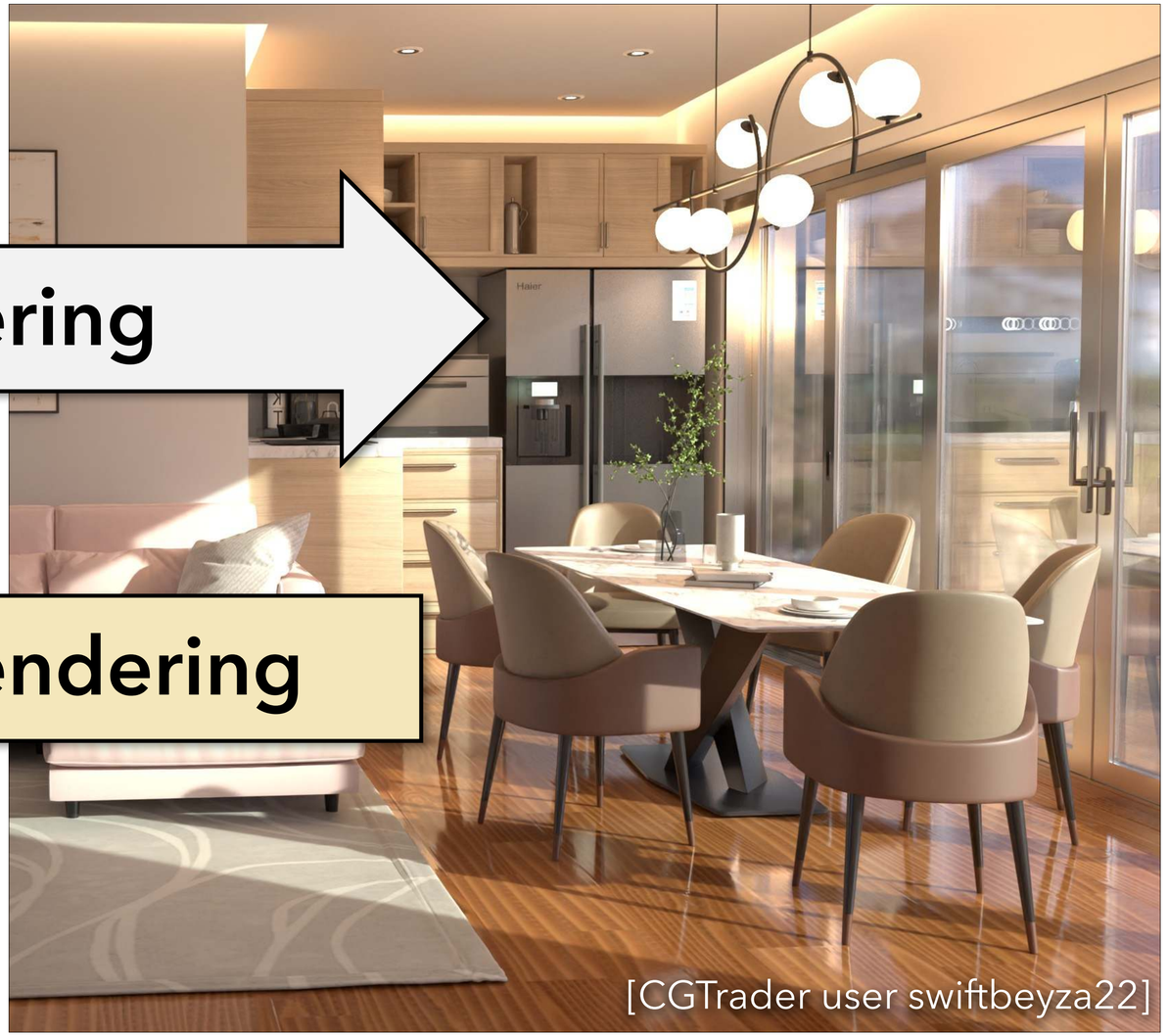
Input scene

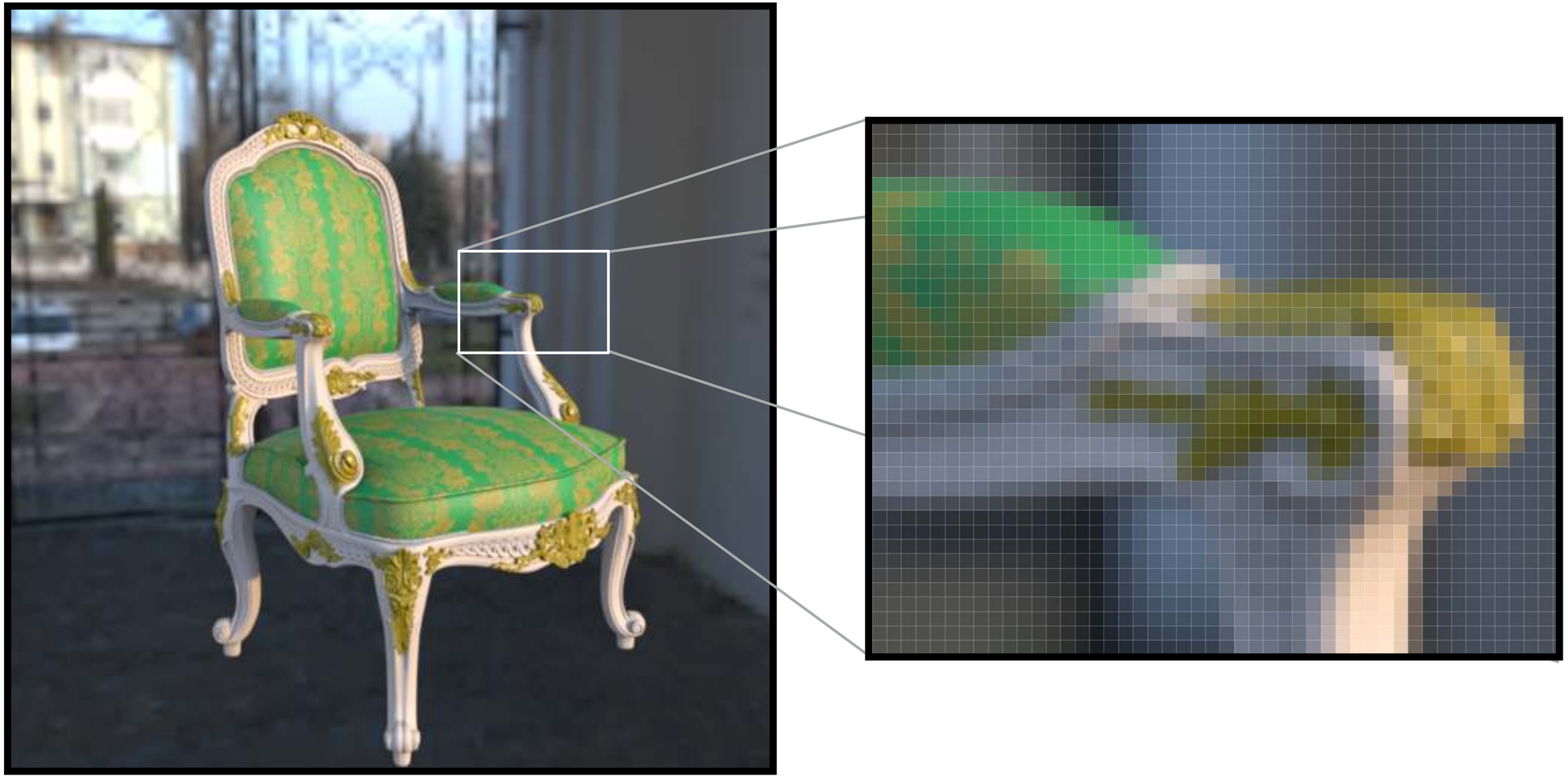


Rendering

Inverse Rendering

Rendered image





[*Antique Chairs Set*, BlenderArtists user 1DInc]

[*Antique Chairs Set*, BlenderArtists user 1DInc]

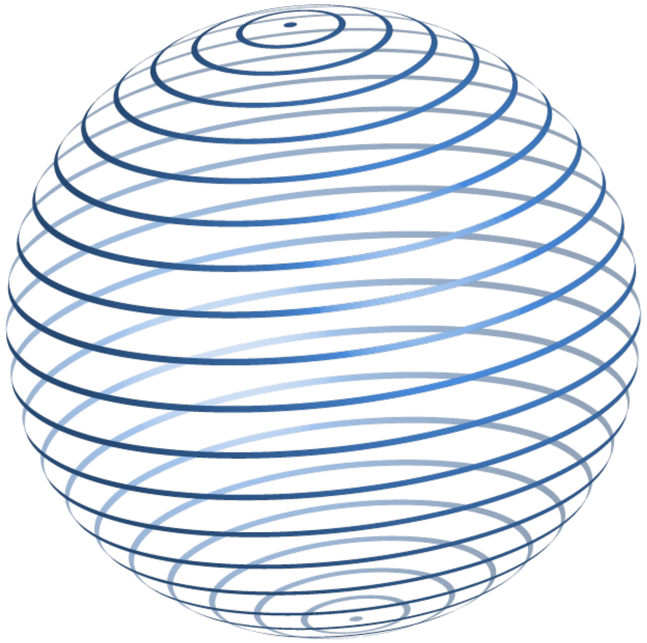
Oh, snap!

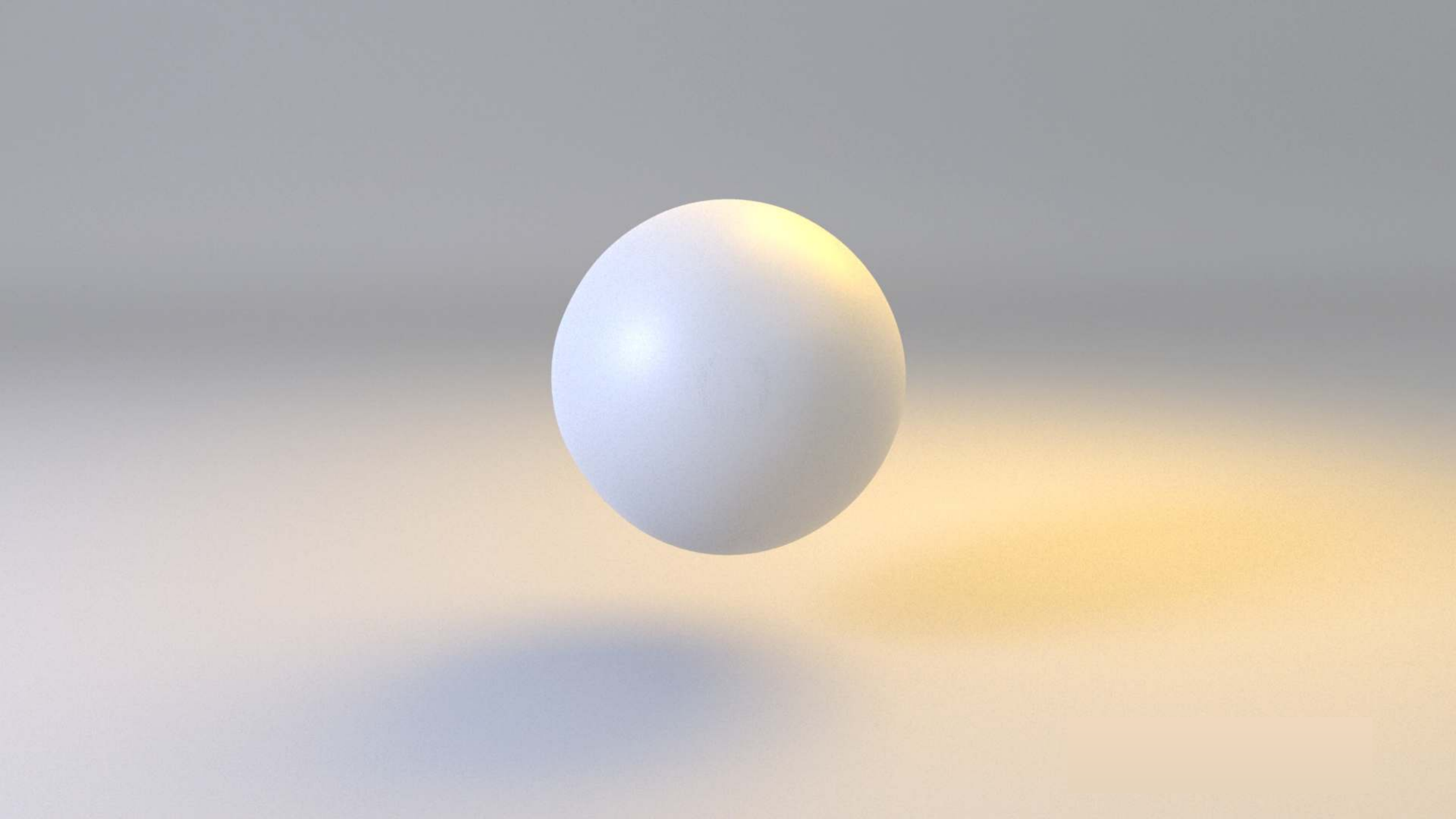
[MidJourney: "smoke coming out of a computer workstation"]

The computer broke, everything is gone 😱.

.. *except*: we still have some pictures of the chair.

Can we recover the 3D model using **only those pictures**?



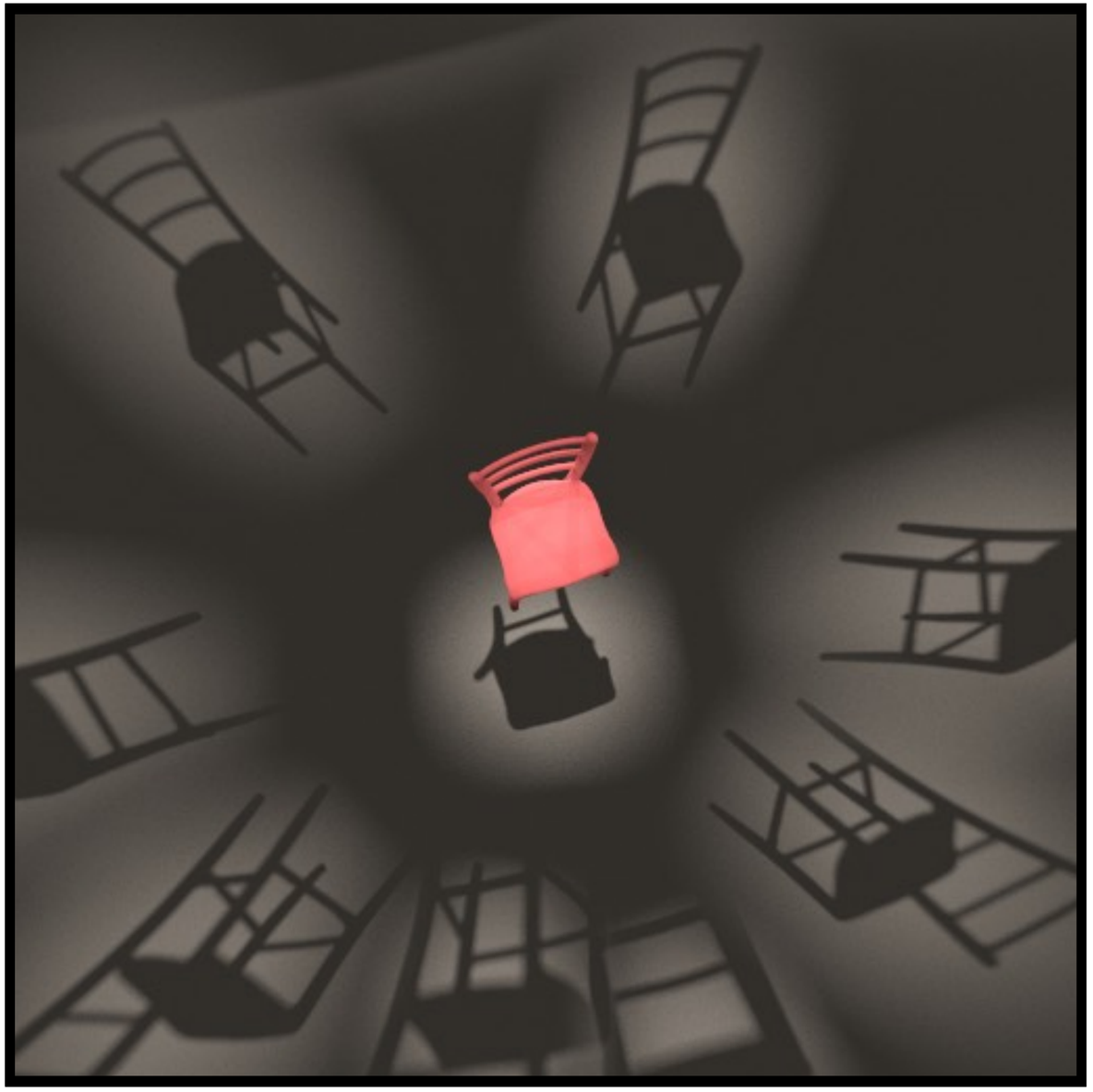


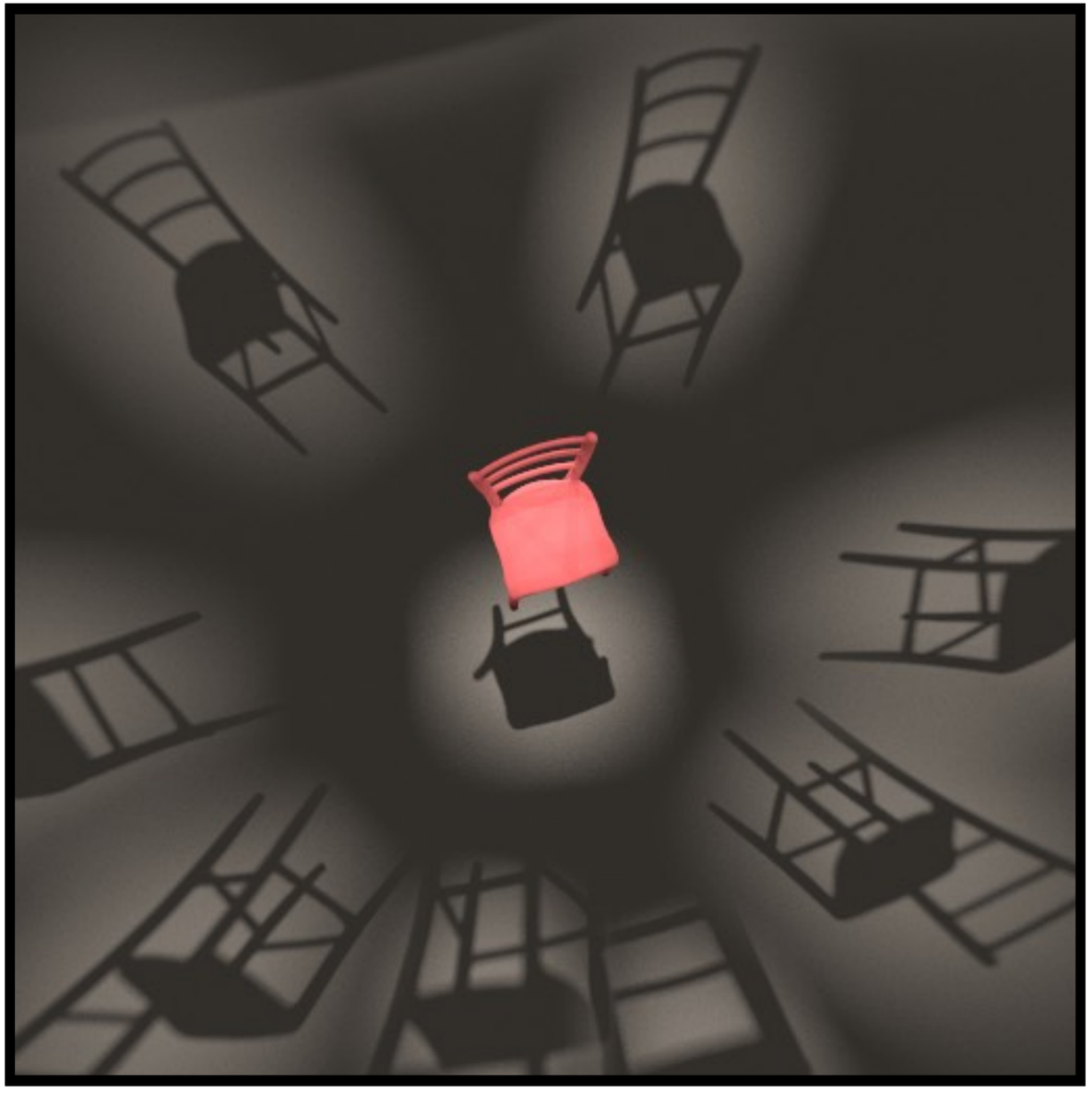
Optimization

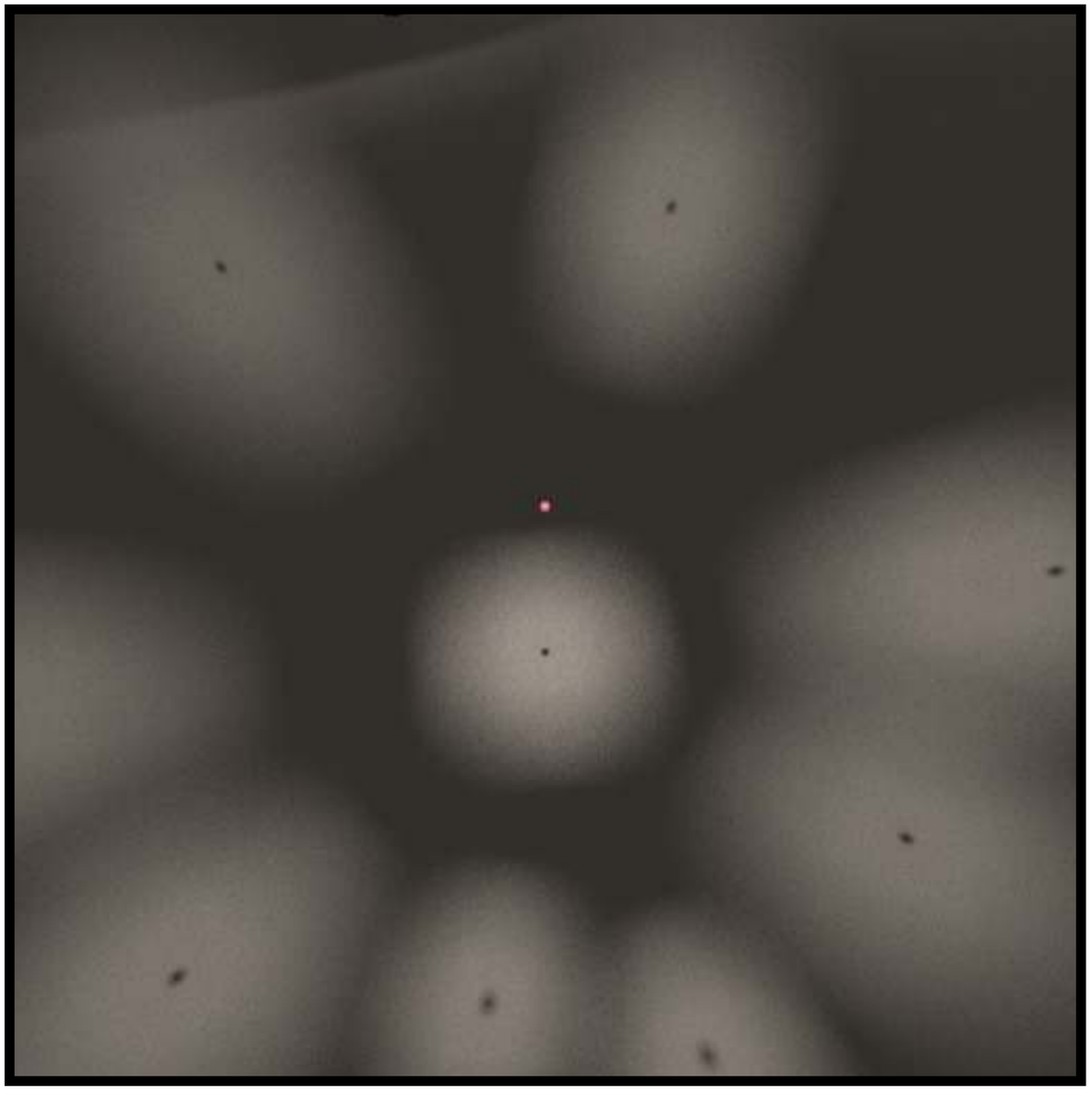
Optimizing shape, albedo
& roughness

Iteration 0

[*Differentiable Signed Distance Function Rendering*, Vicini et al. 2022]







Ground truth

Optimization view

Inference using *indirect* cues (shadows)

[*Projective Sampling for Differentiable Rendering of Geometry*, Zhang, Roussel, and Jakob. 2024]

Ground truth

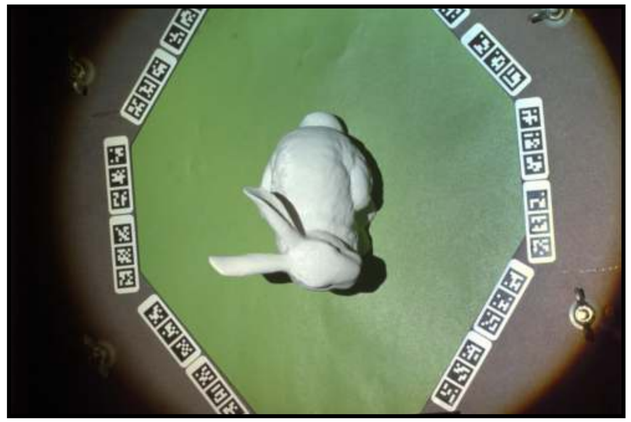
Optimization view

Inference using *indirect* cues (shadows)

[*Projective Sampling for Differentiable Rendering of Geometry*, Zhang, Roussel, and Jakob. 2024]

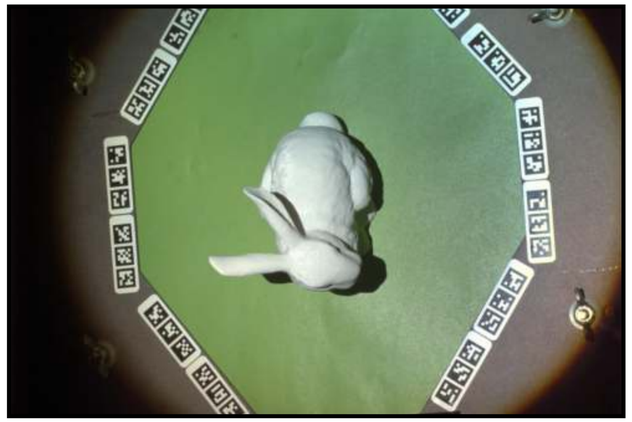
Reconstructing a shape from photos

Photos (4/14 views):




Reconstructing a shape from photos

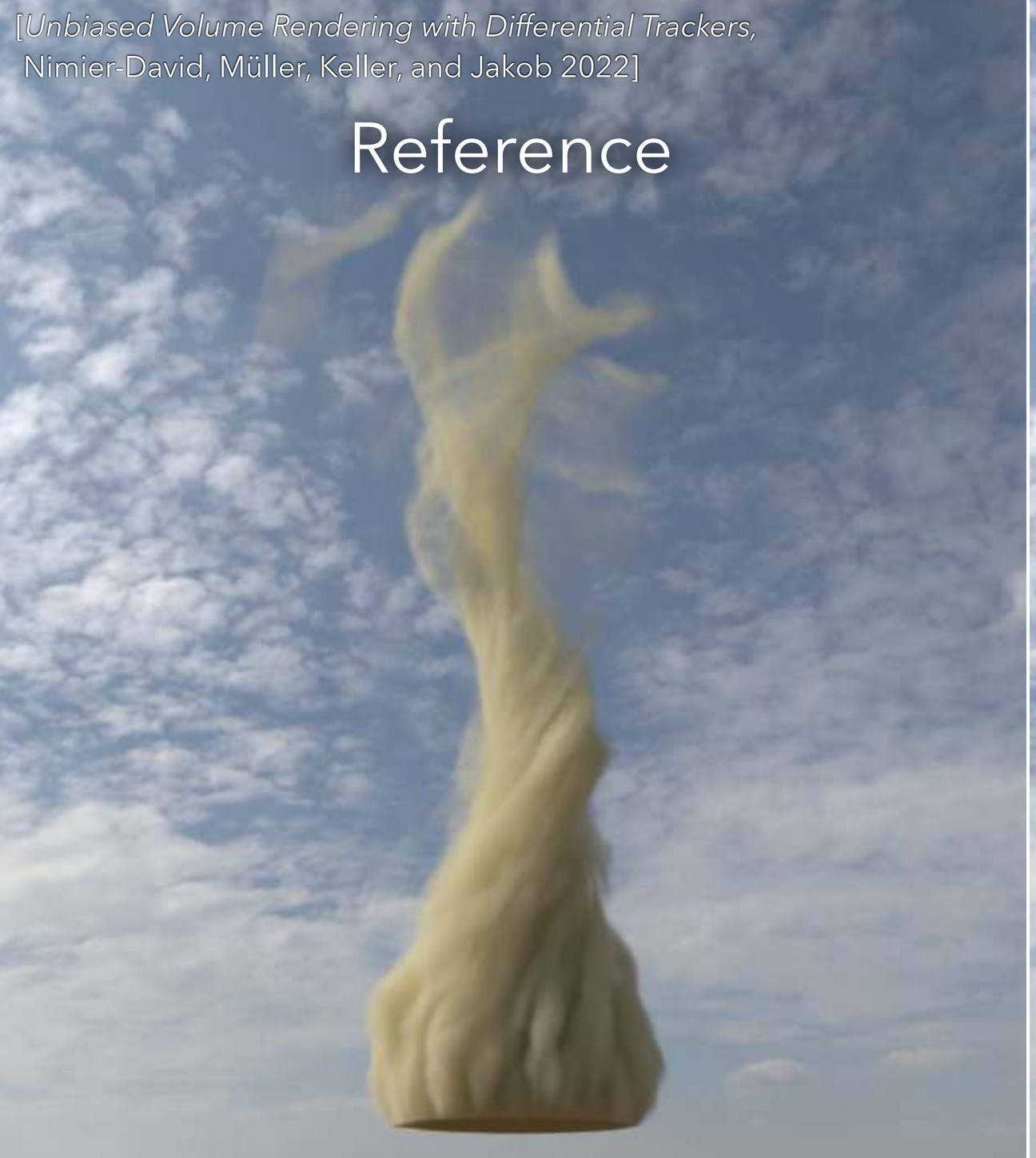
Photos (4/14 views):



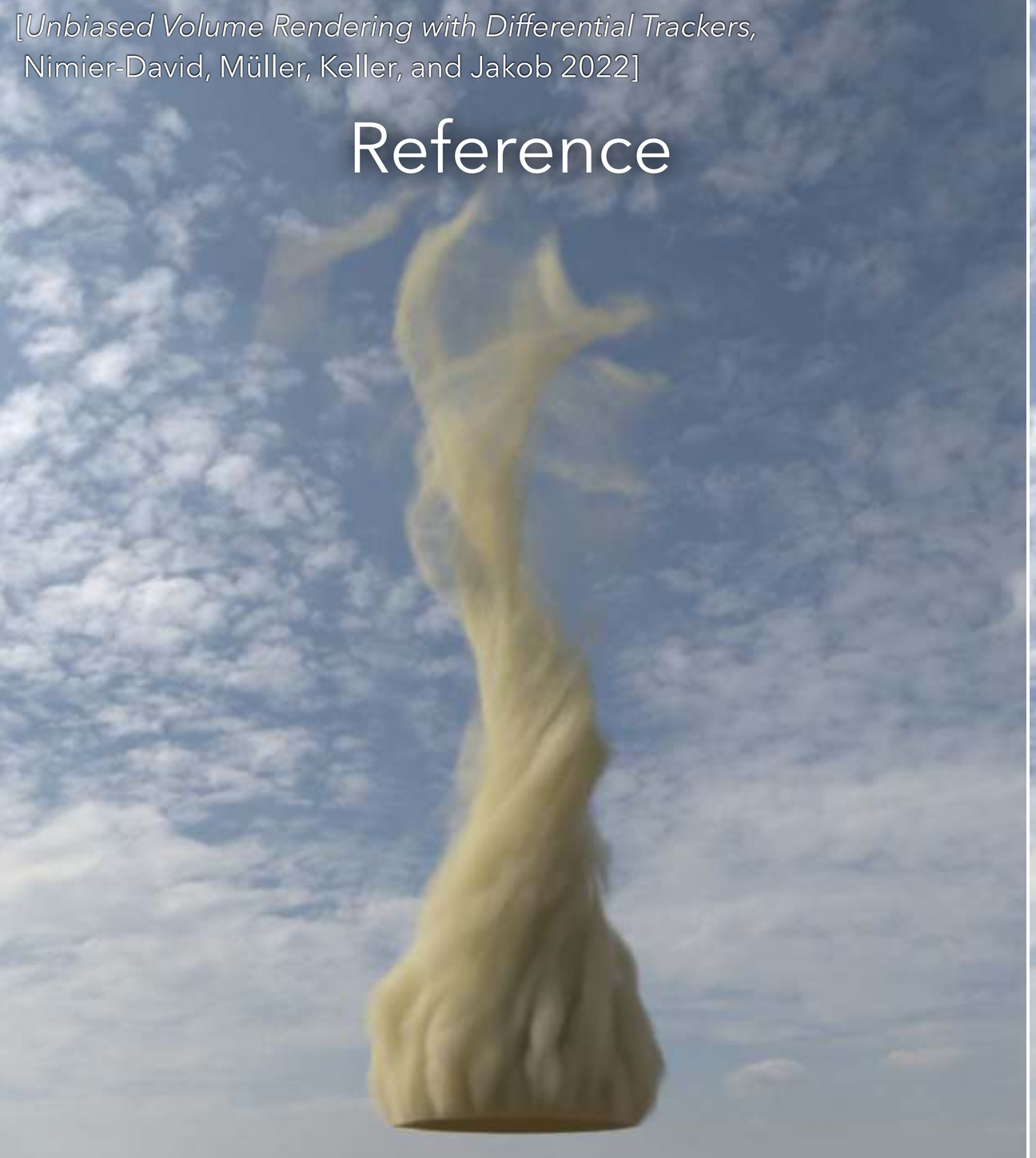

Reconstruction:




[*Unbiased Volume Rendering with Differential Trackers*, Nimier-David, Müller, Keller, and Jakob 2022]



[*Unbiased Volume Rendering with Differential Trackers*, Nimier-David, Müller, Keller, and Jakob 2022]

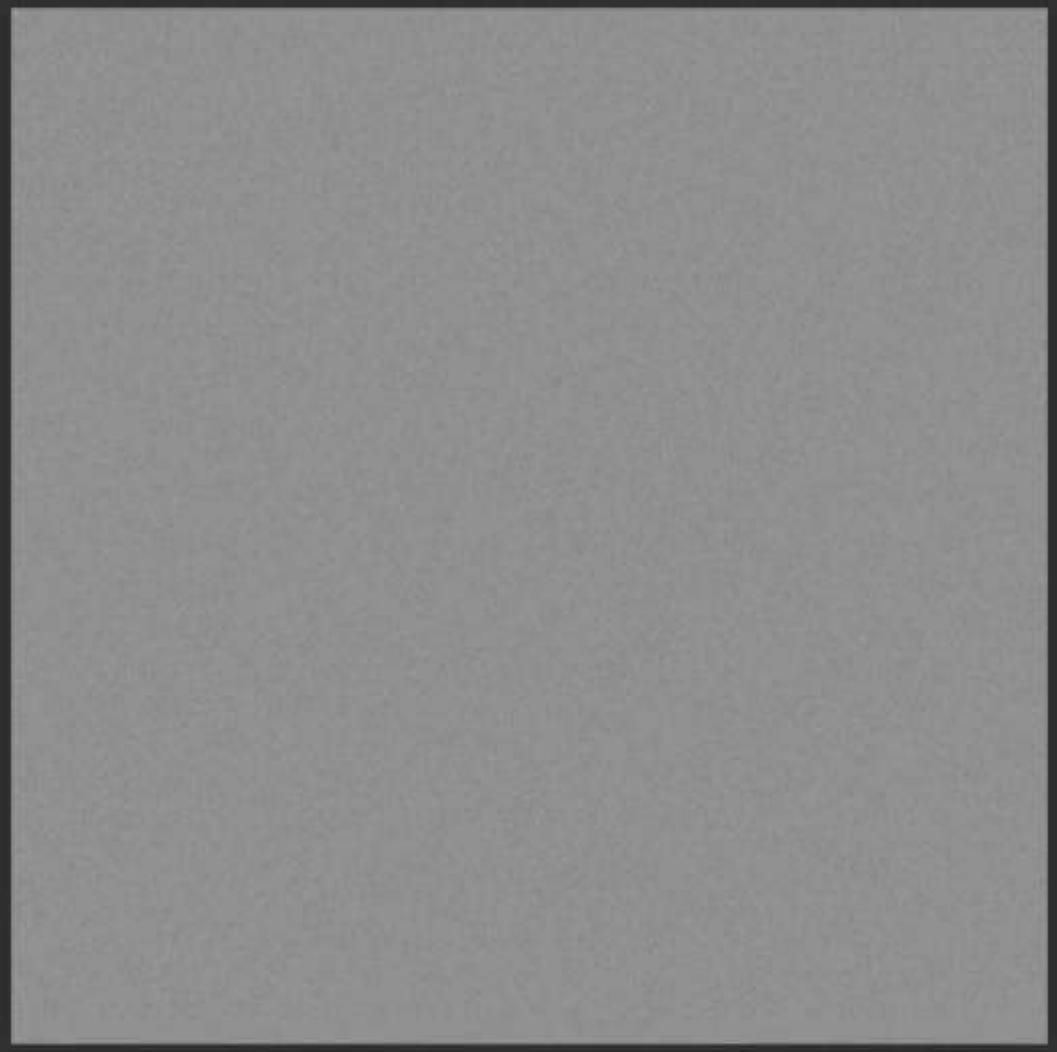



THE DEFINITIVE EDITION
The Feynman
LECTURES ON

Target image

The Great Wave off Kanagawa by Katsushika Hokusai

Initial state

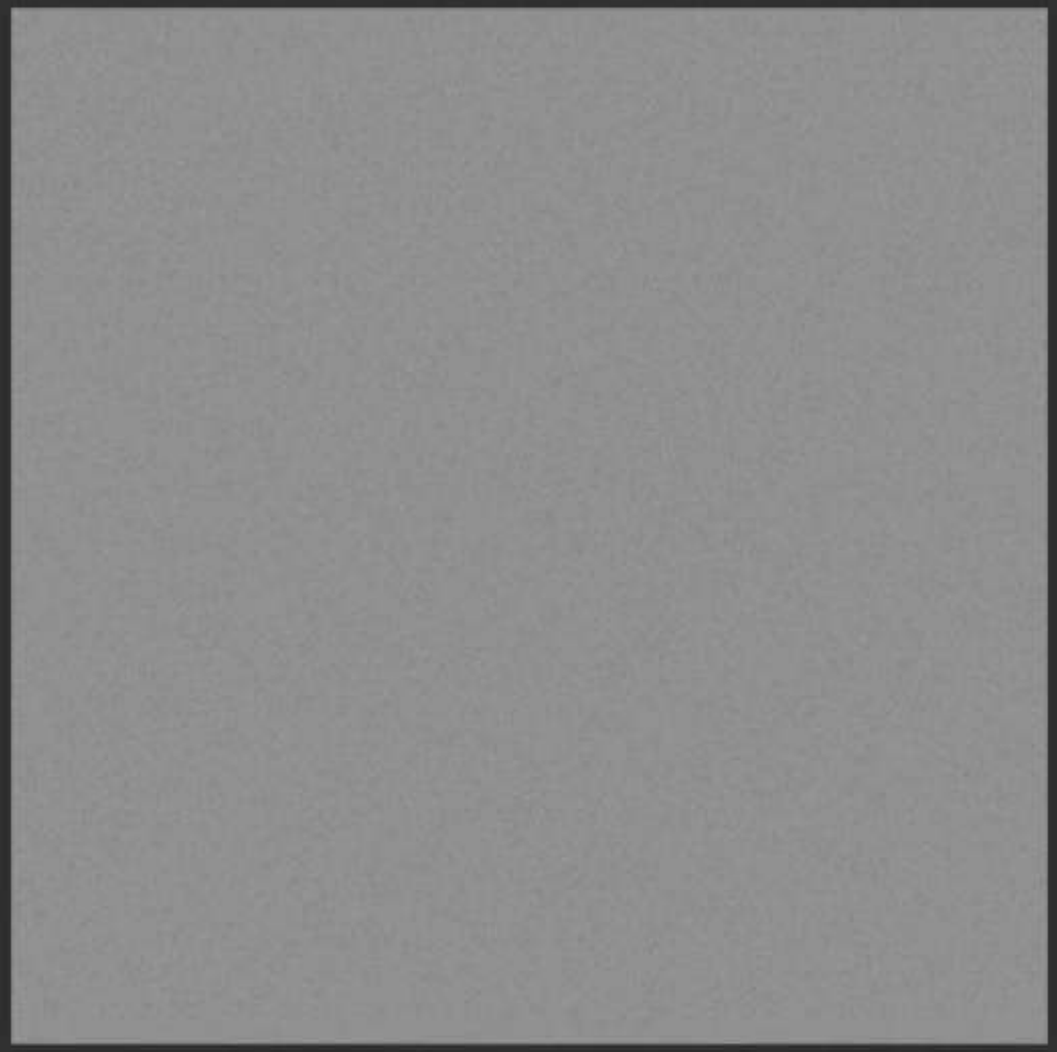


From: Mitsuba 2: A Retargetable Forward and Inverse Renderer [Nimier-David, Vicini, Zleltner, and Jakob 2019]

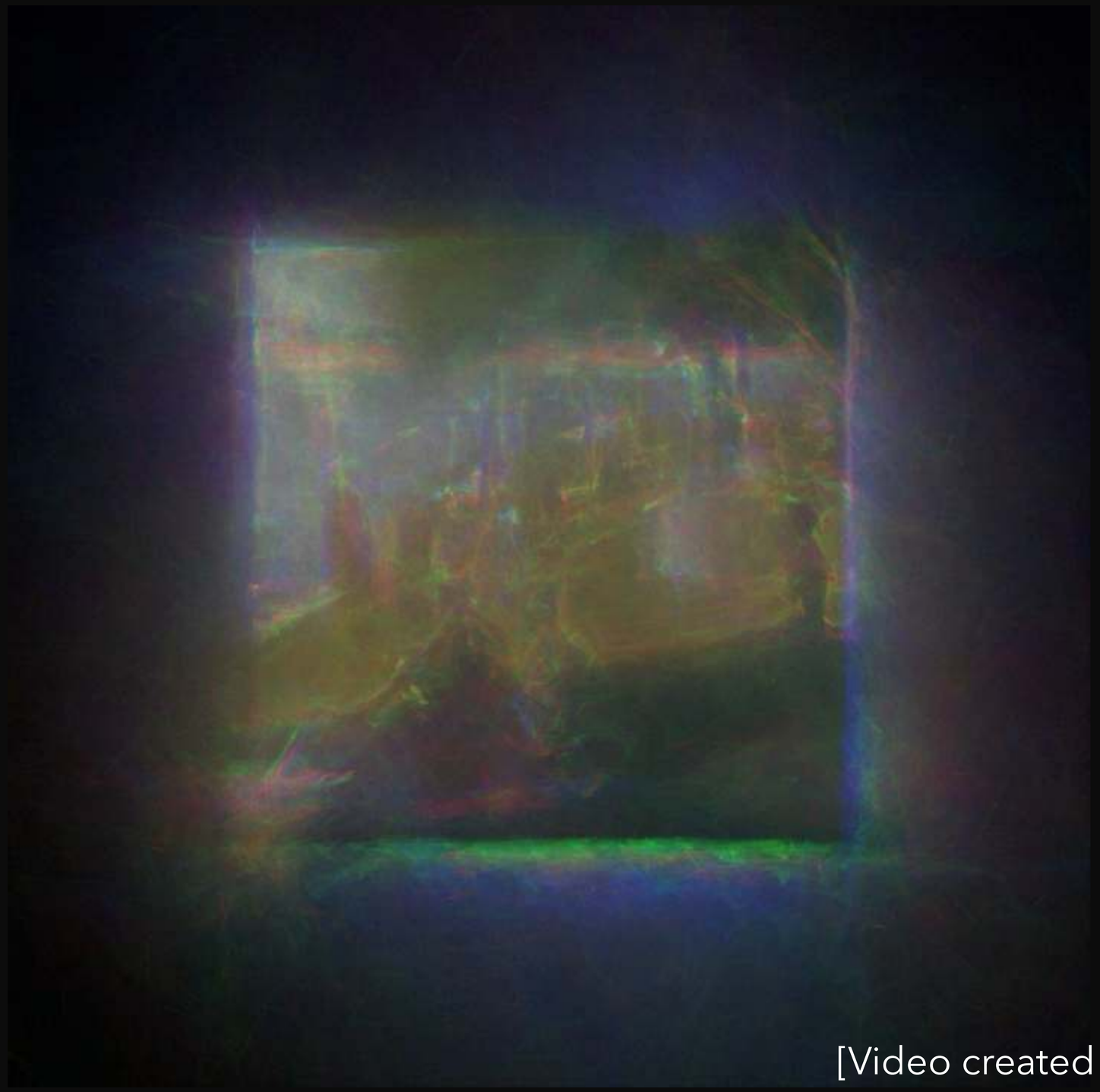
Target image

The Great Wave off Kanagawa by Katsushika Hokusai

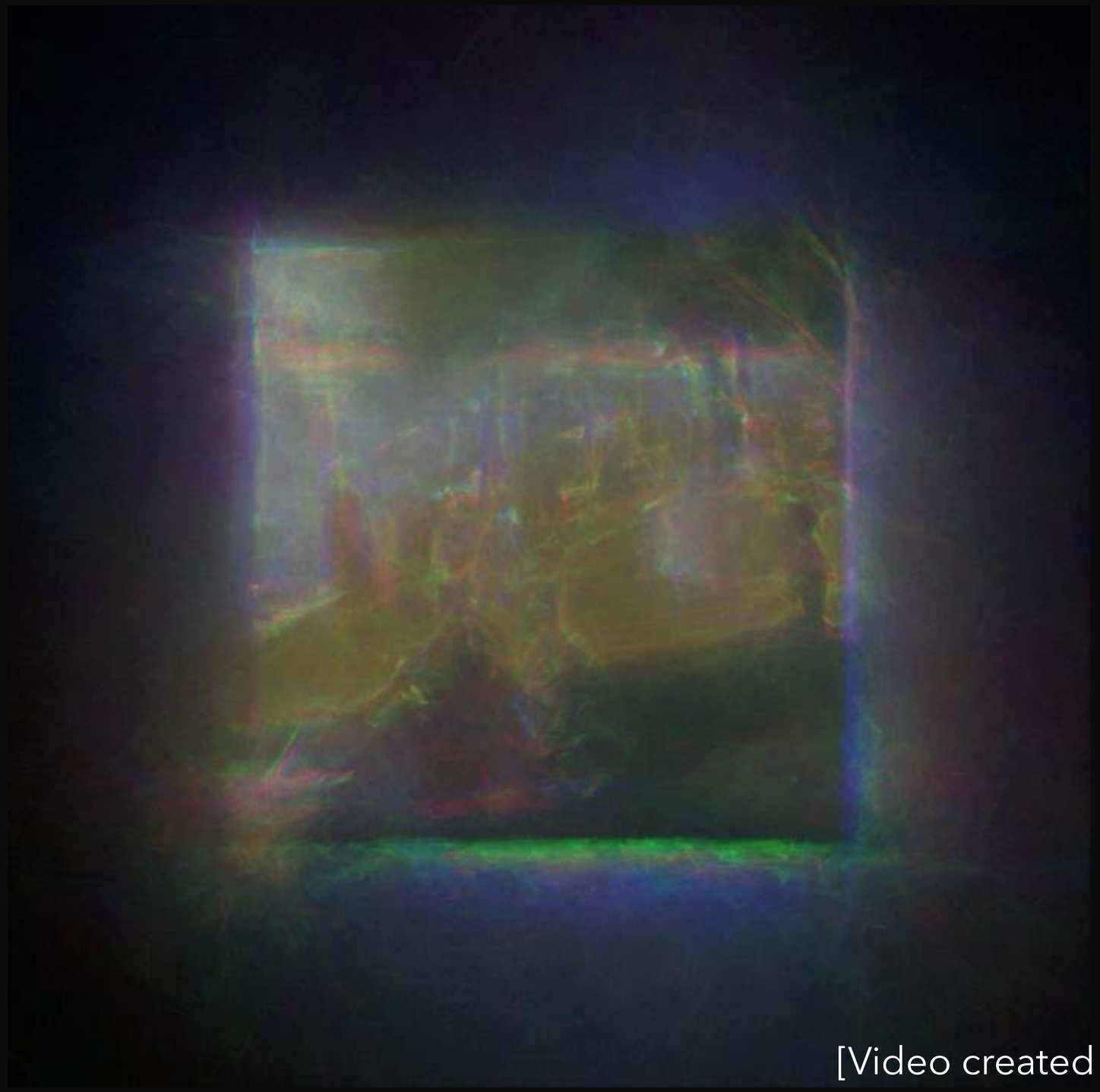
Initial state



From: Mitsuba 2: A Retargetable Forward and Inverse Renderer [Nimier-David, Vicini, Zleltner, and Jakob 2019]

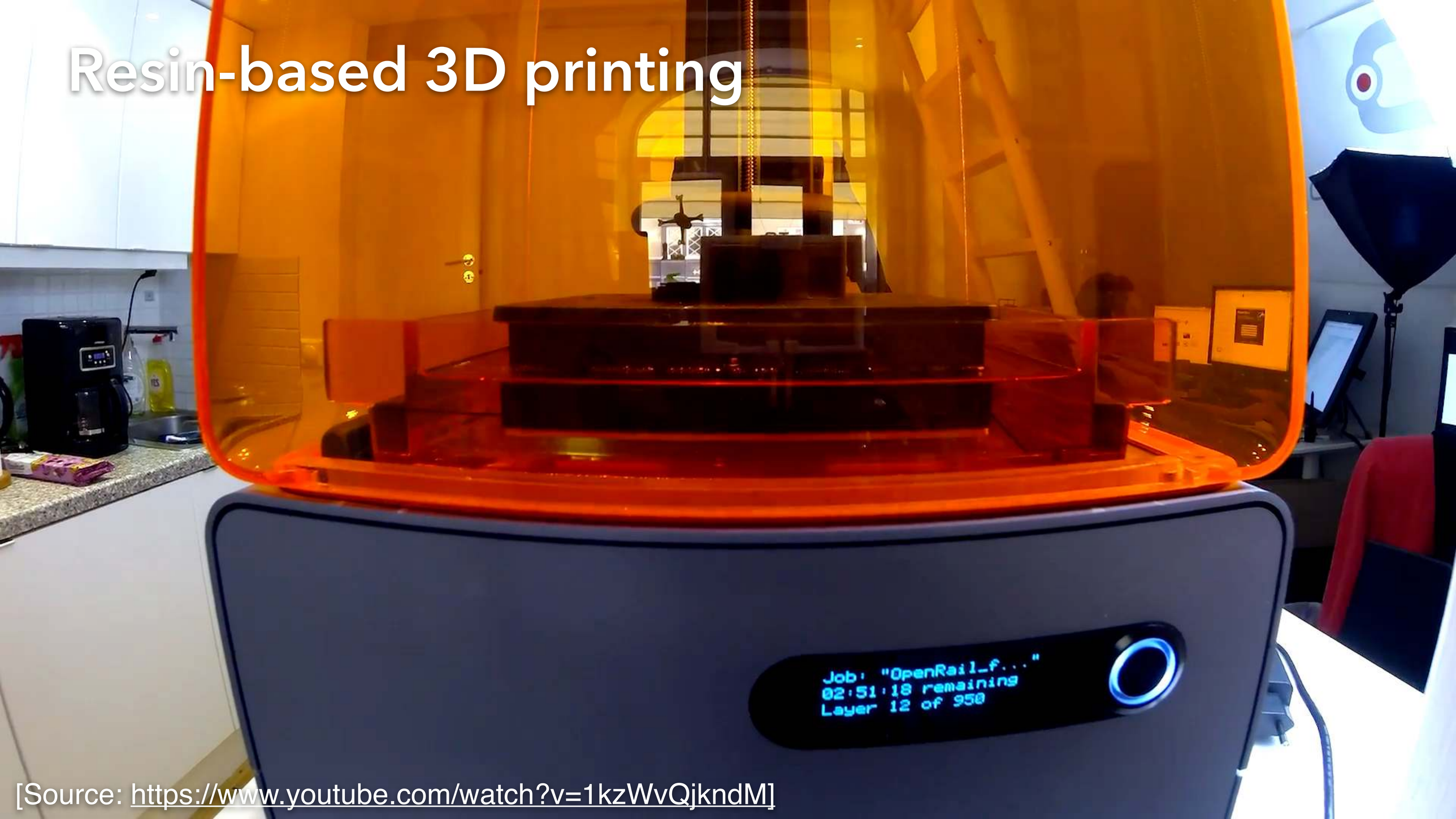


[Video created by Merlin Nimier-David]



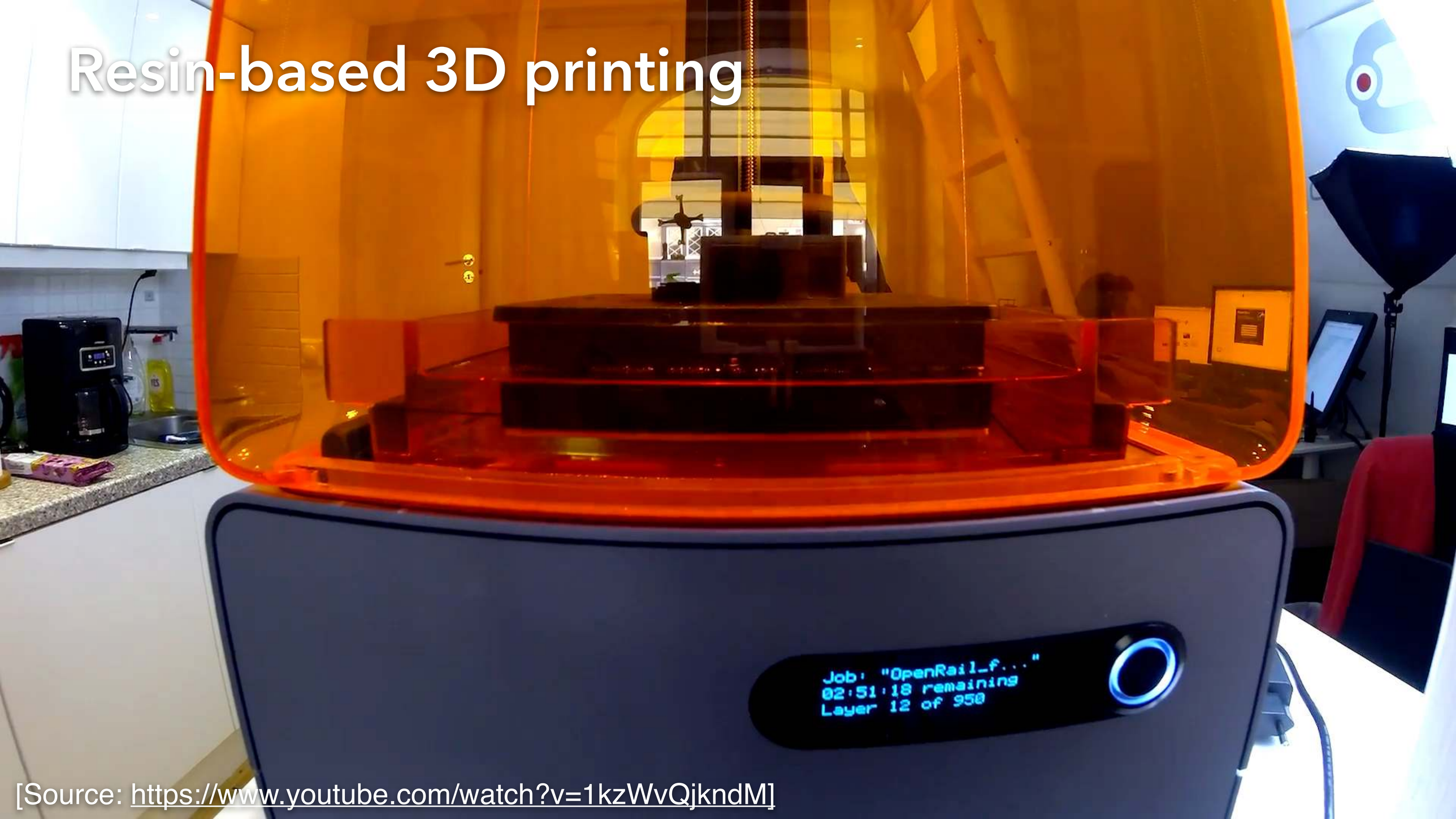
[Video created by Merlin Nimier-David]

Resin-based 3D printing



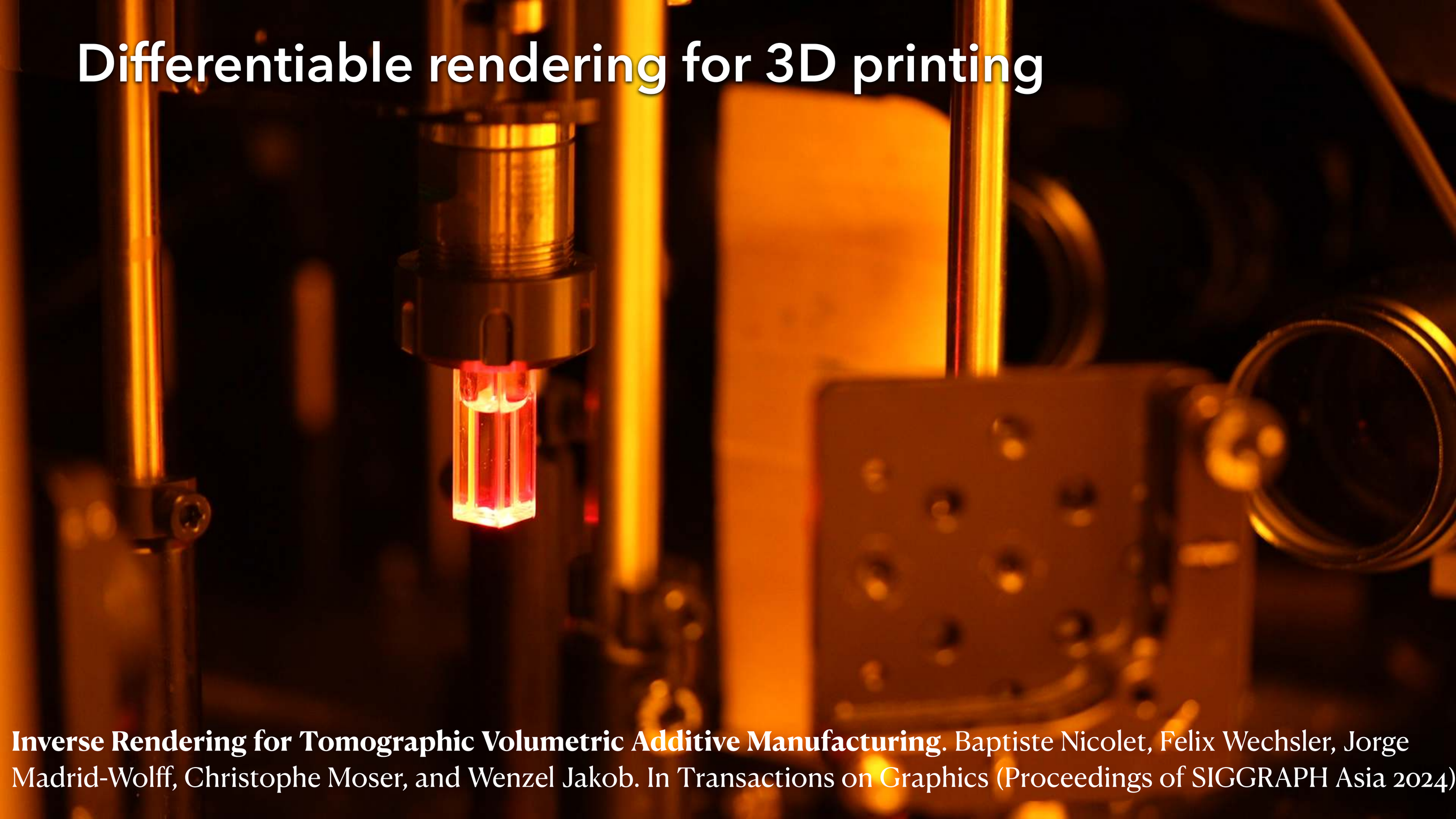
[Source: https://www.youtube.com/watch?v=1kzWvQjkndM]

Resin-based 3D printing



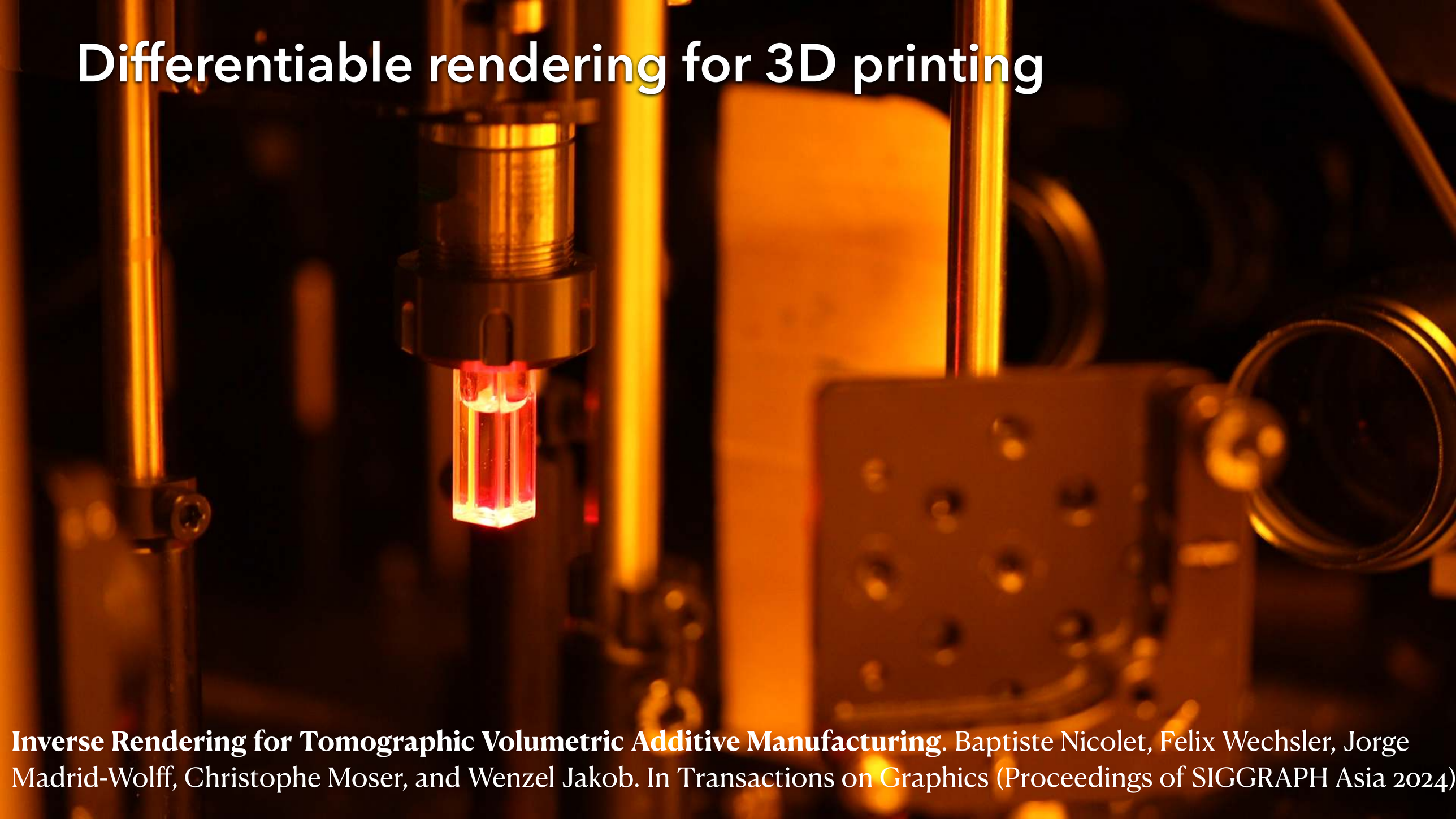
[Source: https://www.youtube.com/watch?v=1kzWvQjkndM]

Differentiable rendering for 3D printing



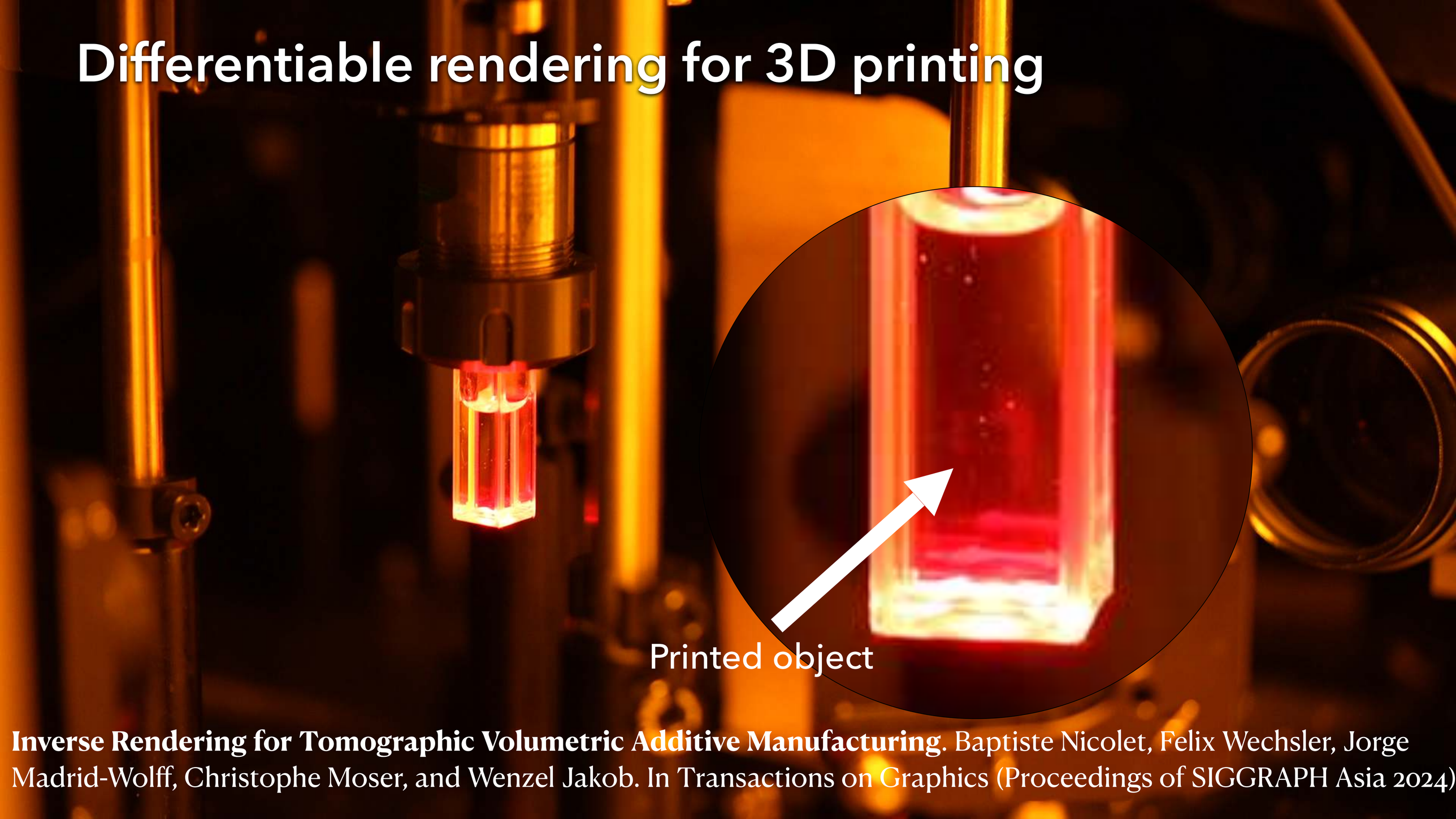
Inverse Rendering for Tomographic Volumetric Additive Manufacturing. Baptiste Nicolet, Felix Wechsler, Jorge Madrid-Wolff, Christophe Moser, and Wenzel Jakob. In Transactions on Graphics (Proceedings of SIGGRAPH Asia 2024)

Differentiable rendering for 3D printing



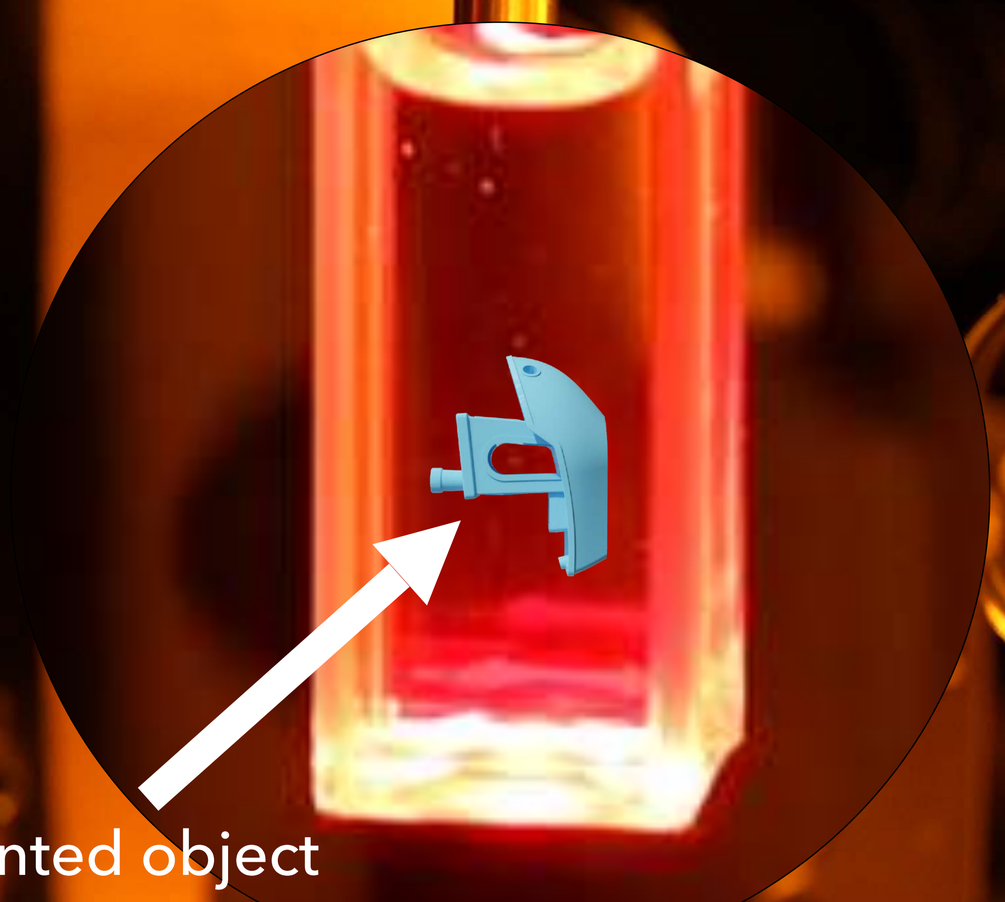
Inverse Rendering for Tomographic Volumetric Additive Manufacturing. Baptiste Nicolet, Felix Wechsler, Jorge Madrid-Wolff, Christophe Moser, and Wenzel Jakob. In Transactions on Graphics (Proceedings of SIGGRAPH Asia 2024)

Differentiable rendering for 3D printing



Inverse Rendering for Tomographic Volumetric Additive Manufacturing. Baptiste Nicolet, Felix Wechsler, Jorge Madrid-Wolff, Christophe Moser, and Wenzel Jakob. In Transactions on Graphics (Proceedings of SIGGRAPH Asia 2024)

Differentiable rendering for 3D printing

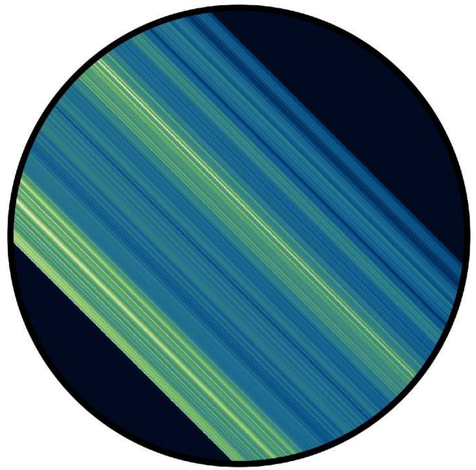


Printed object

Inverse Rendering for Tomographic Volumetric Additive Manufacturing. Baptiste Nicolet, Felix Wechsler, Jorge Madrid-Wolff, Christophe Moser, and Wenzel Jakob. In Transactions on Graphics (Proceedings of SIGGRAPH Asia 2024)

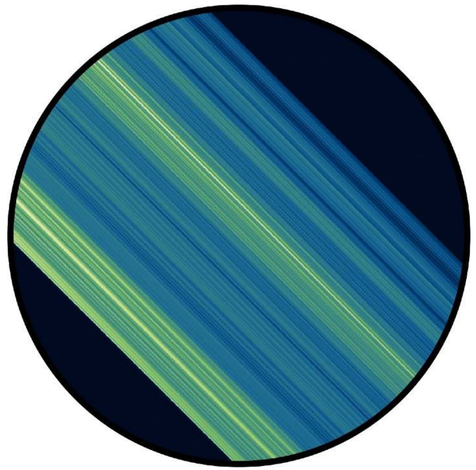
It's more complicated..

Backprojection

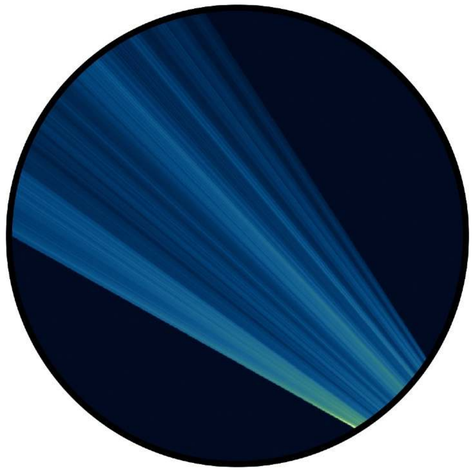


It's more complicated..

Backprojection

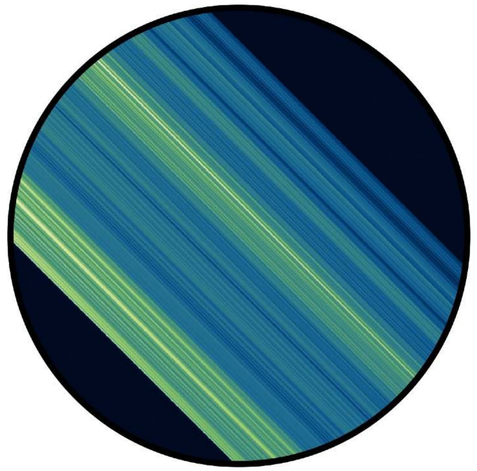


Refraction

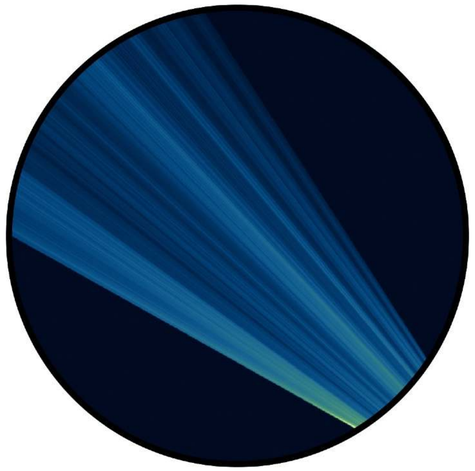


It's more complicated..

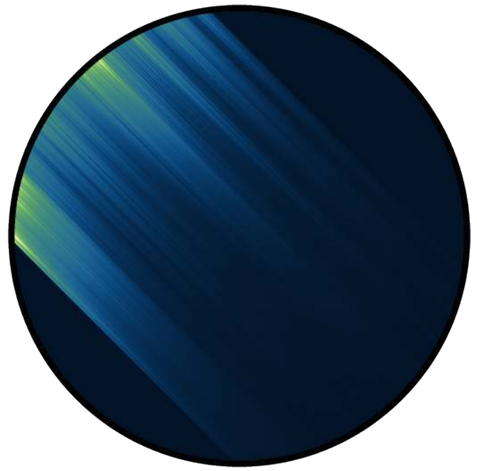
Backprojection



Refraction

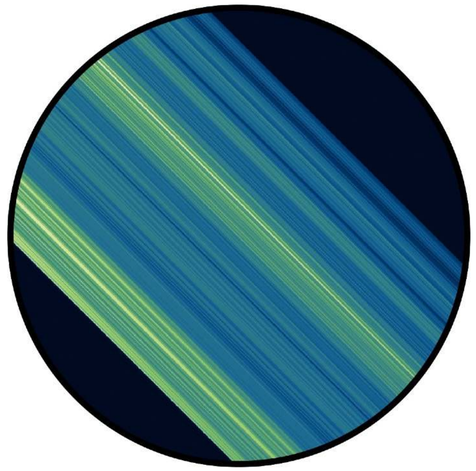


Attenuation

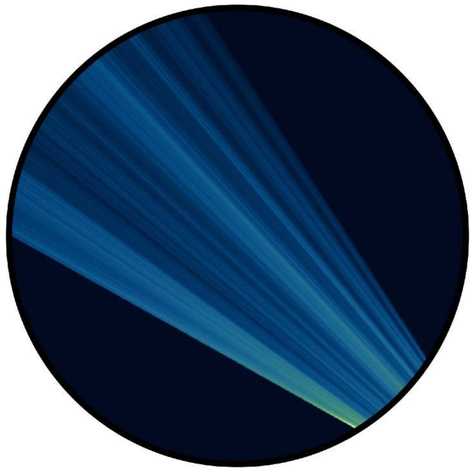


It's more complicated..

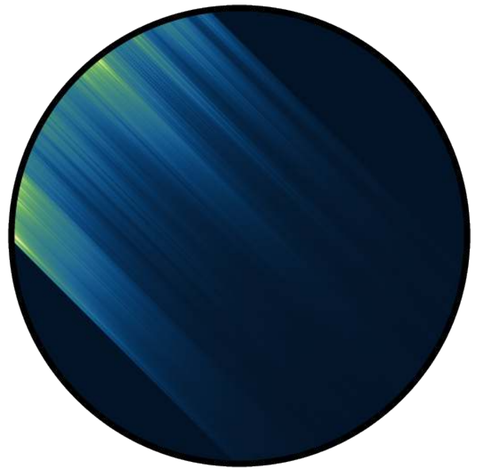
Backprojection



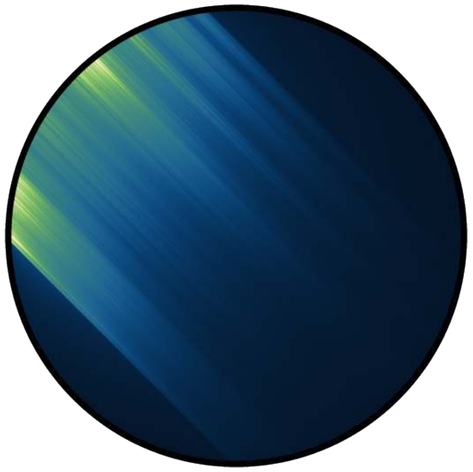
Refraction



Attenuation

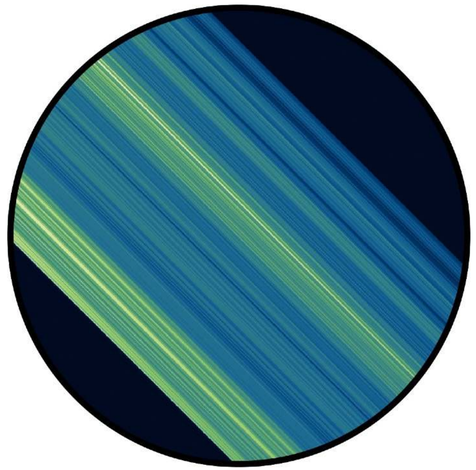


Scattering

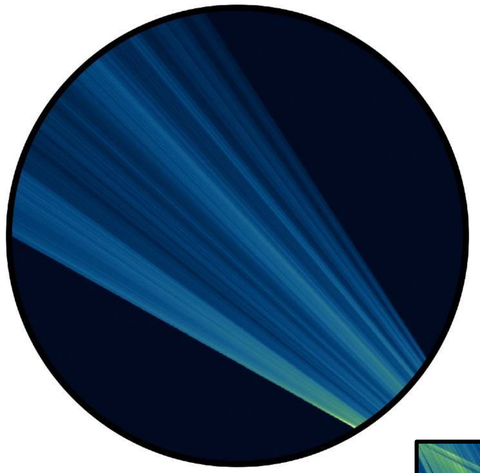


It's more complicated..

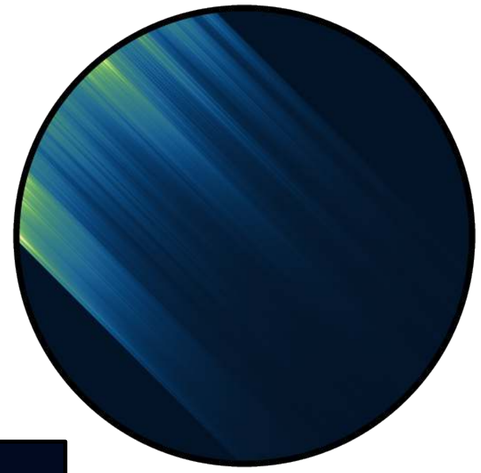
Backprojection

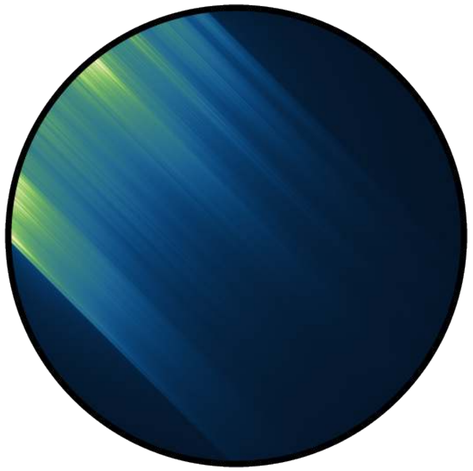


Refraction

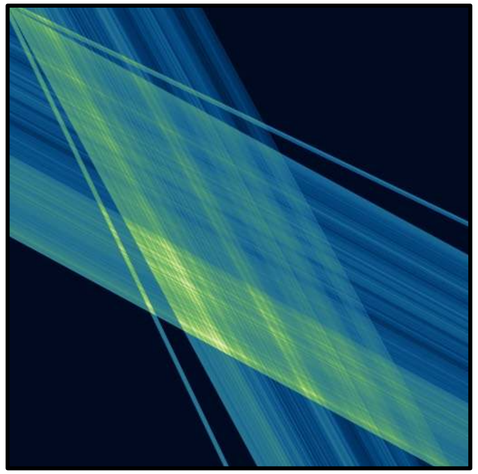


Attenuation



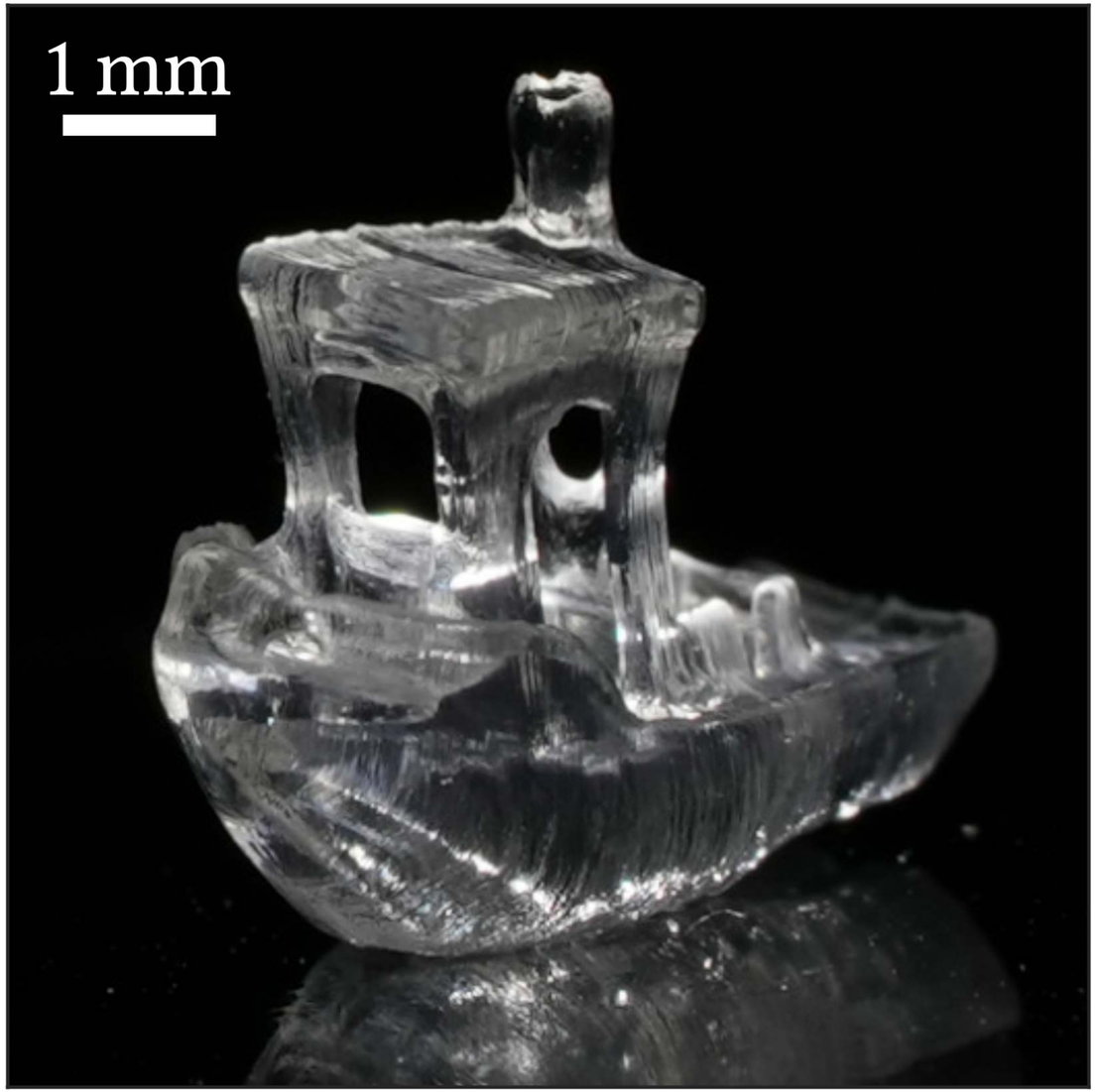


Scattering

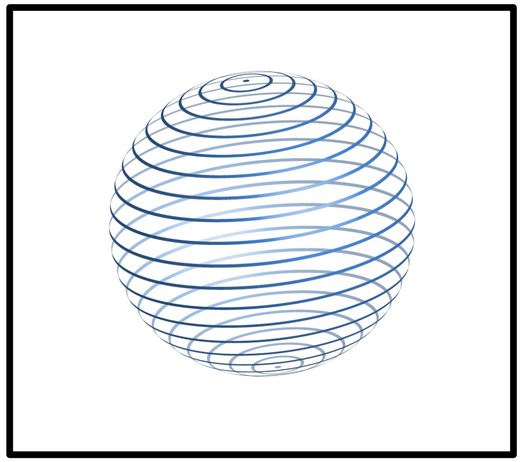


Arbitrary geometry

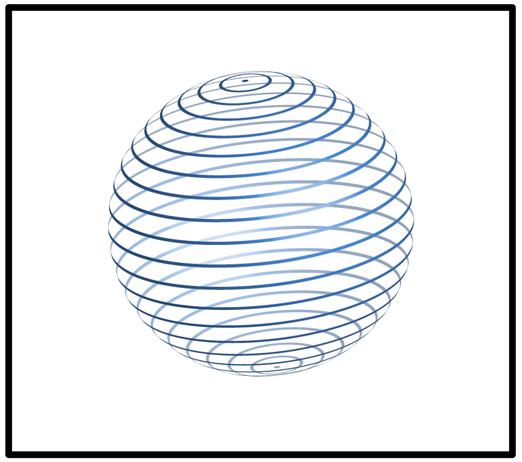
Tomographic print



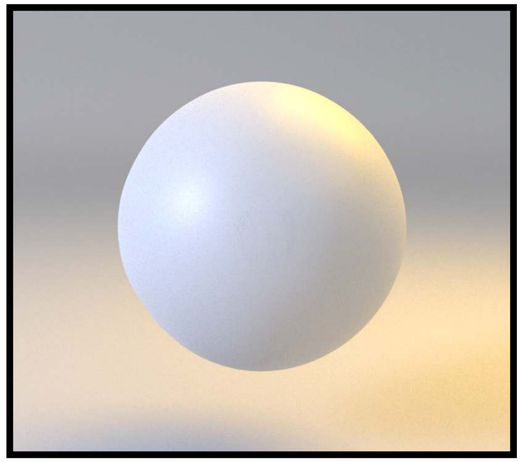
3D model



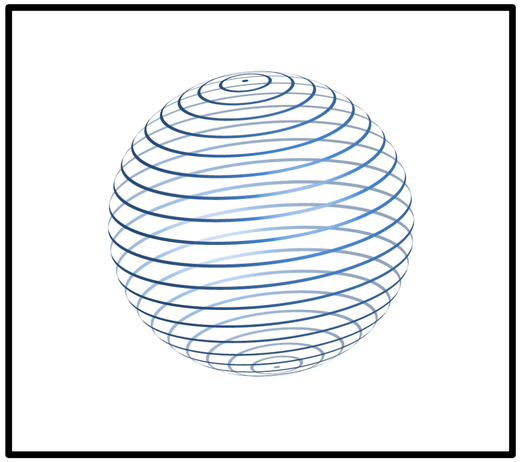
3D model



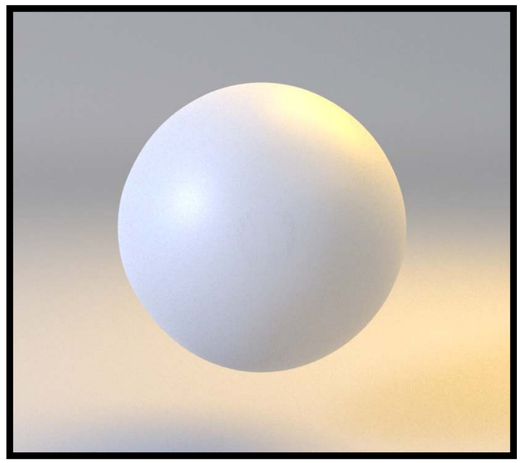
Rendering



3D model

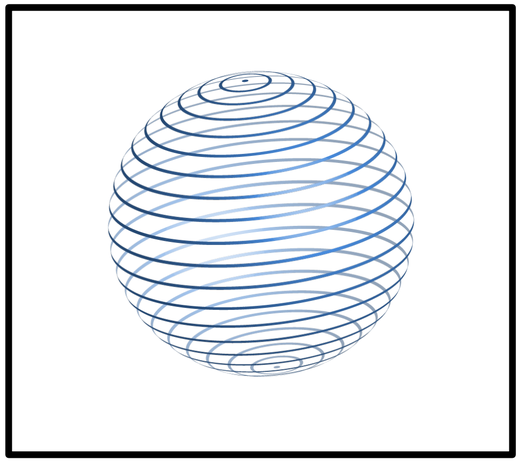


Rendering

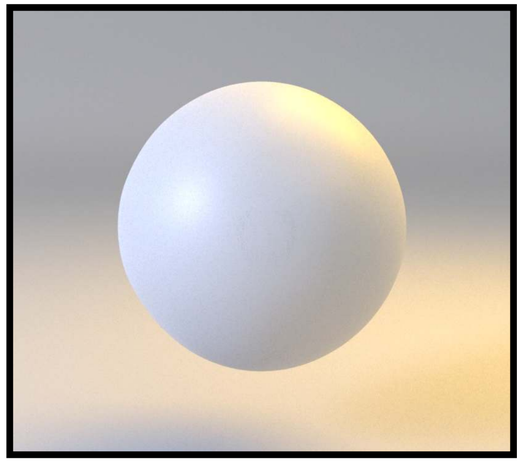


Reference

3D model



Rendering

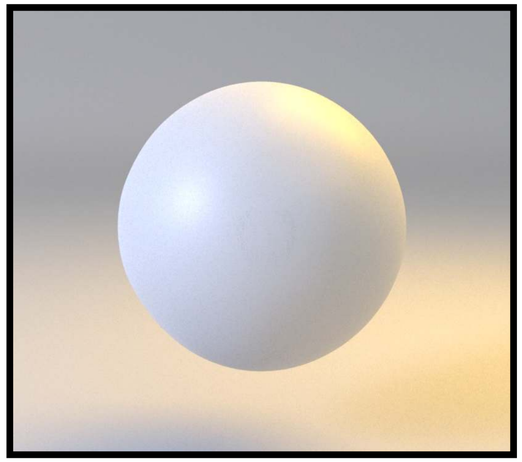


Reference

0.5231

Loss

Rendering



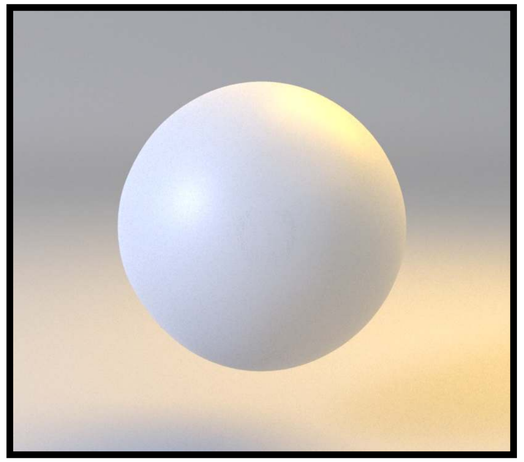
Reference

0.5231

Loss

[Inspired by a slide by Delio Vicini]

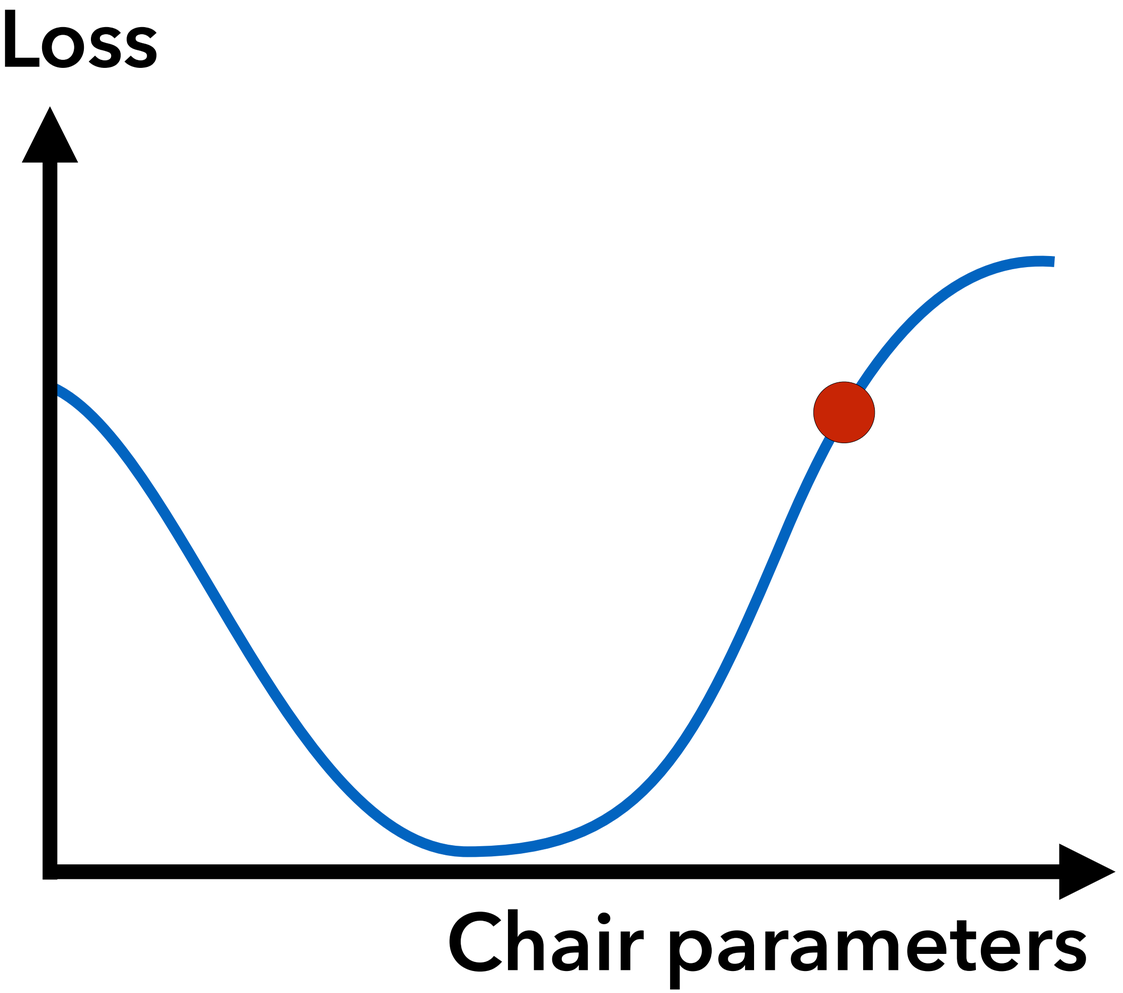
Rendering



Reference

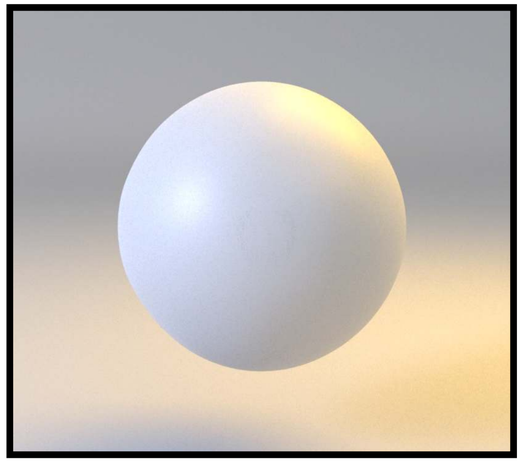
0.5231

Loss



[Inspired by a slide by Delio Vicini]

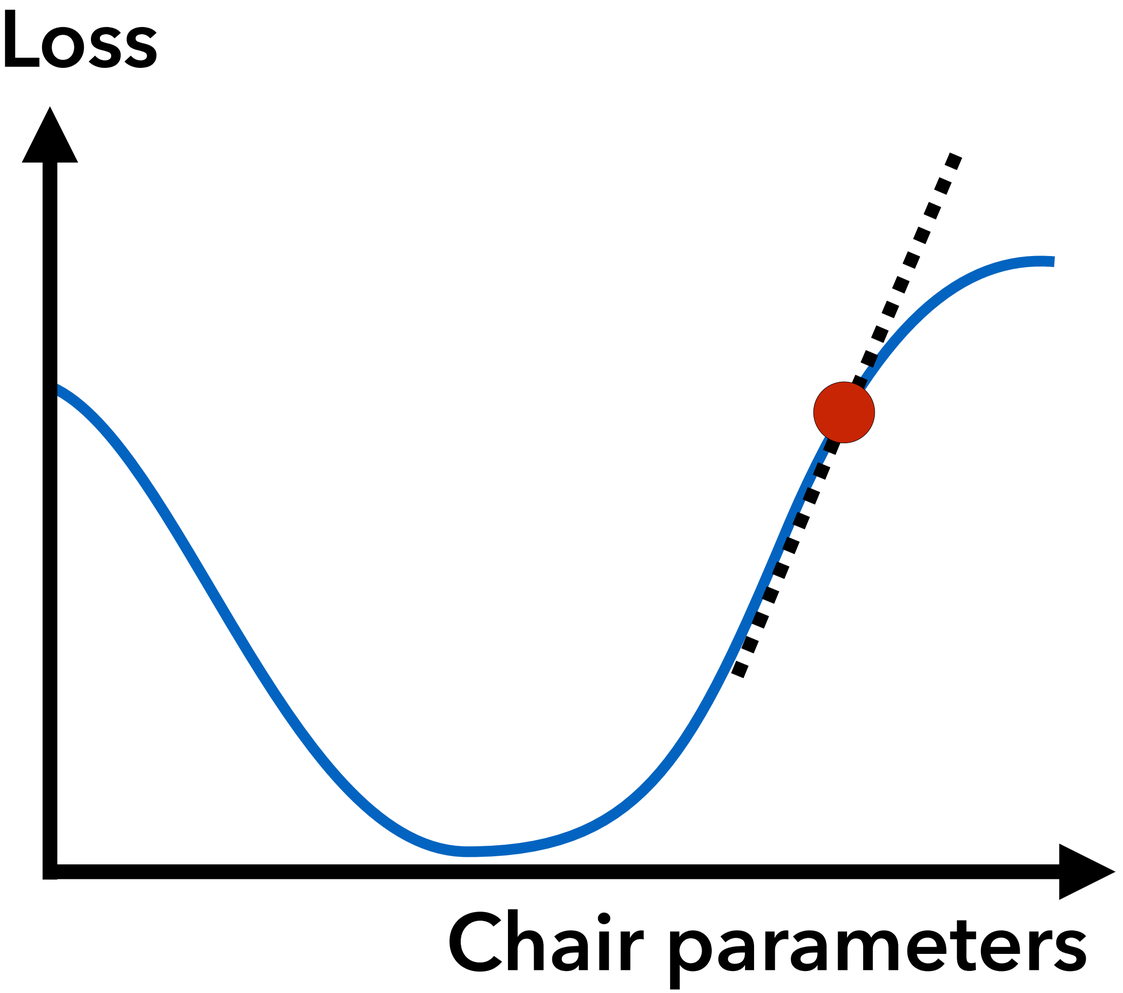
Rendering



Reference

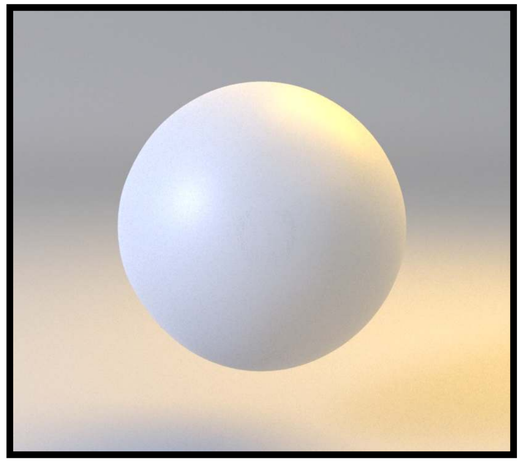
0.5231

Loss



[Inspired by a slide by Delio Vicini]

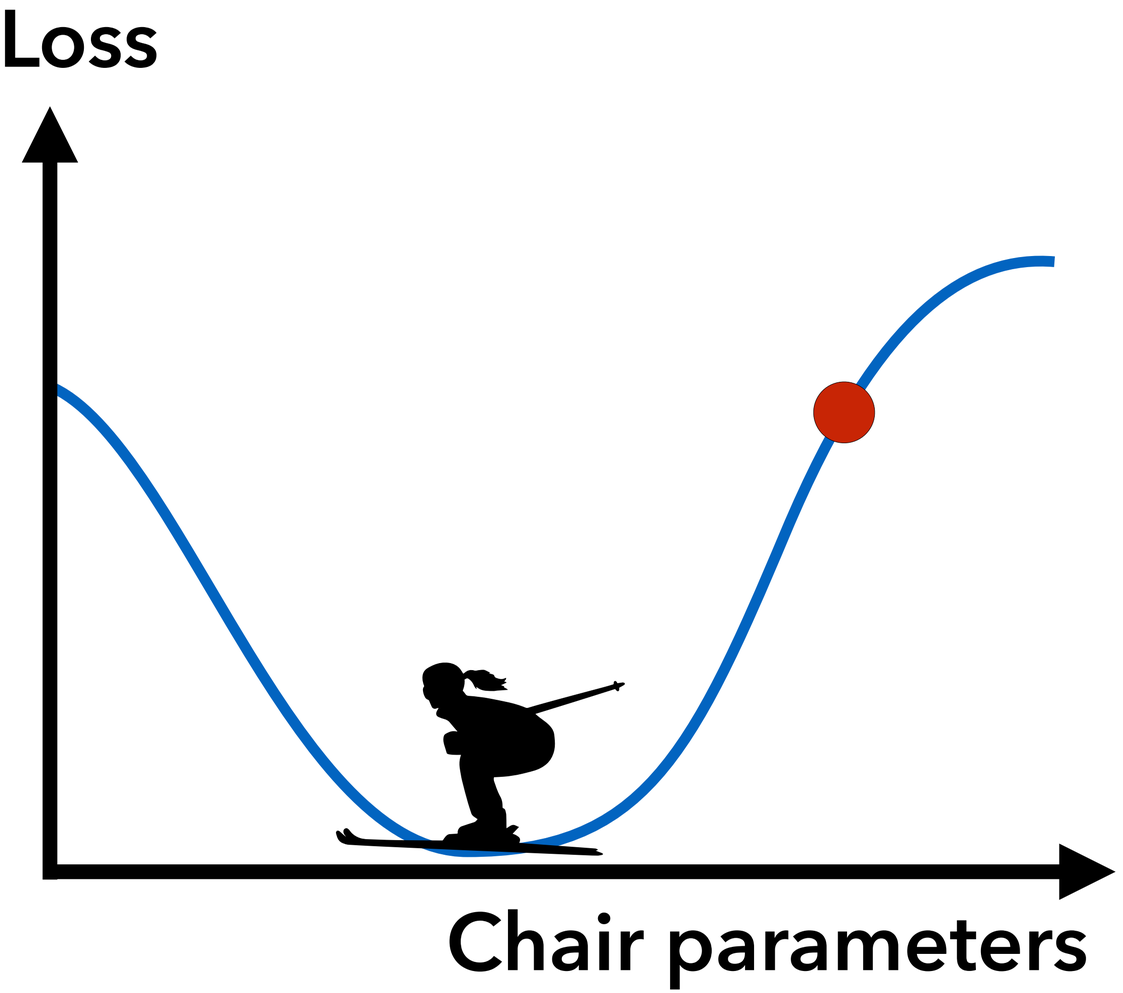
Rendering



Reference

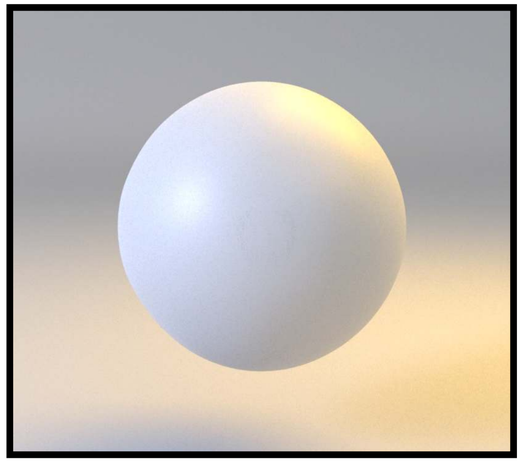
0.5231

Loss



[Inspired by a slide by Delio Vicini]

Rendering

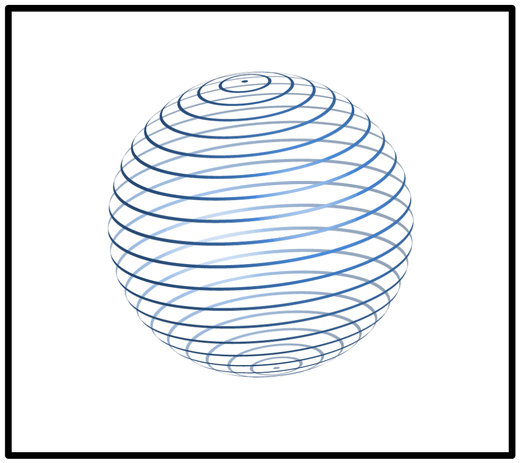


Reference

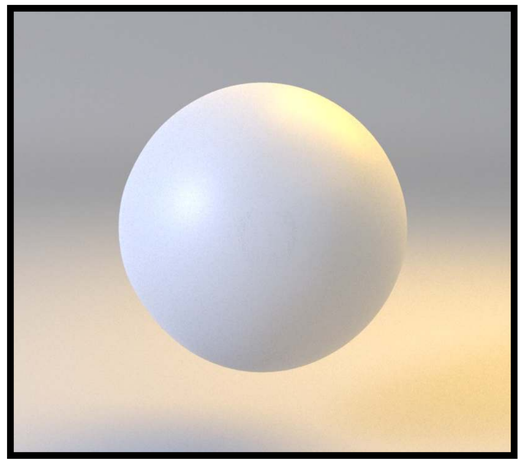
0.5231

Loss

3D model



Rendering

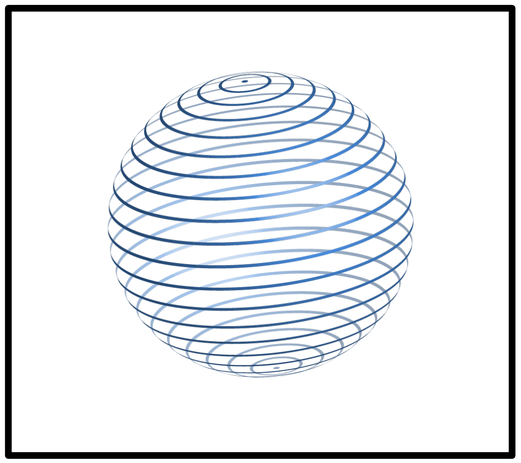


Reference

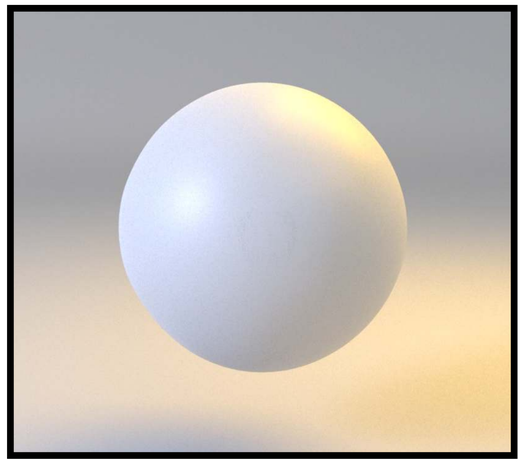
0.5231

Loss

3D model



Rendering



Reference

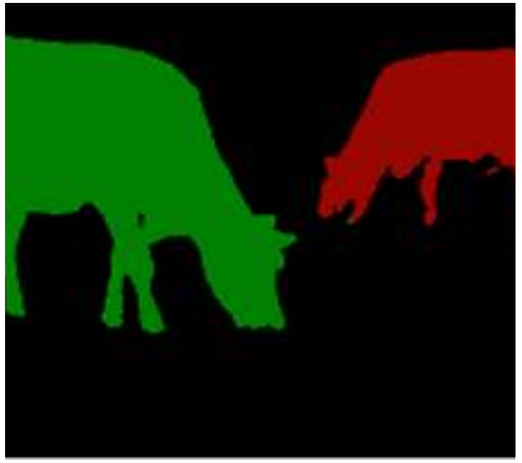
0.5231

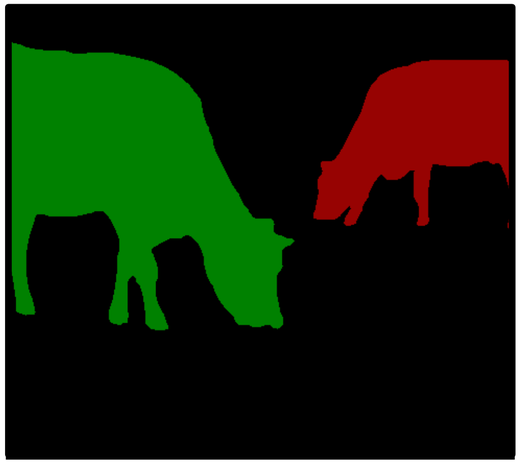
Loss

[Cattle Target Segmentation Method in Multi-Scenes Using Improved DeepLabV3+ method, Feng et al. 2023]

Image

Segmentation





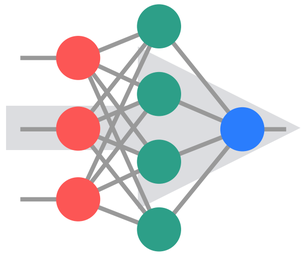
Reference

0.5231

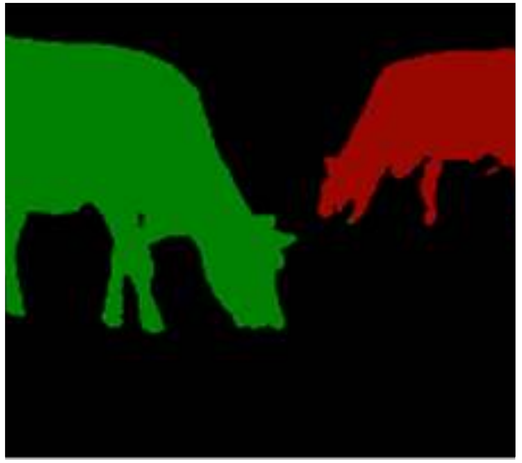
Loss

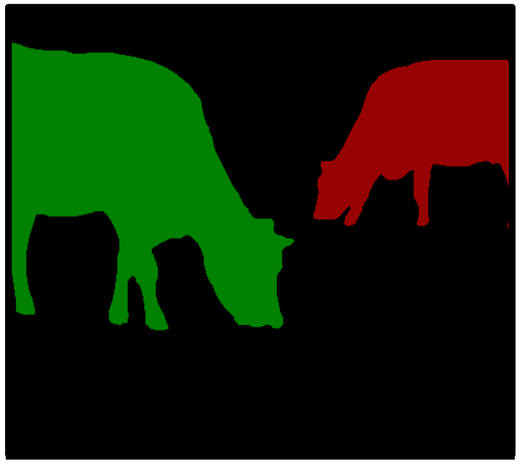
[Cattle Target Segmentation Method in Multi-Scenes Using Improved DeepLabV3+ method, Feng et al. 2023]

Image



Segmentation



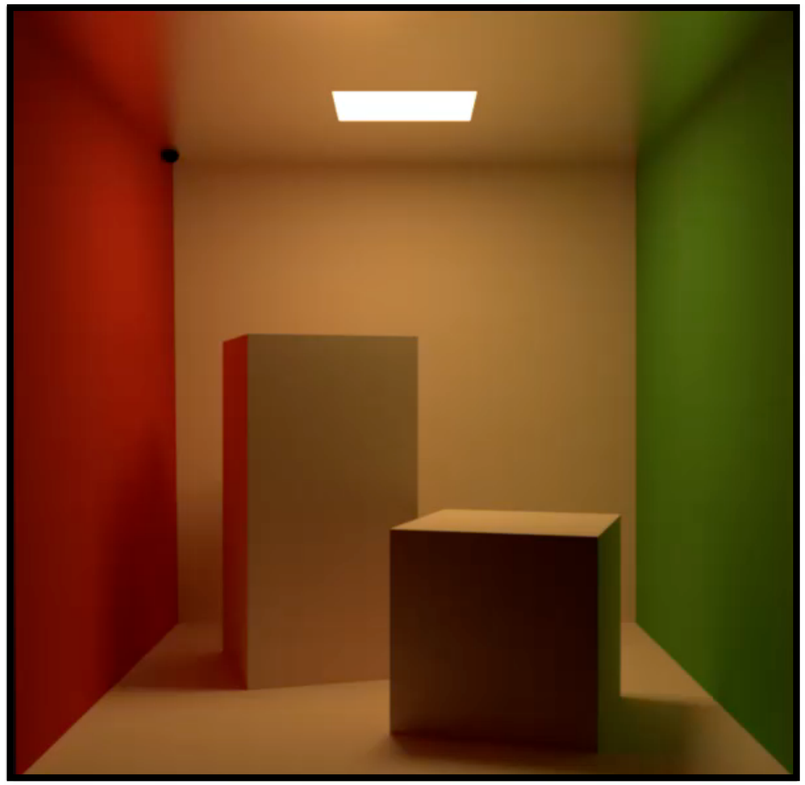


Reference

0.5231

Loss

The rendering equation

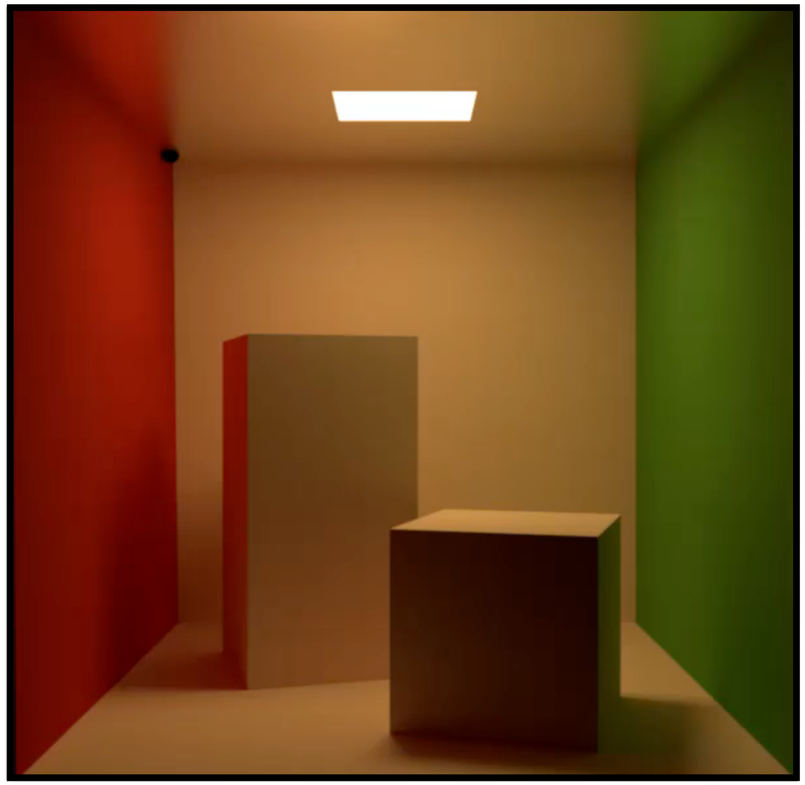


$$\int_{S^2} \cdot \, \mathrm{d}\omega$$

$$= \int_{S^2} \mathrm{d}\omega =$$

Final pixel color

The rendering equation

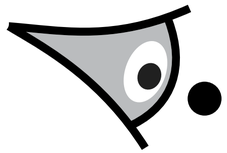


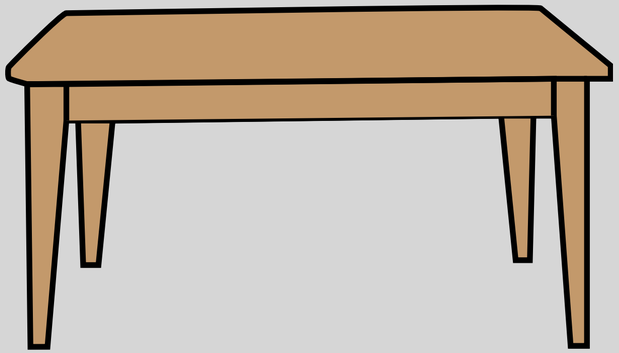
$$\int_{S^2} \cdot \, \mathrm{d}\omega$$

$$= \int_{S^2} \mathrm{d}\omega =$$

Final pixel color

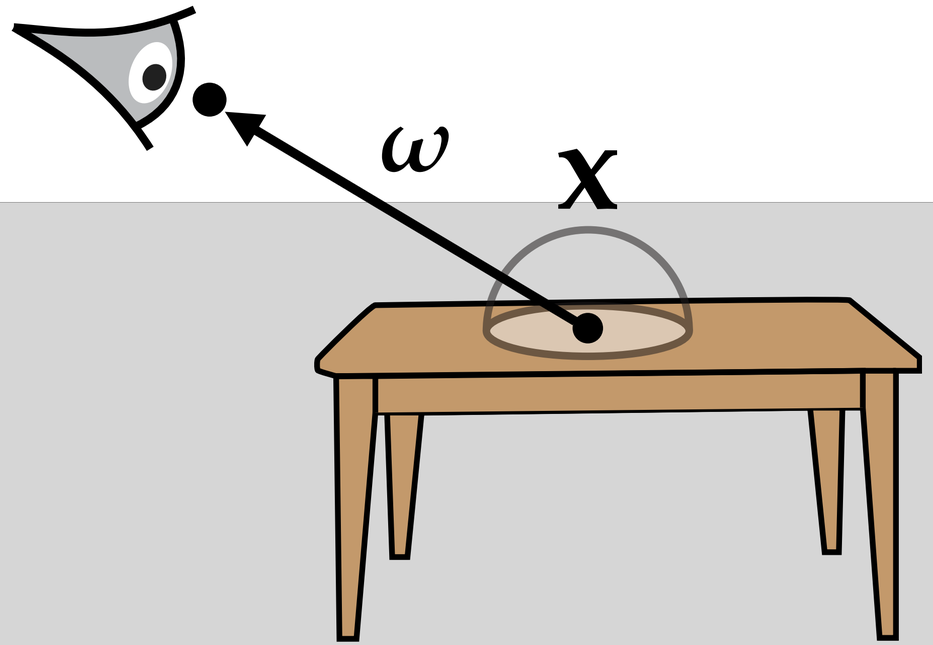
Realistic images via Monte Carlo integration





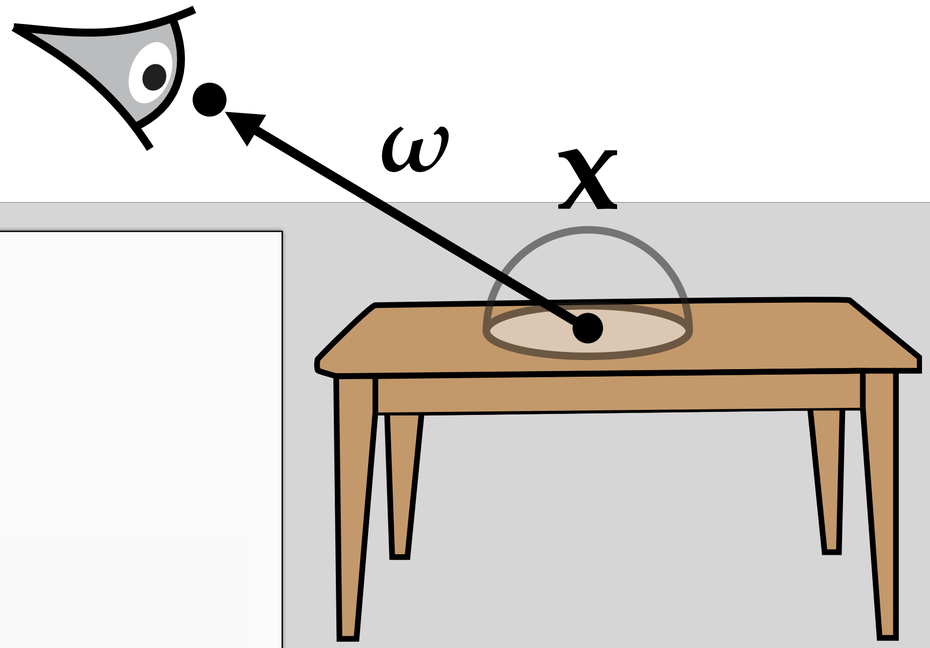
$$L_o(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}'), -\boldsymbol{\omega}')\, M(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\omega$$

Realistic images via Monte Carlo integration



$$L_o(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}'), -\boldsymbol{\omega}')\, M(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\omega$$

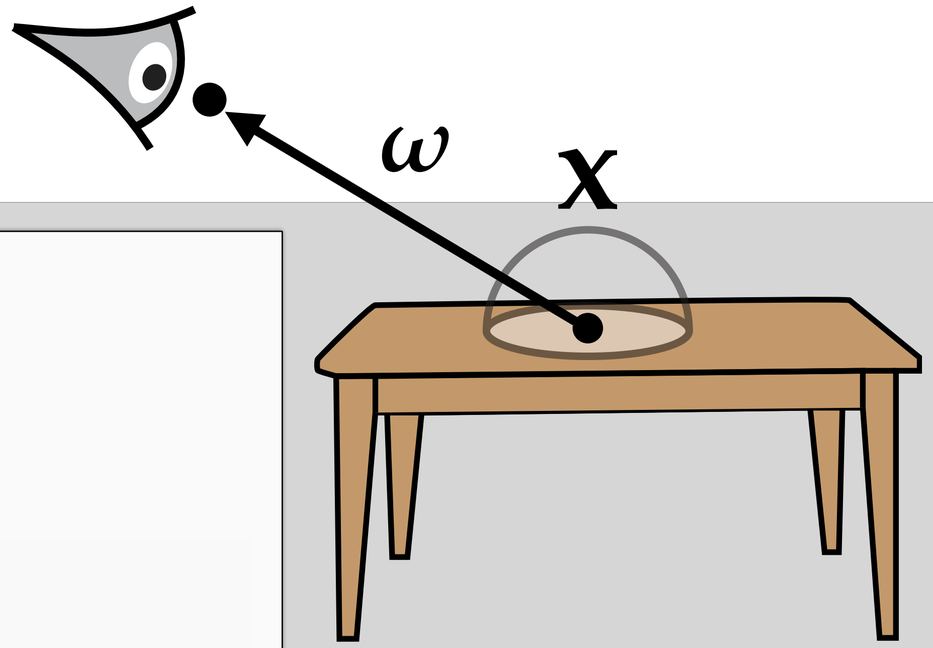
Realistic images via Monte Carlo integration




```
def Lo(x, ω):
```

$$L_o(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}'), -\boldsymbol{\omega}')\, M(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\omega$$

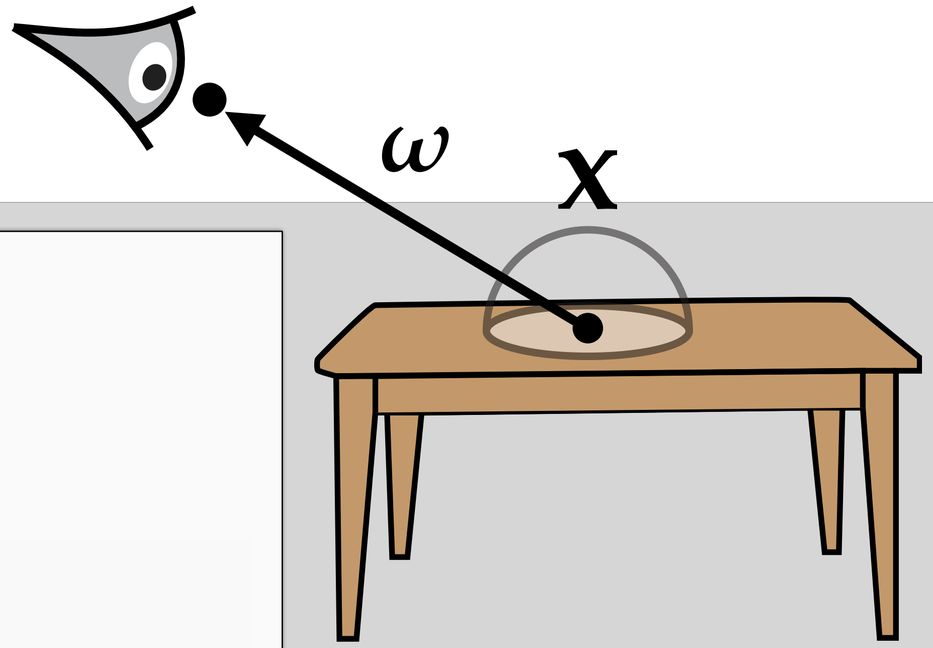
Realistic images via Monte Carlo integration




```
def Lo(x, ω):
```

$$L_o(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}'), -\boldsymbol{\omega}')\, M(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\omega$$

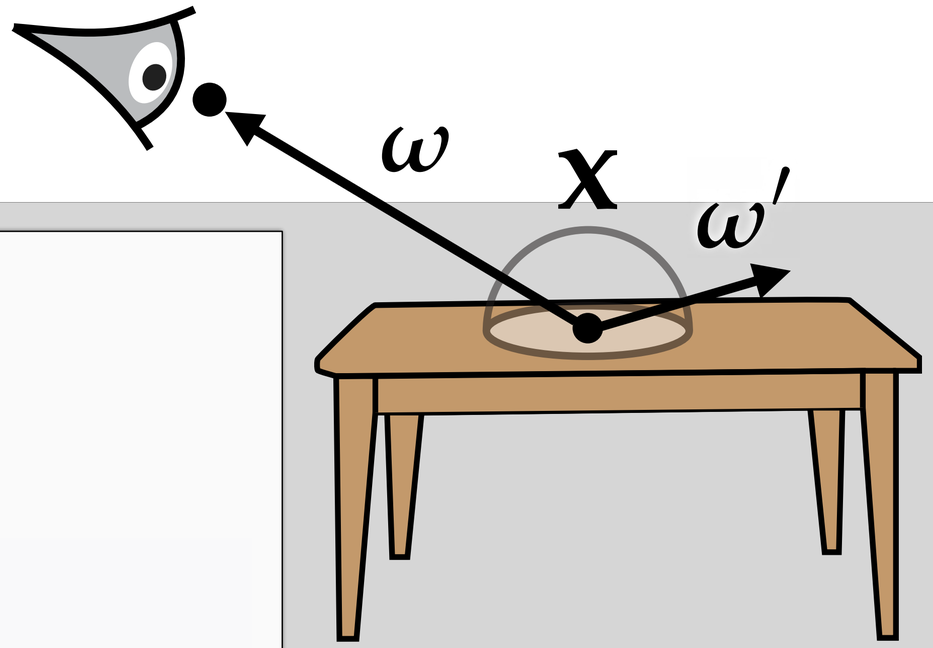
Realistic images via Monte Carlo integration




```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
```

$$L_o(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}'), -\boldsymbol{\omega}')\, M(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\omega$$

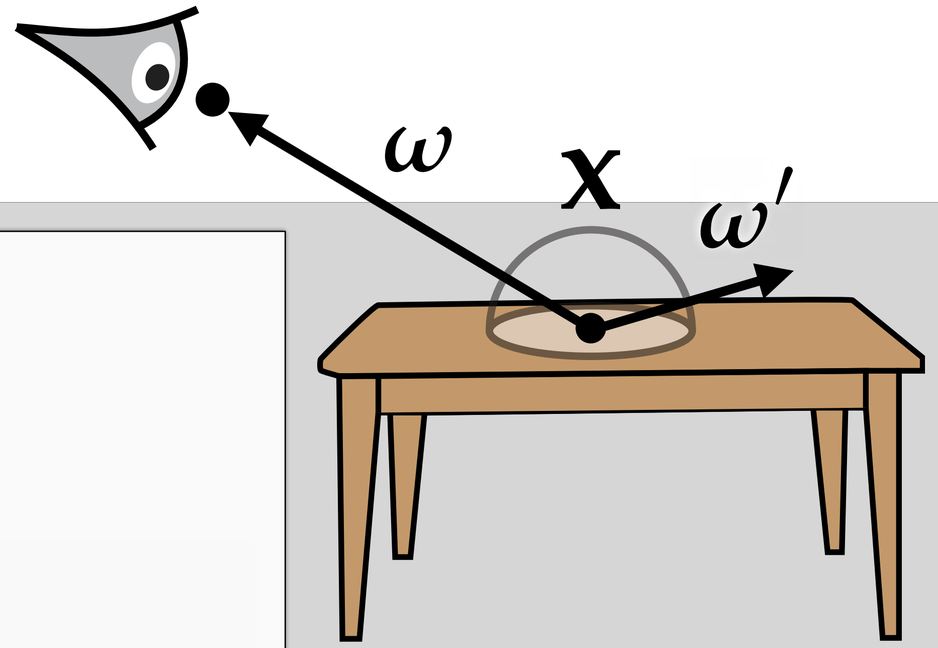
Realistic images via Monte Carlo integration

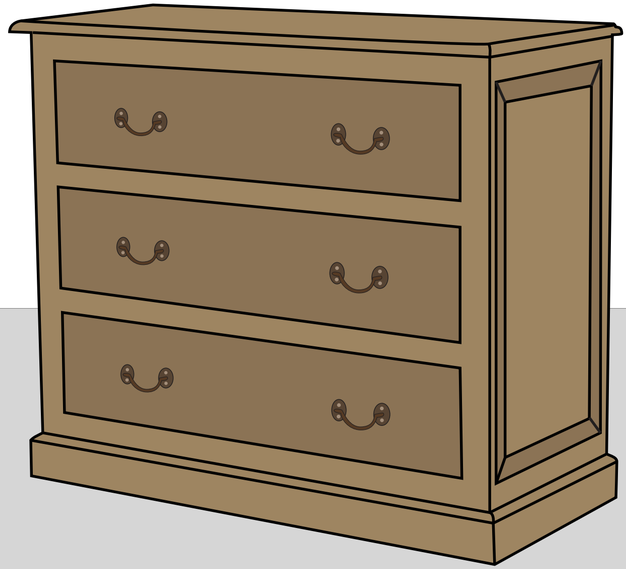



```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
```

$$L_o(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}'), -\boldsymbol{\omega}')\, M(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\omega$$

Realistic images via Monte Carlo integration

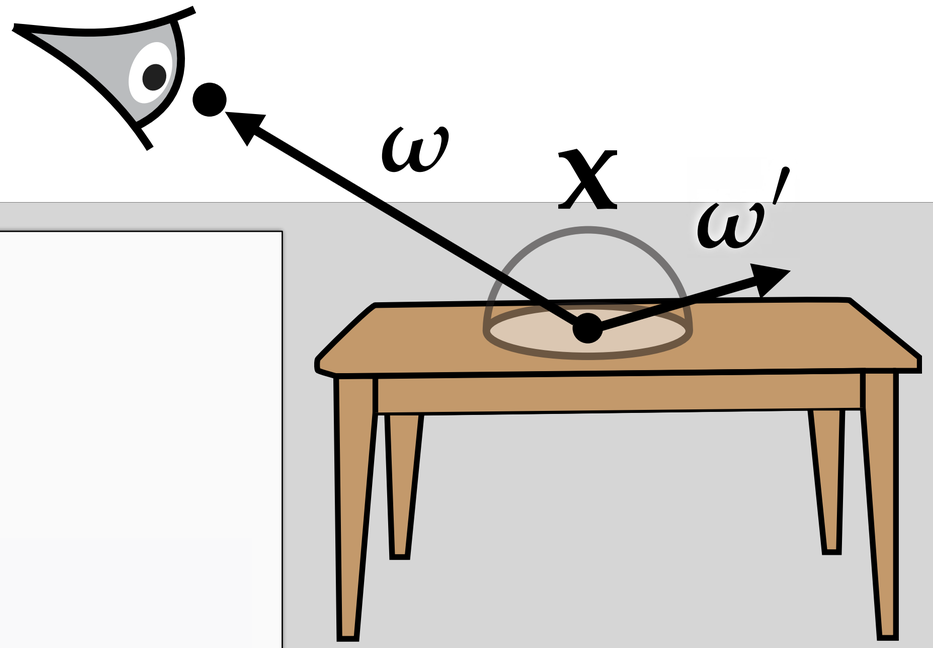


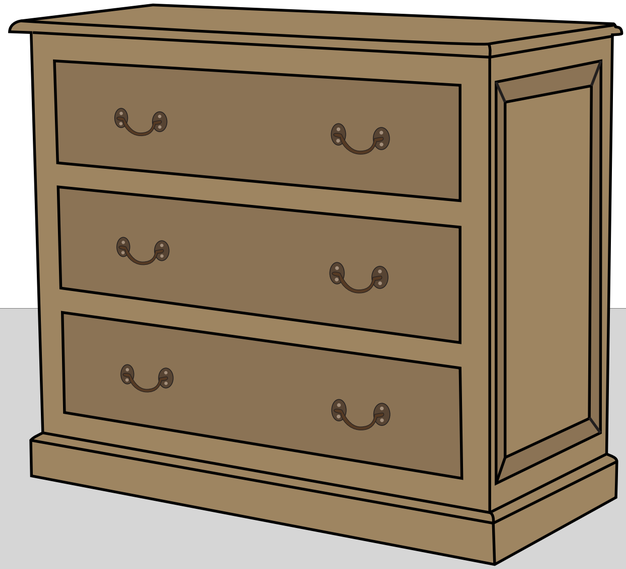


```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
```

$$L_o(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}'), -\boldsymbol{\omega}')\, M(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\omega$$

Realistic images via Monte Carlo integration

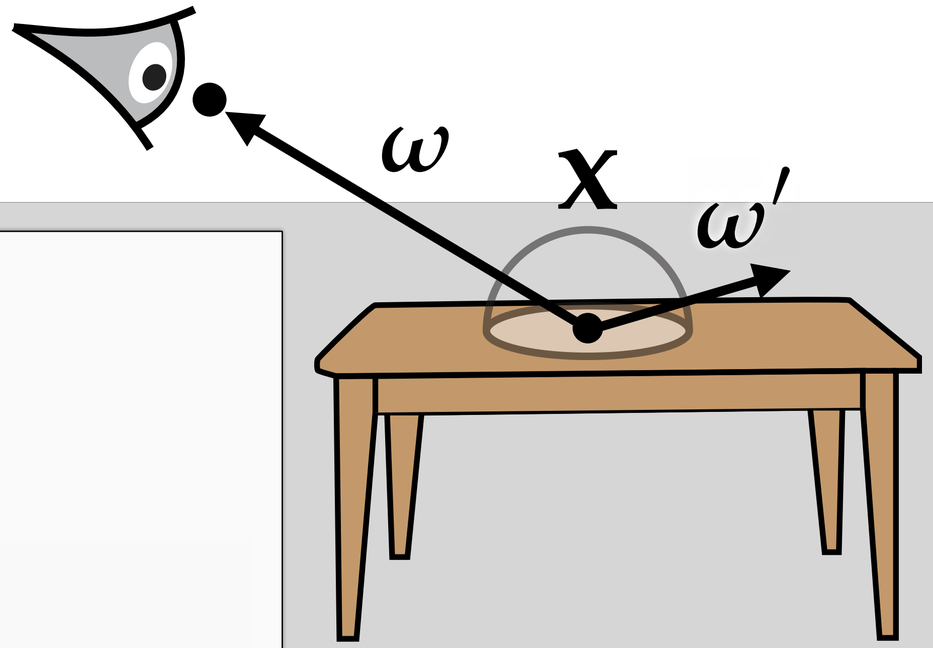


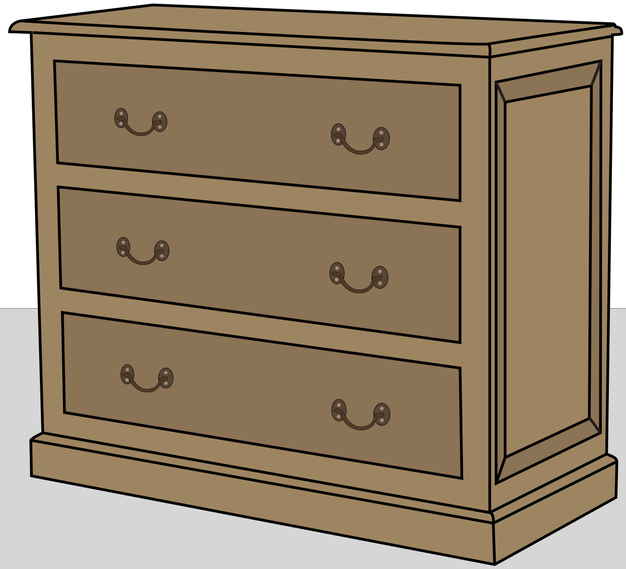


```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
```

$$L_o(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}'), -\boldsymbol{\omega}')\, M(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\omega$$

Realistic images via Monte Carlo integration

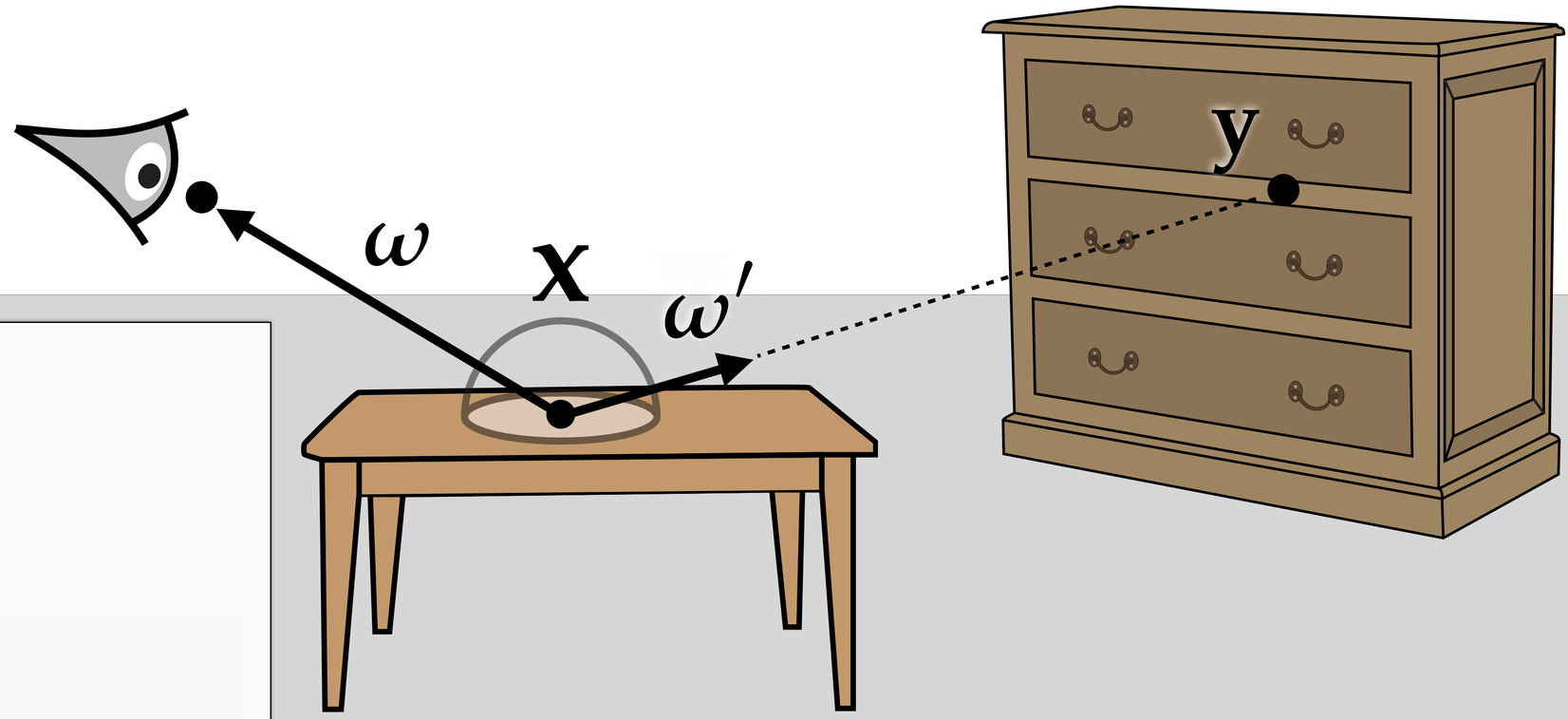




```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
    y = r(x, ω')
```

$$L_o(\mathbf{x},\boldsymbol{\omega}) = L_e(\mathbf{x},\boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x},\boldsymbol{\omega}'),-\boldsymbol{\omega}')\, M(\mathbf{x},\boldsymbol{\omega},\boldsymbol{\omega}')\, \mathrm{d}\omega$$

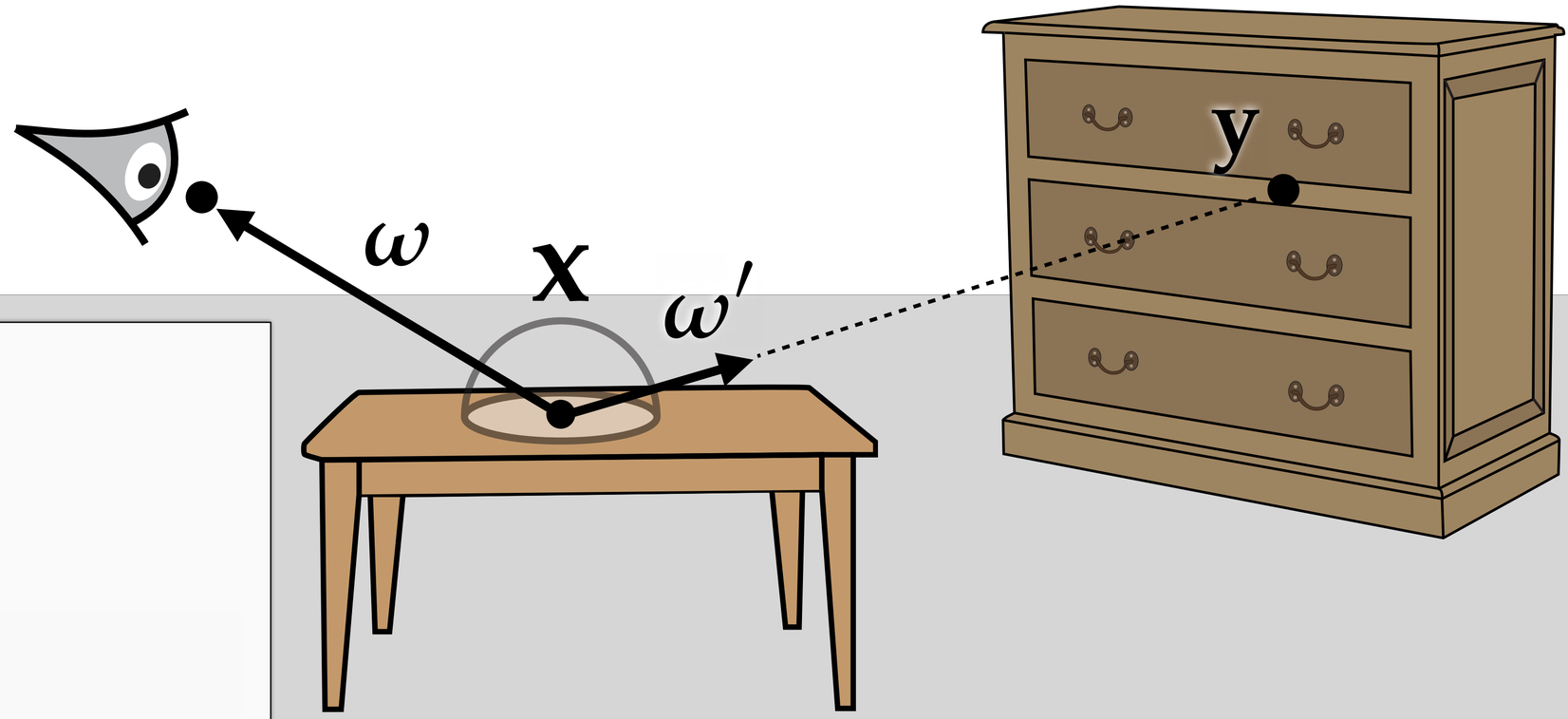
Realistic images via Monte Carlo integration




```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
    y = r(x, ω')
```

$$L_o(\mathbf{x},\boldsymbol{\omega}) = L_e(\mathbf{x},\boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x},\boldsymbol{\omega}'),-\boldsymbol{\omega}')\, M(\mathbf{x},\boldsymbol{\omega},\boldsymbol{\omega}')\, \mathrm{d}\omega$$

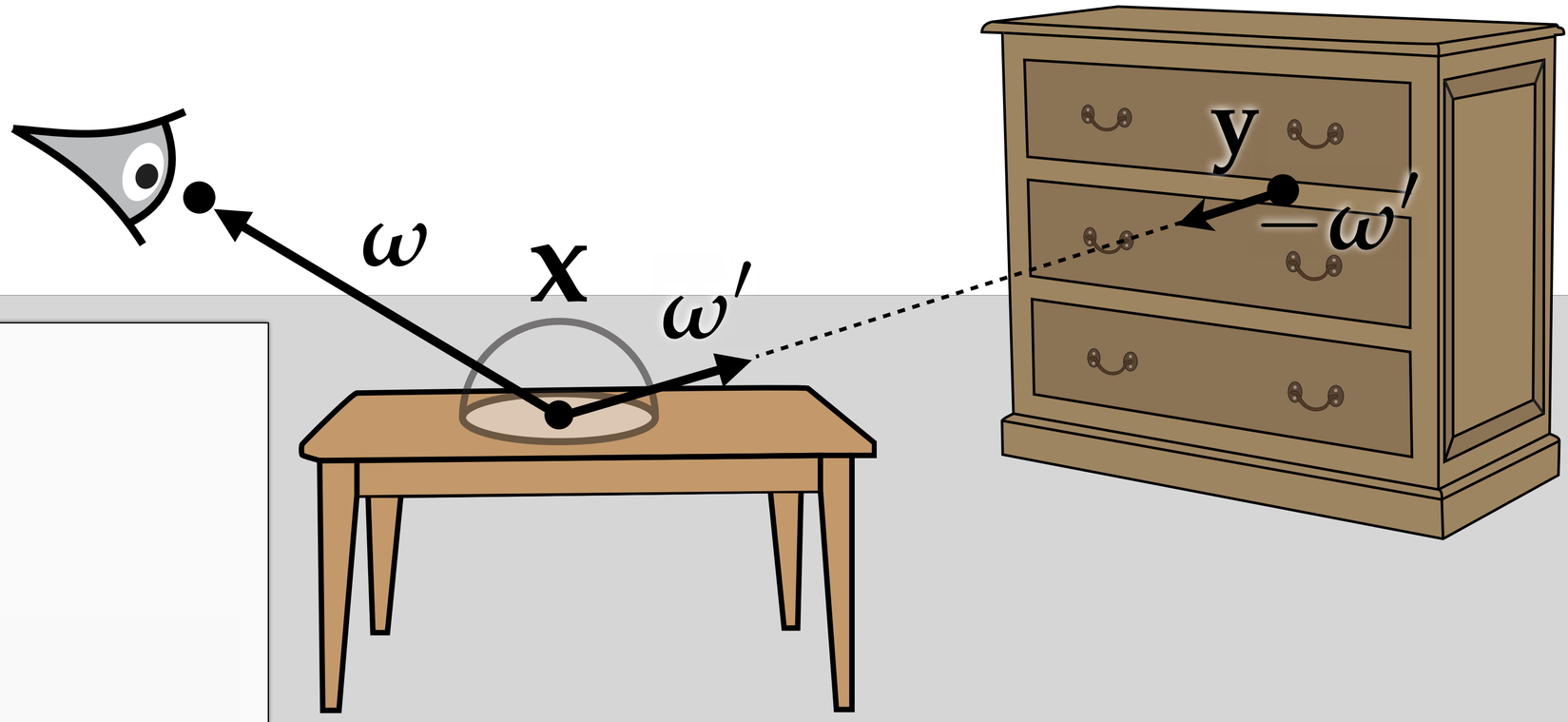
Realistic images via Monte Carlo integration




```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
    y = r(x, ω')
```

$$L_o(\mathbf{x},\omega) = L_e(\mathbf{x},\omega) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x},\omega'), -\omega')\, M(\mathbf{x},\omega,\omega')\, \mathrm{d}\omega$$

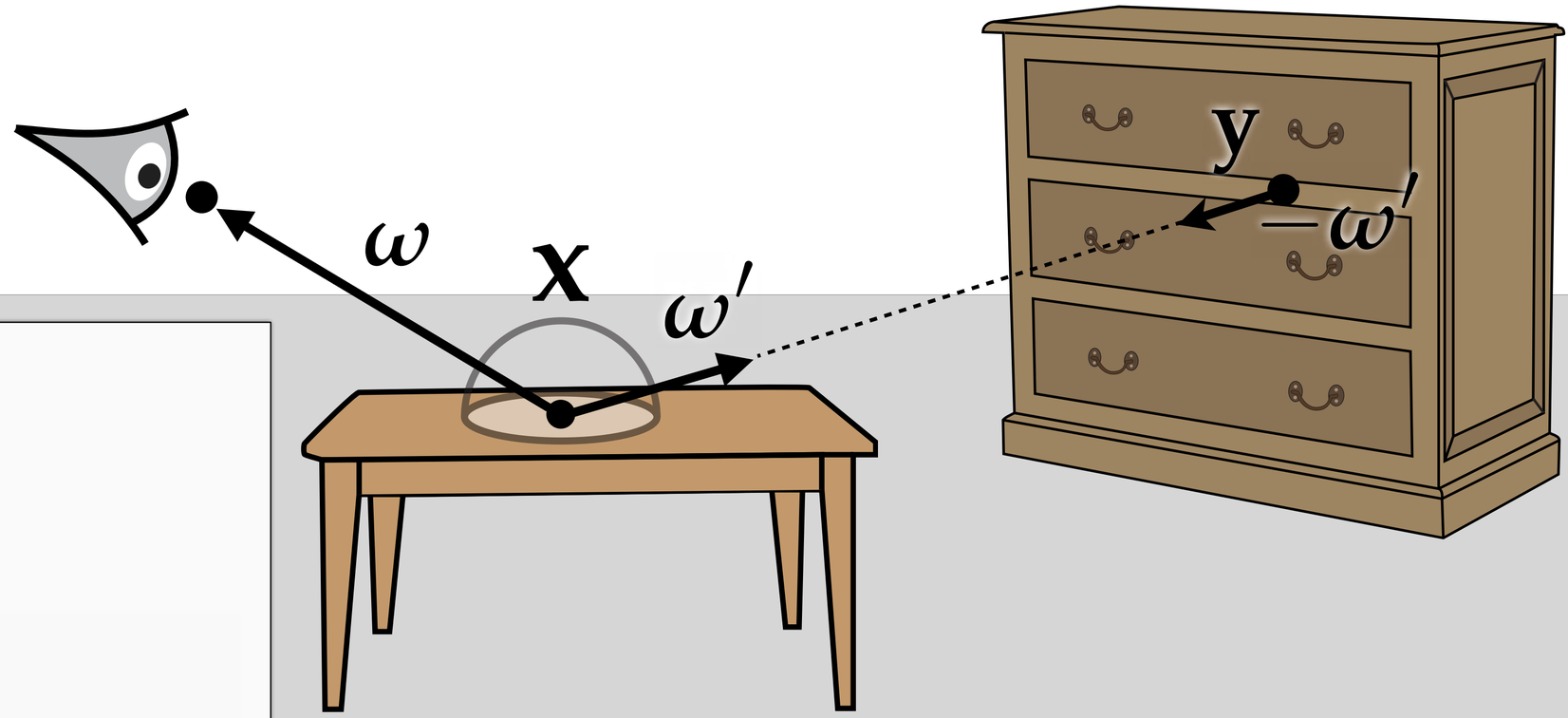
Realistic images via Monte Carlo integration




```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
    y = r(x, ω')
```

$$L_o(\mathbf{x},\omega) = L_e(\mathbf{x},\omega) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x},\omega'),-\omega')\, M(\mathbf{x},\omega,\omega')\, \mathrm{d}\omega$$

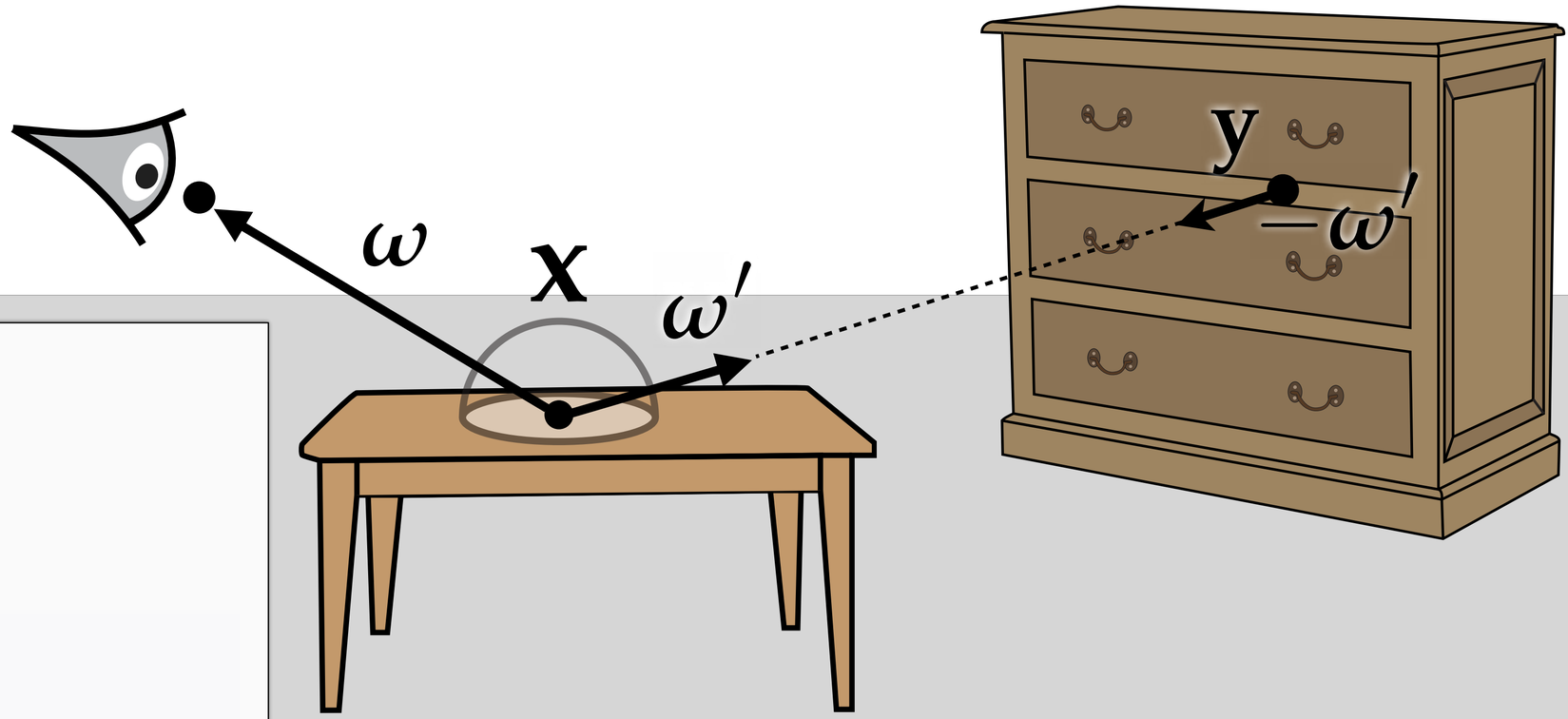
Realistic images via Monte Carlo integration




```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
    y = r(x, ω')
```

$$L_o(\mathbf{x},\boldsymbol{\omega}) = L_e(\mathbf{x},\boldsymbol{\omega}) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x},\boldsymbol{\omega}'),-\boldsymbol{\omega}')\, M(\mathbf{x},\boldsymbol{\omega},\boldsymbol{\omega}')\, \mathrm{d}\omega$$

Realistic images via Monte Carlo integration

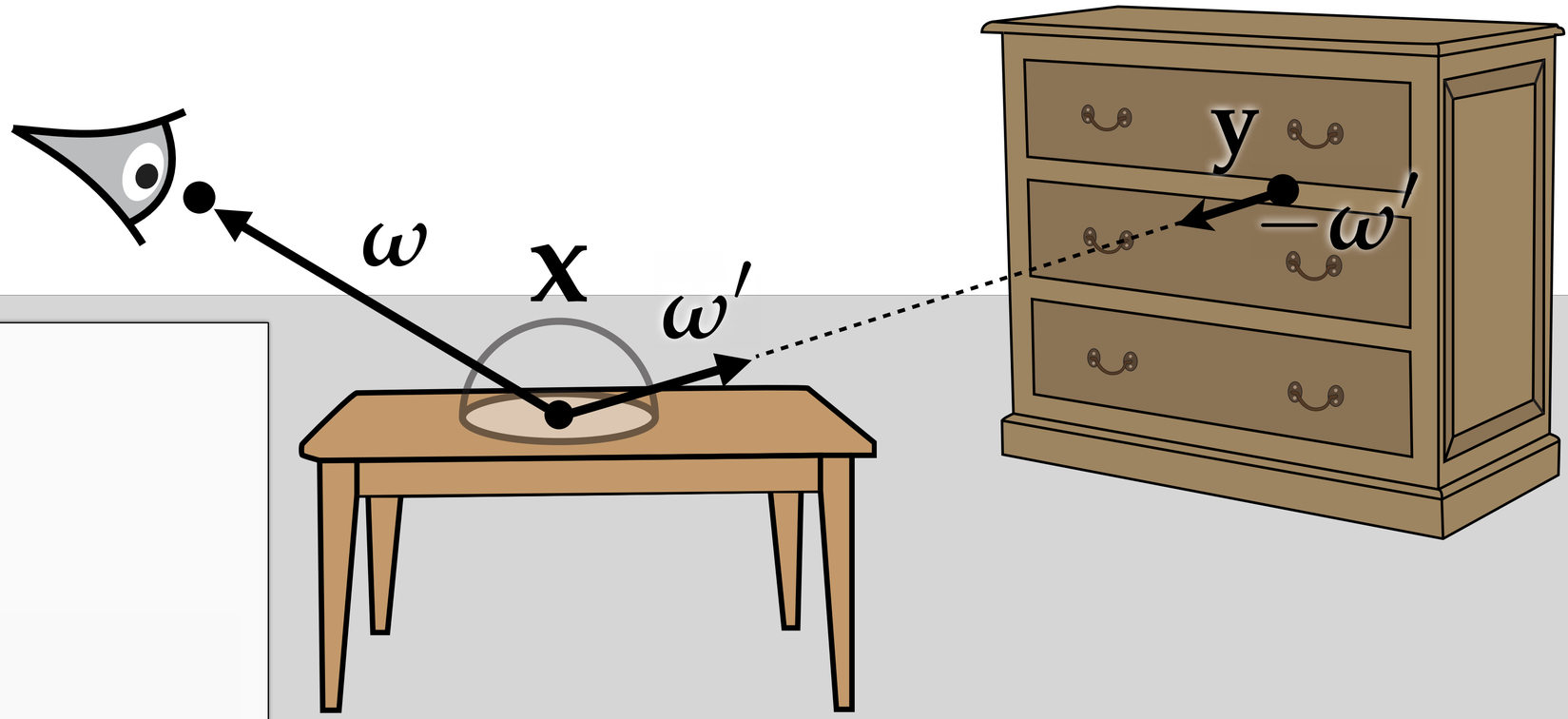



```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
    y = r(x, ω')

    return α*Lo(y, -ω')
```

$$L_o(\mathbf{x},\omega) = L_e(\mathbf{x},\omega) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x},\omega'),-\omega')\, M(\mathbf{x},\omega,\omega')\, \mathrm{d}\omega$$

Realistic images via Monte Carlo integration

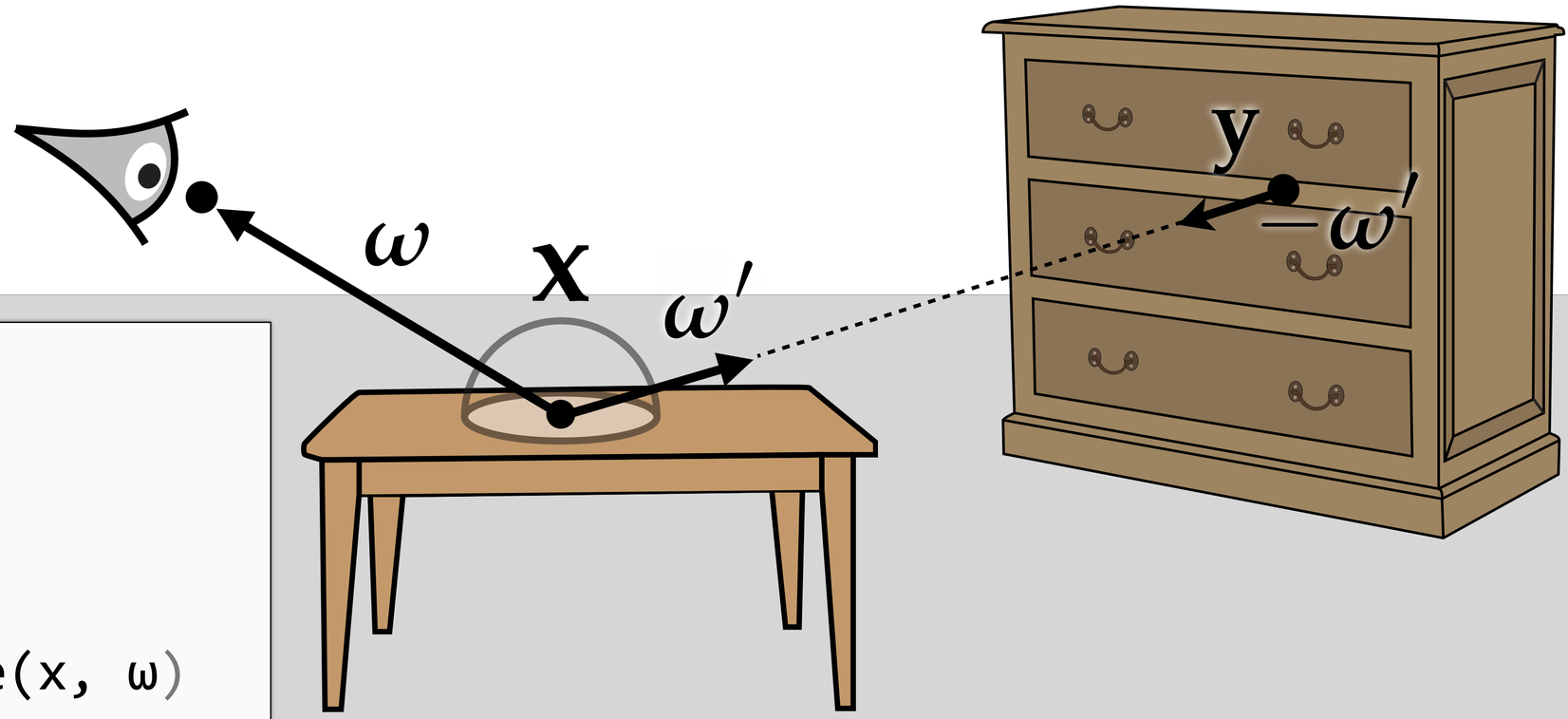



```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
    y = r(x, ω')

    return α*Lo(y, -ω')
```

$$L_o(\mathbf{x},\omega) = L_e(\mathbf{x},\omega) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x},\omega'),-\omega')\, M(\mathbf{x},\omega,\omega')\, \mathrm{d}\omega$$

Realistic images via Monte Carlo integration




```
def Lo(x, ω):
    ω', α = sample_M(x, ω)
    y = r(x, ω')

    return α*Lo(y, -ω') + Le(x, ω)
```

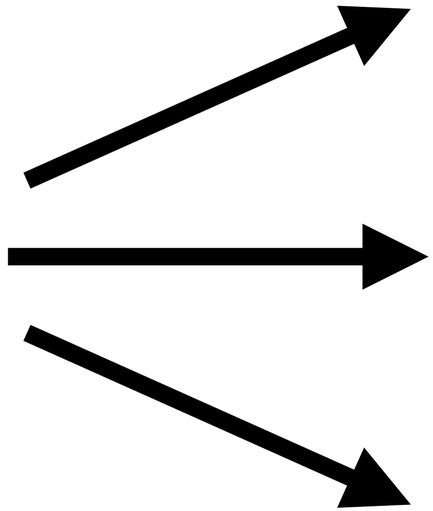
$$L_o(\mathbf{x},\omega) = L_e(\mathbf{x},\omega) + \int_{S^2} L_o(\mathbf{r}(\mathbf{x},\omega'),-\omega')\, M(\mathbf{x},\omega,\omega')\, \mathrm{d}\omega$$

Differentiable rendering

Rendering algorithm

Differentiable rendering

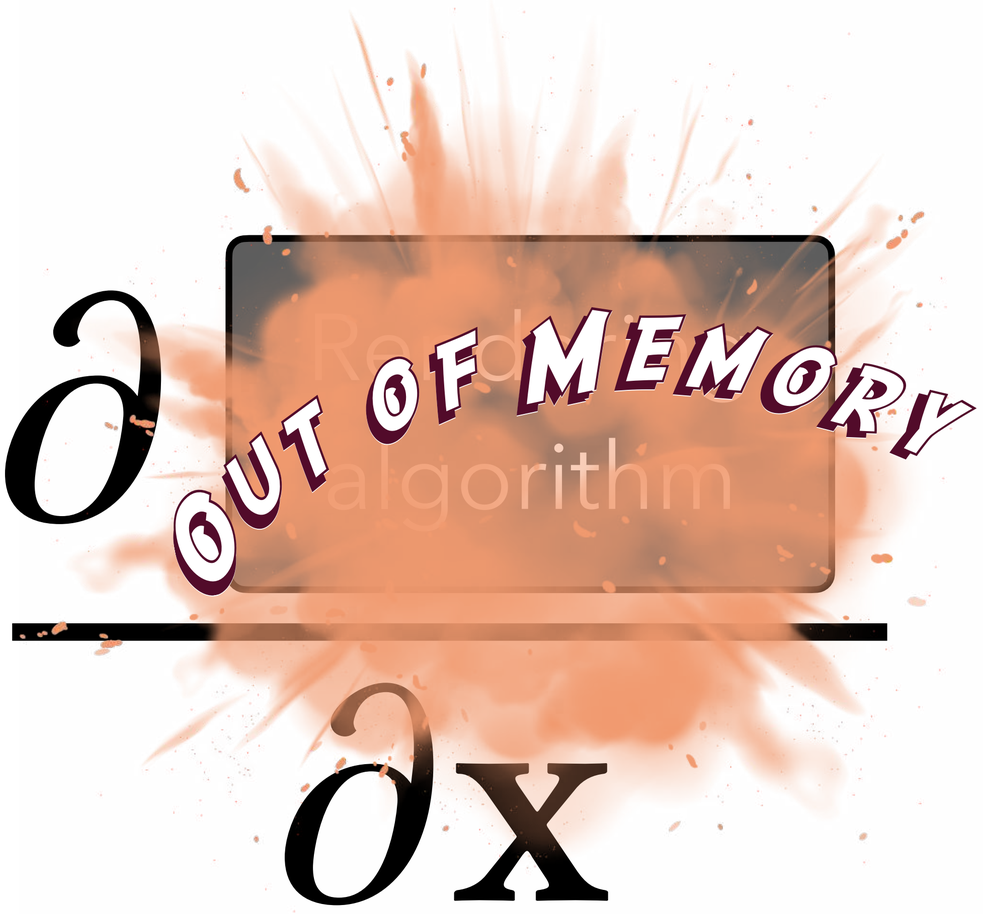
$$\frac{\partial \boxed{\text{Rendering algorithm}}}{\partial \mathbf{x}}$$

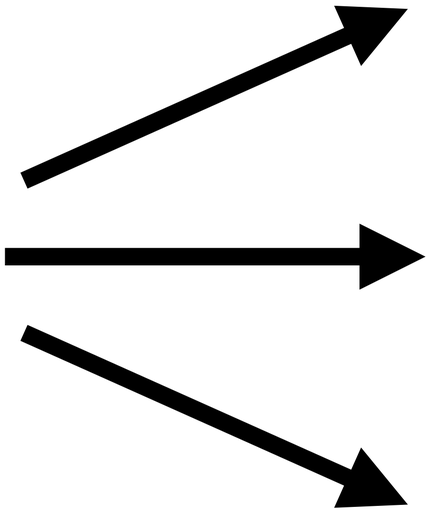


TensorFlow

PyTorch

Differentiable rendering





TensorFlow

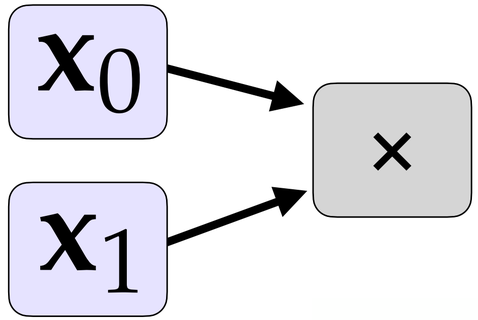
PyTorch

Backpropagation

$$\mathbf{x}_0 \cdot \mathbf{x}_1$$

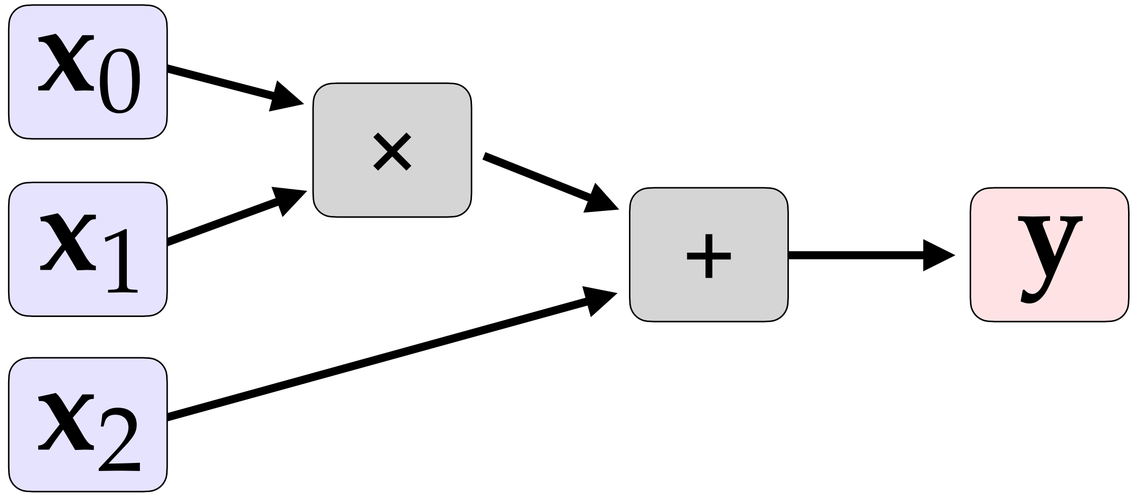
Backpropagation

$$\mathbf{x}_0 \cdot \mathbf{x}_1$$



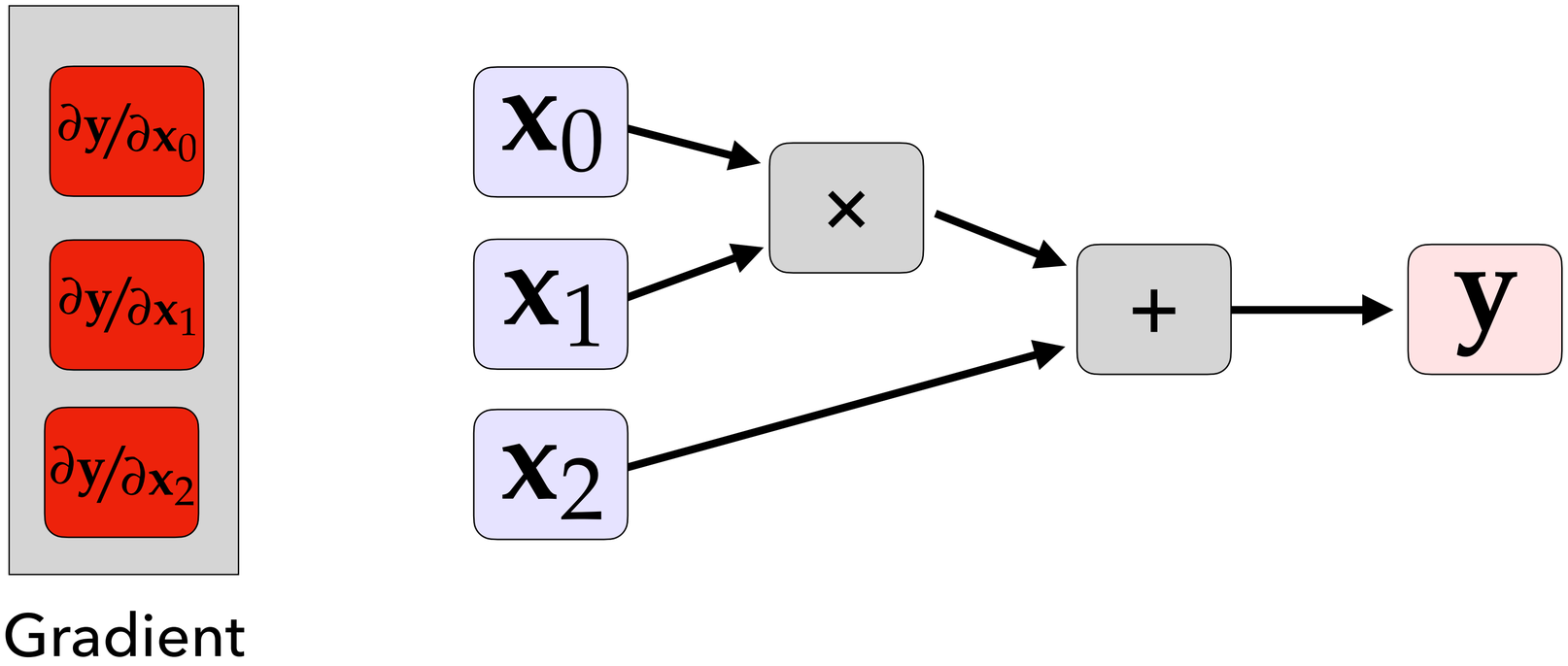
Backpropagation

$$\mathbf{y} = \mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_2$$



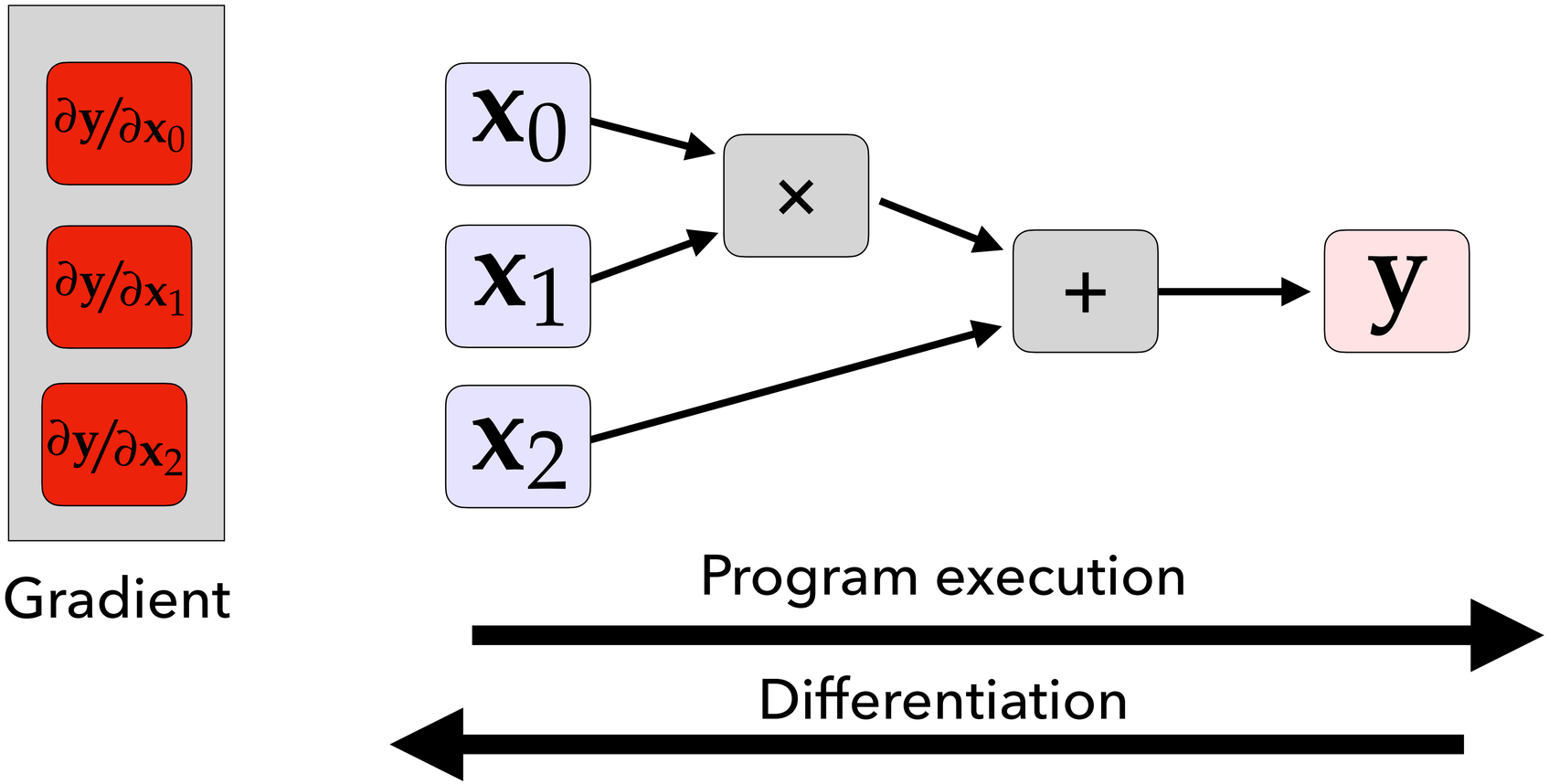
Backpropagation

$$\mathbf{y} = \mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_2$$

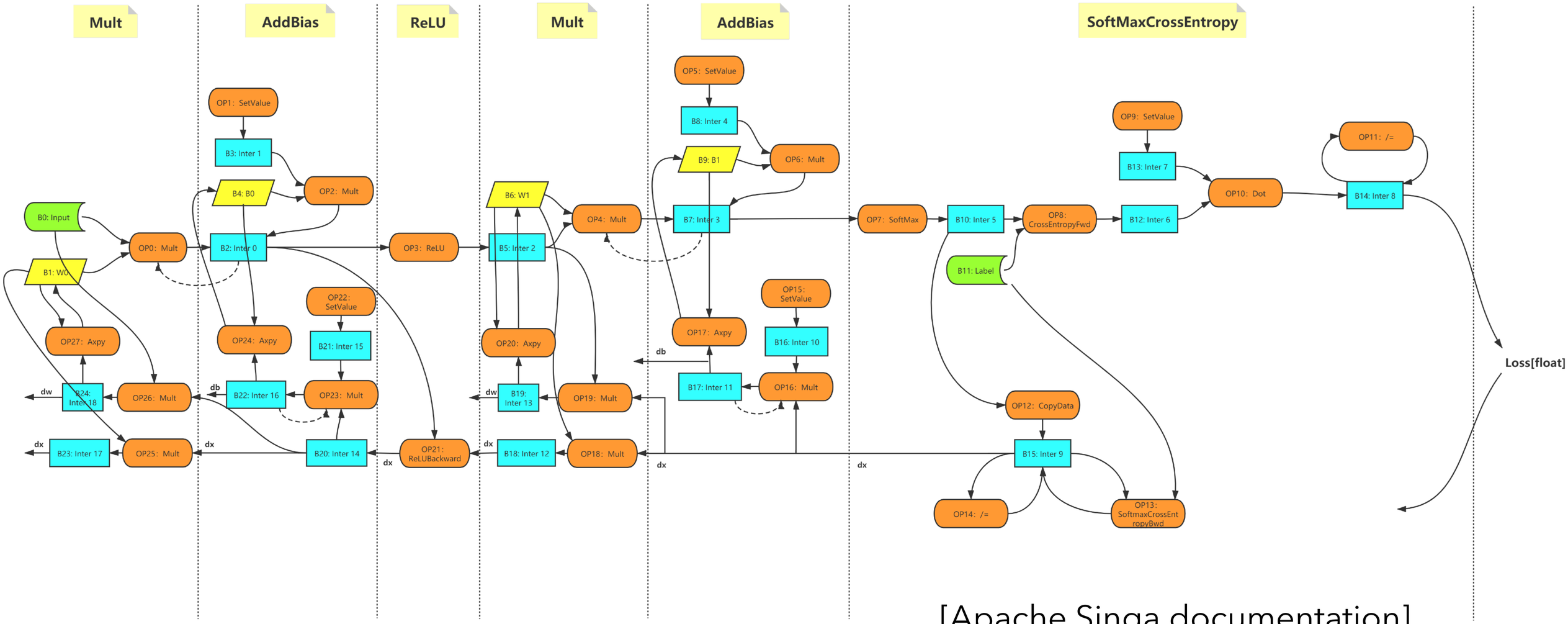


Backpropagation

$$y = x_0 \cdot x_1 + x_2$$

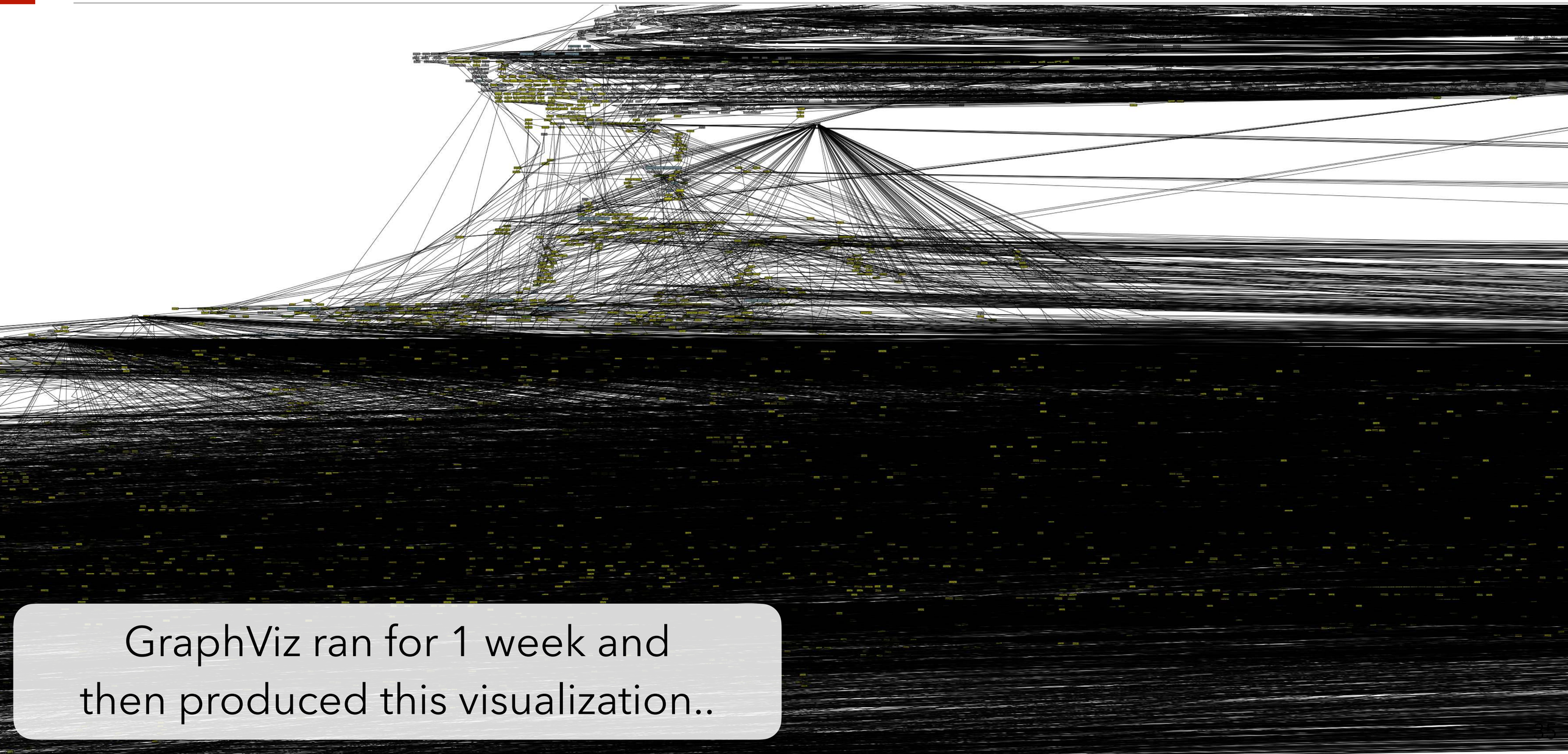


Computation graph of a neural network



[Apache Singa documentation]

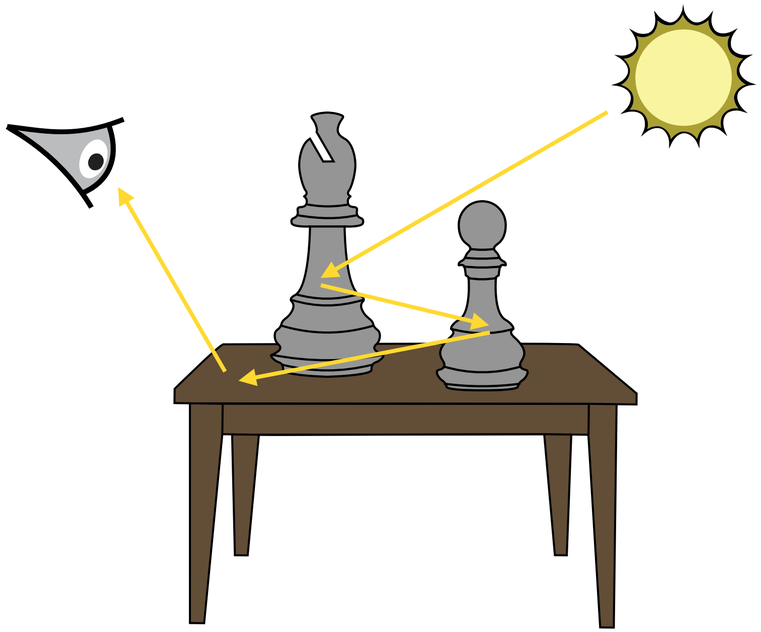
Computation graph of a simple (< 1 sec) rendering job



GraphViz ran for 1 week and
then produced this visualization..

Observations

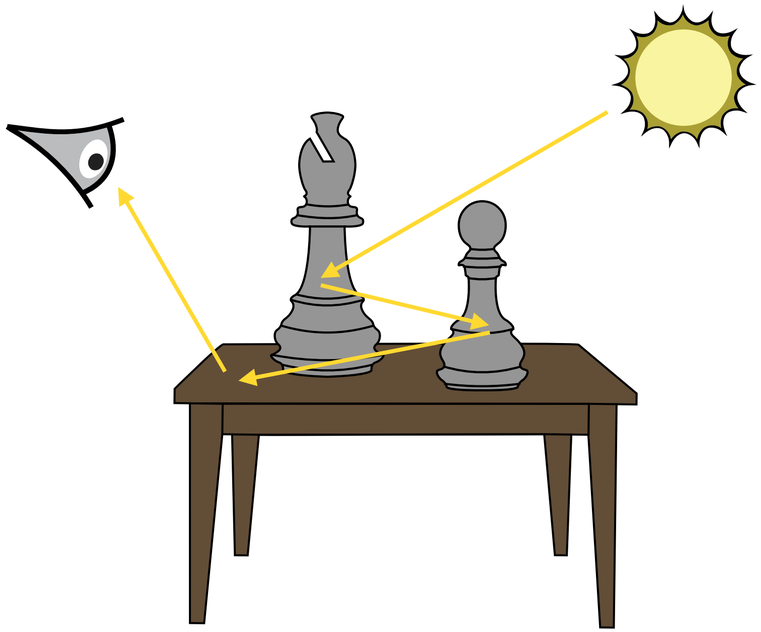
- Reverse-mode differentiation needs to run the algorithm "in reverse"



Observations

- Reverse-mode differentiation needs to run the algorithm "in reverse"

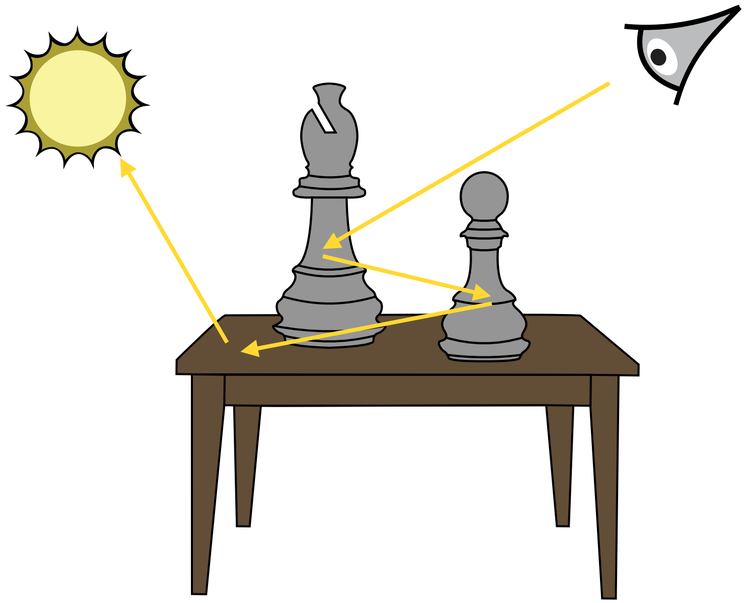
- Light satisfies a physical property called **Helmholtz reciprocity** that is very related!



Observations

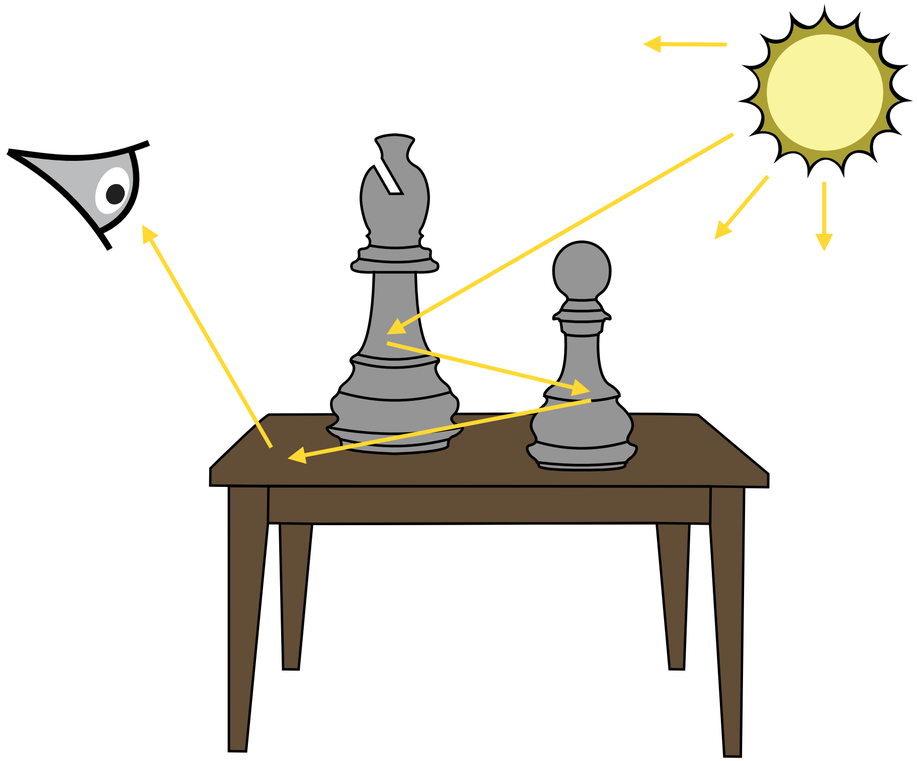
- Reverse-mode differentiation needs to run the algorithm "in reverse"

- Light satisfies a physical property called **Helmholtz reciprocity** that is very related!

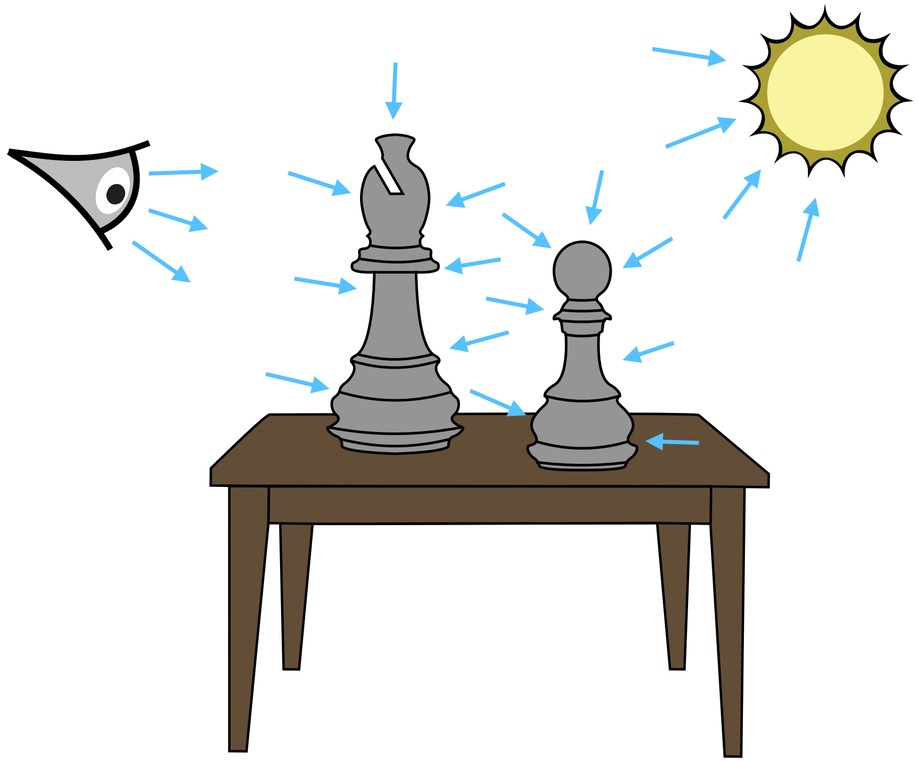


Differentiation as a physical process

Phase 1: simulate light



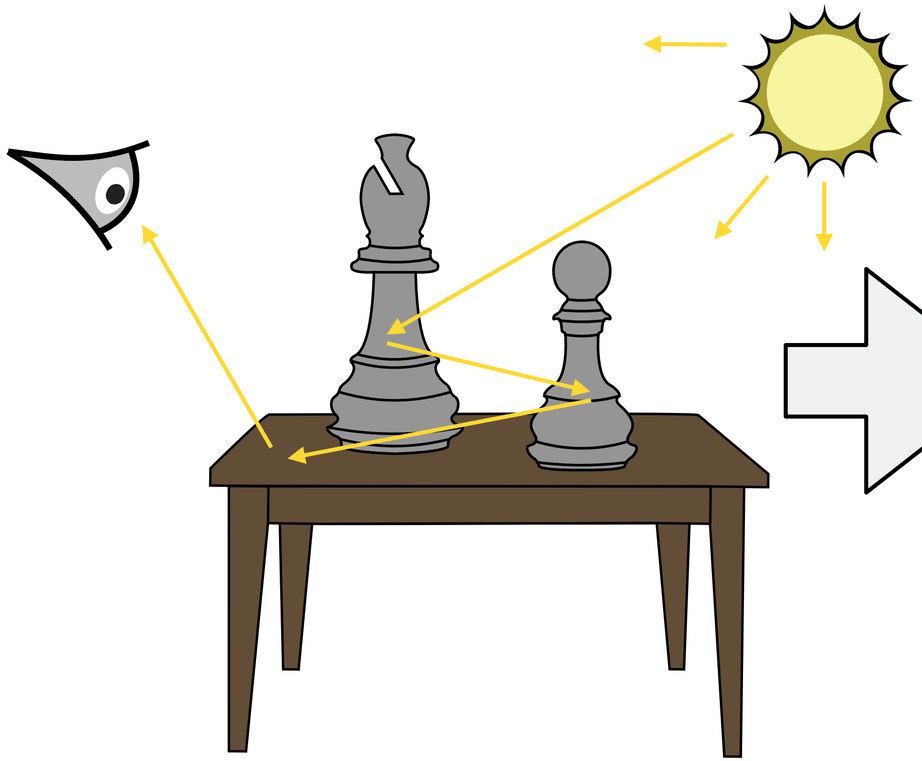
Phase 2: simulate derivative of light

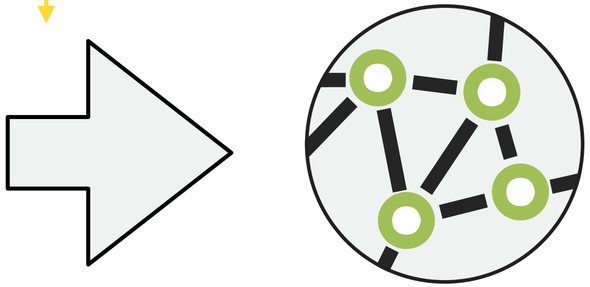


Chess pieces by Tizian Zeltner

Differentiation as a physical process

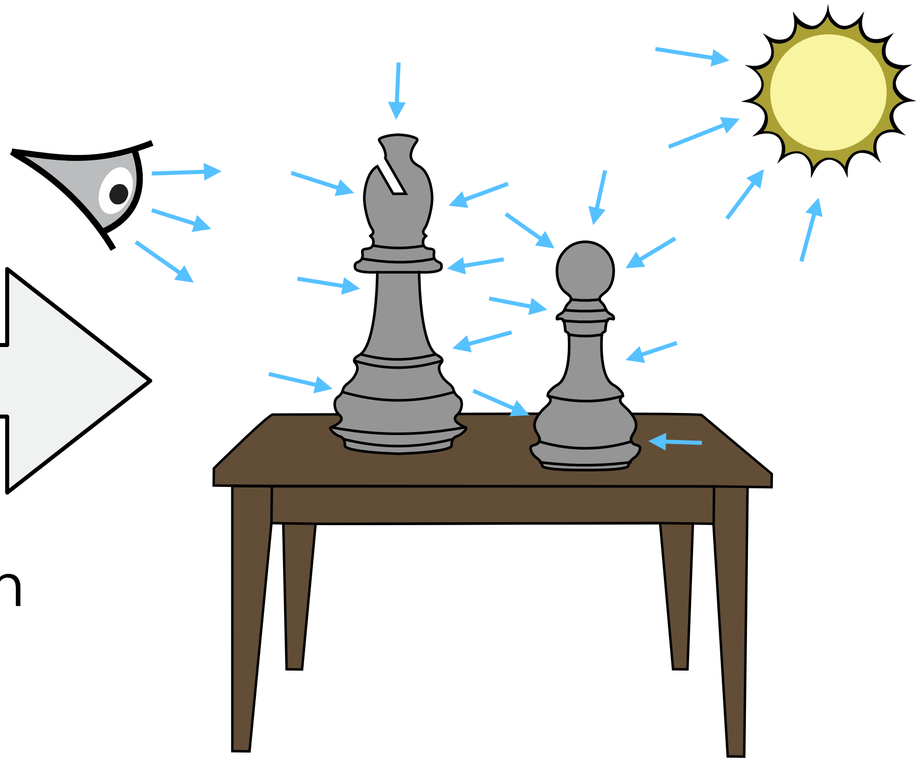
Phase 1: simulate light





Computation graph

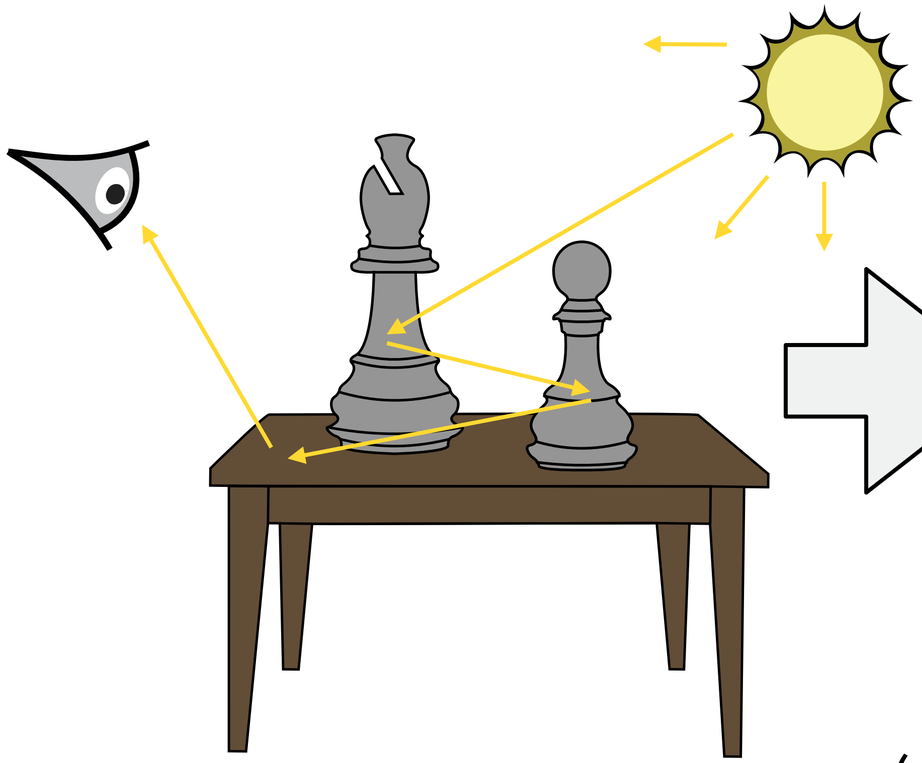
Phase 2: simulate derivative of light



Chess pieces by Tizian Zeltner

Differentiation as a physical process

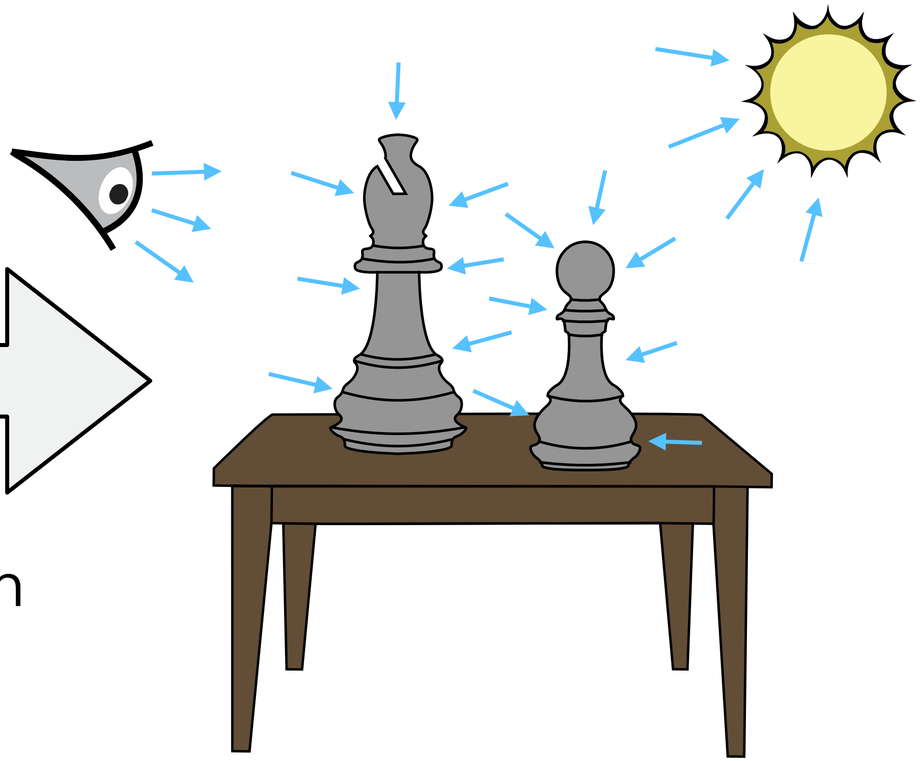
Phase 1: simulate light



Computation graph

Huge speedup
(we observed factors approaching ~1000 ×)

Phase 2: simulate derivative of light

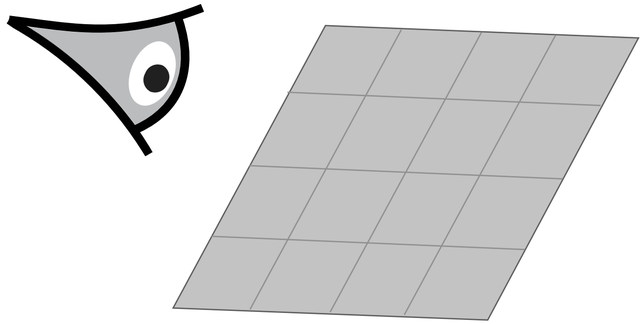


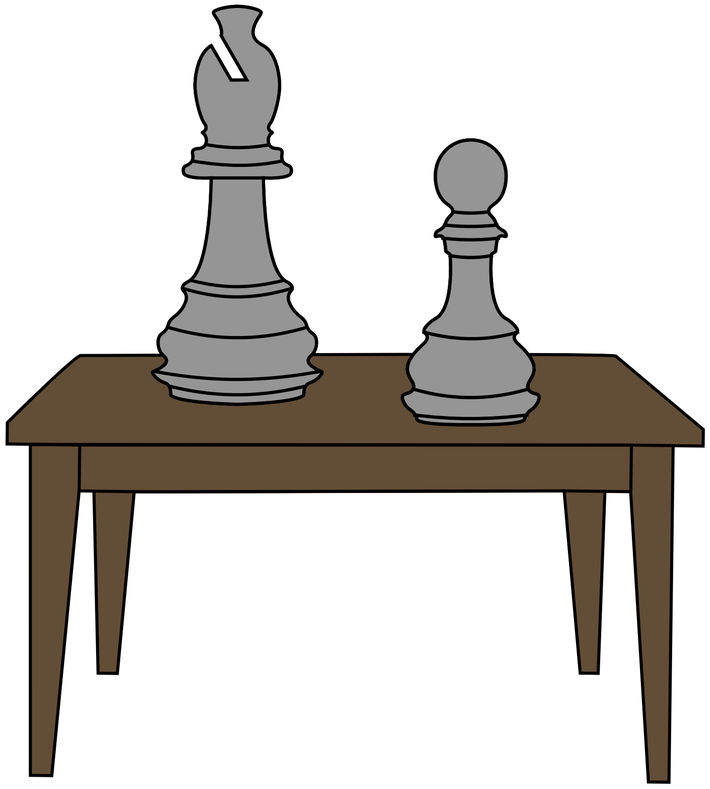
Chess pieces by Tizian Zeltner

Radiative backpropagation

Current state

Target image





Chess pieces by Tizian Zeltner

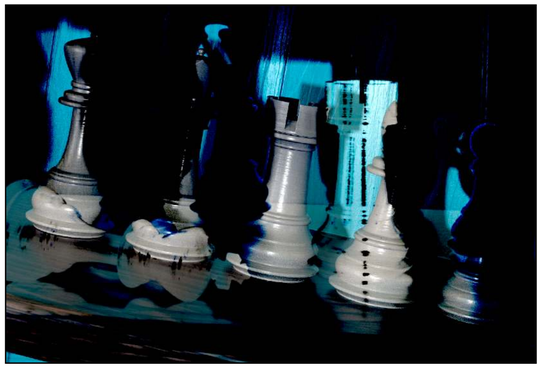
Radiative backpropagation

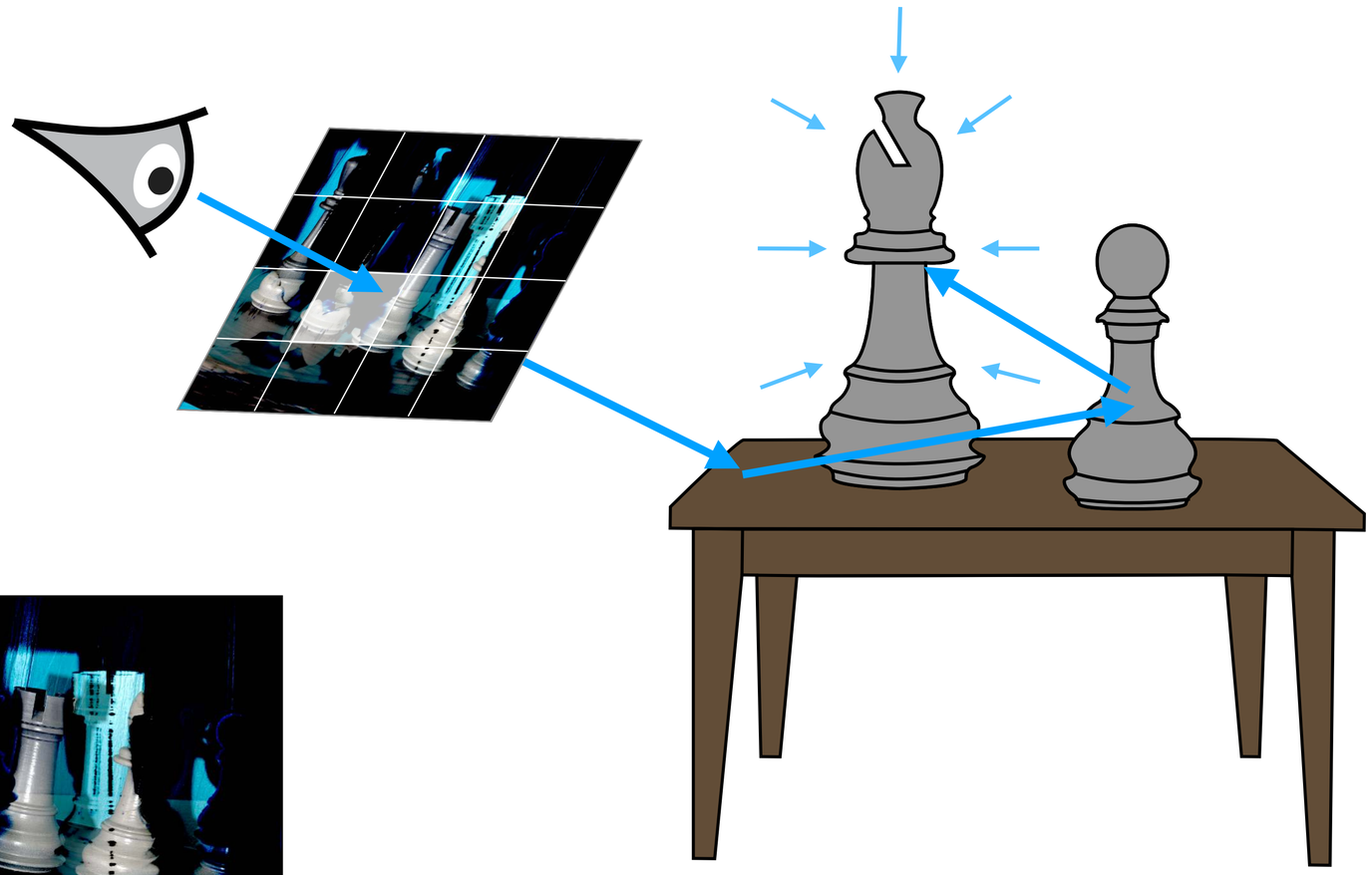
Current state

Target image

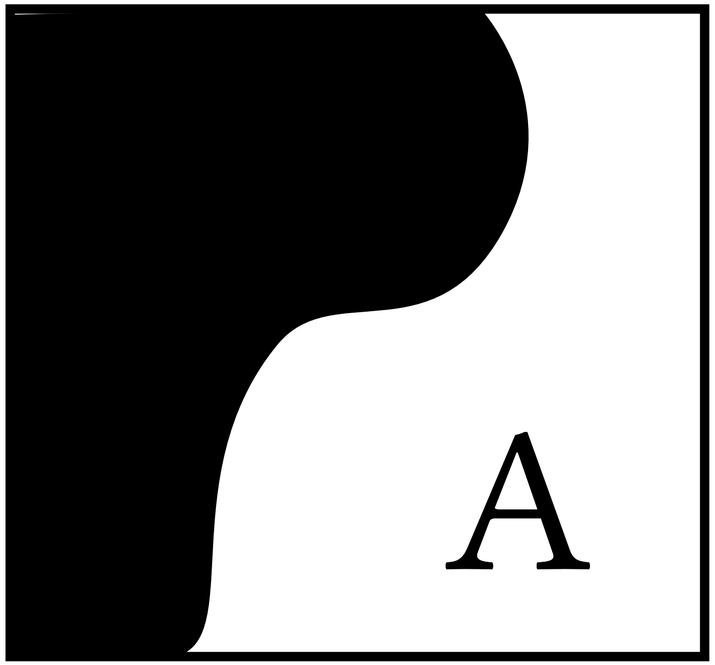
Loss function $g(\cdot)$

$$\frac{\partial g}{\partial I}$$

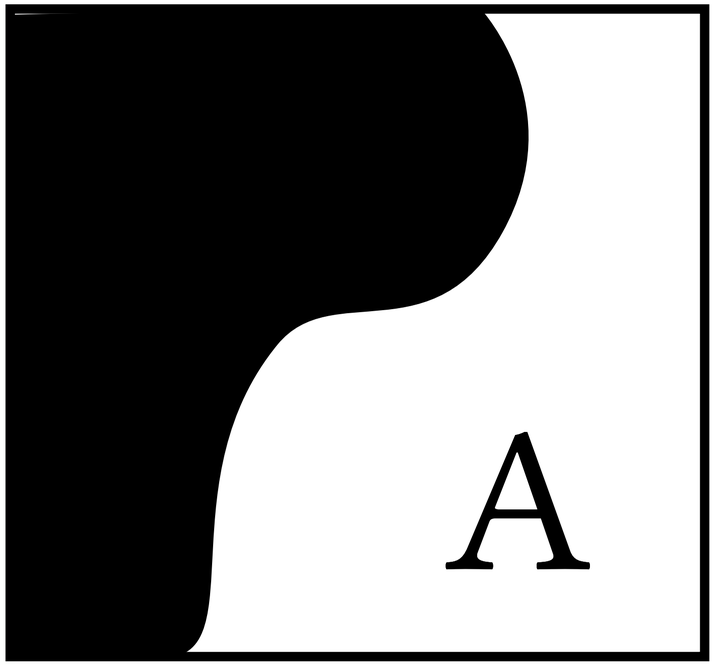




Discontinuous integrals

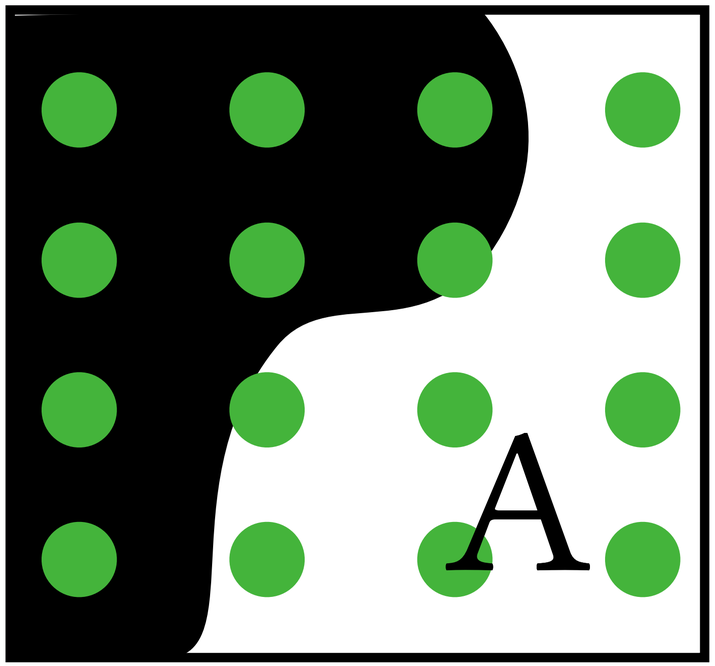


Discontinuous integrals



$$I = \int_{\mathrm{A}} f(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

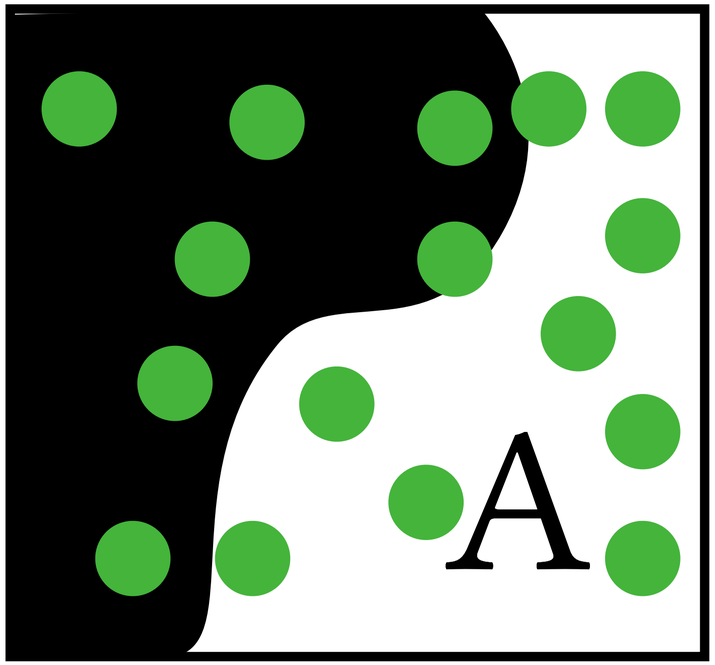
Discontinuous integrals



$$I = \int_{\mathrm{A}} f(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

Integration via: **Quadrature**

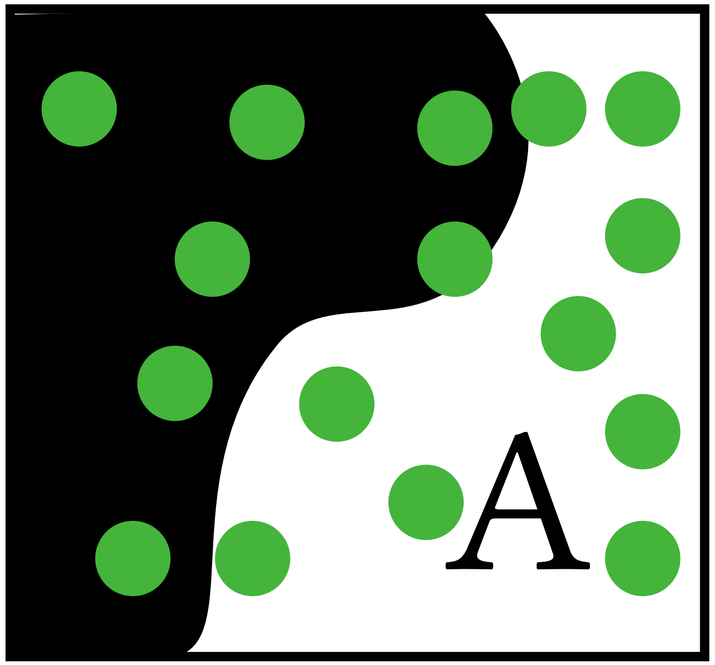
Discontinuous integrals



$$I = \int_{\mathrm{A}} f(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

Integration via: **Monte Carlo**

Discontinuous integrals



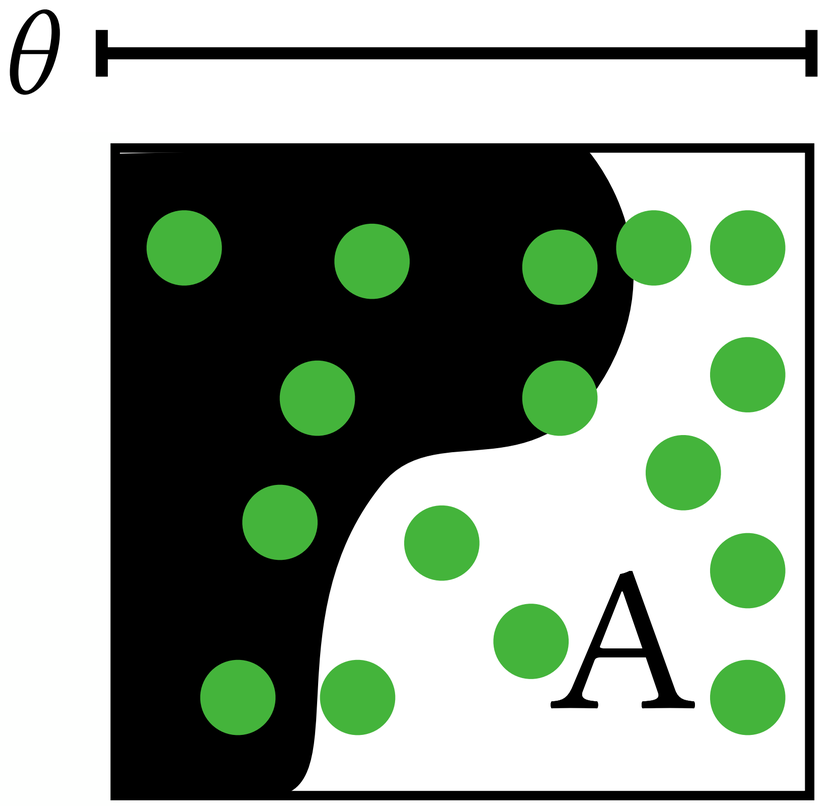
Integration via: **Monte Carlo**

$$I = \int_{\mathrm{A}} f(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

$$\downarrow$$

$$I(\theta) = \int_{\mathrm{A}} f(\mathbf{x}, \theta)\, \mathrm{d}\mathbf{x}$$

Discontinuous integrals



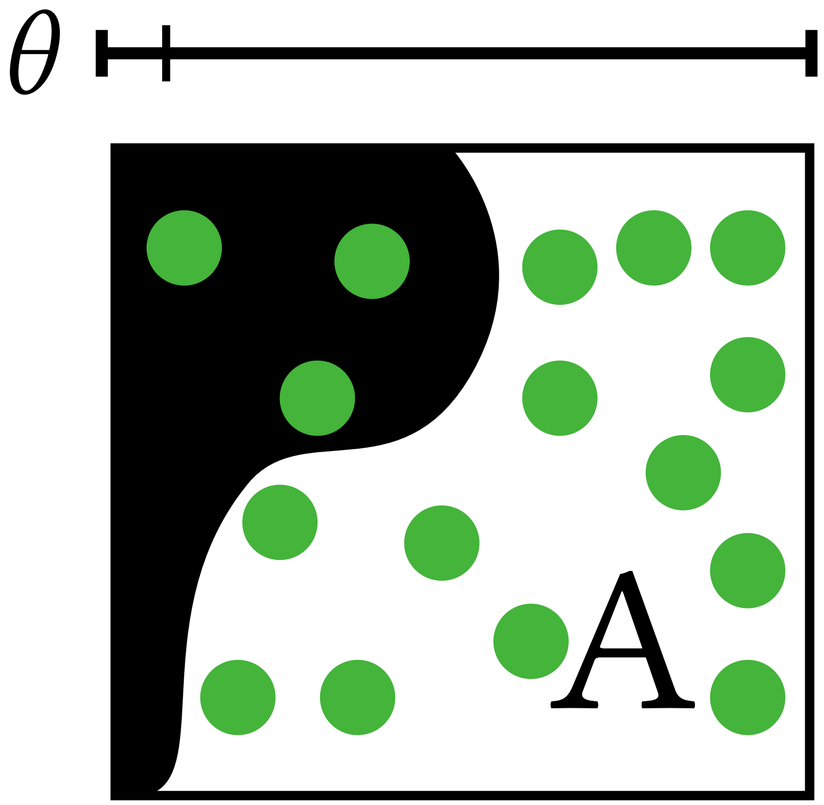
Integration via: **Monte Carlo**

$$I = \int_{\mathrm{A}} f(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

$$\downarrow$$

$$I(\theta) = \int_{\mathrm{A}} f(\mathbf{x}, \theta)\, \mathrm{d}\mathbf{x}$$

Discontinuous integrals

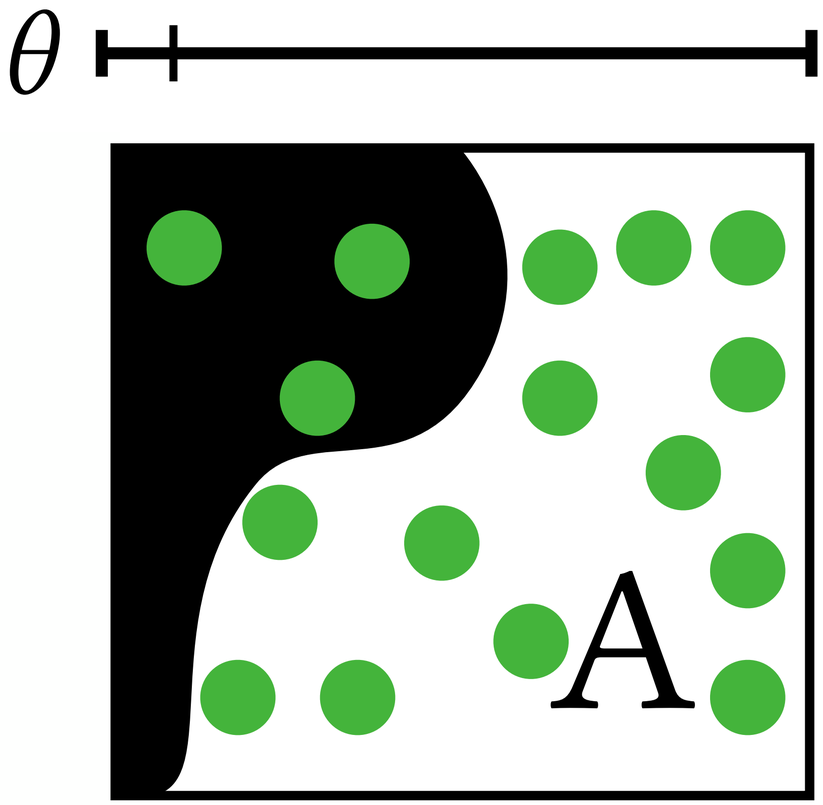


Integration via: **Monte Carlo**

$$I = \int_{\mathrm{A}} f(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

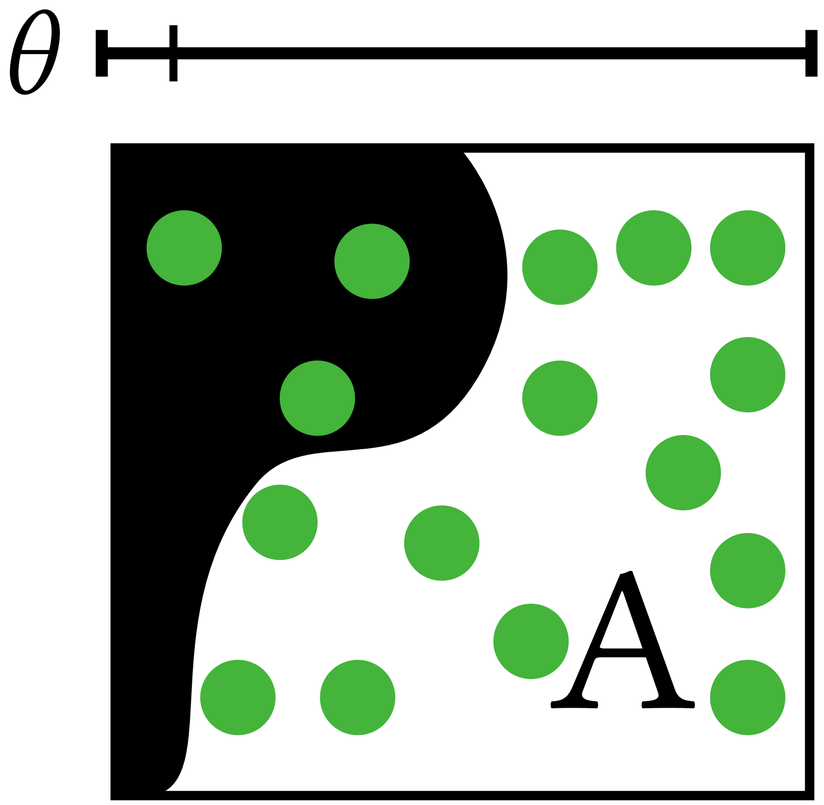
$$I(\theta) = \int_{\mathrm{A}} f(\mathbf{x}, \theta)\, \mathrm{d}\mathbf{x}$$

Discontinuous integrals



$$I(\theta) = \int_{\mathrm{A}} f(\mathbf{x}, \theta)\, \mathrm{d}\mathbf{x}$$

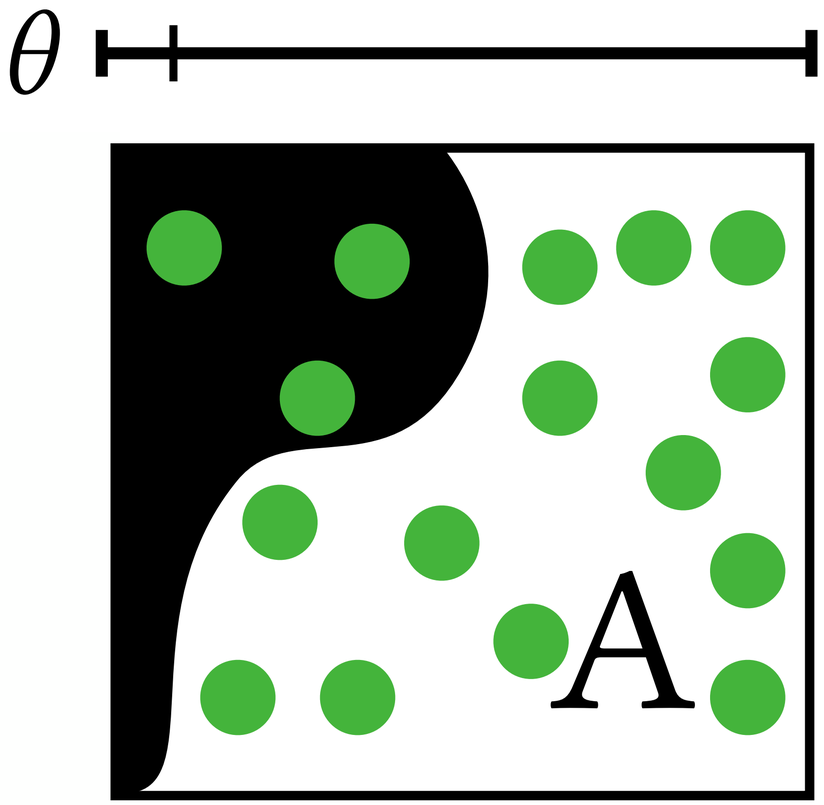
Discontinuous integrals



$$I(\theta) = \int_{\mathrm{A}} f(\mathbf{x}, \theta)\, \mathrm{d}\mathbf{x}$$

$$\frac{\partial}{\partial \theta} I(\theta)$$

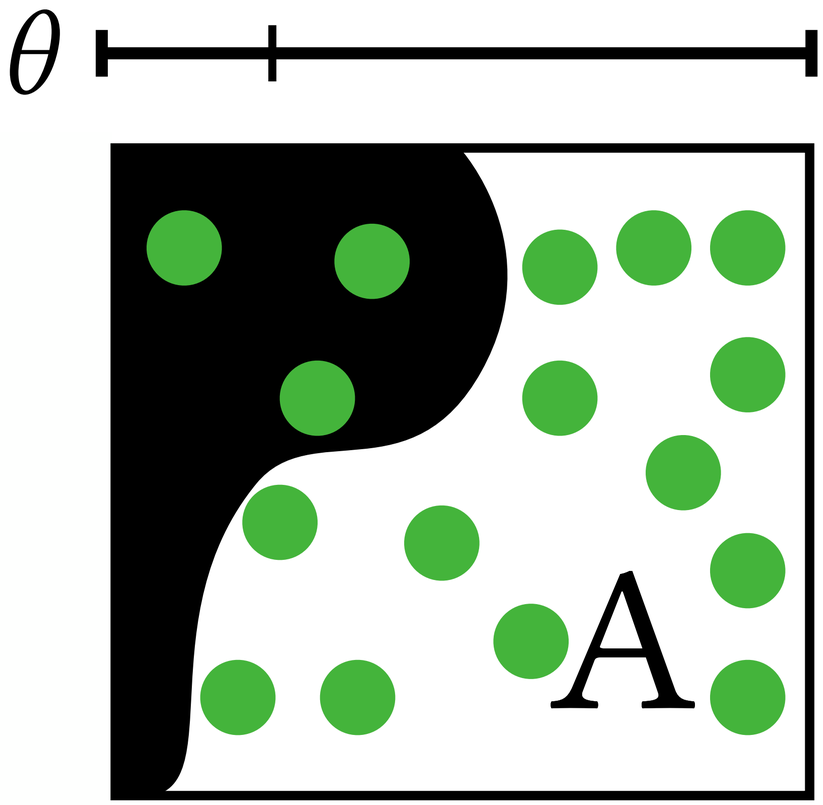
Discontinuous integrals

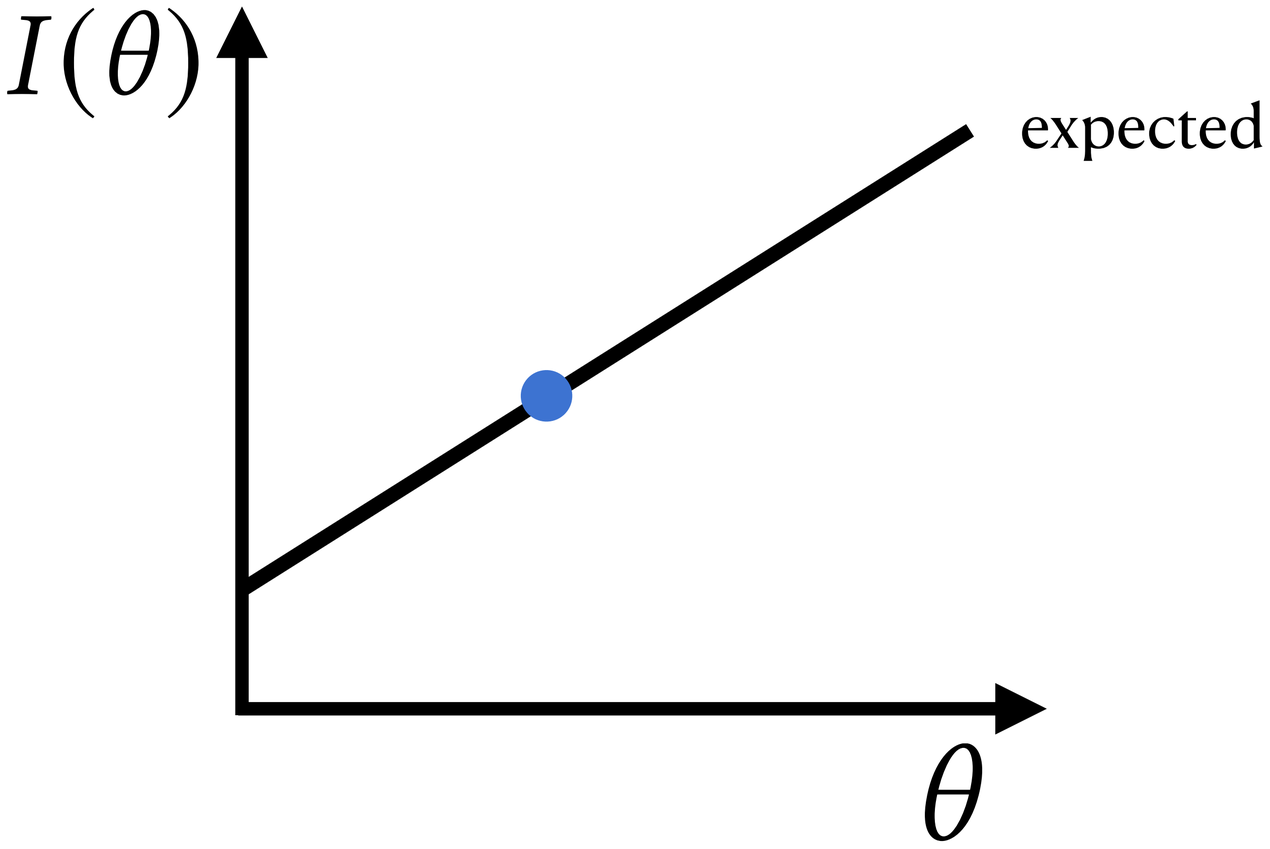


$$I(\theta) = \int_{\mathrm{A}} f(\mathbf{x}, \theta)\, \mathrm{d}\mathbf{x}$$

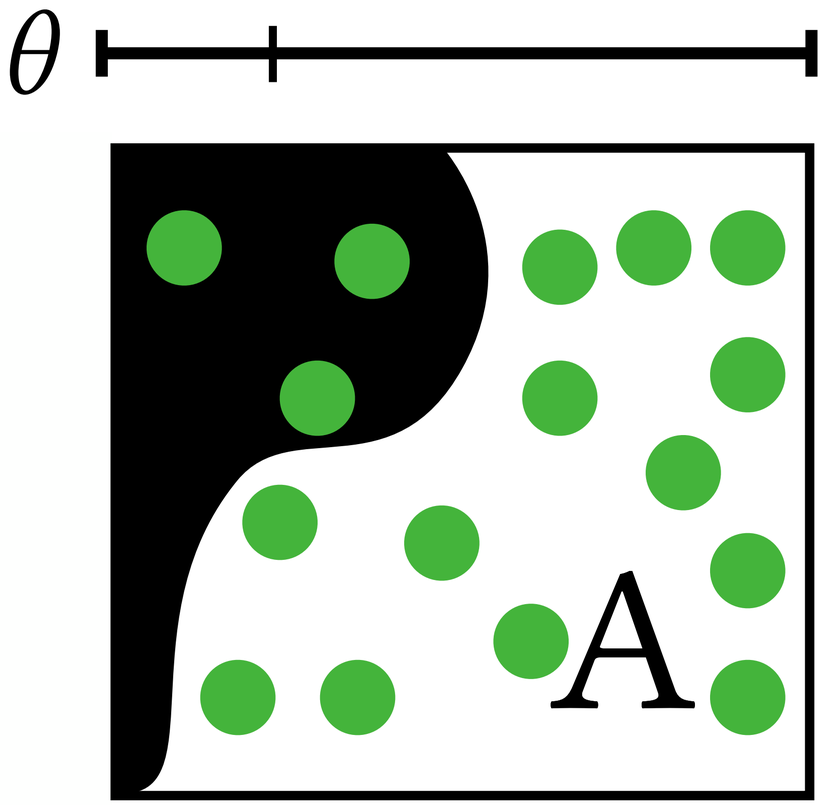
$$\frac{\partial}{\partial \theta} I(\theta)$$

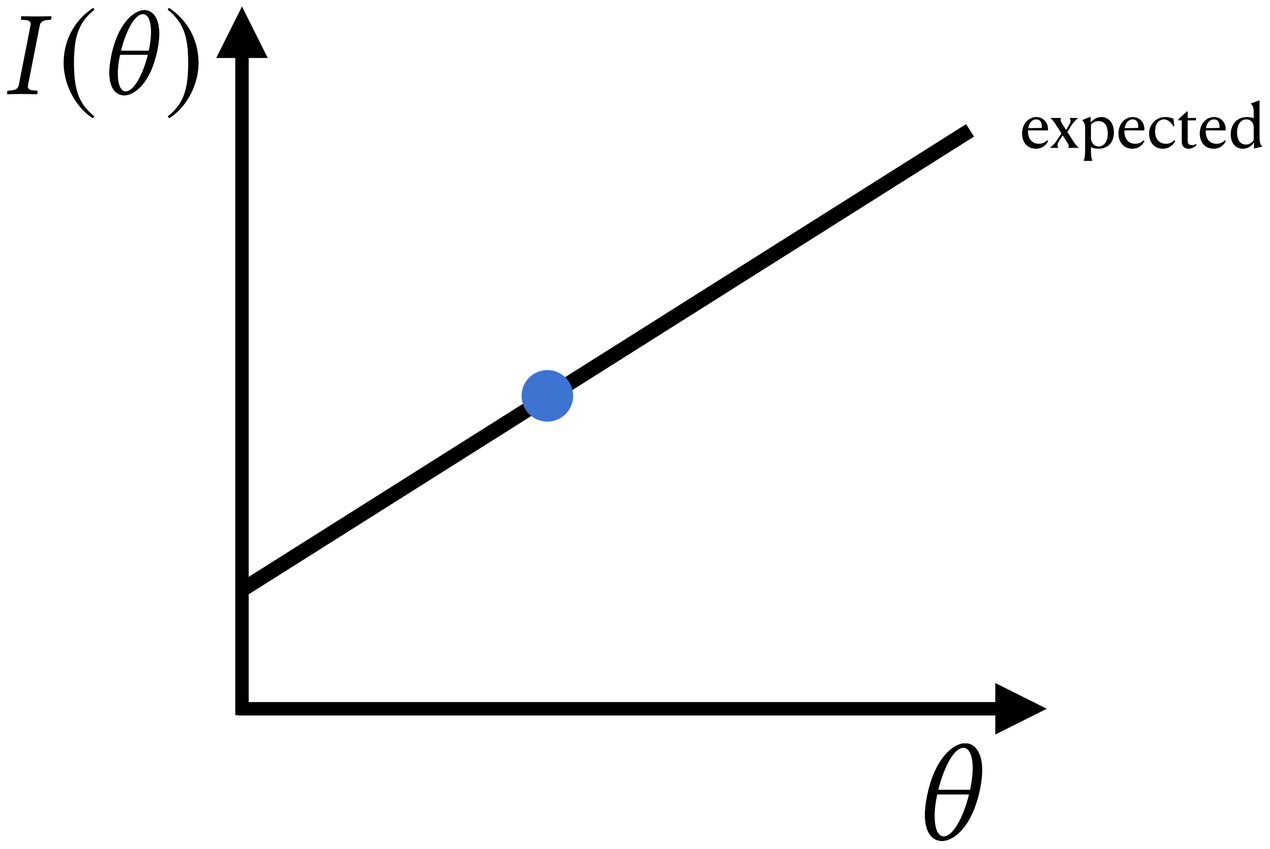
Discontinuous integrals



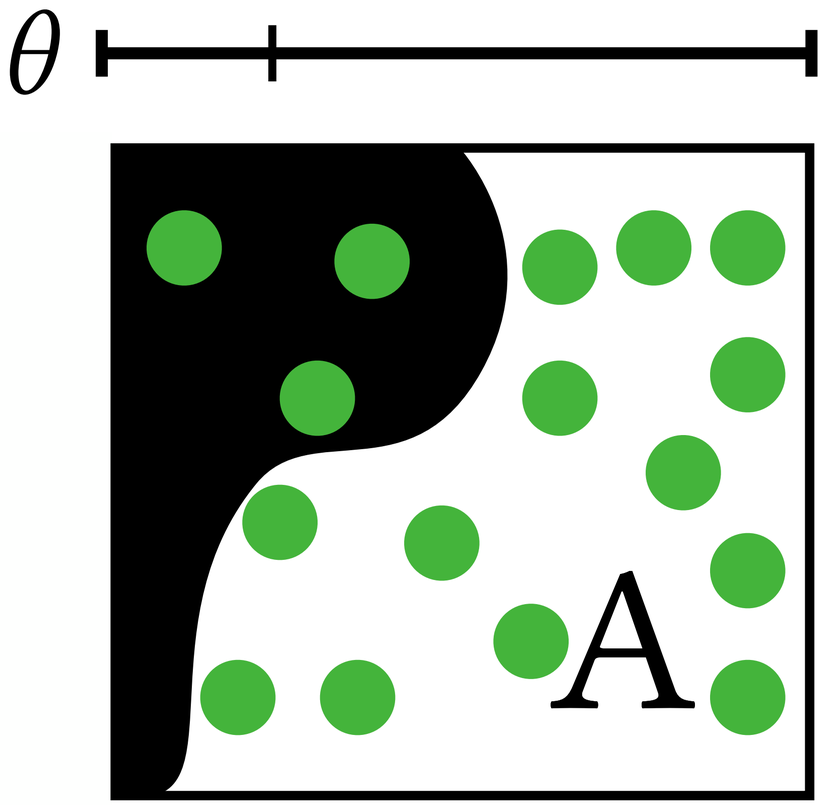


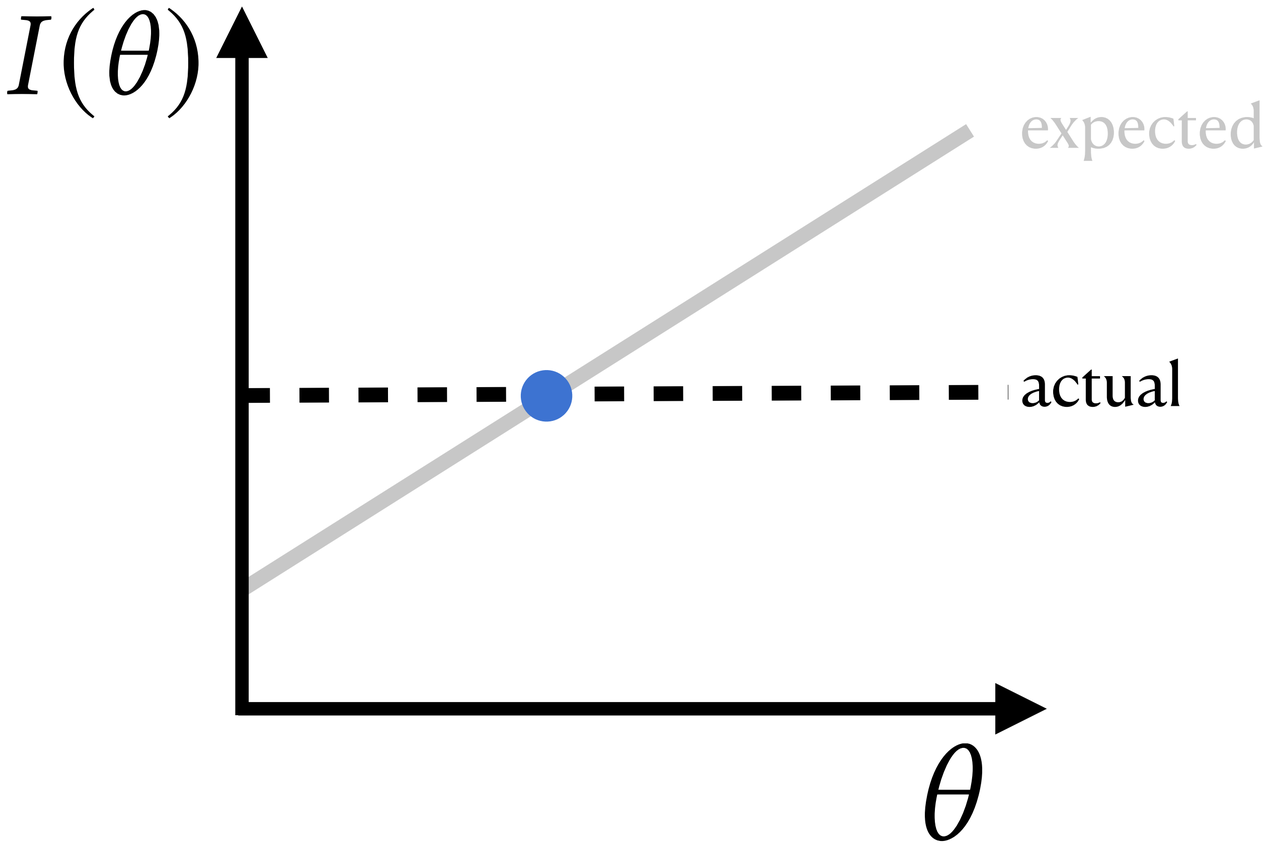
Discontinuous integrals



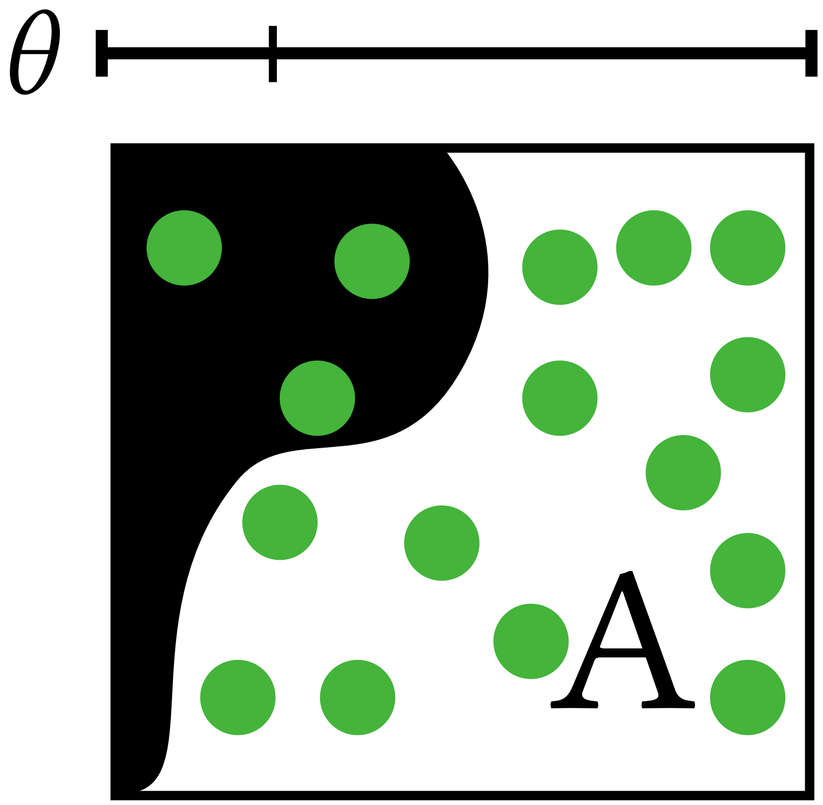


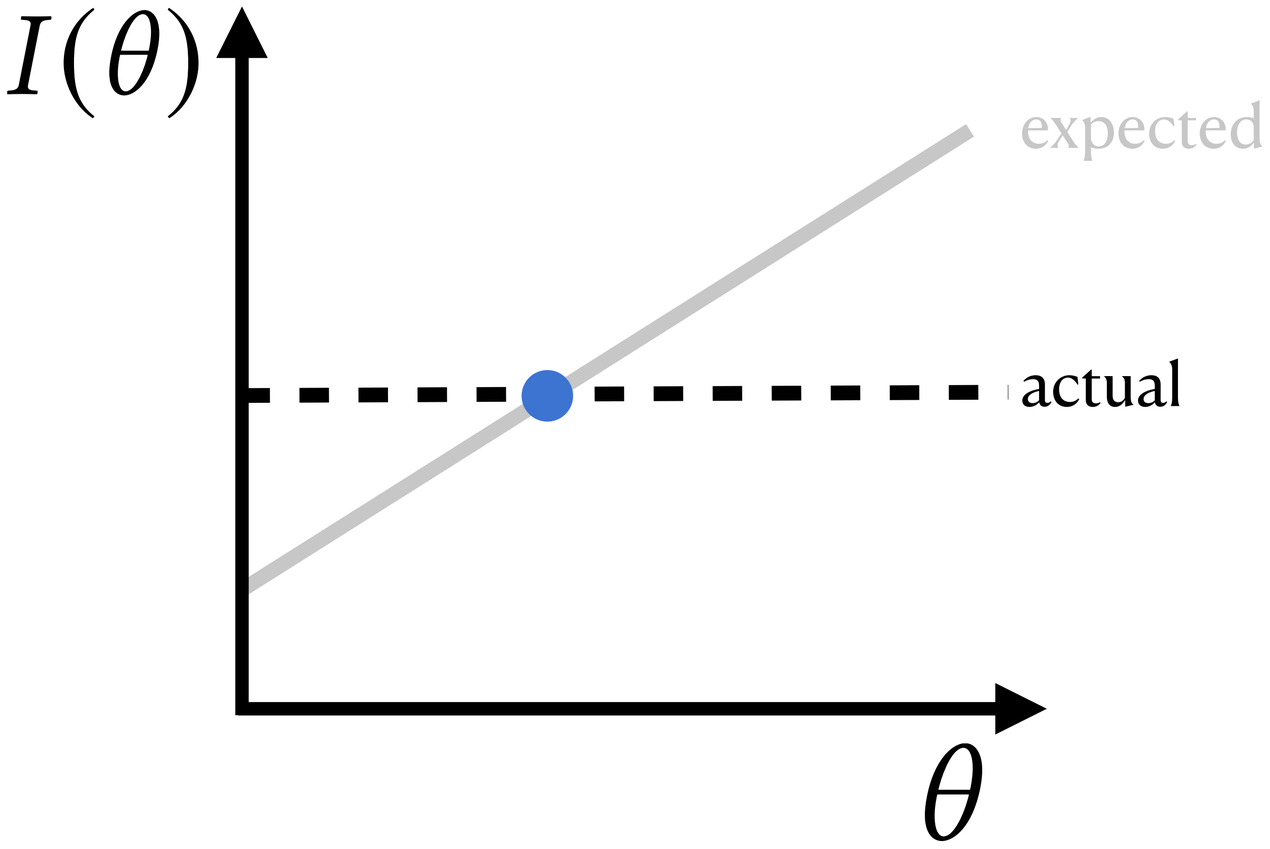
Discontinuous integrals





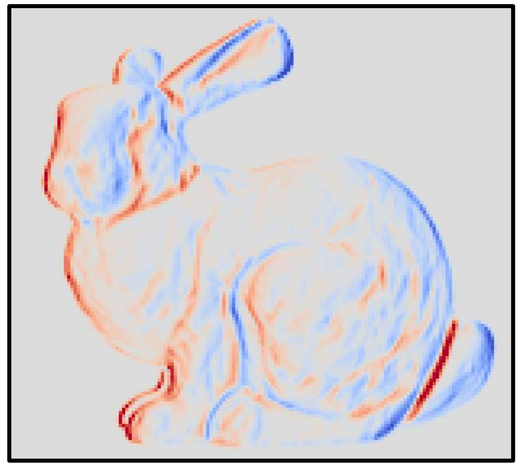
Discontinuous integrals



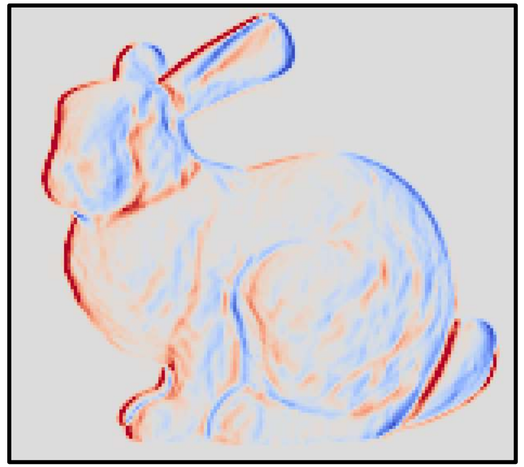


How important is this, really?

Naïve



Ours

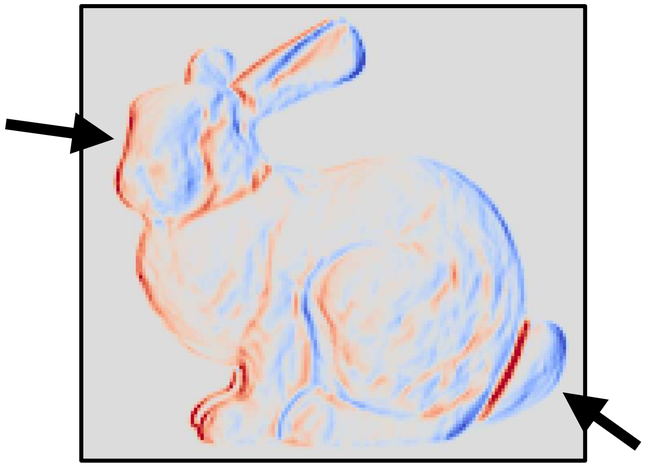


Gradients

From: *Differentiable Signed Distance Rendering* [Vicini, Speierer, and Jakob 2022]

How important is this, really?

Naïve



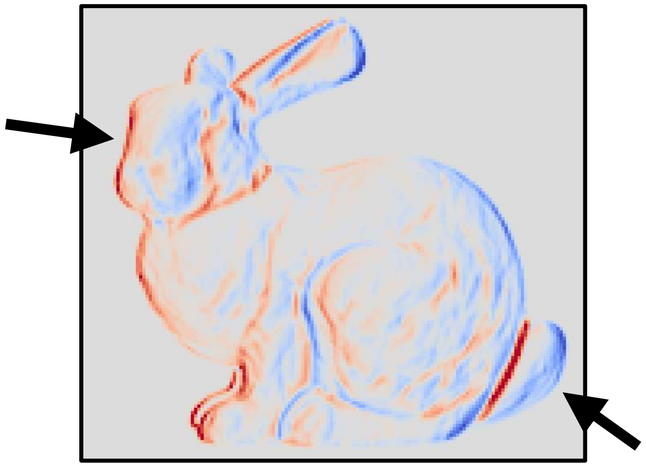
Ours

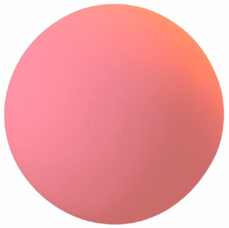
Gradients

From: *Differentiable Signed Distance Rendering* [Vicini, Speierer, and Jakob 2022]

How important is this, really?

Naïve





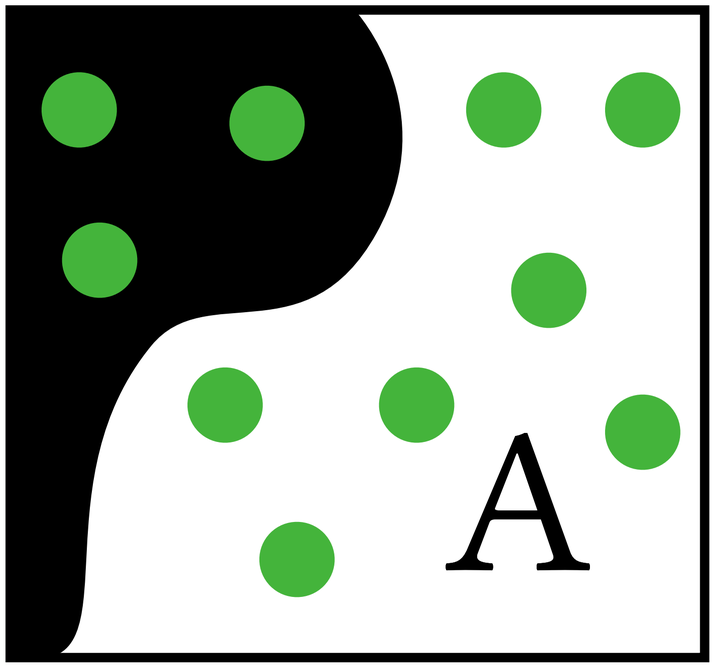
Ours

Gradients

Optimization

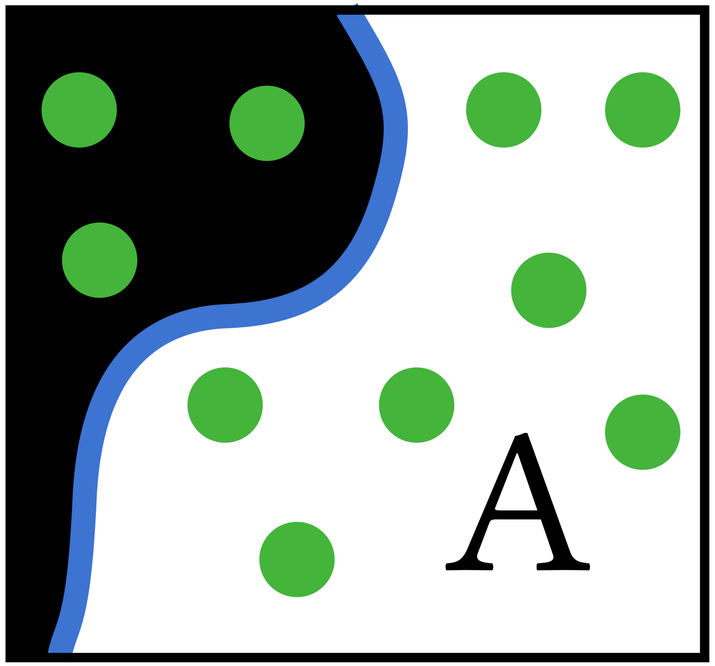
From: *Differentiable Signed Distance Rendering* [Vicini, Speierer, and Jakob 2022]

Discontinuous integrals



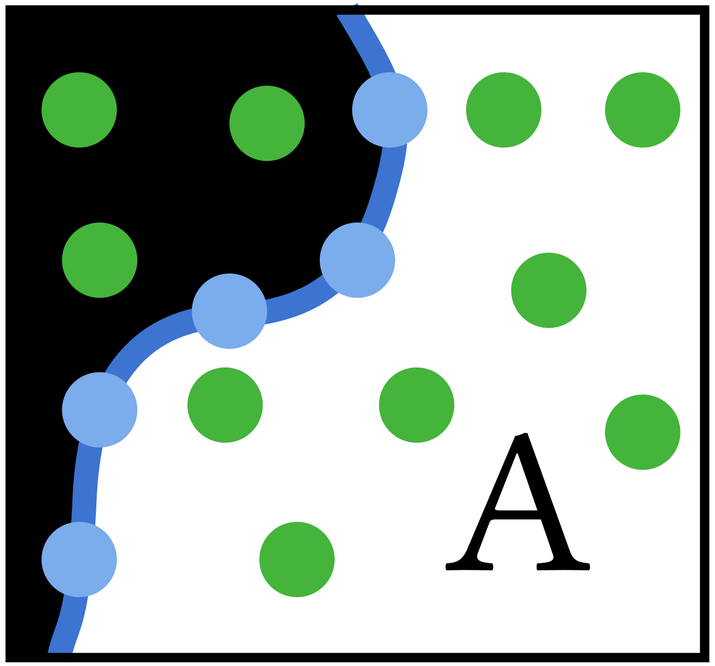
Edge sampling

Discontinuous integrals



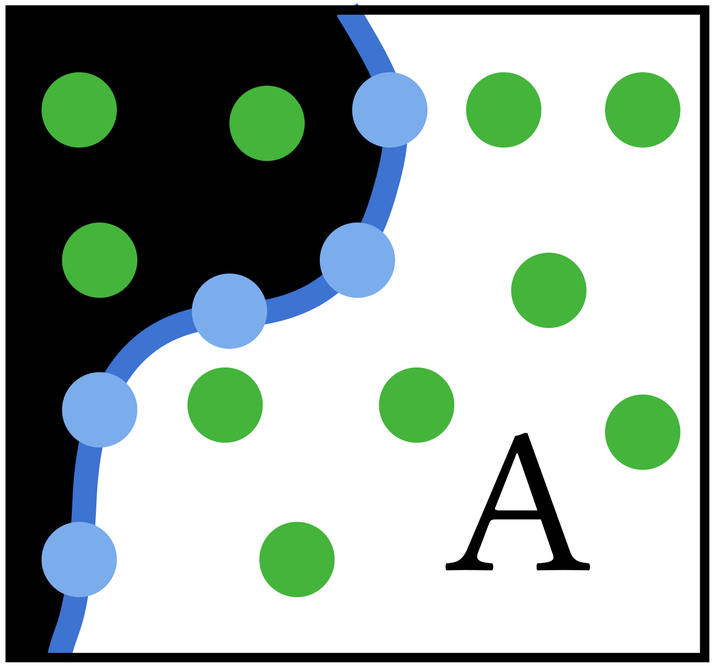
Edge sampling

Discontinuous integrals

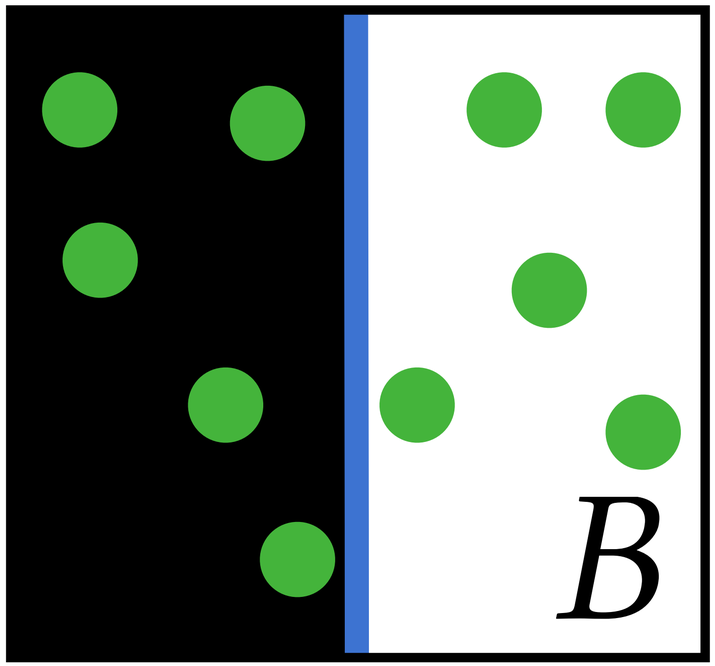


Edge sampling

Discontinuous integrals

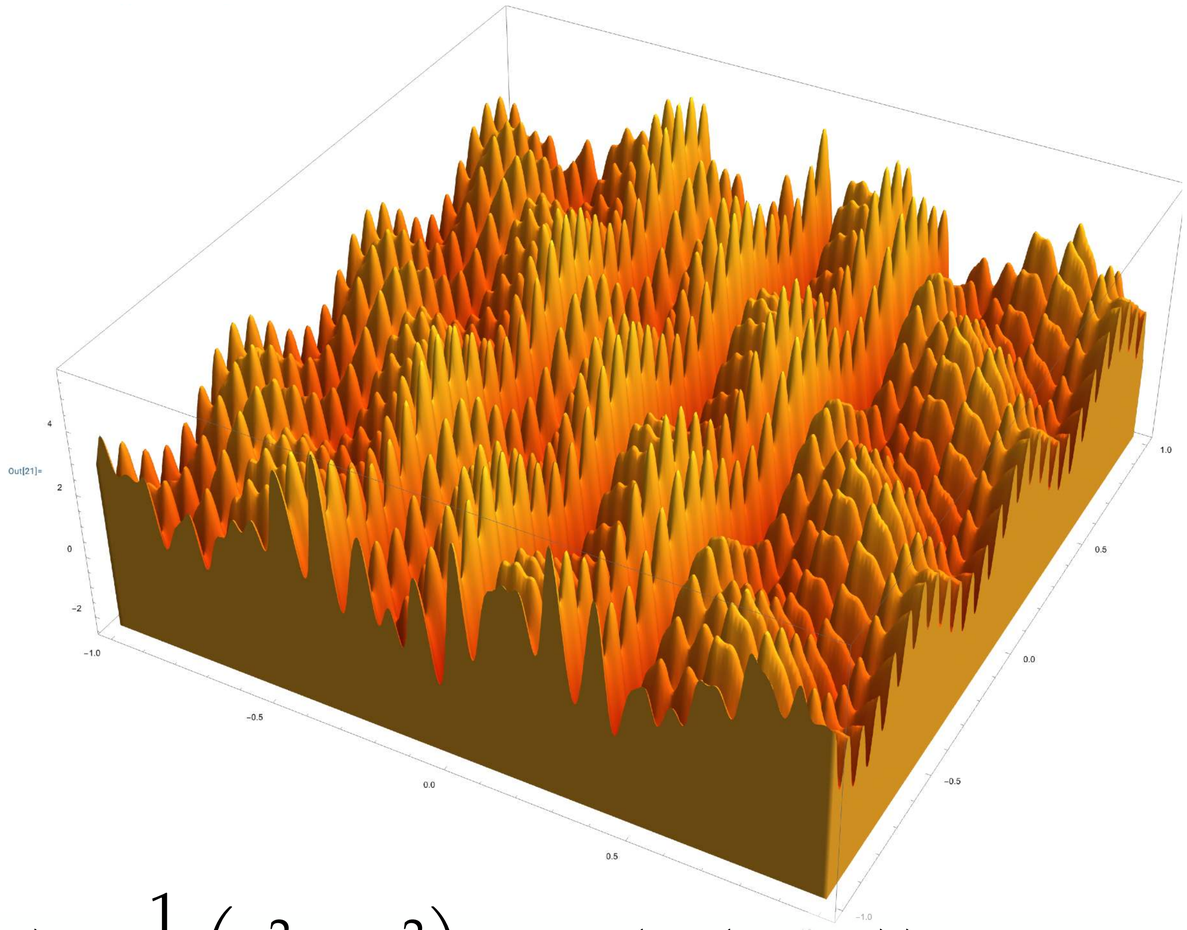


Edge sampling



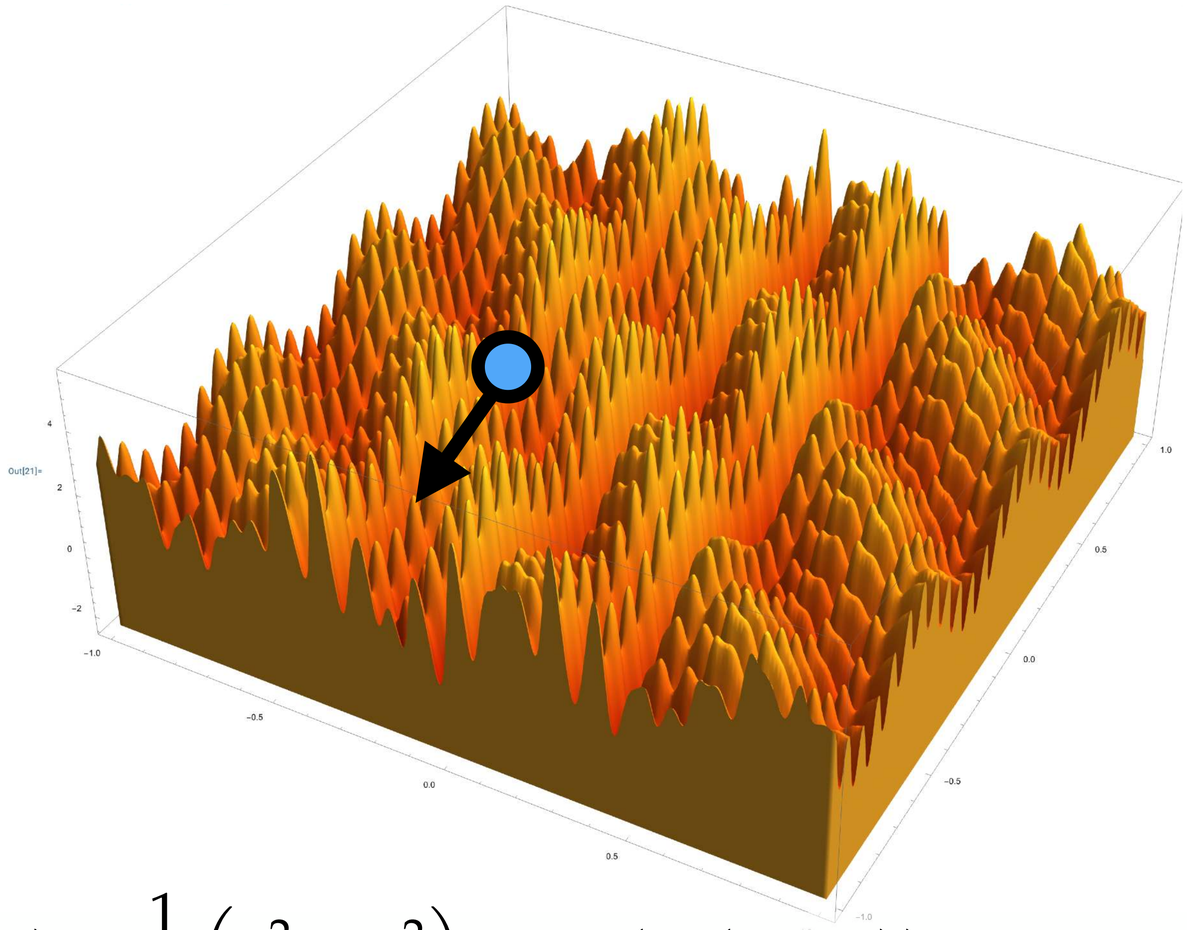
Reparameterization

Gradient-based optimization can be fragile



Plot of: $f(x,y) = \frac{1}{4}\left(x^2+y^2\right) - \sin(10(x+y)) + e^{\sin(50x)} + \sin(70\sin(x)) + \sin(\sin(80y))$

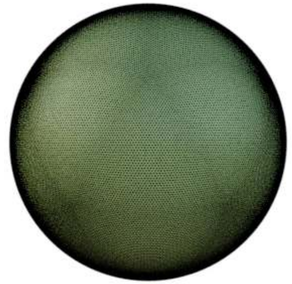
Gradient-based optimization can be fragile



Plot of: $f(x,y) = \frac{1}{4}\left(x^2+y^2\right) - \sin(10(x+y)) + e^{\sin(50x)} + \sin(70\sin(x)) + \sin(\sin(80y))$

Robustness

Target

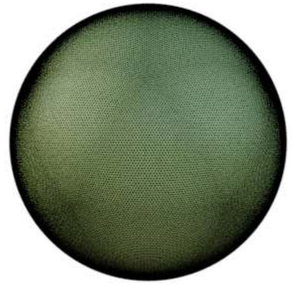


Reconstruction

From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

Robustness

Target



Reconstruction

From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

Robustness

Target

Reconstruction

From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

Regularizers to the rescue

$$\underset{\mathbf{x}\in\Omega}{\text{minimize}}\ \Phi(\mathbf{x})$$

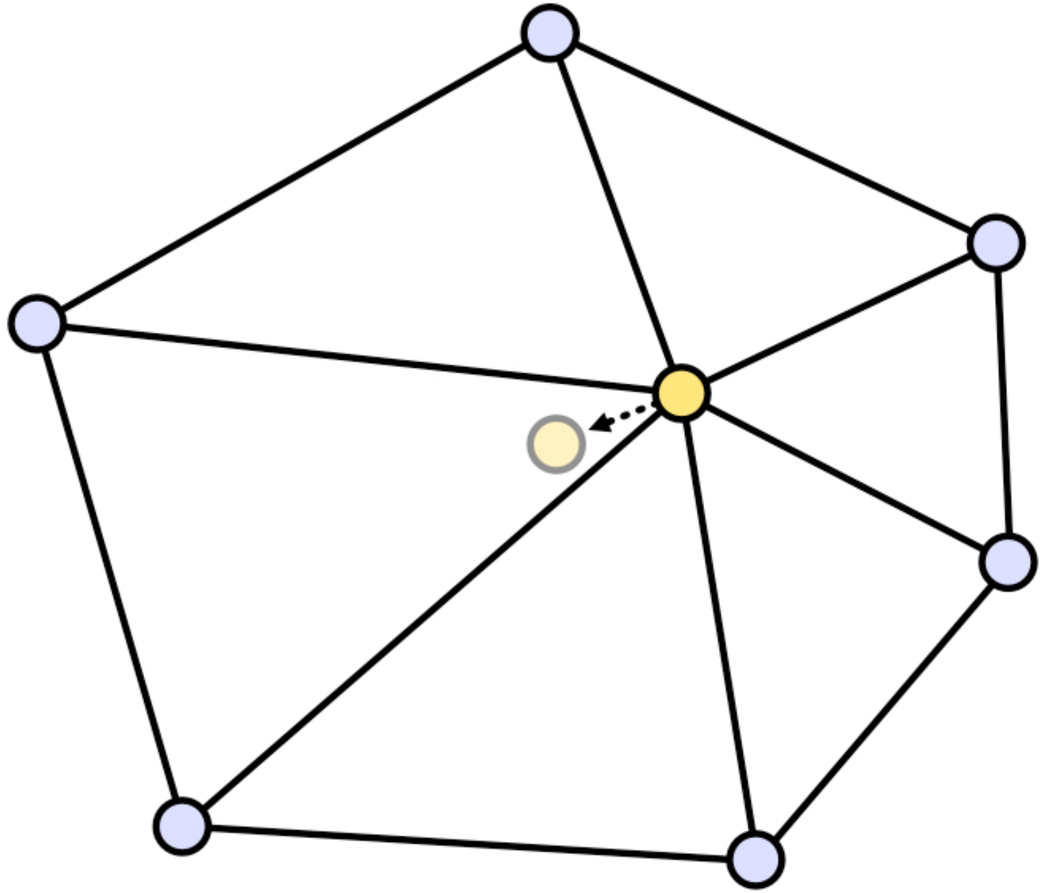
From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

Regularizers to the rescue

Original objective

Laplacian regularizer

$$\underset{\mathbf{x}\in\Omega}{\text{minimize}}\ \Phi(\mathbf{x}) + \lambda/2\,\text{tr}(\mathbf{x}^T\mathbf{L}\mathbf{x})$$



From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

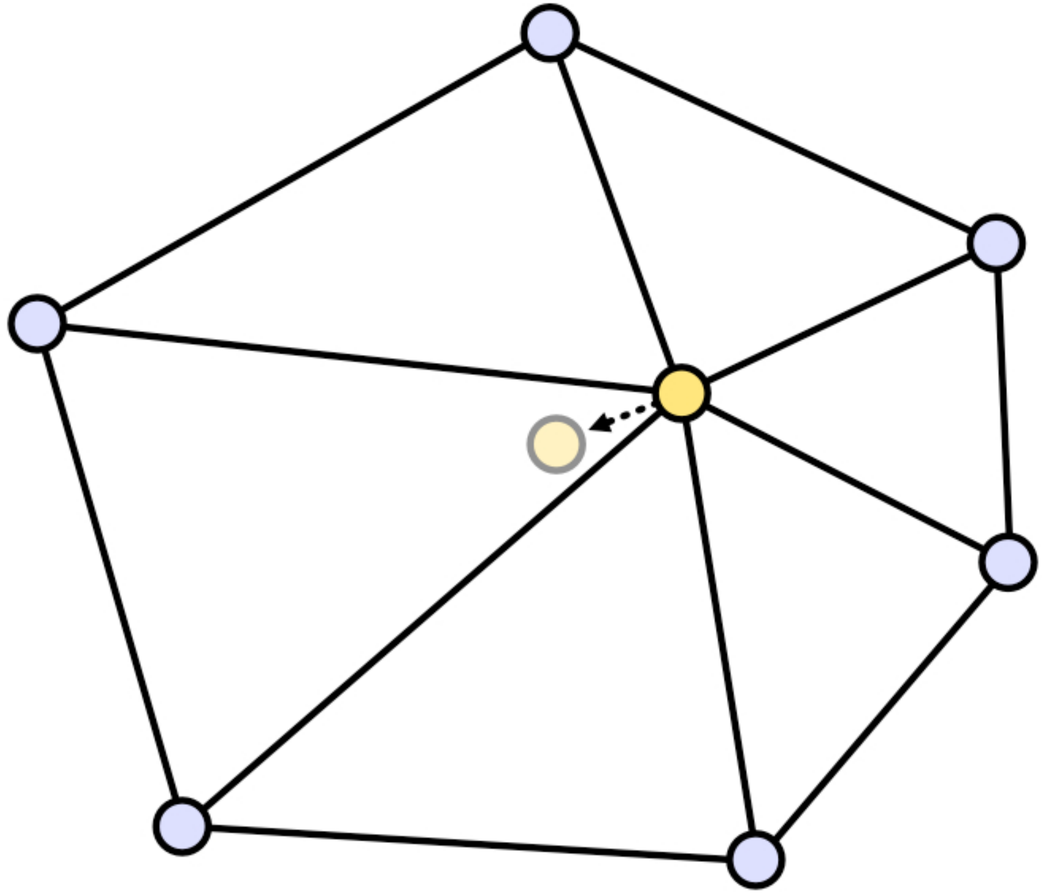
Regularizers to the rescue

Original objective

Laplacian regularizer

$$\underset{\mathbf{x}\in\Omega}{\text{minimize}}\ \Phi(\mathbf{x}) + \lambda/2\,\text{tr}(\mathbf{x}^T\mathbf{L}\mathbf{x})$$

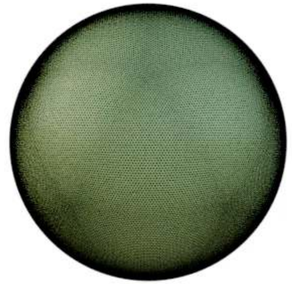
Compromise ☹



From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

Robustness

Target

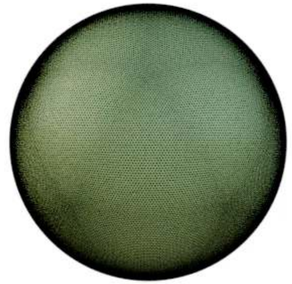


Reconstruction

From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

Robustness

Target



Reconstruction

From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

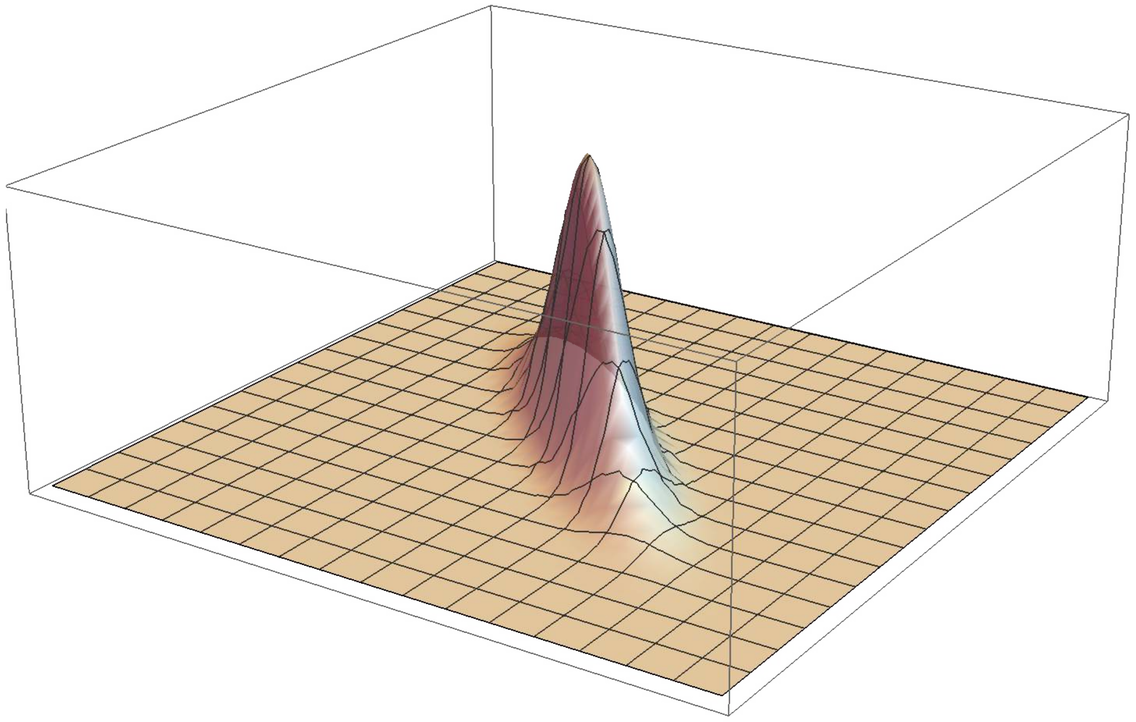
Robustness

Target

Reconstruction

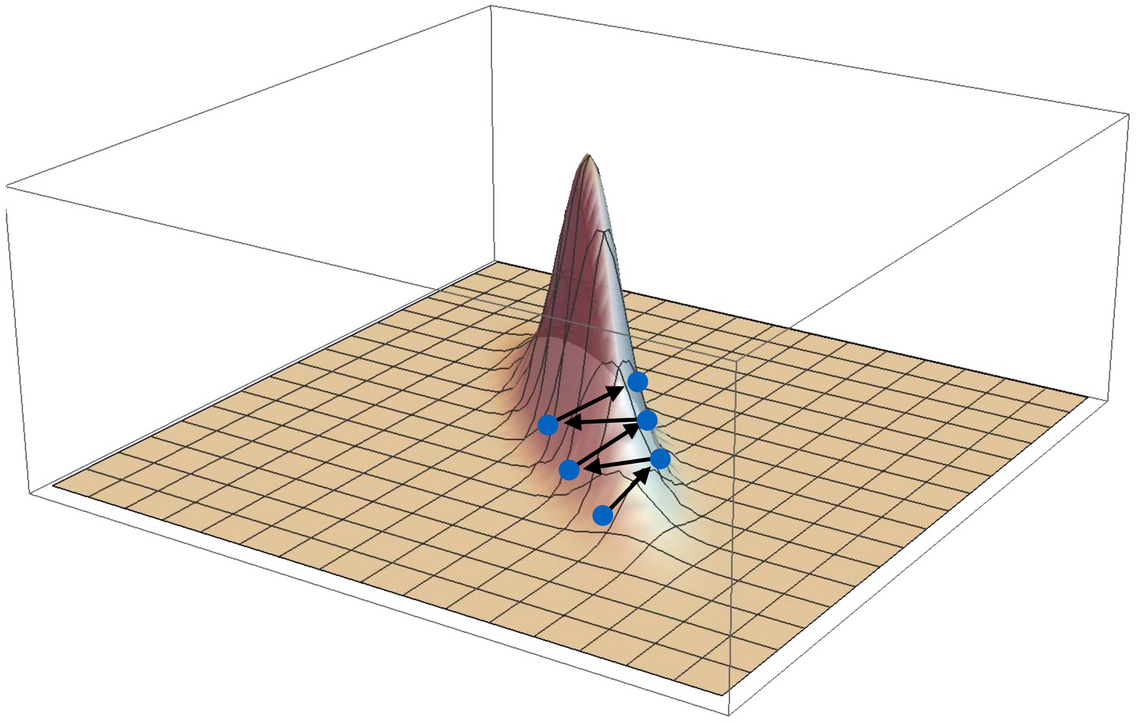
From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

2D analogy of the problem



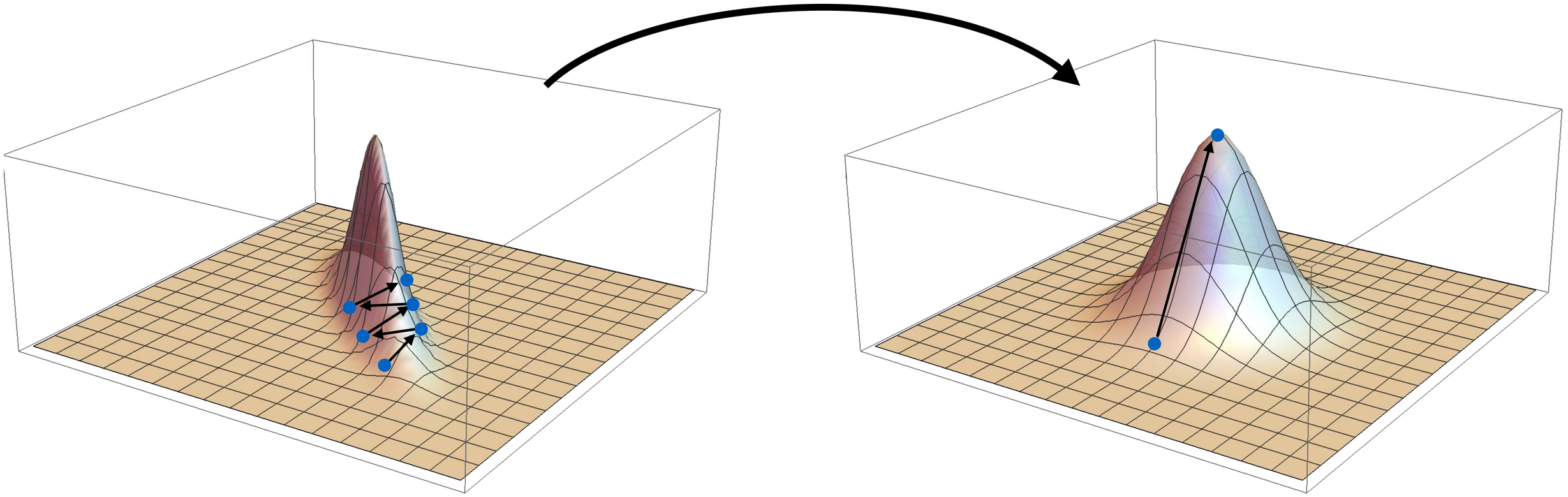
From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

2D analogy of the problem



From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

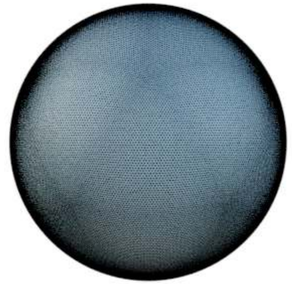
2D analogy of the problem



From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

Quasi-Newton optimization method

Target

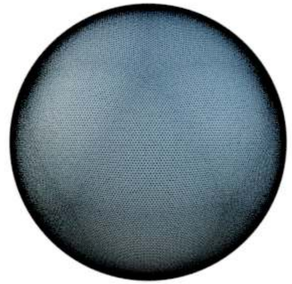


Reconstruction

From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

Quasi-Newton optimization method

Target

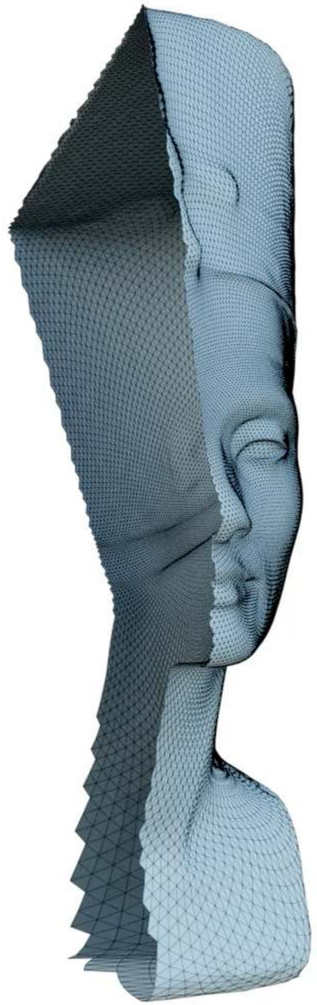


Reconstruction

From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

Quasi-Newton optimization method

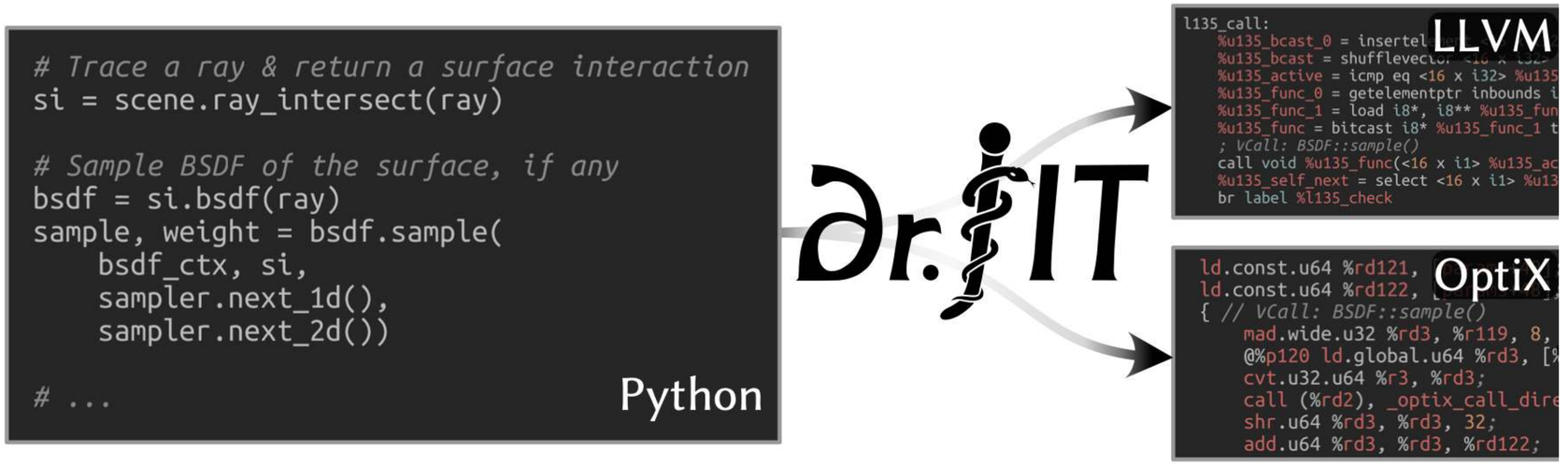
Target



Reconstruction

From: *Large Steps in Inverse Rendering of Geometry* [Nicolet, Jacobson, and Jakob 2021]

Technology



Dr.Jit: Compiler & automatic differentiation engine

Mitsuba 3

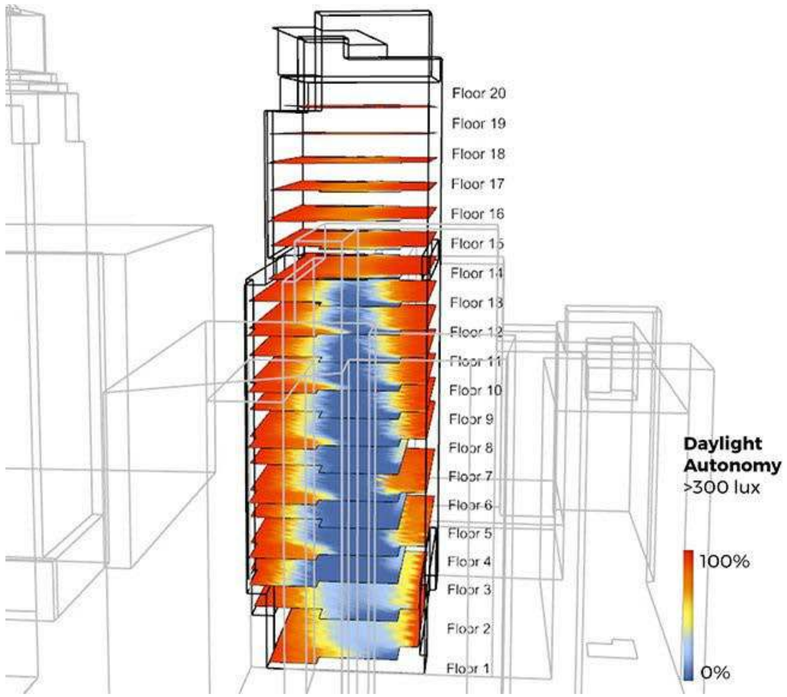
Mitsuba: Differentiable physically-based renderer

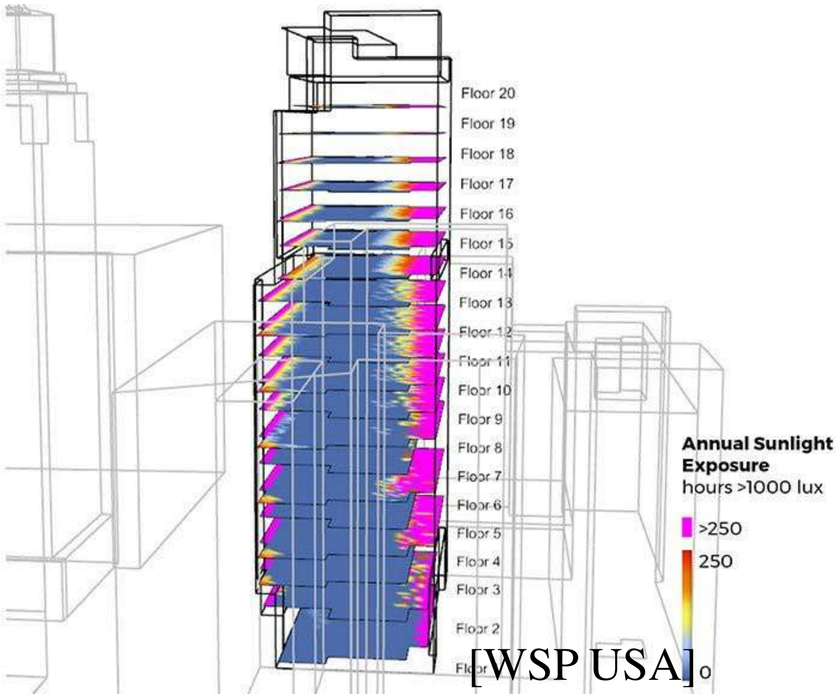
Beyond computer graphics: a world of applications

Many disciplines rely on understanding or controlling the behavior of light in images or other kinds of measurements.

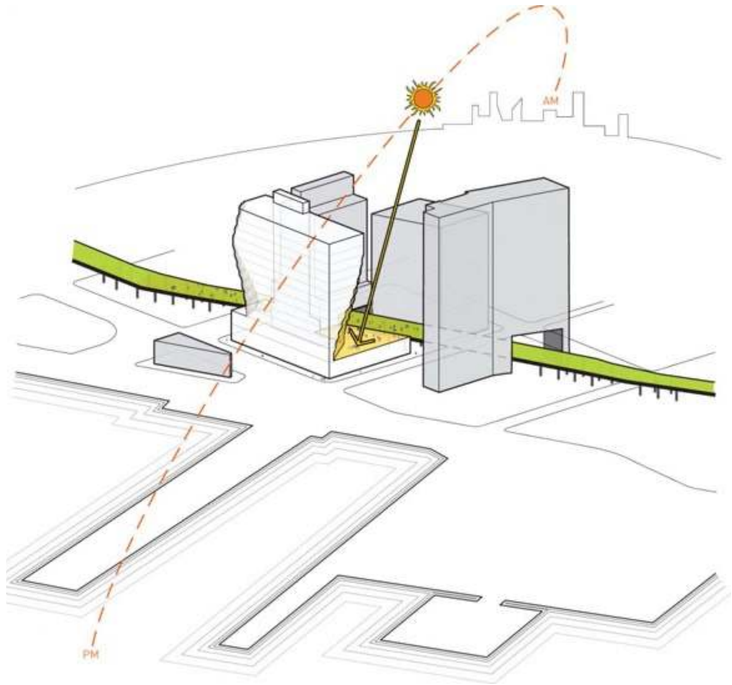
Beyond computer graphics: a world of applications

Many disciplines rely on understanding or controlling the behavior of light in images or other kinds of measurements.





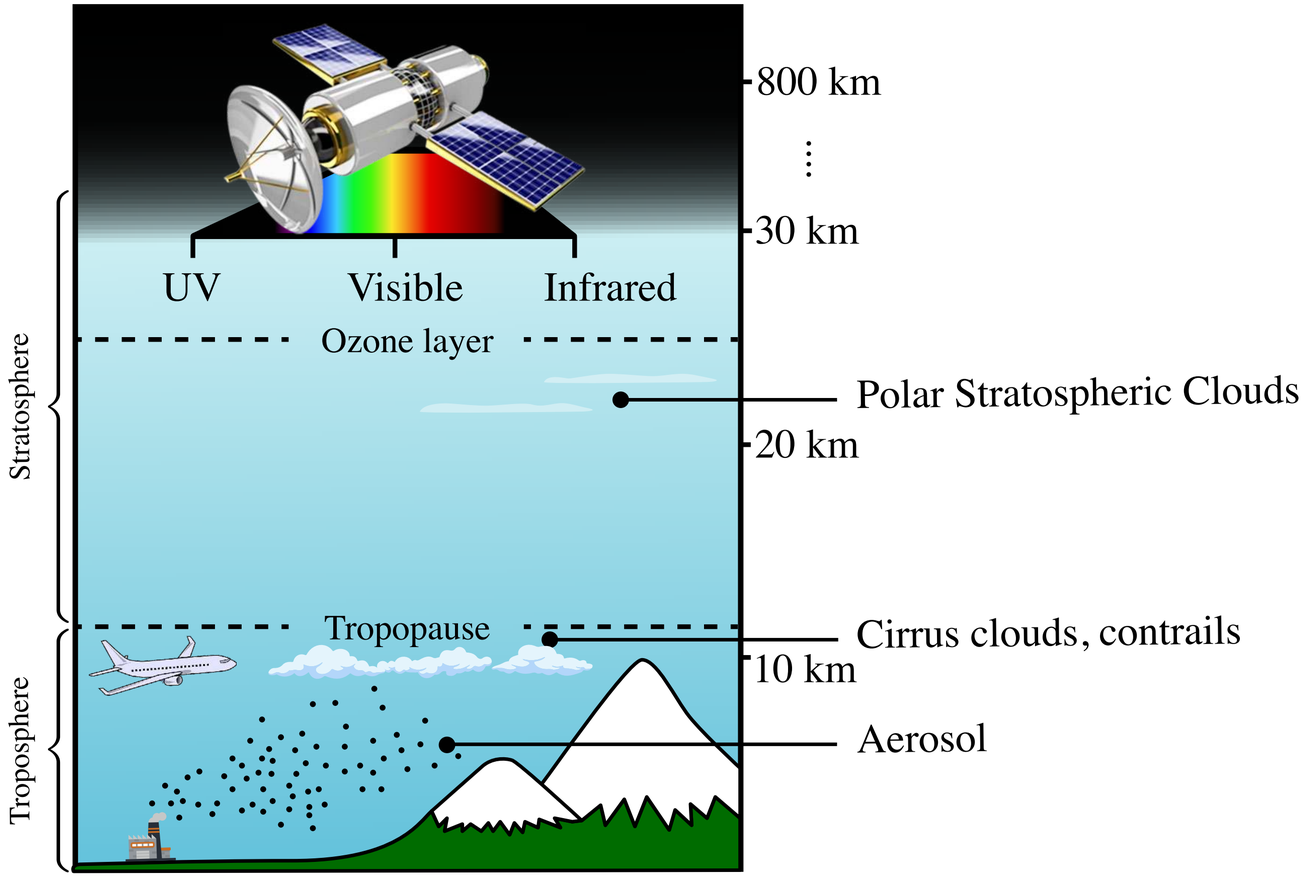
[WSP USA]



[Solar Carve Tower - Studio Gang]

Earth Observation

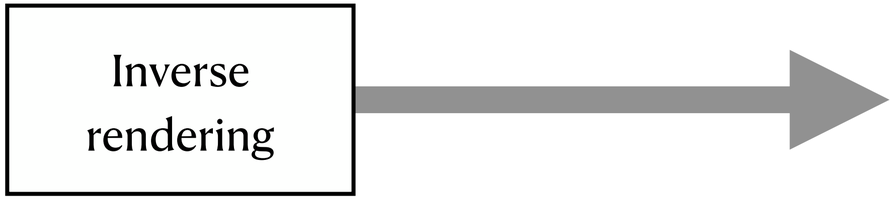
Shanghai and Yangtze river mouth
[ESA, Copernicus Sentinel-3A, OCLI instrument]



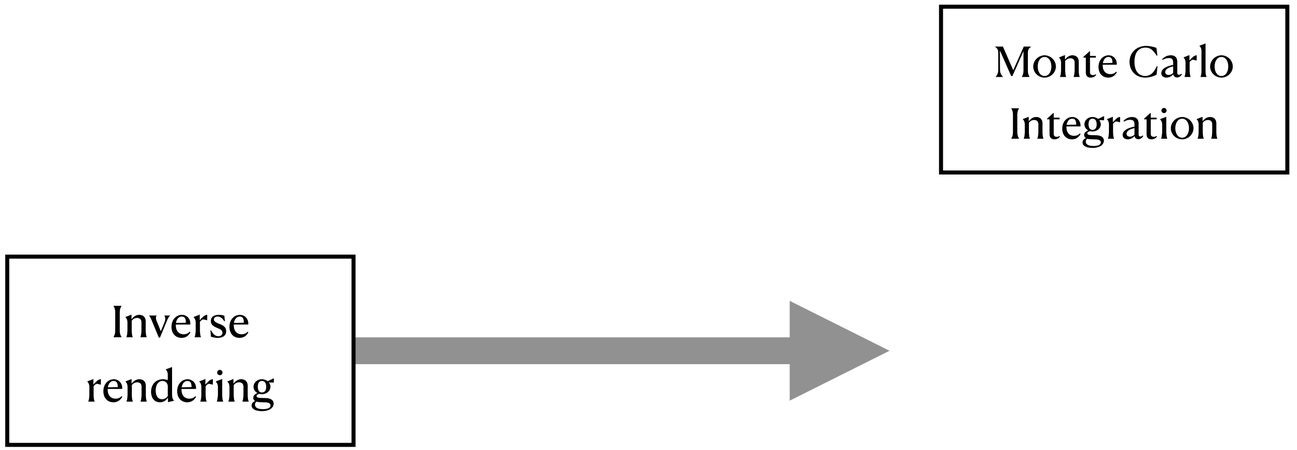
So many connections

Inverse rendering

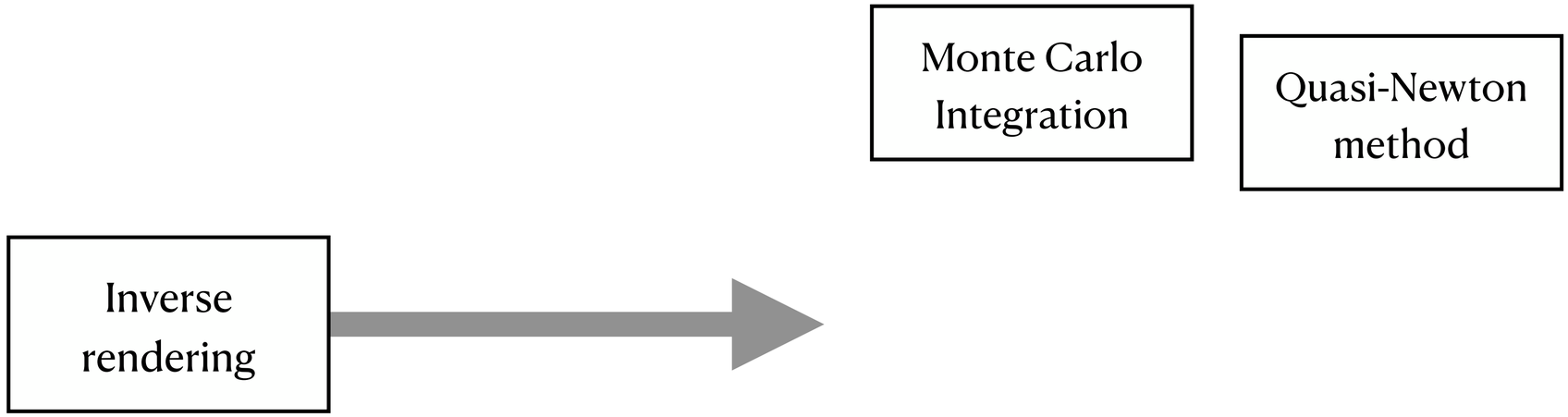
So many connections



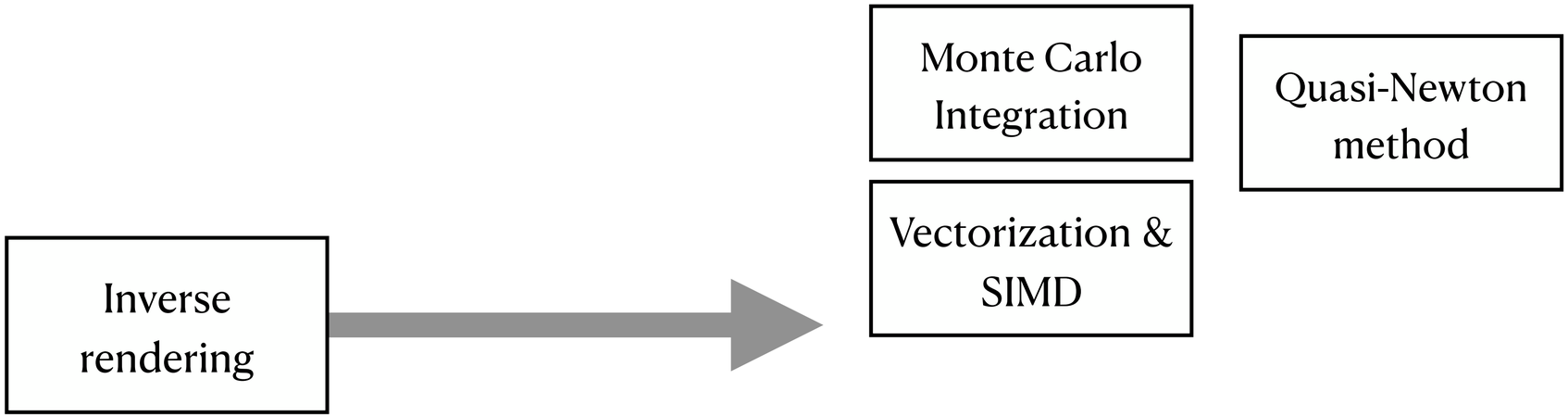
So many connections



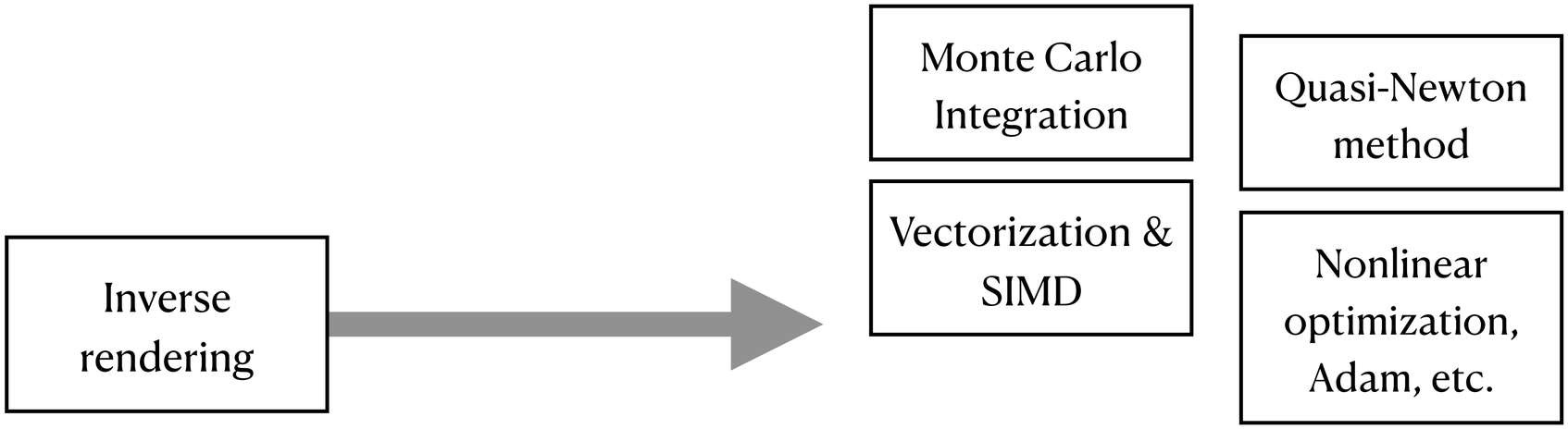
So many connections



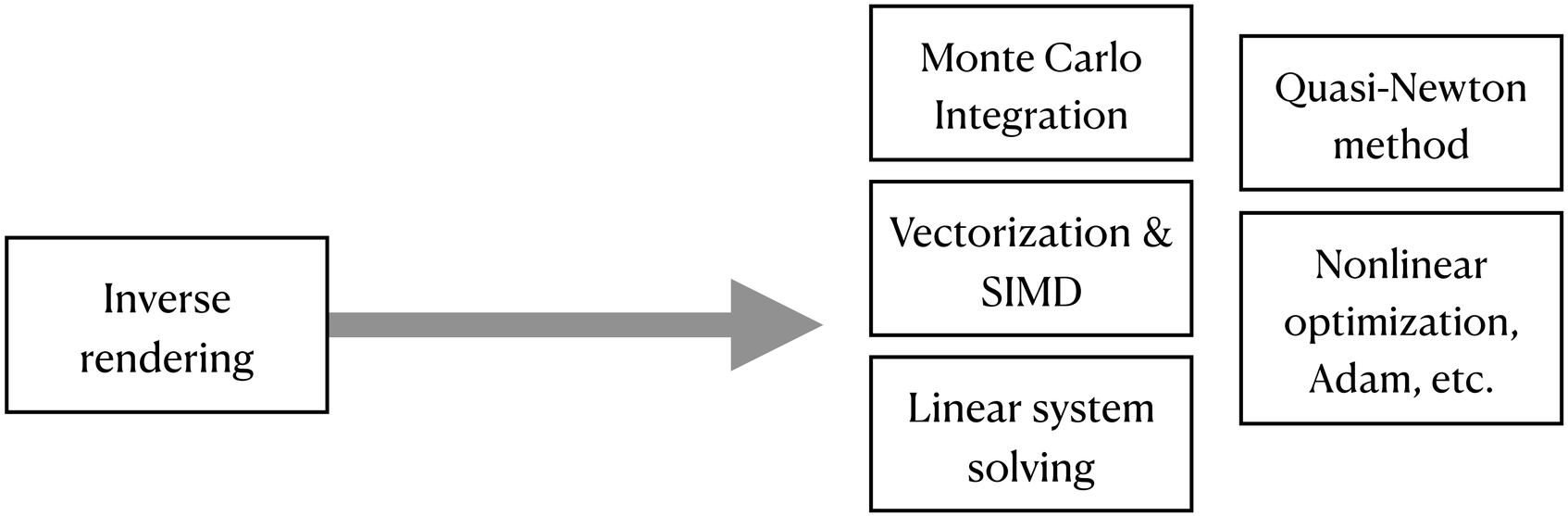
So many connections



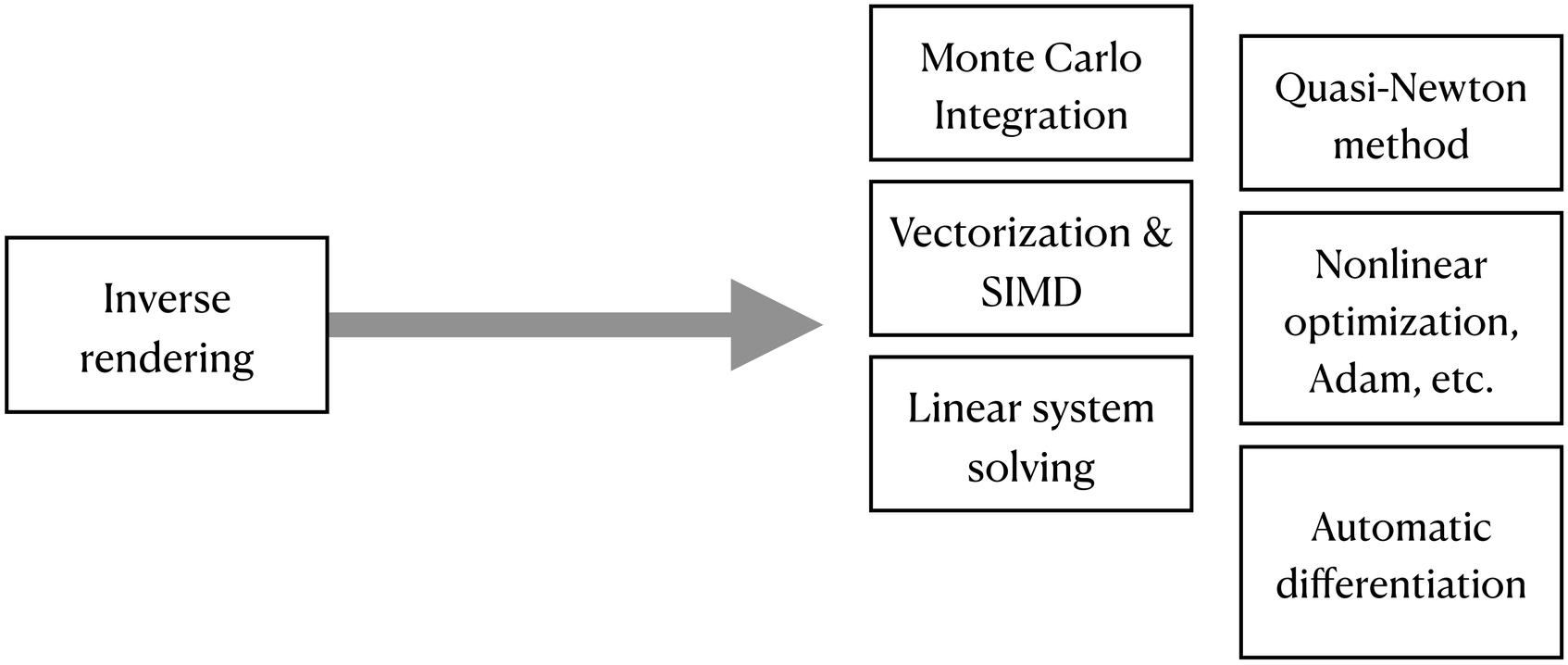
So many connections



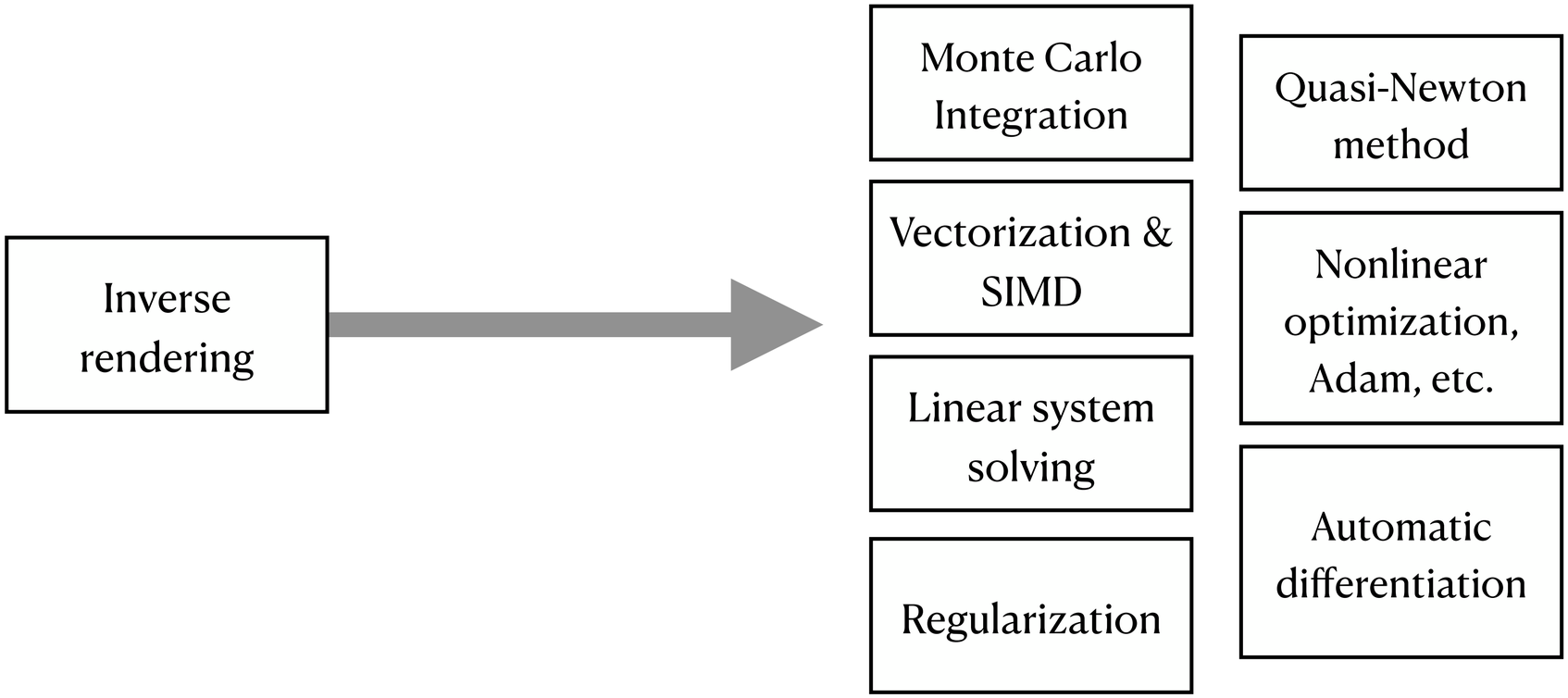
So many connections



So many connections



So many connections



EPFL RGL
Realistic Graphics Lab

I hope you enjoyed the course!