Embodied Vision

Rendering and 3D Scene Reconstruction

Tsung-Wei Ke

Spring 2026

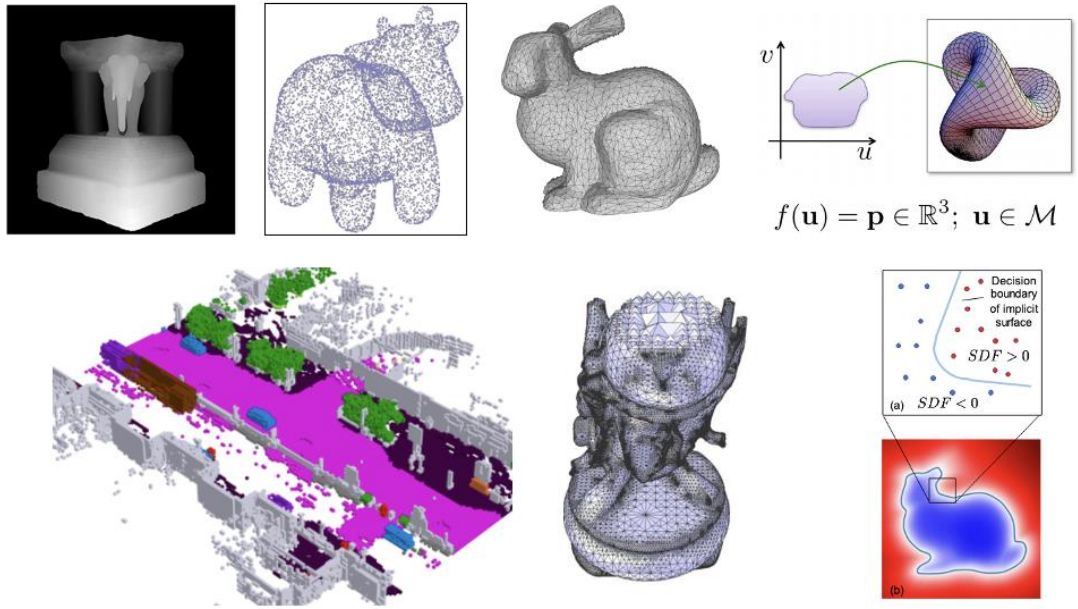
Disclaimer

- This lecture borrows contents heavily from
  - Learning for 3D Vision by Shubam Tulsiani at CMU
  - Image Synthesis Techniques by Kayvon Fatahalian, Doug James and Matt Pharr at Stanford
  - Computer Graphics and Imaging by Ren Ng at UC Berekely
  - Machine Learning for Inverse Graphics by Vincent Sitzmann at MIT

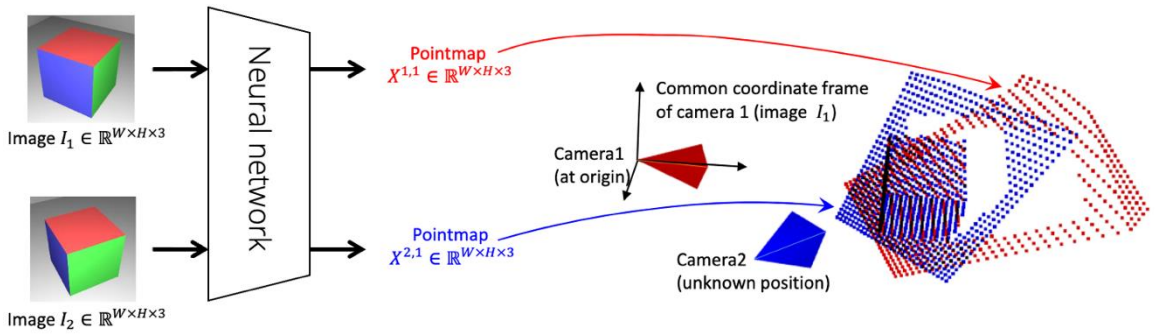
Recap

Structure from Motion

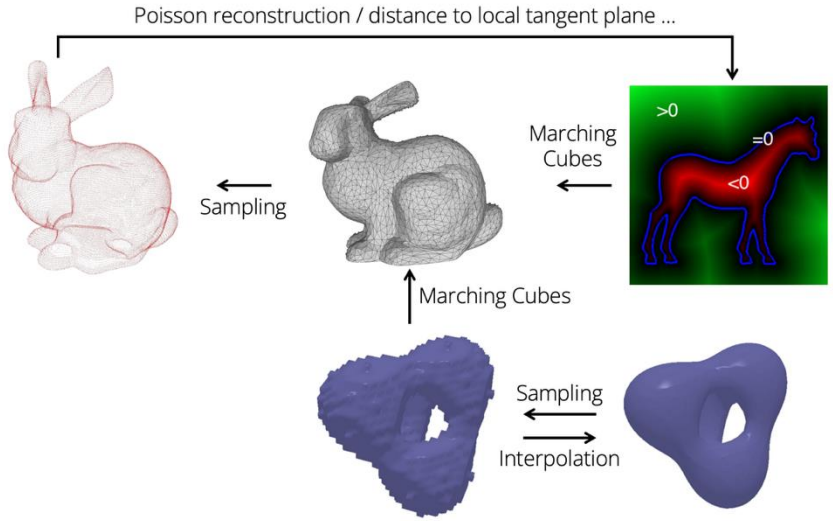
3D Representations



DUSt3R



Conversion between 3D Representations



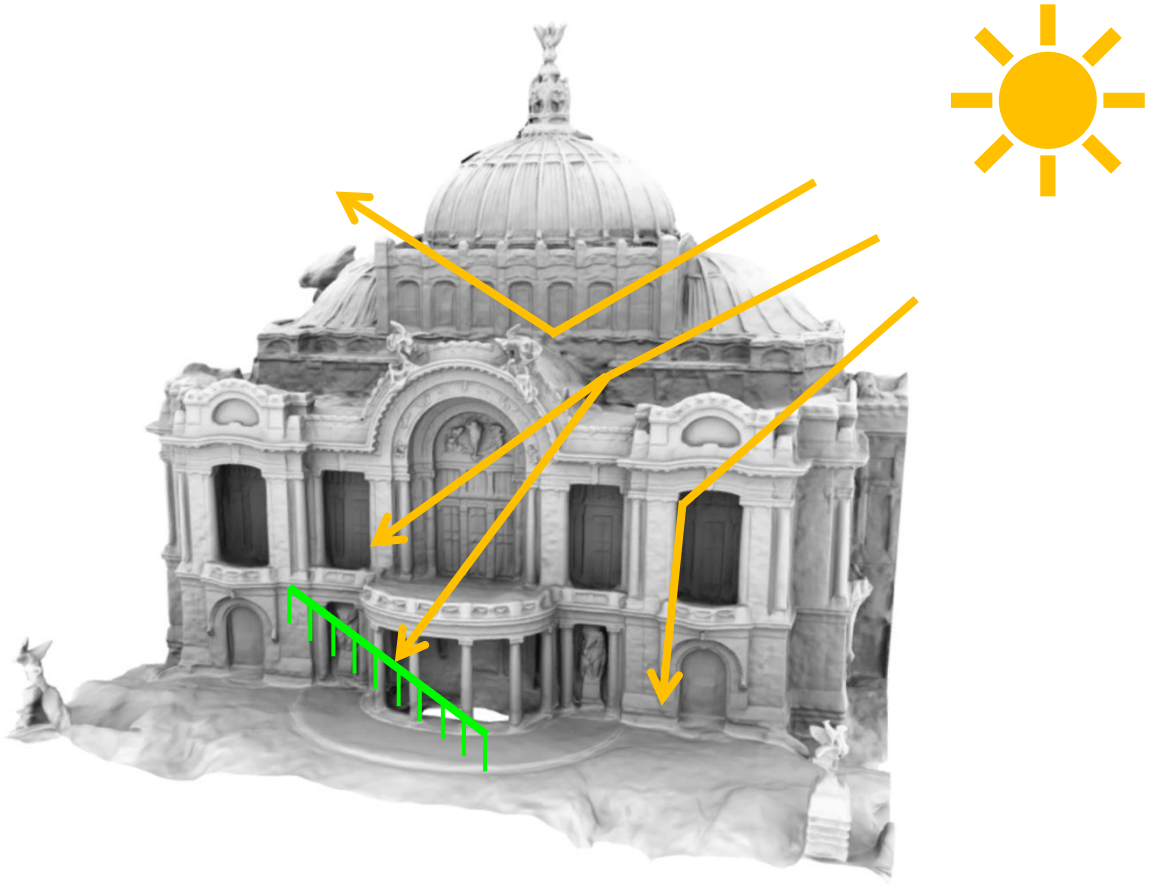
Content

- Light Sources
- Light Measurement
- A mathematical model for image rendering
- Volume Rendering
  - Voxel
  - Nerf
  - Gaussian Splatting

Content

- Light Sources
- Light Measurement
- A mathematical model for image rendering
- Volume Rendering
  - Voxel
  - Nerf
  - Gaussian Splatting

We Know How to Model the 3D World, Let's Revisit Image Formation



Recap: 2D Image Formation from the 3D World--A simplified model of photometric image formation

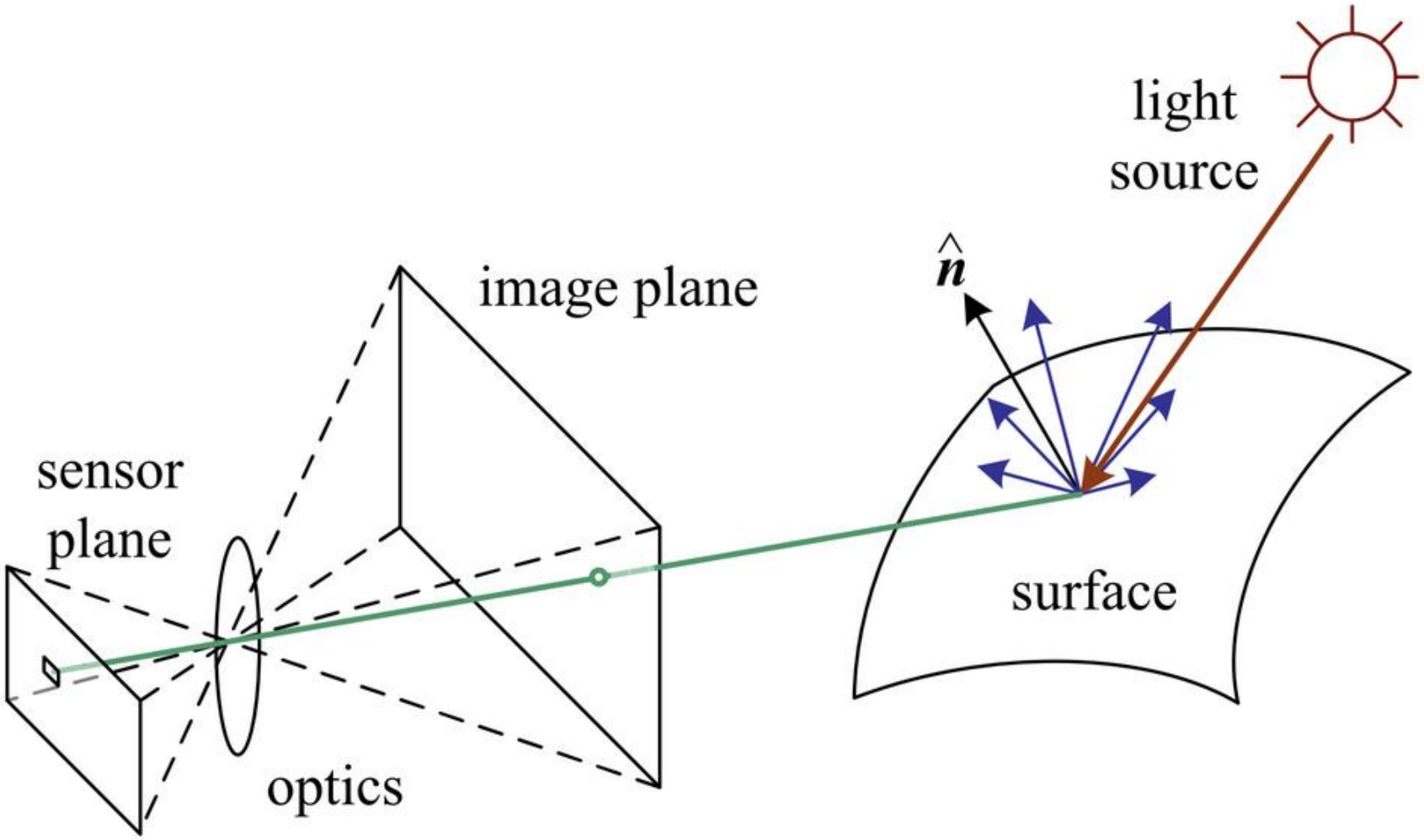
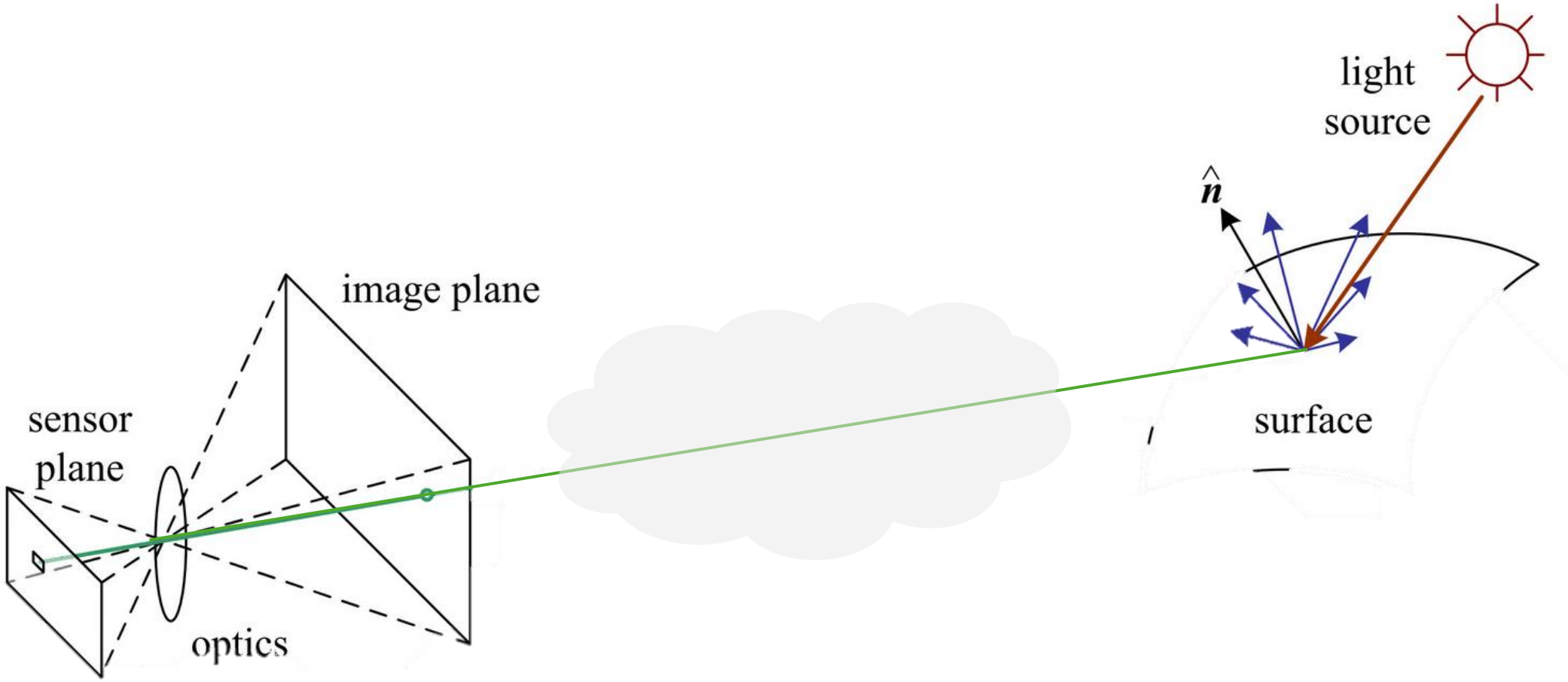


Image source S. Seitz text book

Absorption: Light Gets Absorbed when Traveling In a Medium



Absorption: Light Gets Absorbed when Traveling In a Medium

Low density smoke

High density smoke

Image source Walt Disney Animation Studios

Absorption: Light Gets Absorbed when Traveling In a Medium

Image source https://www.bbc.com/future/article/20151221-the-lethal-effects-of-london-fog

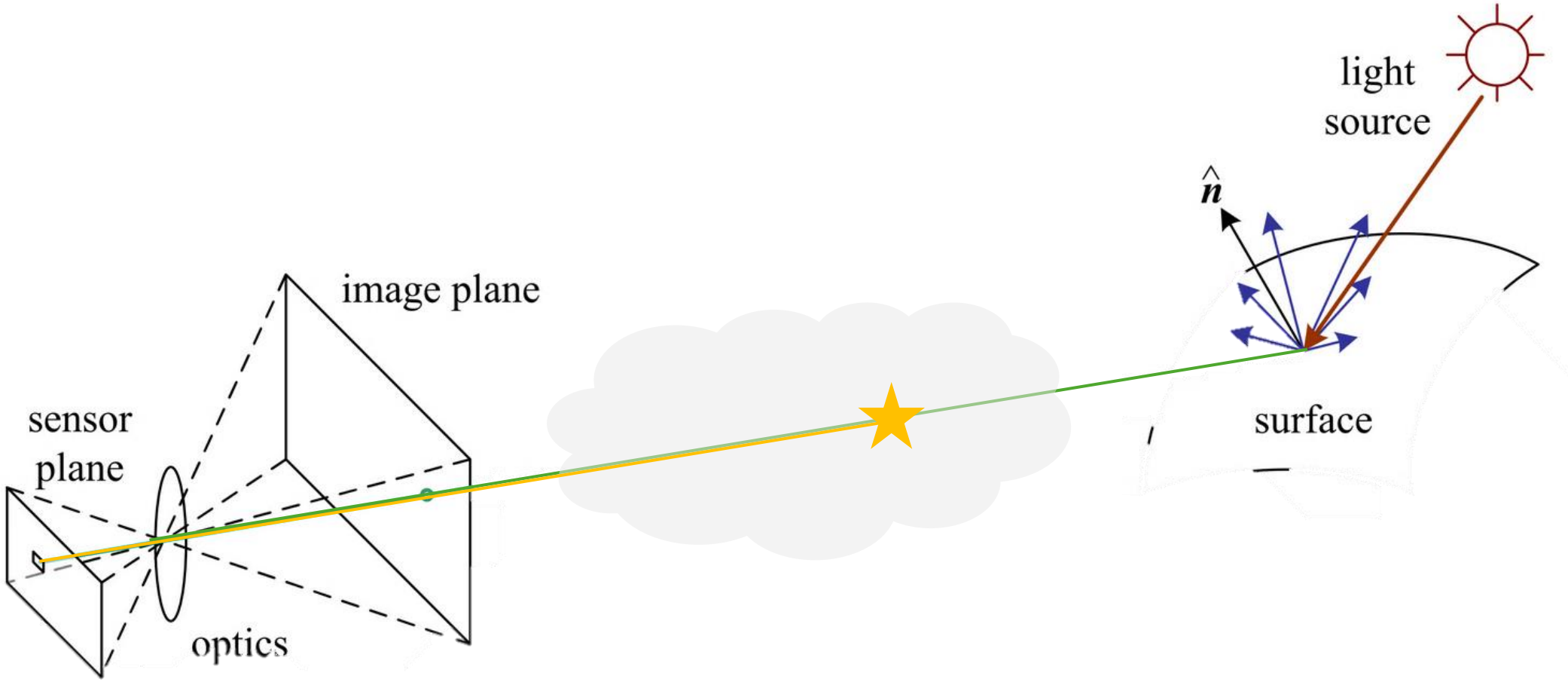
Absorption: Light Gets Absorbed when Traveling In a Medium

Thus, if one is to be five times as distant, make it five times bluer.

—Treatise on Painting, Leonardo Da Vinci, pp 295, circa 1480.

Slide credit S. Tulsiani

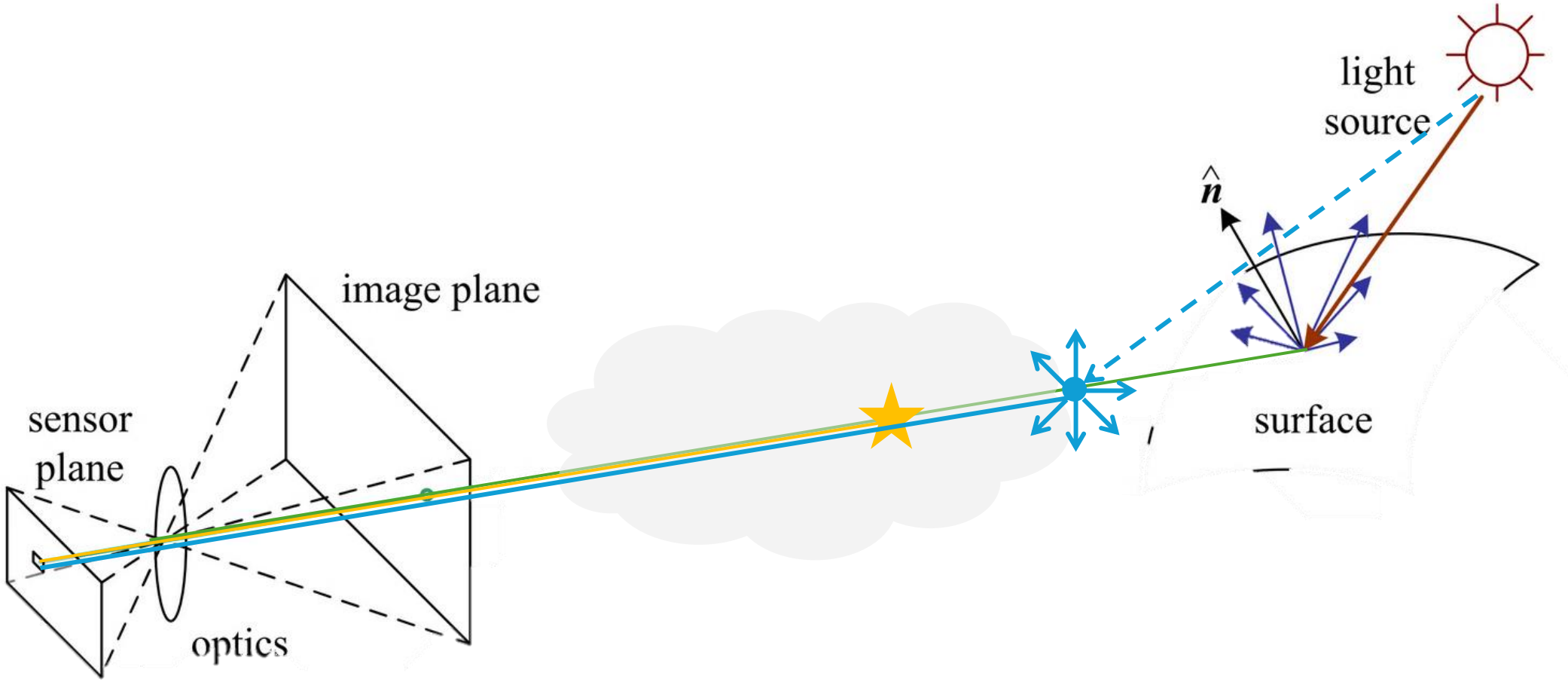
Emission: The Medium Emits Light



Emission: The Medium Emits Light

Slide credit S. Tulsiani

Scattering: Light Along the Ray Comes from Different Source



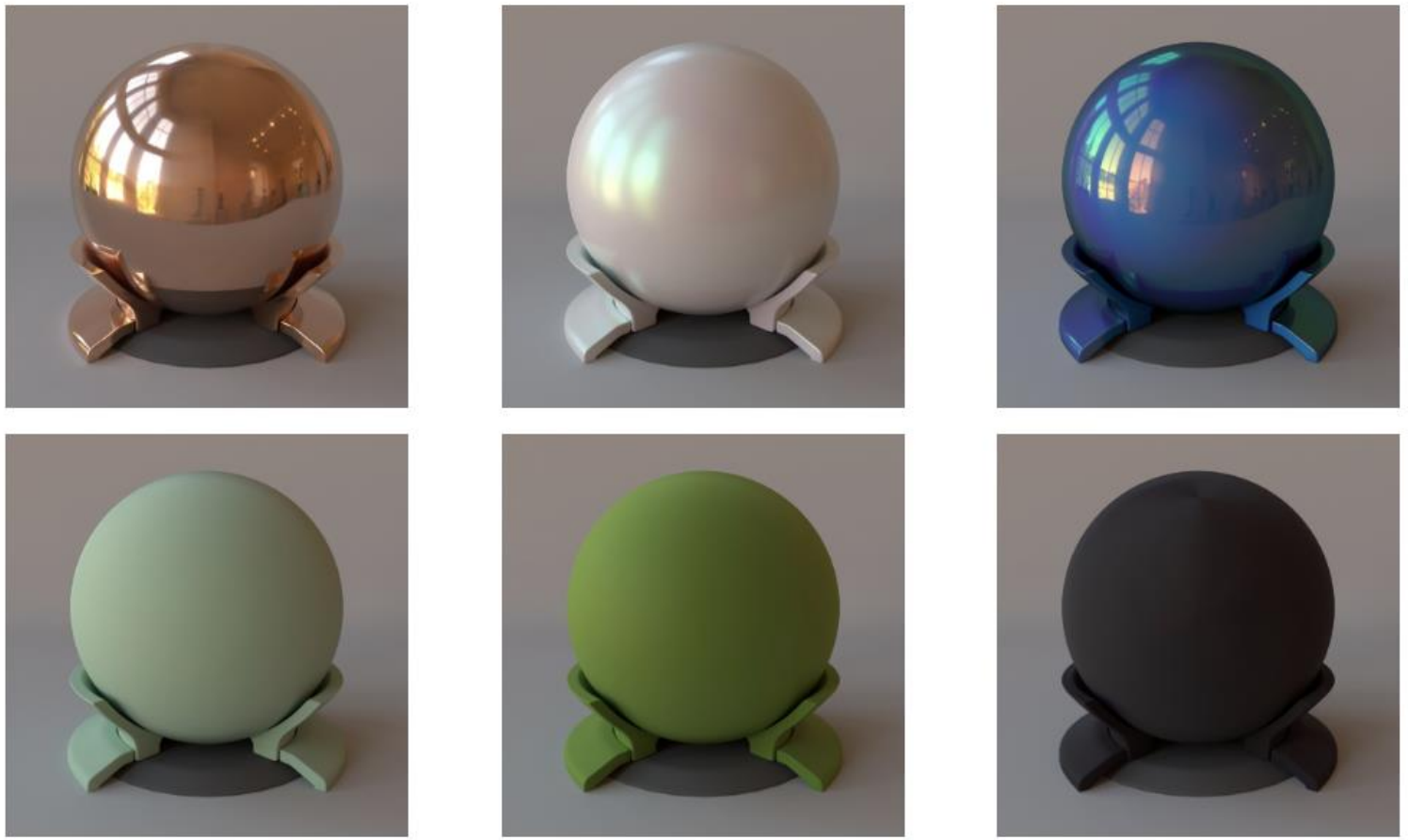
Scattering: Light Along the Ray Comes from Different Source

More Backward Scattering

More Forward Scattering

Image source K. Fatahalian, D. James and M. Pharr

Scattering: Light Along the Ray Comes from Different Source



EPFL RGL Material Database

Image source K. Fatahalian, D. James and M. Pharr

Surface Reflection Oversimplifies Image Formation

Greg Zaal: https://polyhaven.com/a/rathaus

Image source K. Fatahalian, D. James and M. Pharr

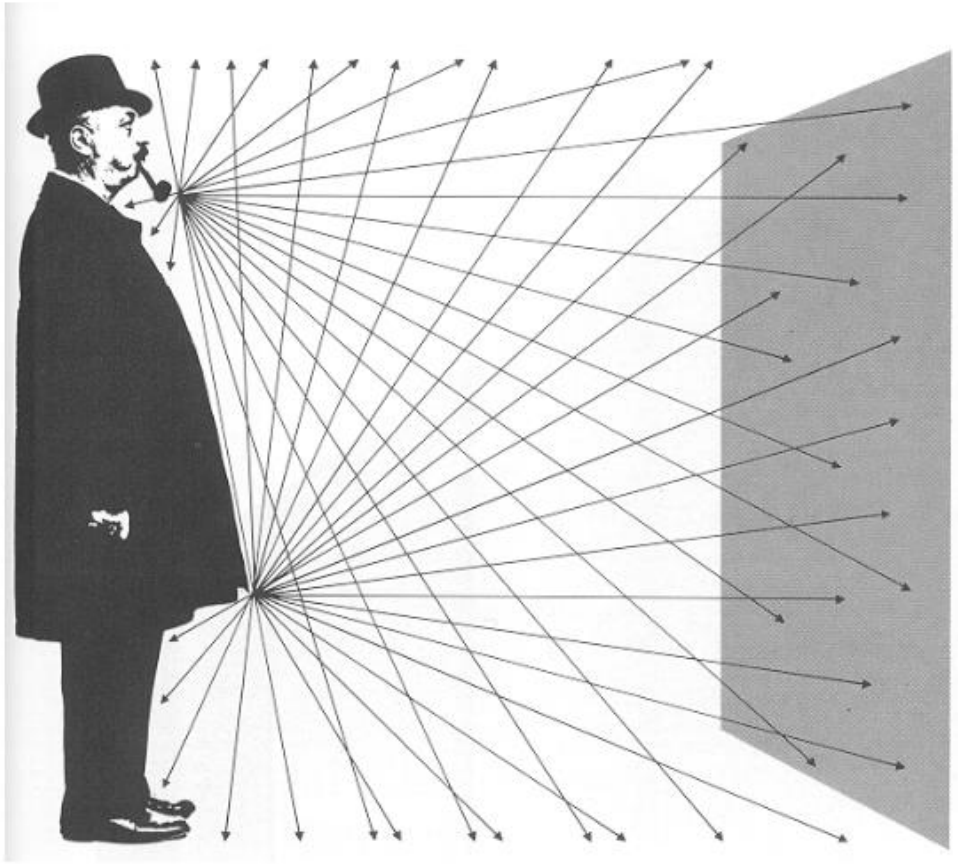
Content

- Light Sources
- Light Measurement
- A mathematical model for image rendering
- Volume Rendering
  - Voxel
  - Nerf
  - Gaussian Splatting

Let’s Derive the Mathematical Model for Image Rendering

How to Measure Light?

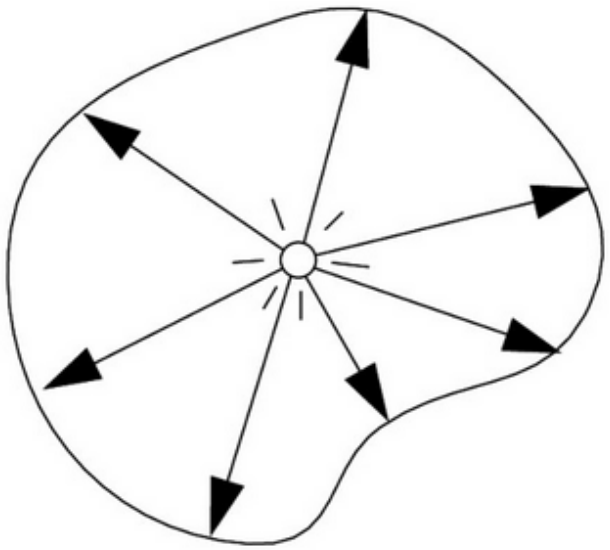
Flux – What's the Density of Photons Flowing Through a Sensor?



From London and Upton

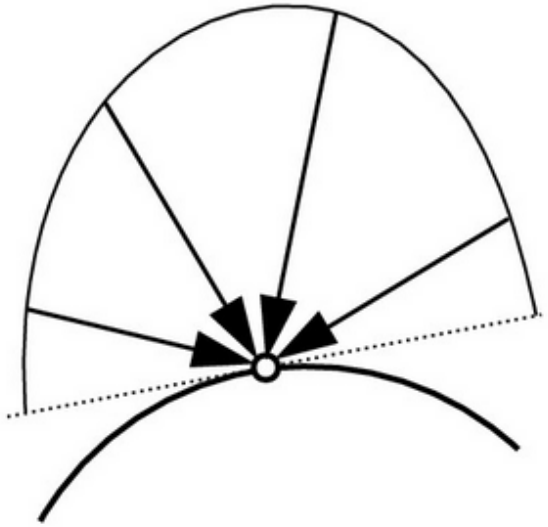
Slide credit R. Ng

Different Types of Light Measurements



Light Emitted From A Source

"Radiant Intensity"



Light Falling On A Surface

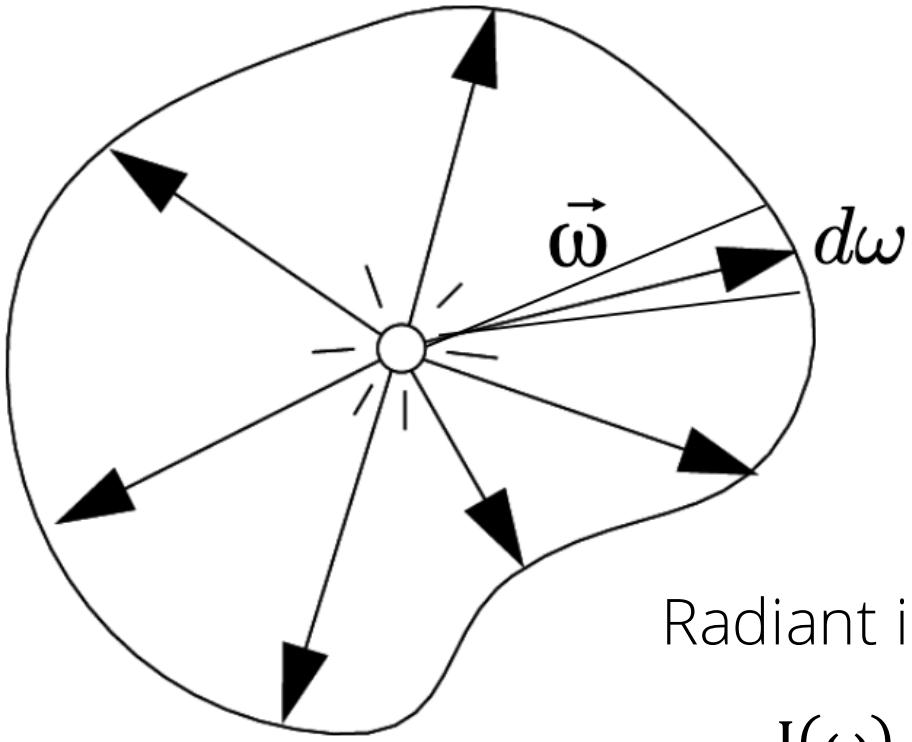
"Irradiance"

Light Traveling Along A Ray

"Radiance"

Slide credit R. Ng

Radiant Intensity: The Power Per Unit Solid Angle Emanating from a Point Source

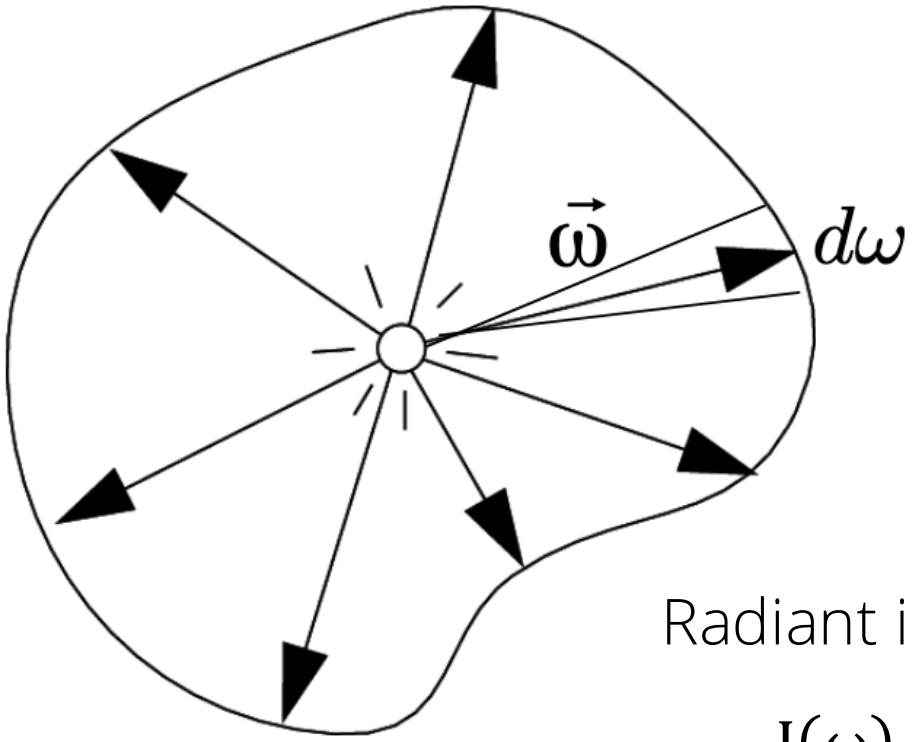


Radiant intensity:

$$\mathrm{I}(\omega) \equiv \frac{d\Phi}{d\omega}$$

Φ : the radiant energy emitted, reflected, transmitted and received per unit time
ω: solid angle

Radiant Intensity: The Power Per Unit Solid Angle Emanating from a Point Source



Radiant intensity:

$$\mathrm{I}(\omega) \equiv \frac{d\Phi}{d\omega}$$

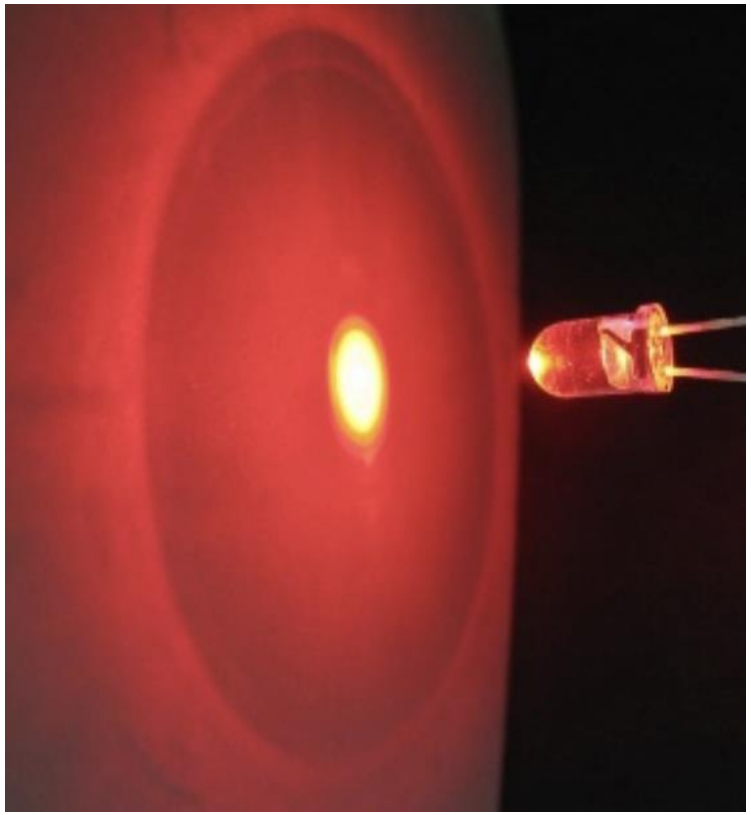
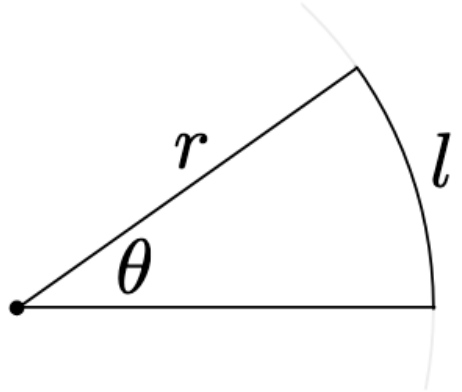


Image source
https://www.jensign.com/LEDIntensity

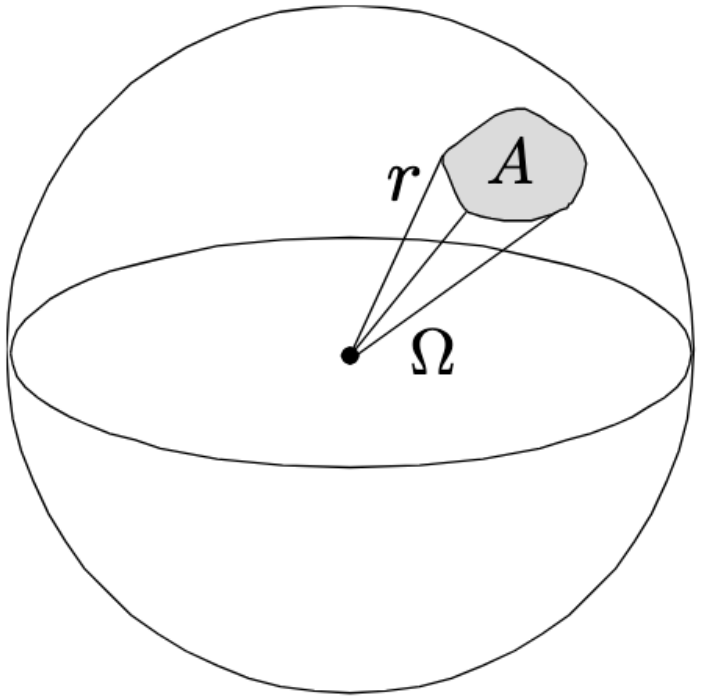
Solid Angle

Angle $\theta = \frac{l}{r}$



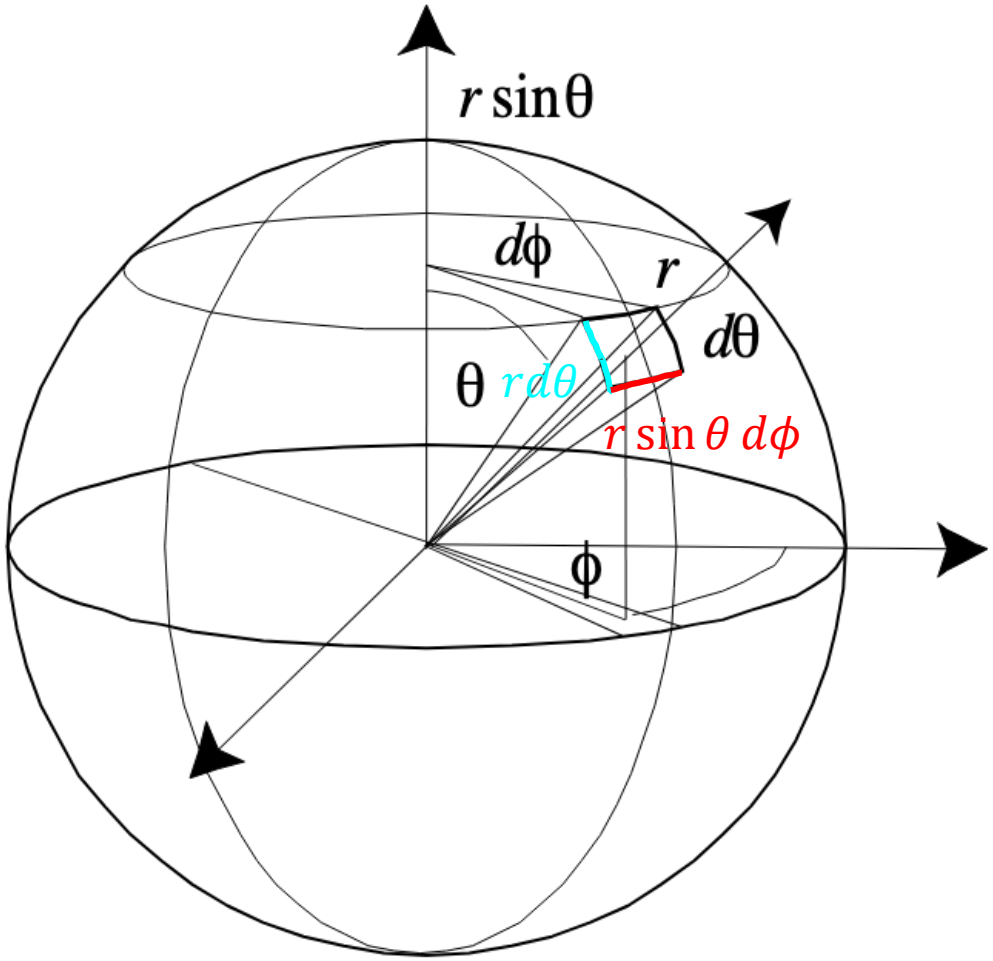
⇒ circle has 2 π radians

Solid angle $\Omega = \frac{A}{r^2}$



⇒ sphere has 4 π steradians

Infinitesimal Solid Angle



$$dA = (rd\theta)(r\sin\theta\, d\phi)$$
$$= r^2 \sin\theta\, d\theta d\phi$$

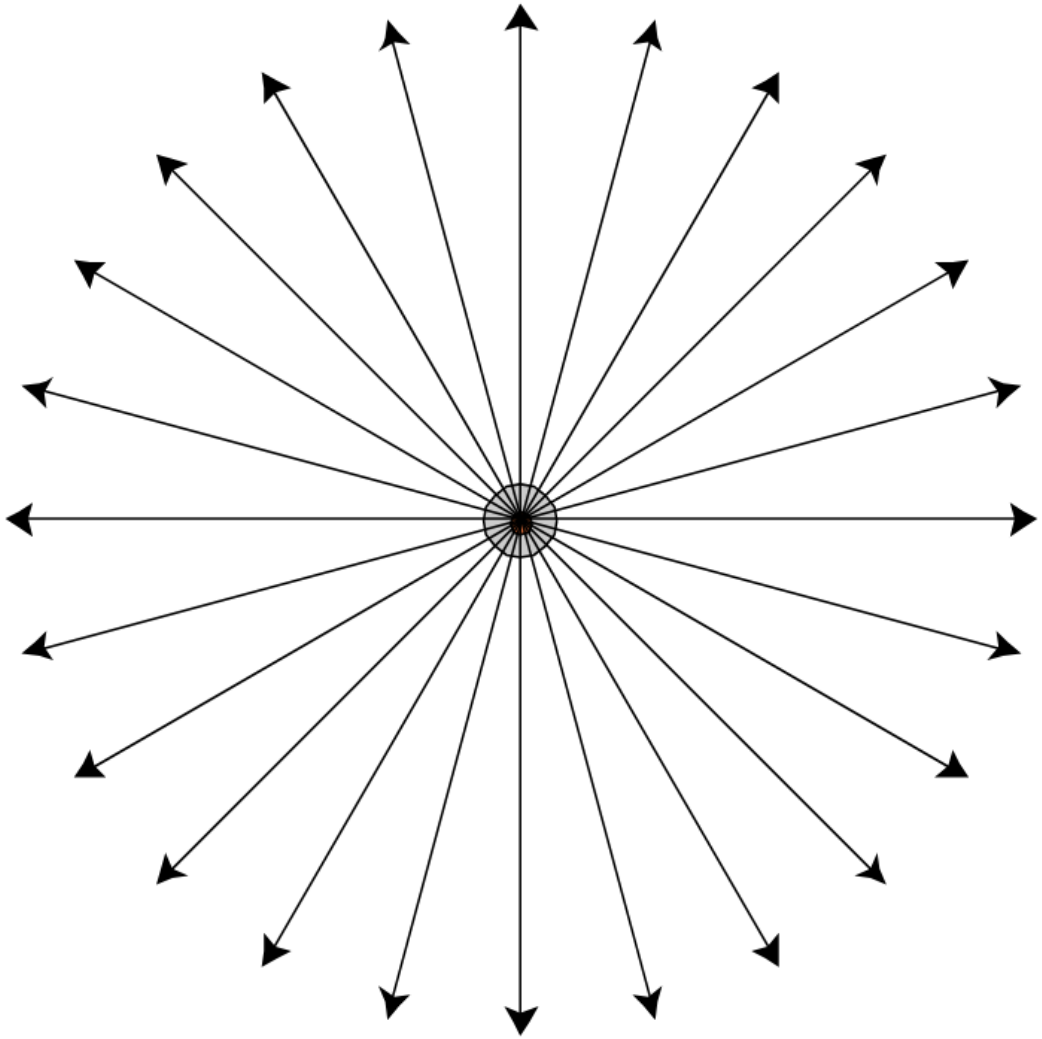
Infinitesimal solid angle:

$$d\omega = \frac{dA}{r^2} = \sin\theta\, d\theta d\phi$$

Total solid angle:

$$\Omega = \oint_{S^2} d\omega = \int_0^{\pi}\int_0^{2\pi} \sin\theta\, d\theta d\phi$$
$$= \int_{-1}^{1}\int_0^{2\pi} d\cos\theta\, d\phi = 4\pi$$

Isotropic Point Source

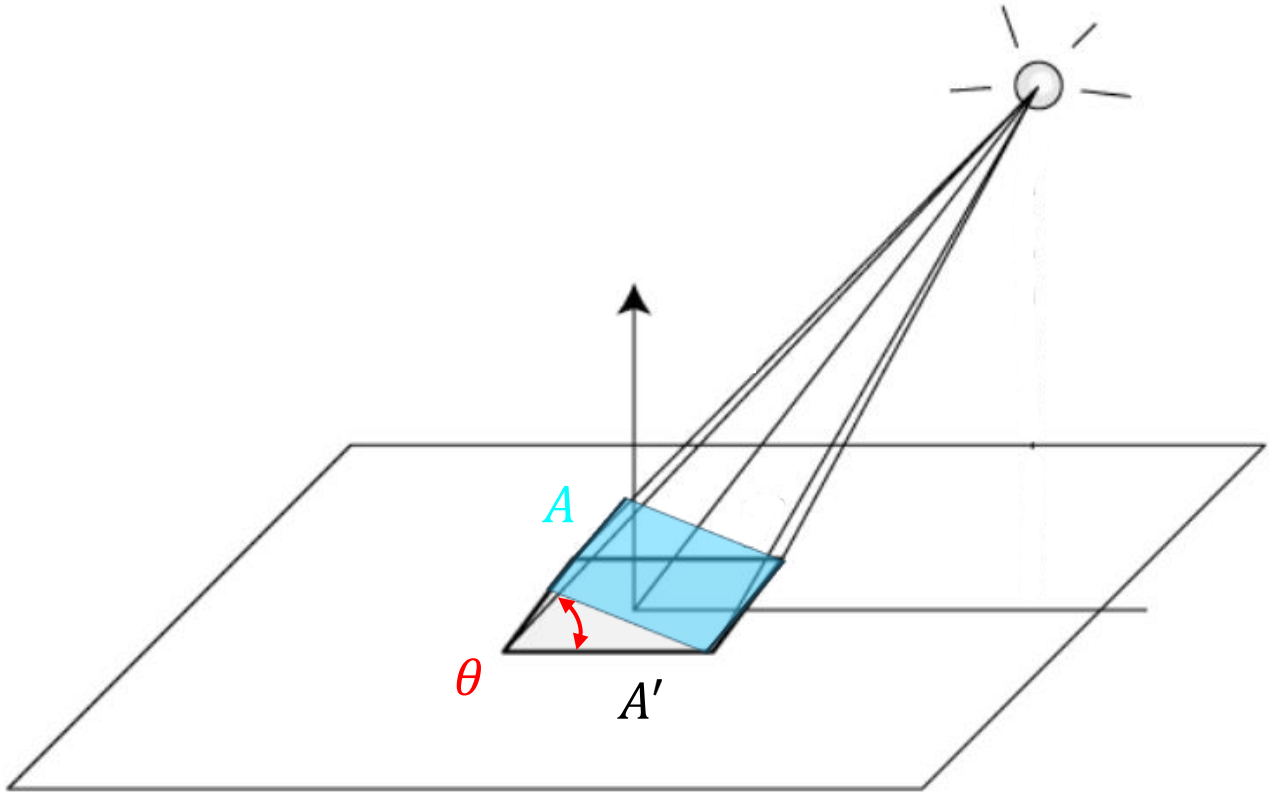


$$\Phi = \int_{S^2} I \, d\omega$$

$$= 4\pi I$$

$$I = \frac{\Phi}{4\pi}$$

Irradiance: The Power Per Unit Area Incident on a Surface



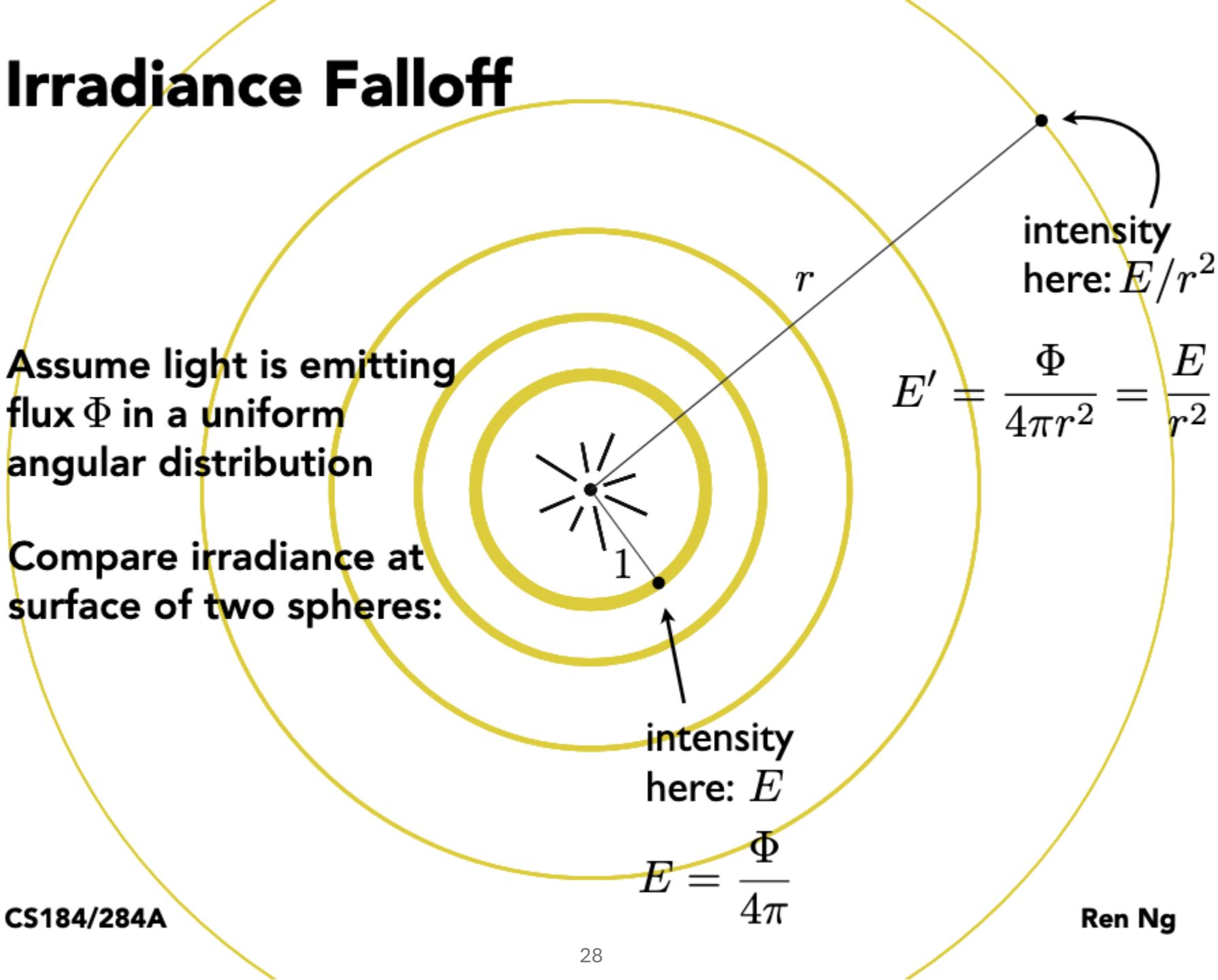
Irradiance

$$\mathrm{E}(x) \equiv \frac{d\Phi}{dA}$$

$$A = A' \cos\theta$$

$$E' = E \cos\theta$$

Irradiance Falloff



Assume light is emitting flux Φ in a uniform angular distribution

Compare irradiance at surface of two spheres:

intensity here: E

$$E = \frac{\Phi}{4\pi}$$

intensity here: E/r^2

$$E' = \frac{\Phi}{4\pi r^2} = \frac{E}{r^2}$$

Radiance: The Power Per Unit Area Per Unit Solid Angle Leaving/Received by a Surface

- Radiance

$$\mathrm{L}(x,\omega) \equiv \frac{dI(x,\omega)}{dA_{\perp}} = \frac{d^2\Phi(x,\omega)}{d\omega dA\cos\theta}$$

- For geometric optics, the fundamental carrier of light is a ray
- The amount of light traveling along a ray is radiance

Slide credit R. Ng

Incident Surface Radiance

$$L_i(x,\omega) \equiv \frac{dE(x,\omega)}{d\omega \cos\theta}$$

Exiting Surface Radiance

$$L_o(x,\omega) \equiv \frac{dI(x,\omega)}{dA \cos\theta}$$

Slide credit R. Ng

Irradiance from the Environment

Computing flux per unit area on surface, due to incoming light from all directions.

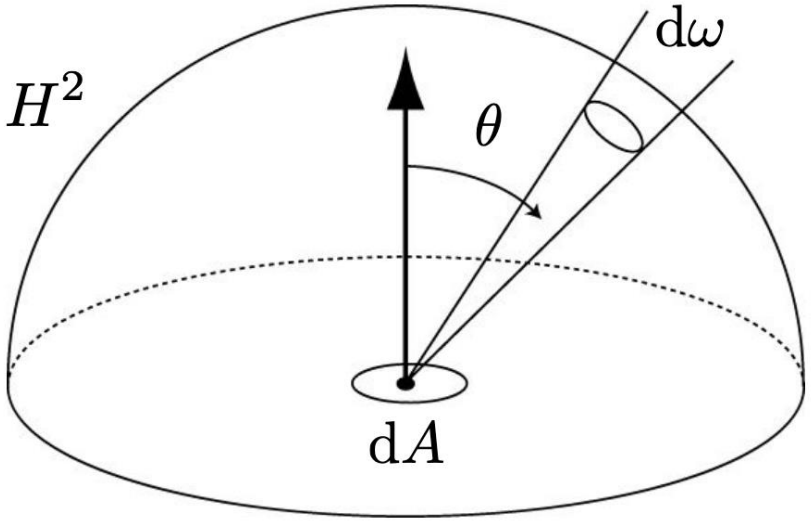
$$dE(\mathrm{p},\omega) = L_i(\mathrm{p},\omega)\cos\theta\,\mathrm{d}\omega$$

← **Contribution to irradiance from light arriving from direction ω**

$$E(\mathrm{p}) = \int_{H^2} L_i(\mathrm{p},\omega)\cos\theta\,\mathrm{d}\omega$$

Light meter

Hemisphere: H^2



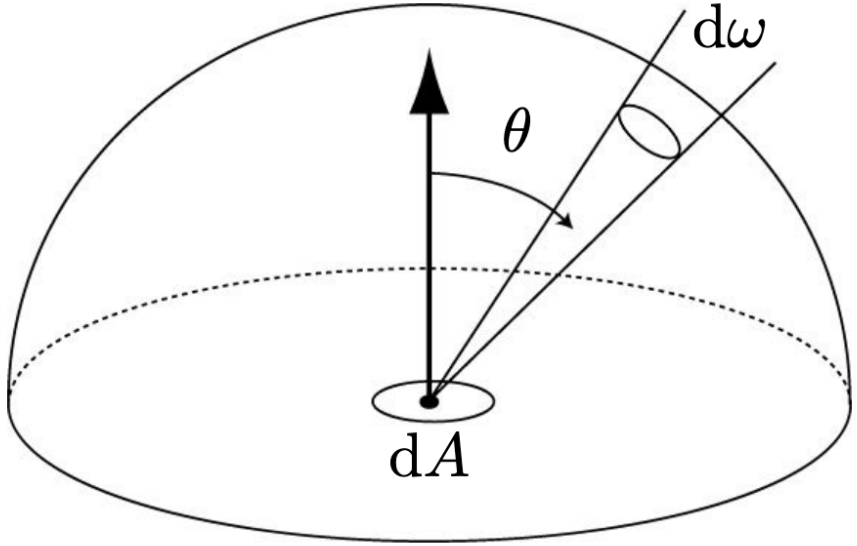
Irradiance from the Environment

$$E(\mathrm{p}) = \int_{H^2} L \cos\theta \, \mathrm{d}\omega$$

$$= L \int_0^{2\pi} \int_0^{\frac{\pi}{2}} \cos\theta \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi$$

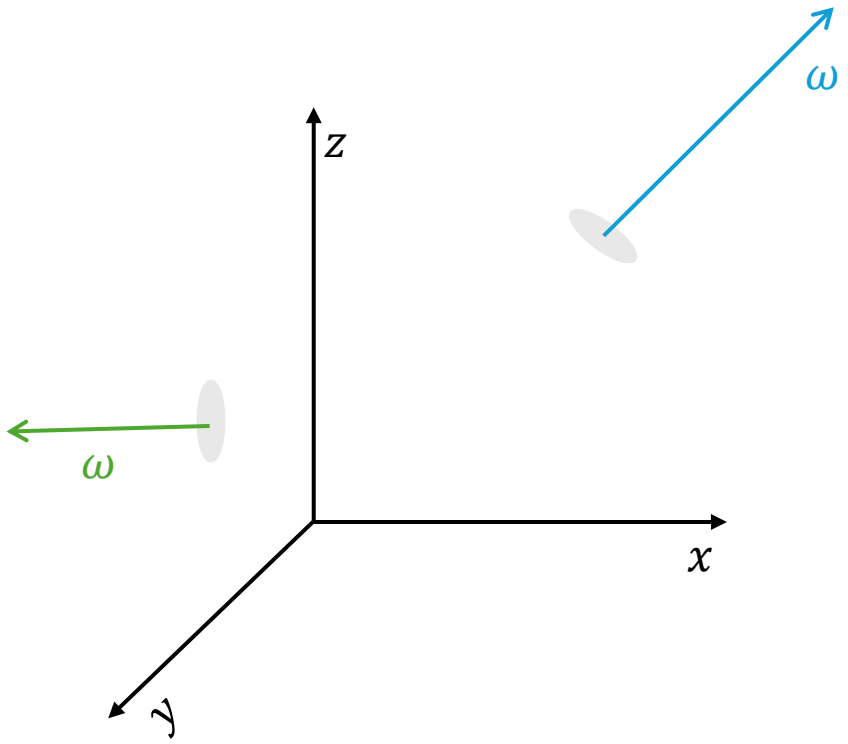
$$= L\,\pi$$

Note: integral of cosine over hemisphere is only 1/2 the area of the hemisphere.



Slide credit R. Ng

Light Field

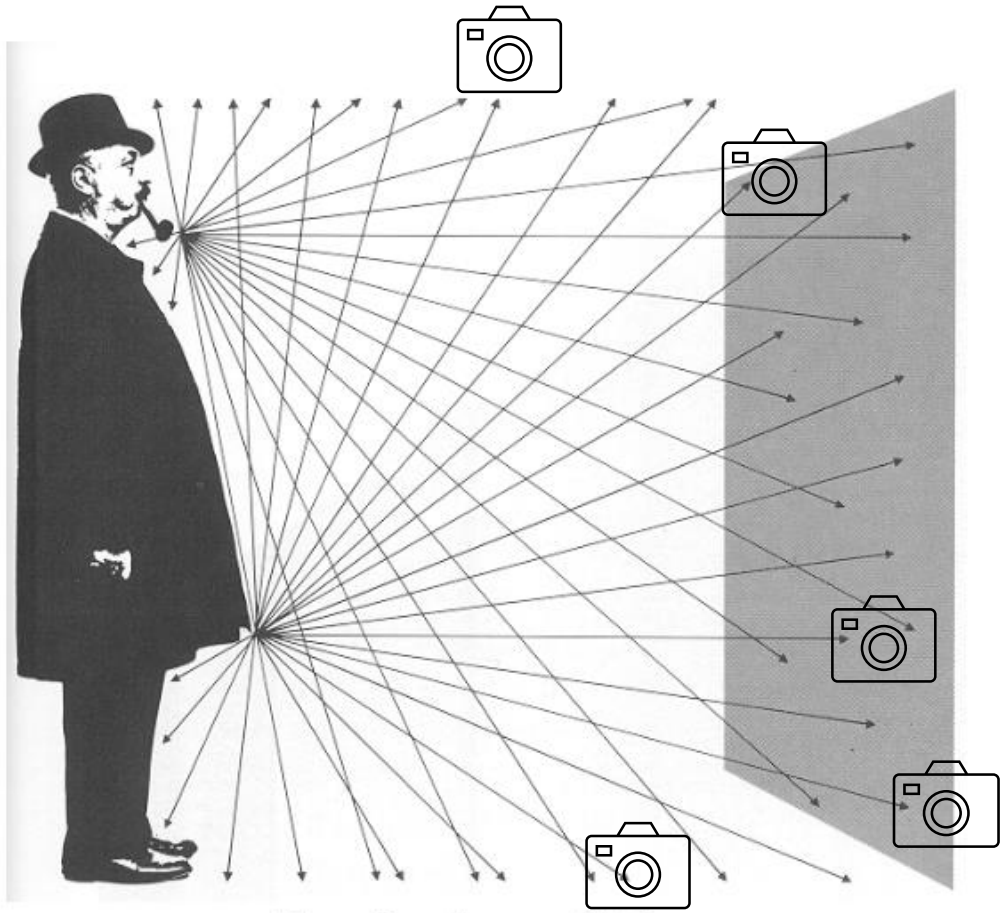


- Definition: The field radiance (luminance) at a point in space in a given direction is the power per unit solid angle per unit area perpendicular to the direction

Content

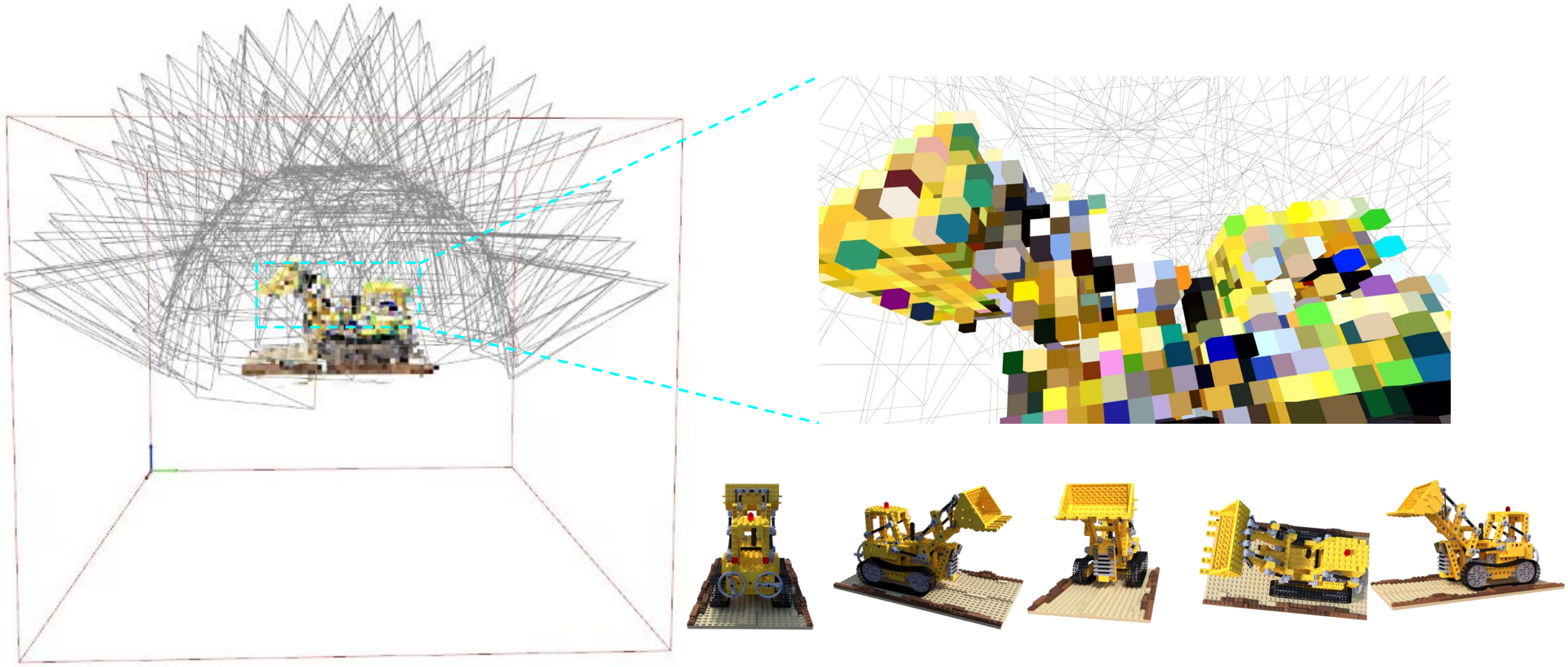
- Light Sources
- Light Measurement
- A mathematical model for image rendering
- Volume Rendering
  - Voxel
  - Nerf
  - Gaussian Splatting

Capture the Light Field and Reconstruct the Scene



Slide credit R. Ng

Volume Rendering from Multi-View Images

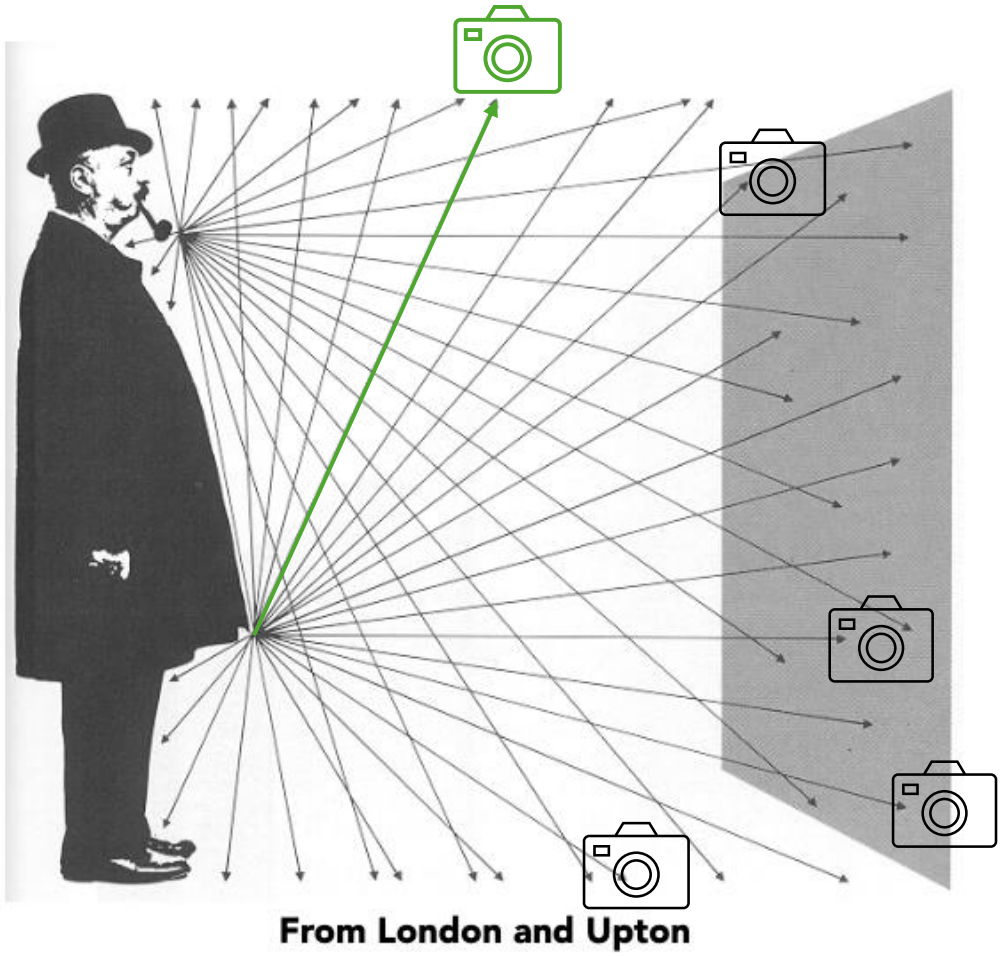


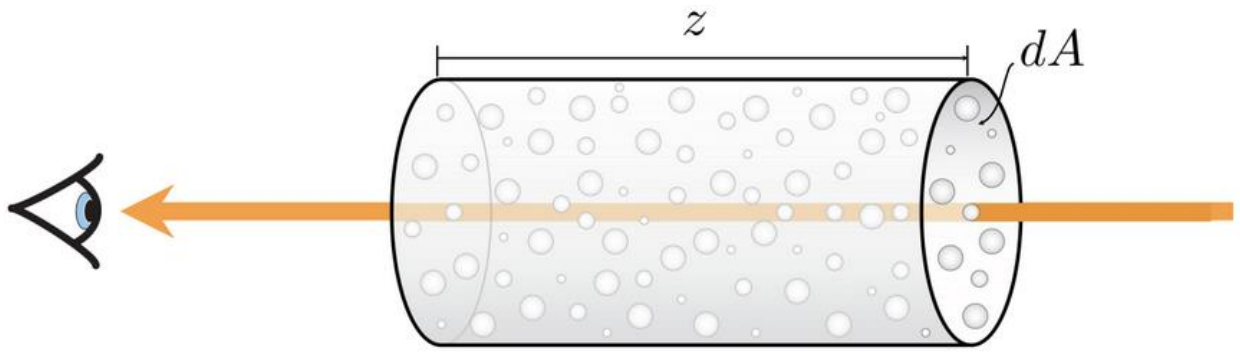
Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. Sun et al.

Volume Rendering vs. Multi-View Stereo

- Multi-view stereo obtains 3D by correspondence
- Volume rending obtains 3D by modeling reception of light

Volume Rendering a Ray



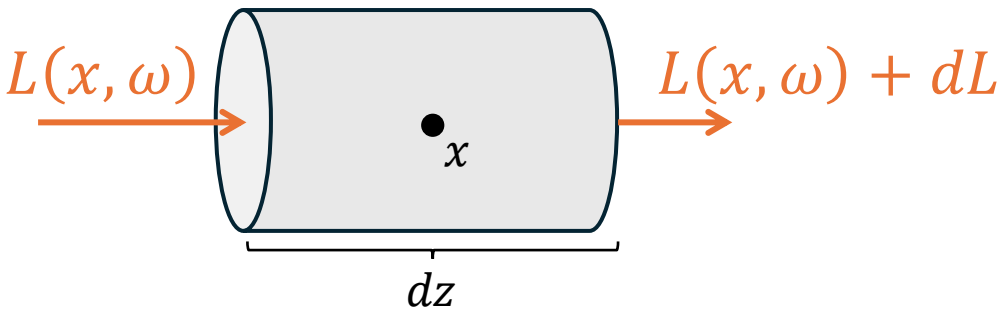


How much light is lost/gained along the differential beam due to interactions of light with the medium?

How does $L(x, \omega)$ vary along a ray?

Slide credit S. Tulsiani

Absorption

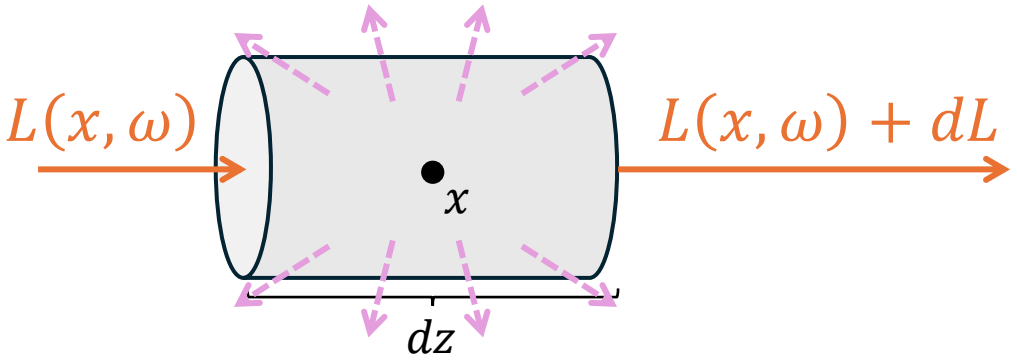


$$dL(x,\omega) = -\sigma_a(x)L(x,\omega)dz$$

Absorption across section σ_a :

- Probability of being absorbed per unit length
- Units: 1 / distance

Emission

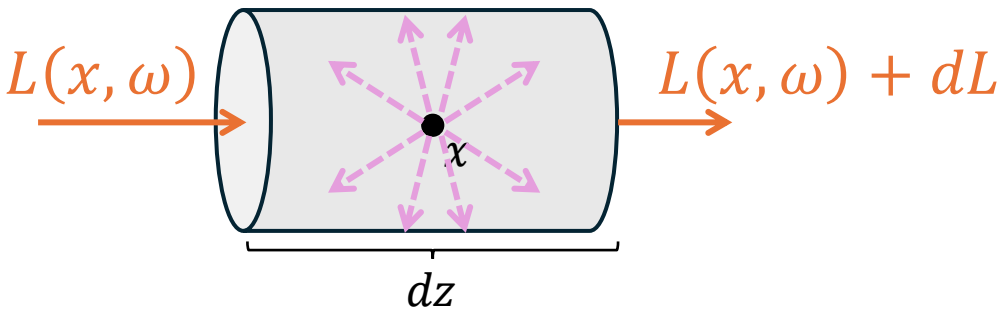


$$dL(x,\omega) = \sigma_a(x)L_e(x,\omega)dz$$

Emission across section σ_a :

- Probability of emitting light per unit length
- Units: 1 / distance

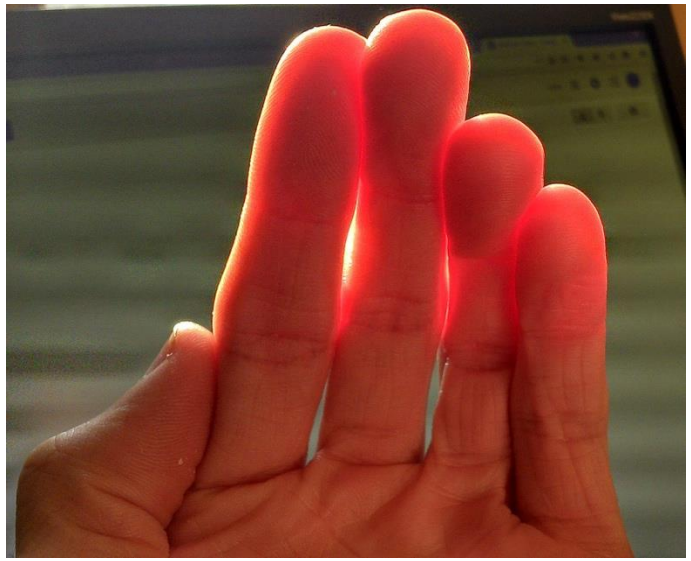
Out Scattering

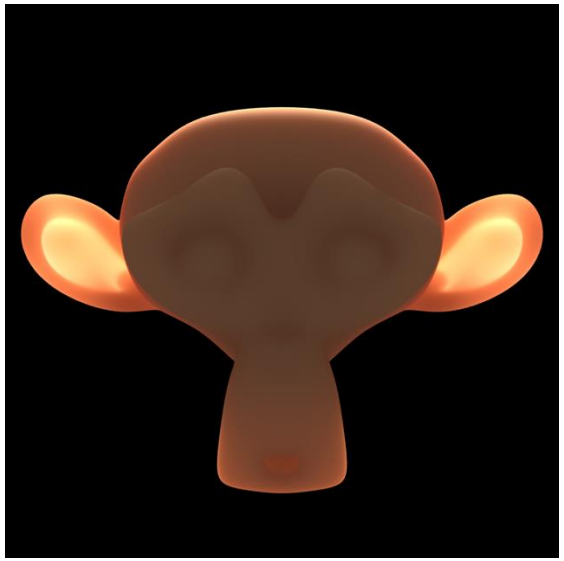


$$dL(x,\omega) = -\sigma_s(x)L(x,\omega)dz$$

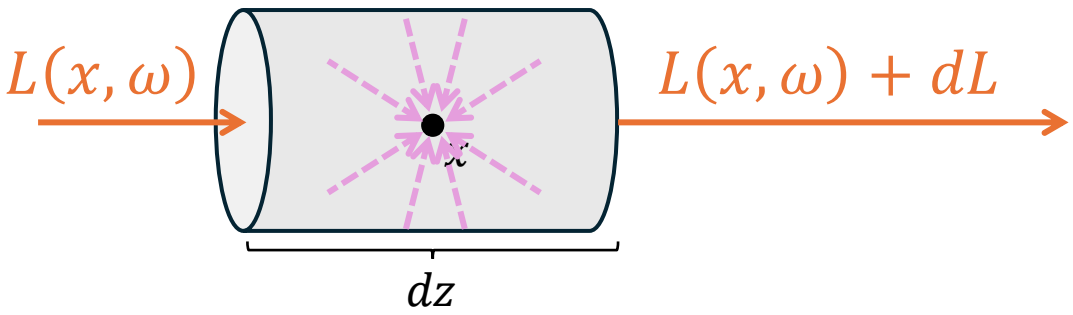
Scattering across section σ_s :

- Probability of being scattered per unit length
- Units: 1 / distance





In Scattering



$$dL(x,\omega) = \sigma_s(x) L_s(x,\omega) dz$$

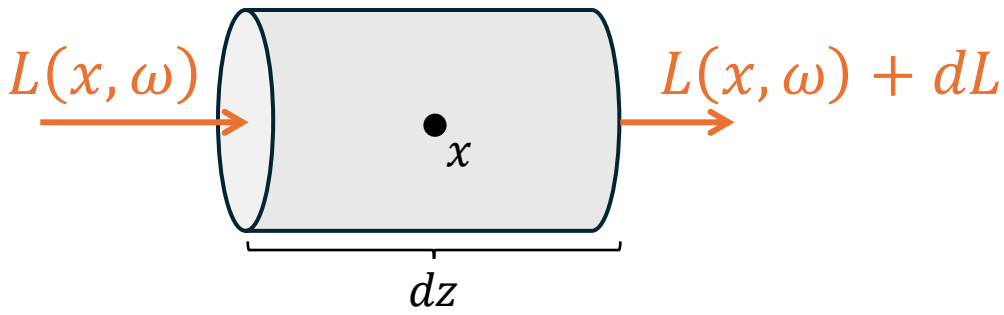
Scattering across section σ_s :

- Probability of gathering scattered light per unit length
- Units: 1 / distance

$$L_s(x,\omega) = \int_{S^2} p(\omega' \rightarrow \omega) L_i(x,\omega') d\omega'$$

Where $p(\omega' \rightarrow \omega)$ denotes the phase function and $L_i(x,\omega)$ also depend on other scattered light

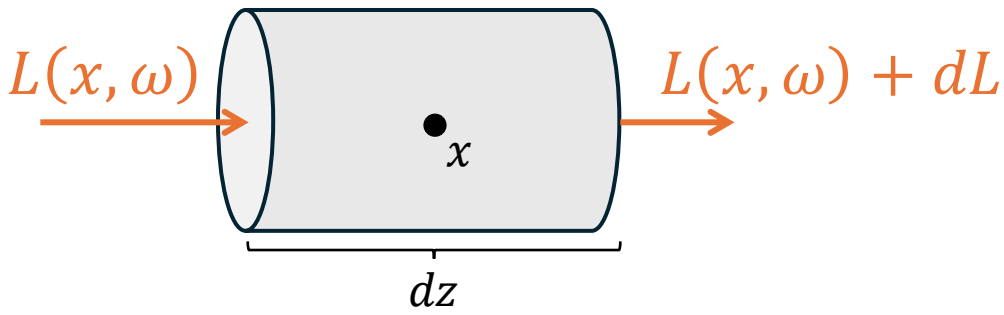
Radiative Transfer Equation



$$dL(x,\omega) = -\sigma_a(x)L(x,\omega)dz - \sigma_s(x)L(x,\omega)dz + \sigma_a(x)L_e(x,\omega)dz + \sigma_s(x)L_s(x,\omega)dz$$

Absorption | Out scattering | Emission | In scattering

Radiative Transfer Equation



$$dL(x,\omega) = -\sigma_a(x)L(x,\omega)dz - \sigma_s(x)L(x,\omega)dz + \sigma_a(x)L_e(x,\omega)dz + \sigma_s(x)L_s(x,\omega)dz$$
$$= -\sigma_a(x)L(x,\omega)dz + \sigma_a(x)L'(x,\omega)dz$$
$$= \sigma(x)\big(L'(x,\omega) - L(x,\omega)\big)dz$$

$$L'(x,\omega) = L_e(x,\omega) + \frac{\sigma_s}{\sigma_a}\big(L_s(x,\omega) - L(x,\omega)\big)$$

- Model absorption-only case
- Allow emission

Absorption-only Volume Rendering

- Volume rendering in a heterogeneous medium:

$$dL(x,\omega) = -\sigma(x)L(x,\omega)dz \Rightarrow L(x_0 + \omega z, \omega) = e^{-\int_{t=0}^{z}\sigma(x+\omega t)dt}L(x_0,\omega)$$

- Homogeneous medium: $\sigma(x) = \sigma$
- Volume rendering in a homogeneous medium:

$$dL(x,\omega) = -\sigma L(x,\omega)dz \Rightarrow L(x_0 + \omega z, \omega) = e^{-\sigma z}L(x_0,\omega)$$

Transmittance

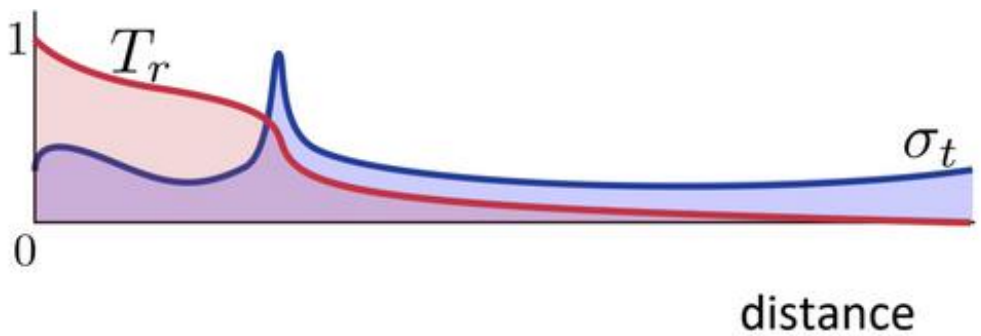
$T(\mathbf{x}, \mathbf{y})$

What fraction of radiance at **x** in direction of **y**, reaches **y** ?
(along a straight line under absorption-only model)

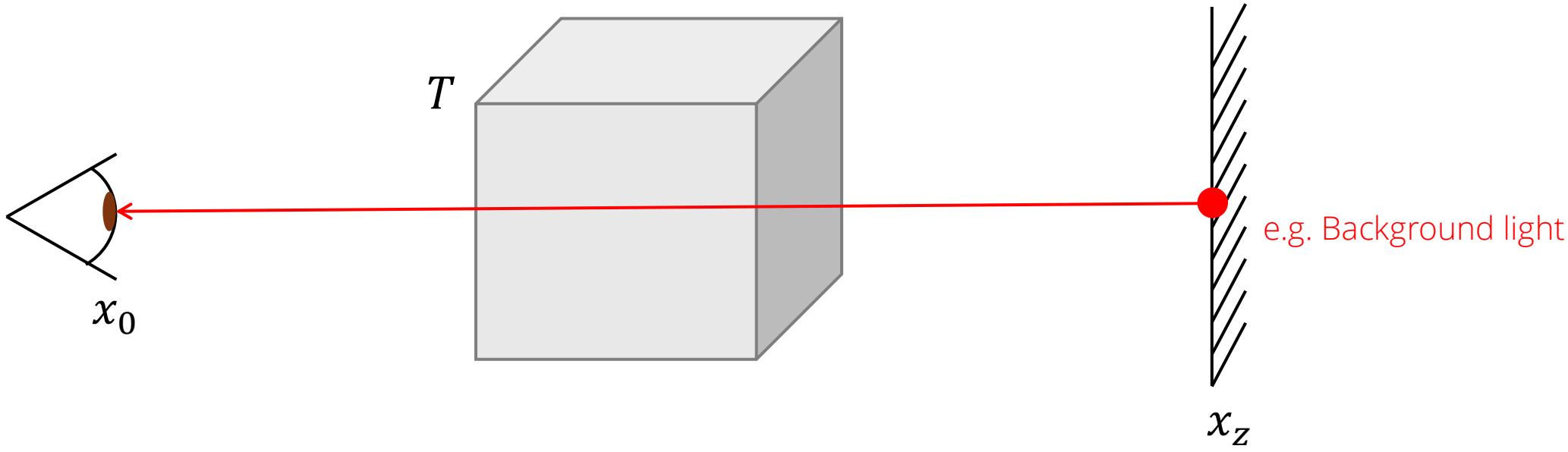
Q: should this be symmetric?

Homogeneous Medium: $e^{-\sigma\|\mathbf{x}-\mathbf{y}\|}$

Heterogeneous Medium: $e^{-\int_{t=0}^{\|\mathbf{x}-\mathbf{y}\|}\sigma(\mathbf{x}+\omega\mathbf{t})}$

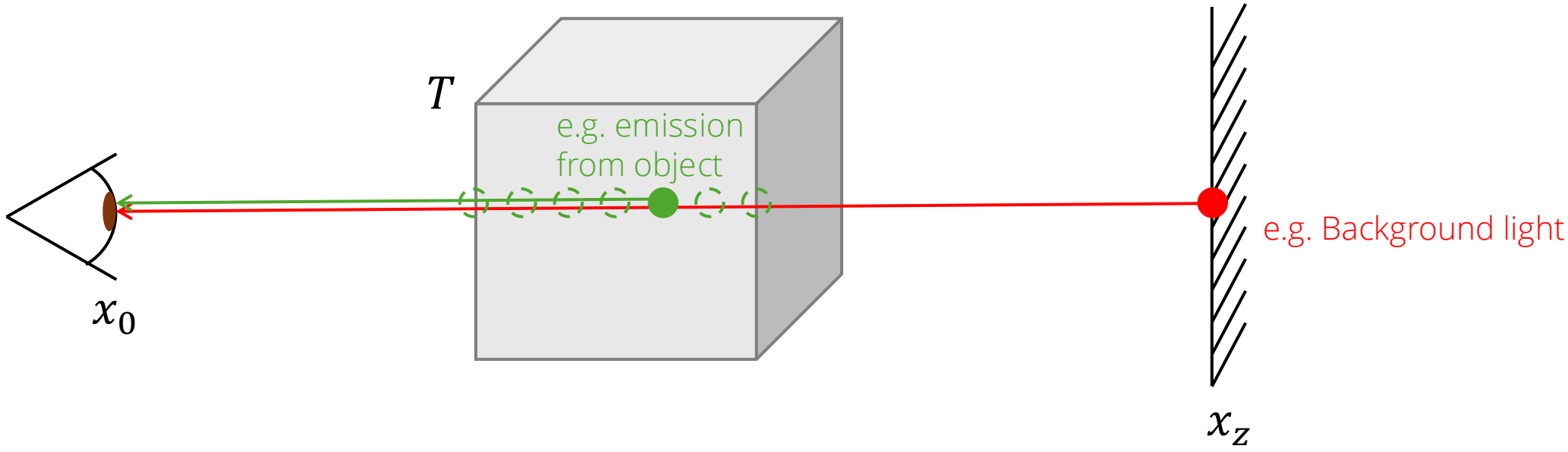


Volume Rendering



$$L(x, \omega) = T(x_0, x_z) L(x_z, \omega)$$

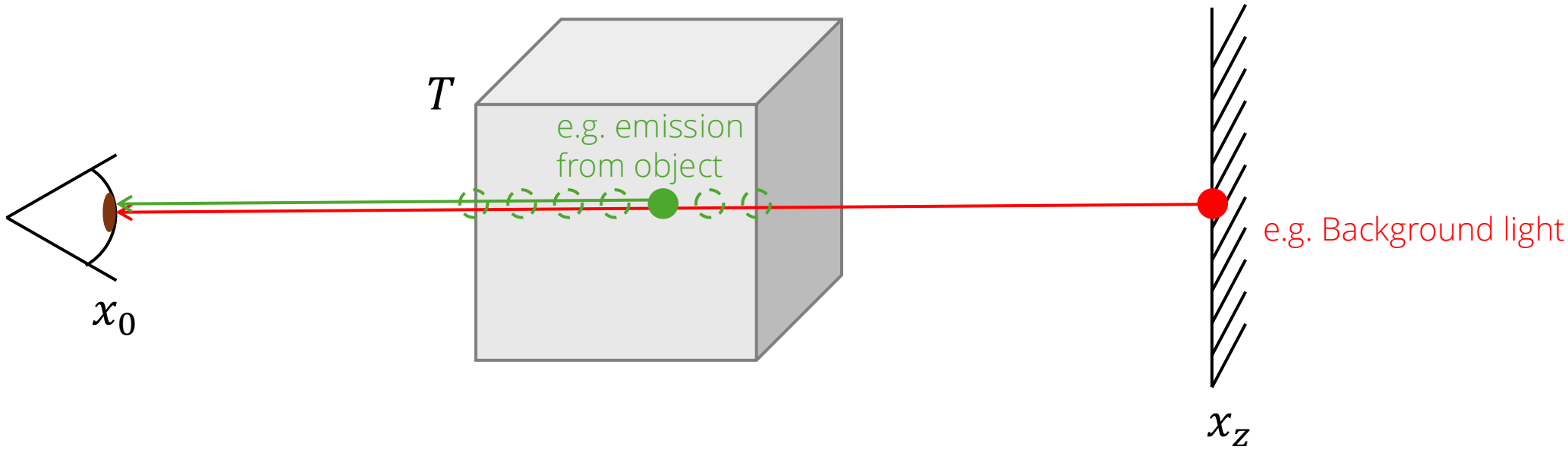
Volume Rendering



$$L(x,\omega) = T(x_0, x_z)L(x_z,\omega)$$

$$+ \int_0^z T(x_0, x_t)\sigma_t L_e(x_t,\omega)dt$$

Volume Rendering in a Homogeneous Emitting Medium



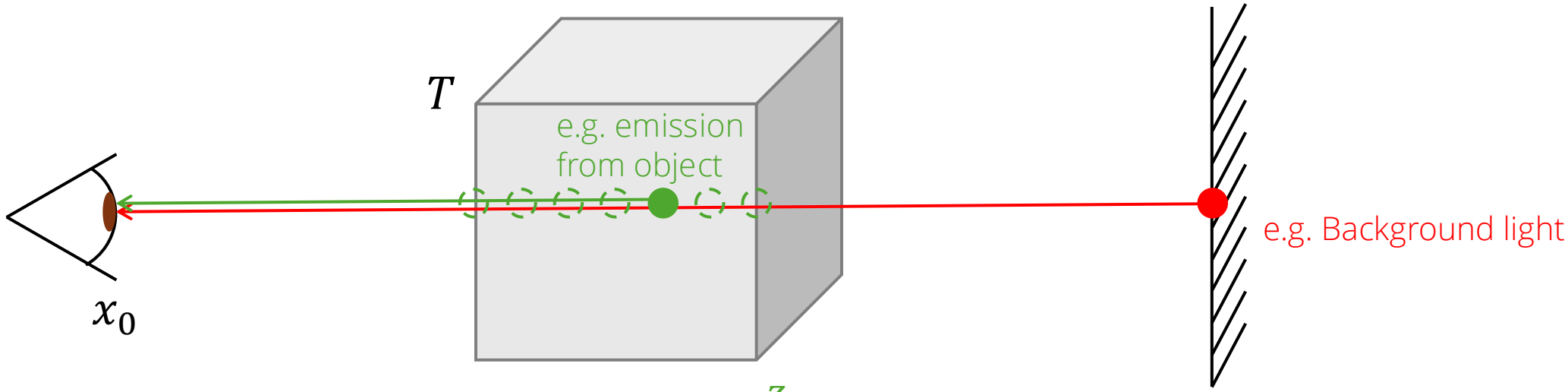
$$L(x,\omega) = e^{-\sigma z} L(x_z,\omega) + \int_0^z e^{-\sigma t} \sigma L_e(x_t,\omega) dt$$

Homogeneous emitting medium:

$C \equiv L_e(x_t,\omega)$

$T(x,y) = e^{-\sigma \|x-y\|}$

Volume Rendering in a Homogeneous Emitting Medium



$$L(x,\omega) = e^{-\sigma z} L(x_z,\omega) + \int_0^z e^{-\sigma t}\sigma L_e(x_t,\omega)dt$$

$$= e^{-\sigma z} L(x_z,\omega) + \int_0^z e^{-\sigma t}\sigma C dt \qquad \int \lambda e^{-\lambda x} dx = -e^{\lambda x} + c$$

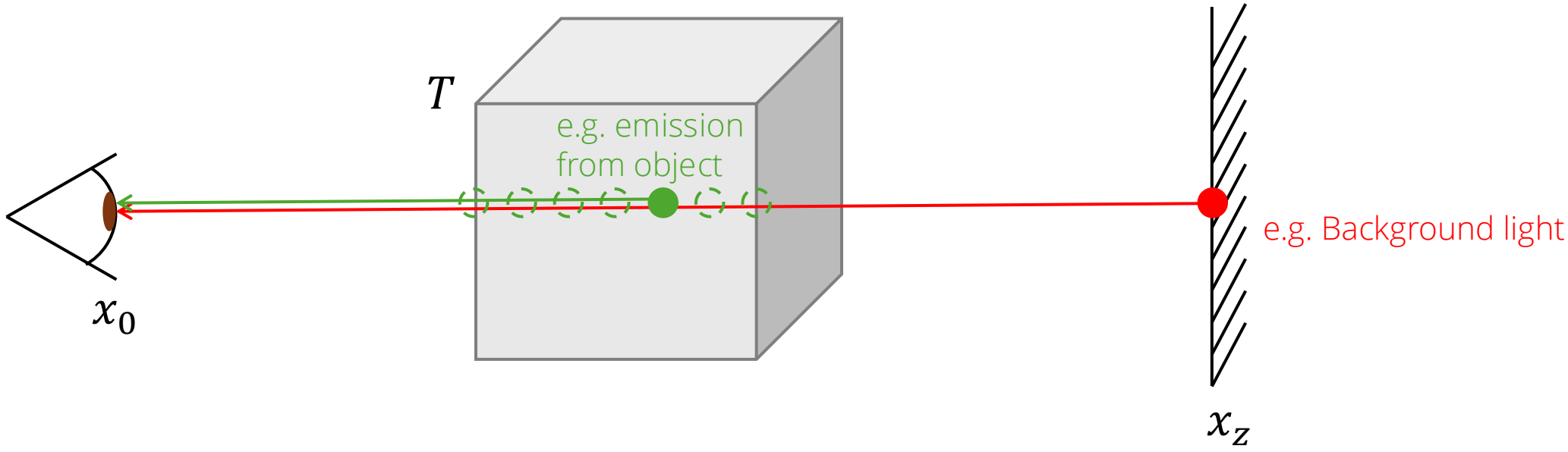
$$= e^{-\sigma z} L(x_z,\omega) + (1 - e^{-\sigma z})C$$

Homogeneous emitting medium:

$C \equiv L_e(x_t,\omega)$

$T(x,y) = e^{-\sigma\|x-y\|}$

Volume Rendering in a Heterogeneous Medium

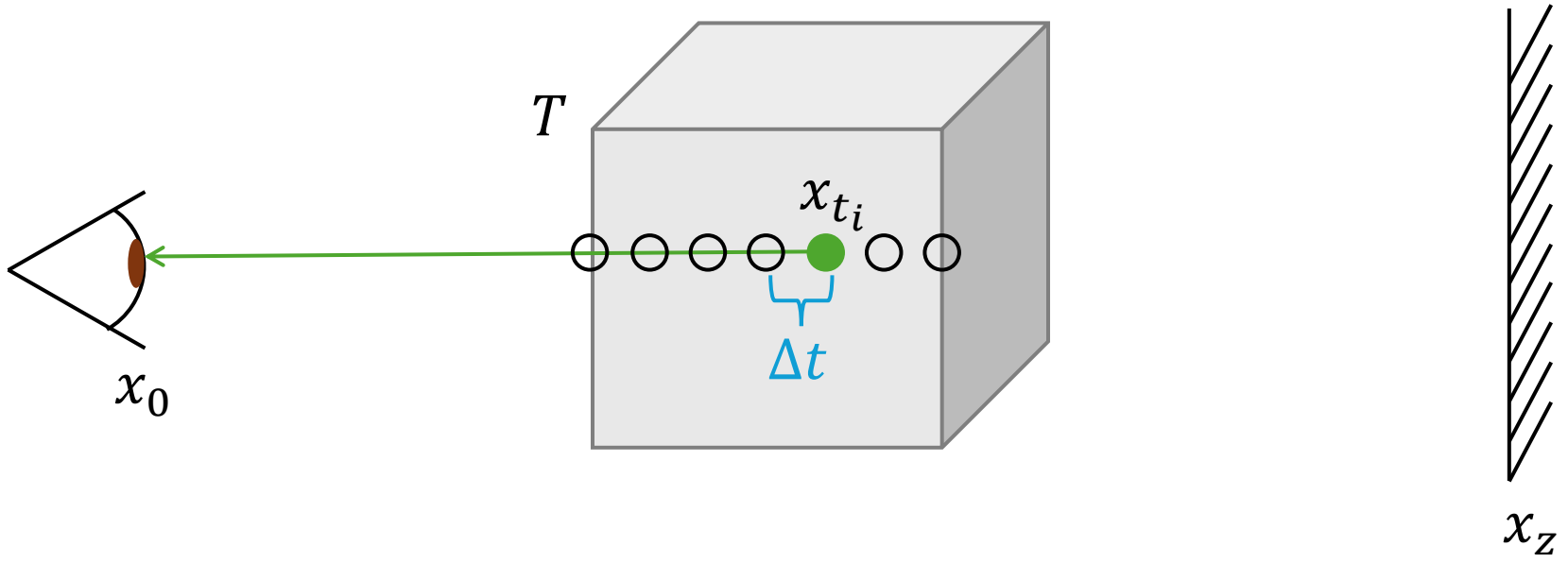


$$L(x,\omega) = T(x_0, x_z)L(x_z,\omega)$$

$$+ \int_0^z T(x_0, x_t)\sigma_t L_e(x_t,\omega)dt$$

Very difficult to solve!

Let's Ignore Background Light But Emission from the Medium

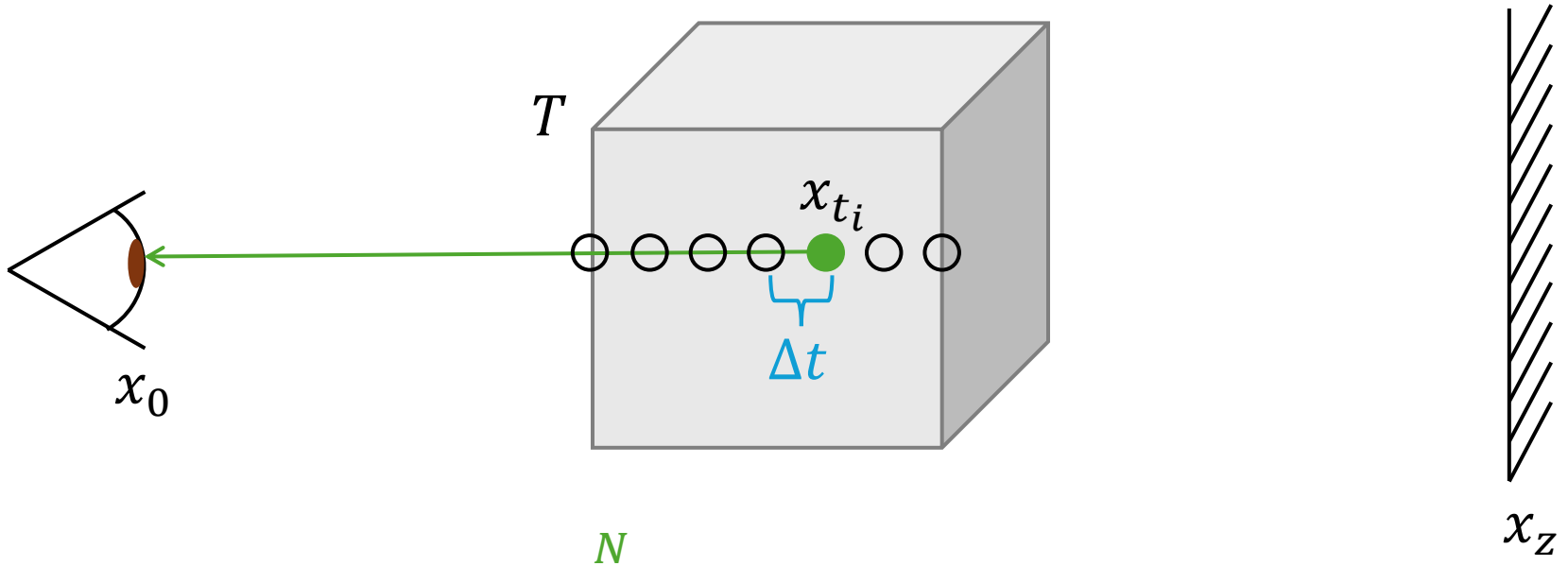


$$L(x,\omega) = \int_0^z T(x_0, x_t)\sigma_t L_e(x_t,\omega)dt$$

Approximate with discrete sum

$$\approx \sum_{i=1}^{N} T(x_0, x_{t_i})\left(emission_{t_{i+1}}^{t_i}\right)$$

Let's Ignore Background Light But Emission from the Medium

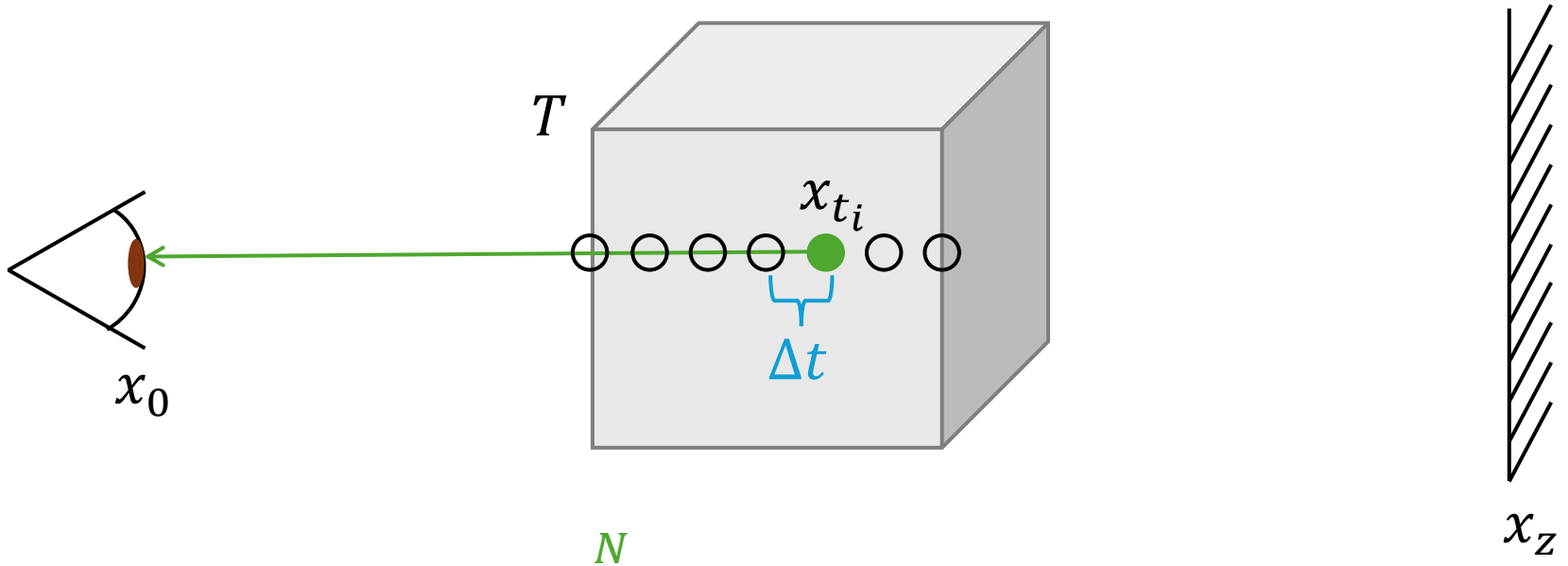


$$L(x,\omega) \approx \sum_{i=1}^{N} T(x_0, x_{t_i}) \left(radiance_{t_{i+1}}^{t_i} \right)$$

$$= \sum_{i=1}^{N} T(x_0, x_{t_i}) (1 - e^{-\sigma_{t_i} \Delta t}) L_e(x_{t_i}, \omega)$$

Assume a homogeneous emitting medium between t_i and t_{i+1}

Let's Ignore Background Light But Emission from the Medium



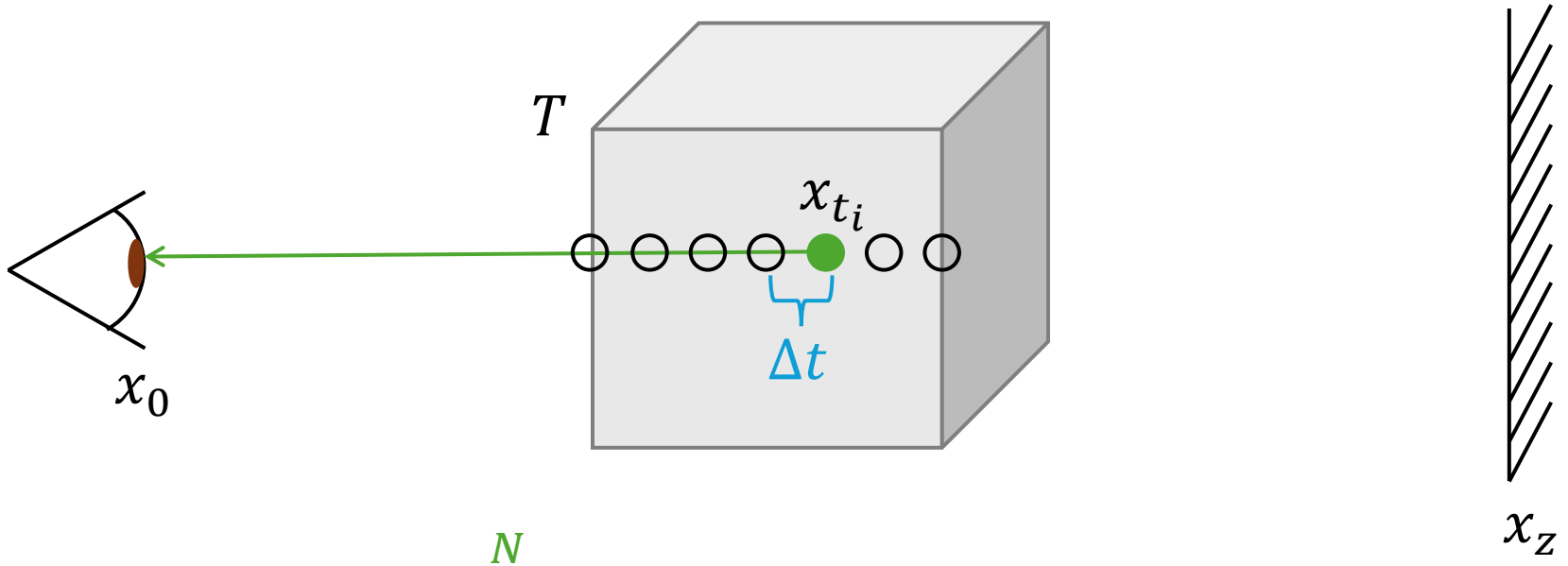
$$L(x,\omega) \approx \sum_{i=1}^{N} T(x_0, x_{t_i}) \left(radiance_{t_{i+1}}^{t_i} \right)$$

$$= \sum_{i=1}^{N} T(x_0, x_{t_i}) \left(1 - e^{-\sigma_{t_i} \Delta t}\right) L_e(x_{t_i}, \omega)$$

This is unclear

Assume a homogeneous emitting medium between t_i and t_{i+1}

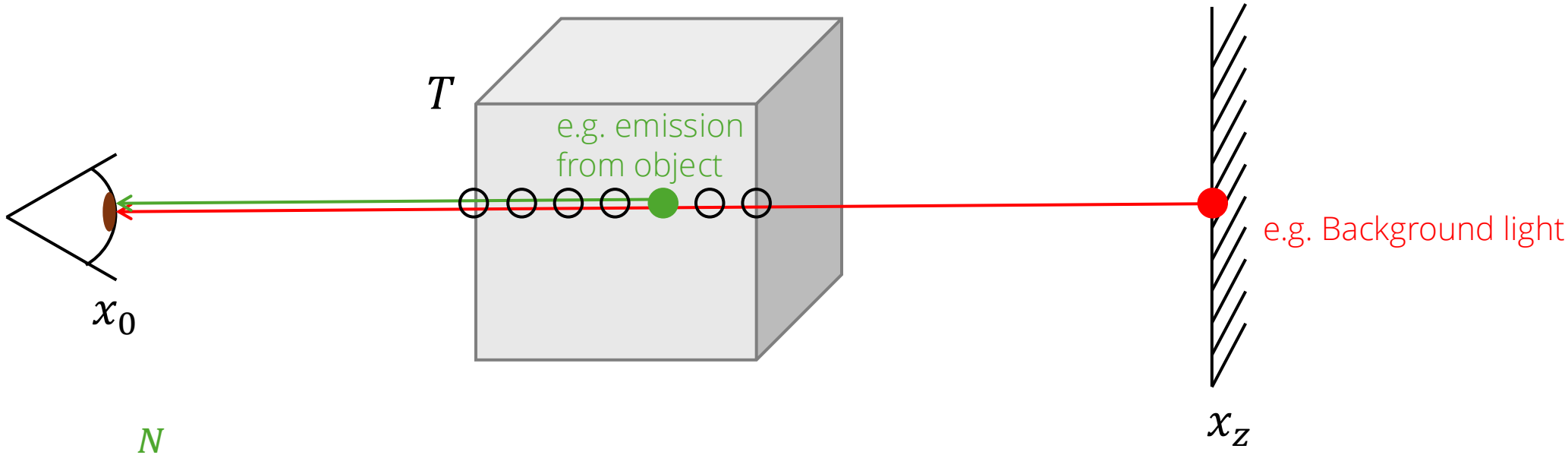
Let's Ignore Background Light But Emission from the Medium



$$L(x,\omega) \approx \sum_{i=1}^{N} T(x_0, x_{t_i})(1 - e^{-\sigma_{t_i}\Delta t}) L_e(x_{t_i}, \omega)$$

$$T(x_0, x_{t_i}) = e^{-\int_0^{t_i} \sigma_t dt} = e^{-\int_0^{t_{i-1}} \sigma_t dt} e^{-\int_{t_{i-1}}^{t_i} \sigma_t dt} = T(x_0, x_{t_i-1}) e^{-\sigma_{t_{i-1}}\Delta t}$$

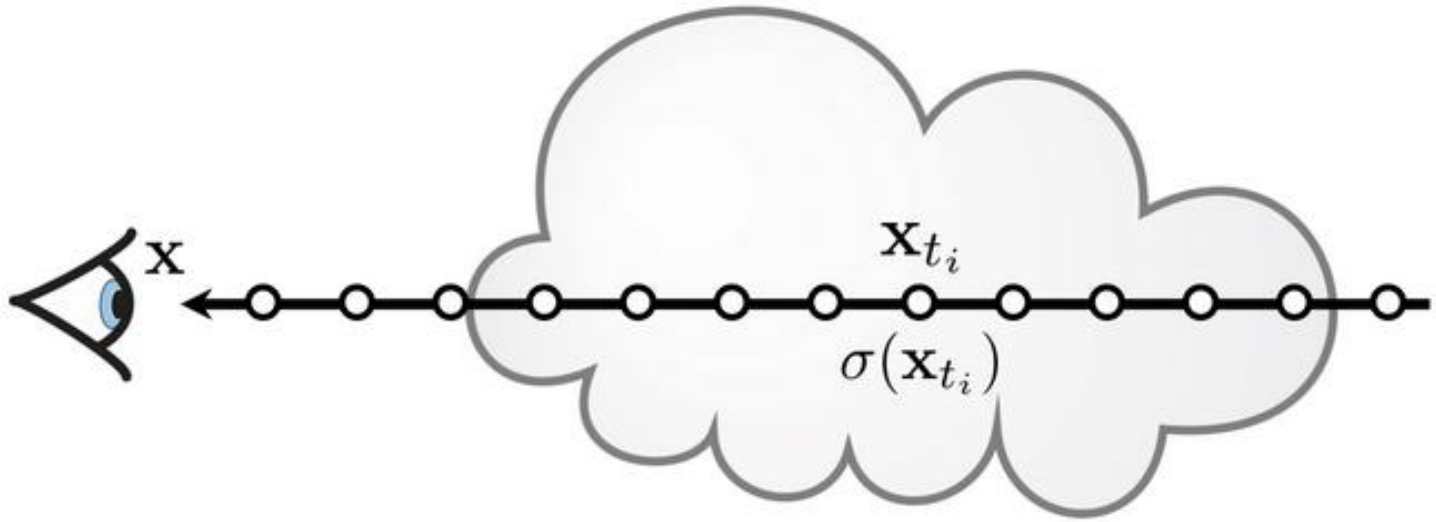
Volume Rendering in a Homogeneous Emitting Medium



$$L(x,\omega) \approx \sum_{i=1}^{N} T(x_0, x_{t_i})(1 - e^{-\sigma_{t_i}\Delta t}) L_e(x_{t_i}, \omega) + T(x_0, x_{t_{N+1}}) L(x_z, \omega)$$

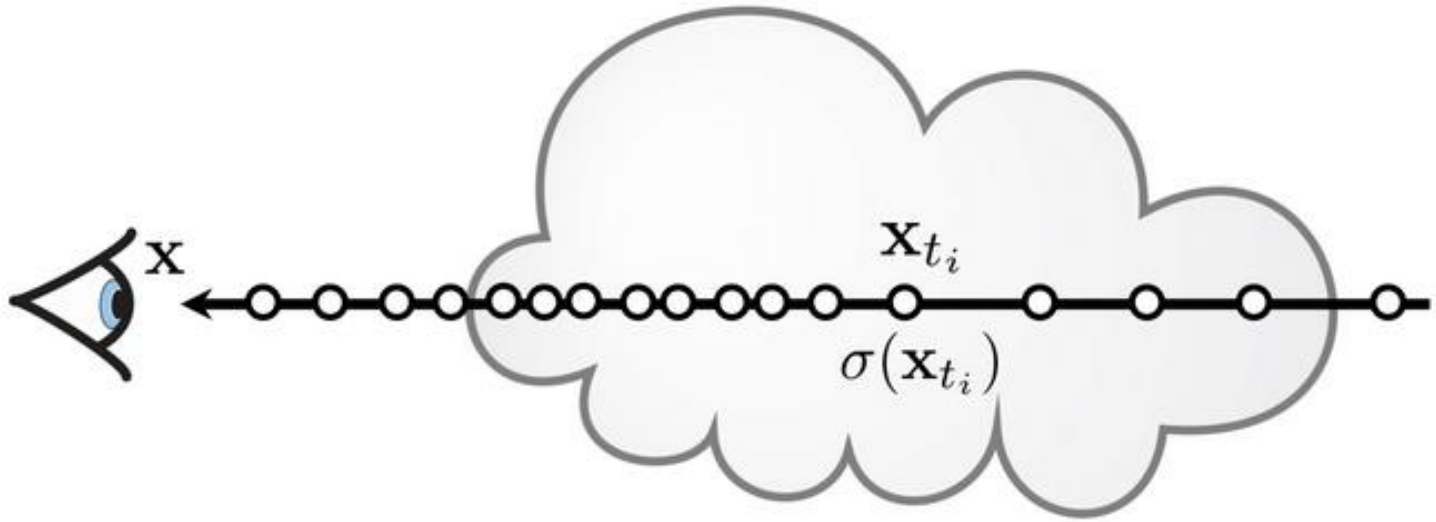
$$T(x_0, x_{t_i}) = e^{-\int_0^{t_i} \sigma_t dt} = e^{-\int_0^{t_{i-1}} \sigma_t dt} e^{-\int_{t_{i-1}}^{t_i} \sigma_t dt} = T(x_0, x_{t_i - 1}) e^{-\sigma_{t_{i-1}}\Delta t}$$

Computational Volume Rendering: Ray Marching



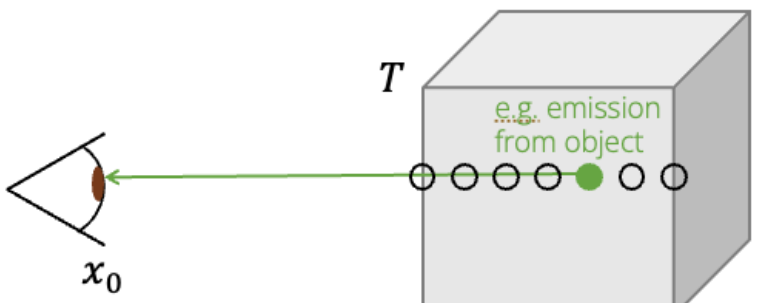
1. Draw uniform samples along a ray (N segments, or N+1 points)
2. Compute transmittance between camera and each sample
3. Aggregate contributions across segments to get overall radiance (color)

Computational Volume Rendering: Ray Marching



1. Draw **non-uniform** samples along a ray
2. Compute transmittance between camera and each sample
3. Aggregate contributions across segments to get overall radiance (color)

Computational Volume Rendering: Ray Marching

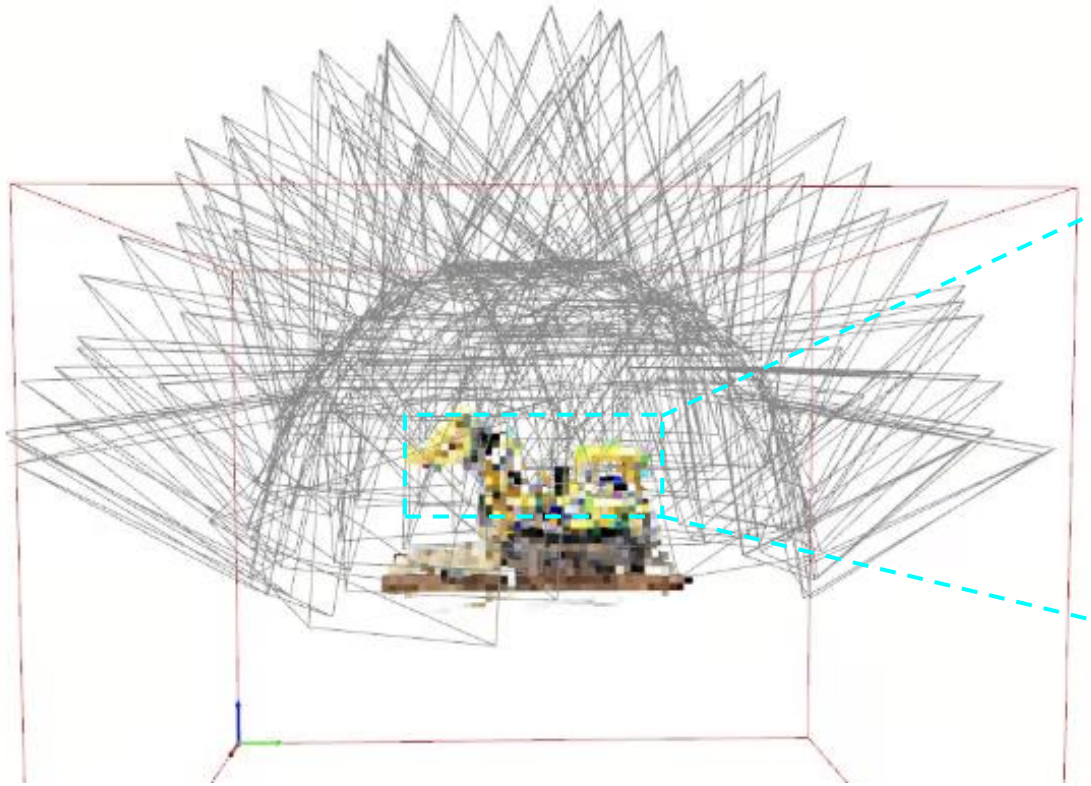


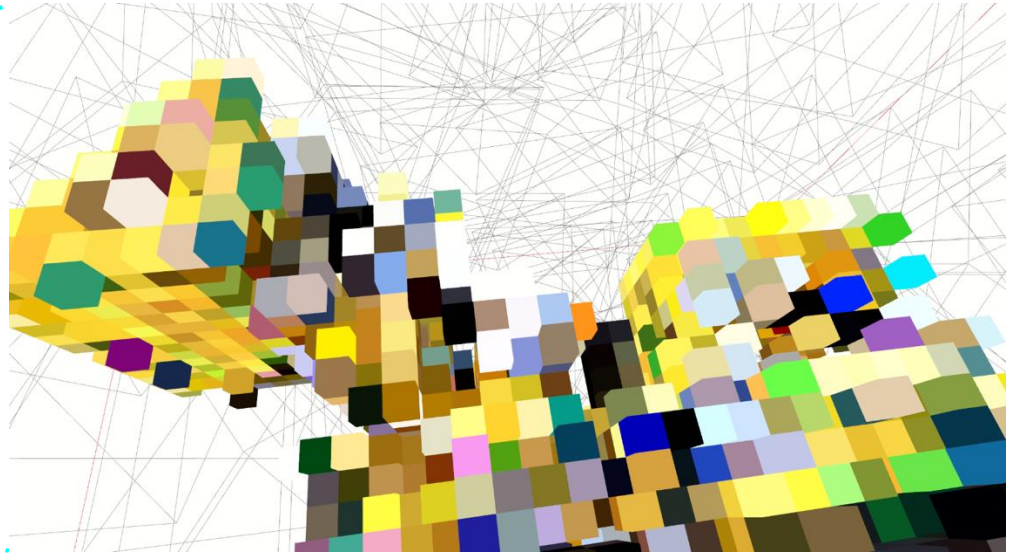
$$L(x,\omega) \approx \sum_{i=1}^{N} T(x_0, x_{t_i})\left(1 - e^{-\sigma_{t_i}\Delta t}\right) L_e(x_{t_i}, \omega)$$

$$T(x_0, x_{t_i}) = T(x_0, x_{t_i-1}) e^{-\sigma_{t_i-1}\Delta t}$$

- We can render any ray through the medium, if we know at each point t_i:
  - Absorption coefficient σ_{t_i}
  - Emitted light $L_e(x_{t_i}, \omega)$

How to Render Volumes from Multi-View Images?

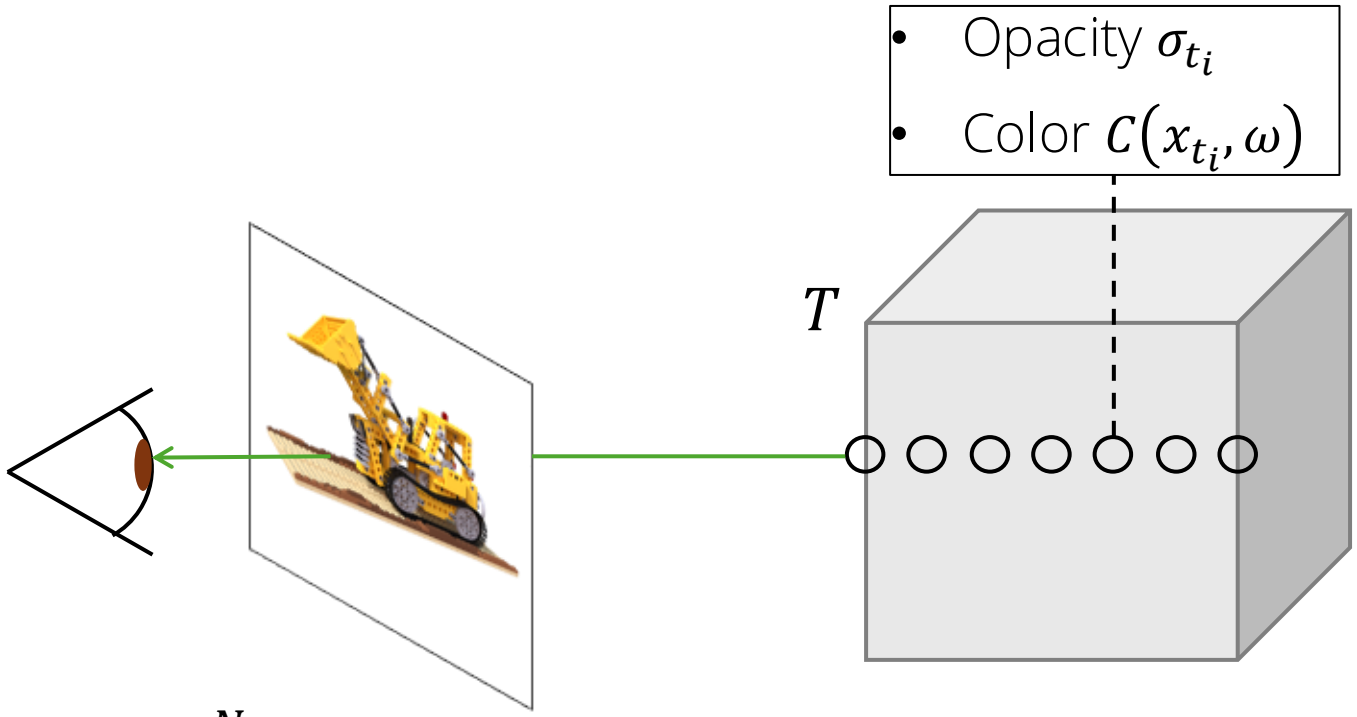


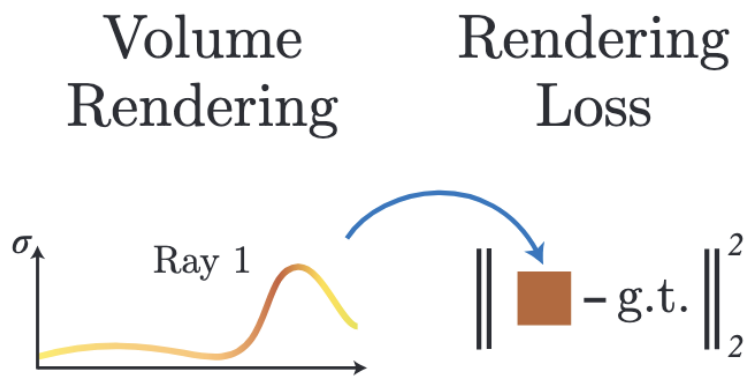


- Given a set of multi-view images, we want to learn at each point:
  - Opacity (Absorption coefficient σ_{t_i})
  - Color (Emitted light $L_e(x_{t_i}, \omega)$)
- Loss: image reconstruction

Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. Sun et al.

How to Render Volumes from Multi-View Images?

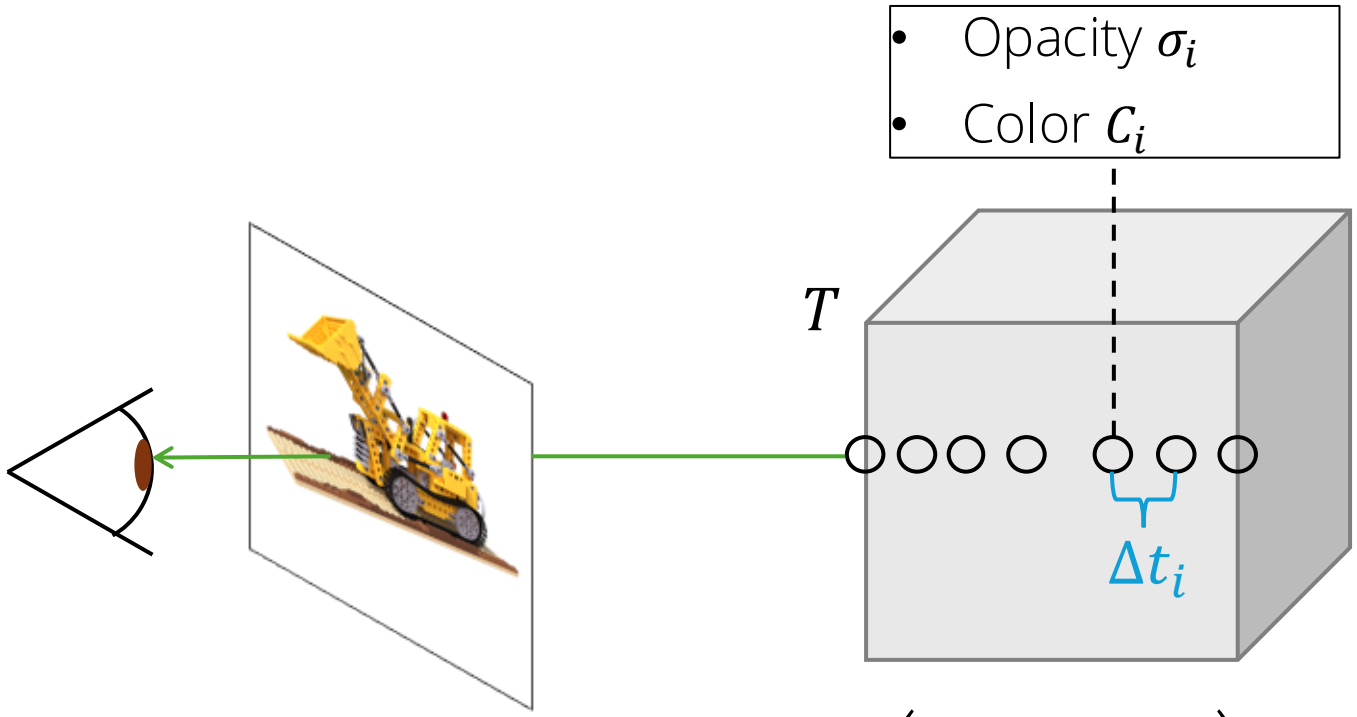


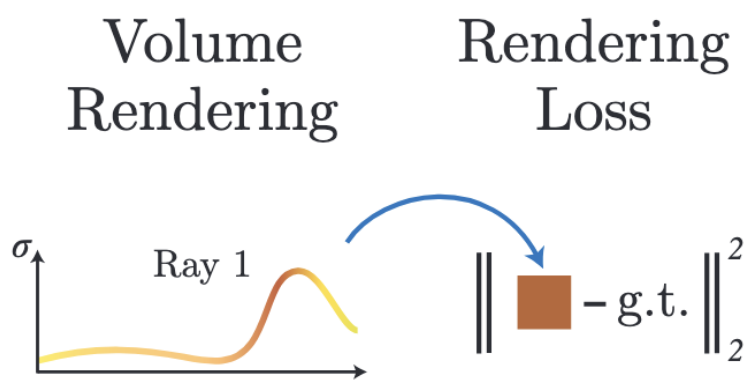


$$C(x,\omega) \approx \sum_{i=1}^{N} T(x_0, x_{t_i})\left(1 - e^{-\sigma_{t_i}\Delta t}\right) C(x_{t_i}, \omega)\,, \text{where } T(x_0, x_{t_i}) = T(x_0, x_{t_{i-1}})\, e^{-\sigma_{t_{i-1}}\Delta t}$$

Learning objective: $\min_{\sigma_{t_i}, C(x_{t_i},\omega)} \|C(x,\omega) - \text{g.t.}\|$

How to Render Volumes from Multi-View Images?



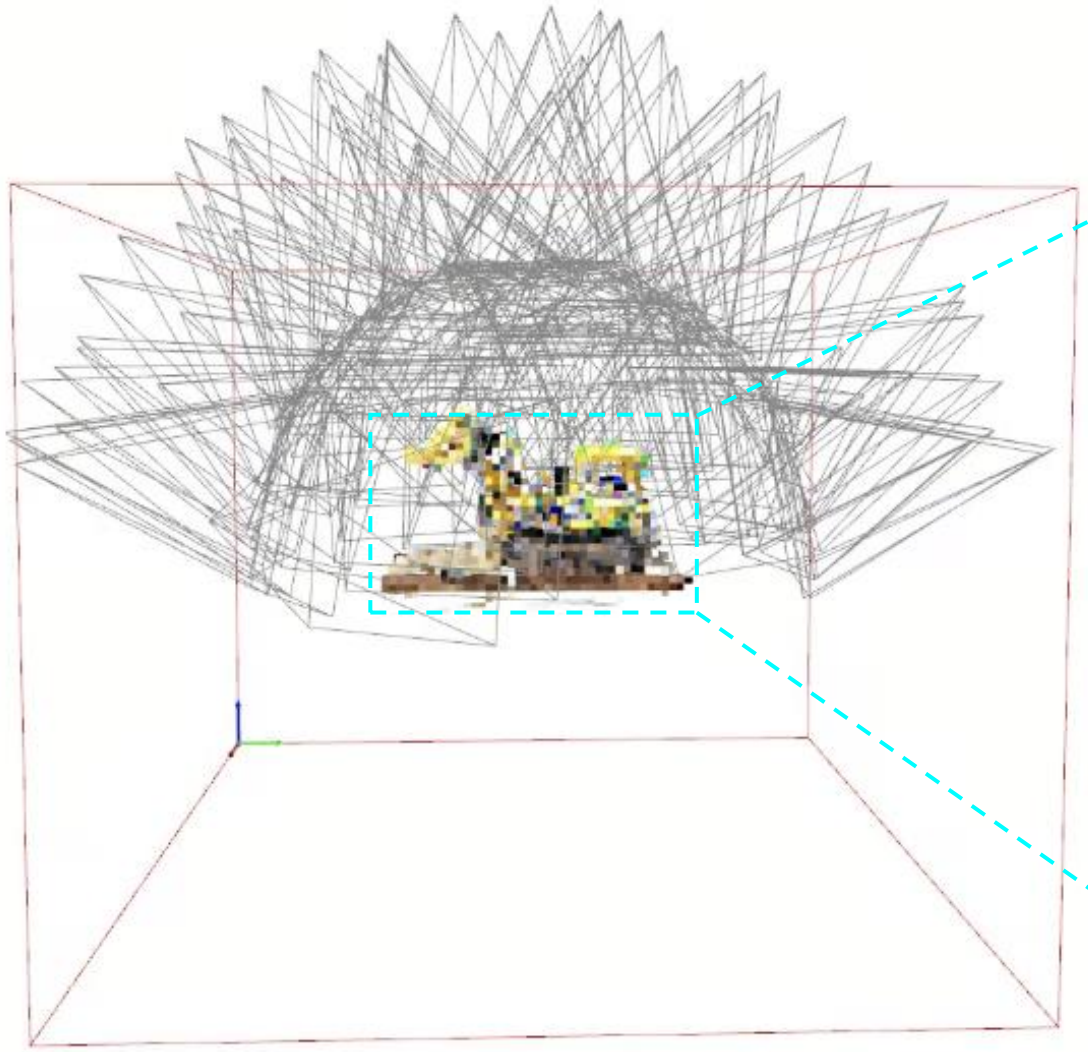


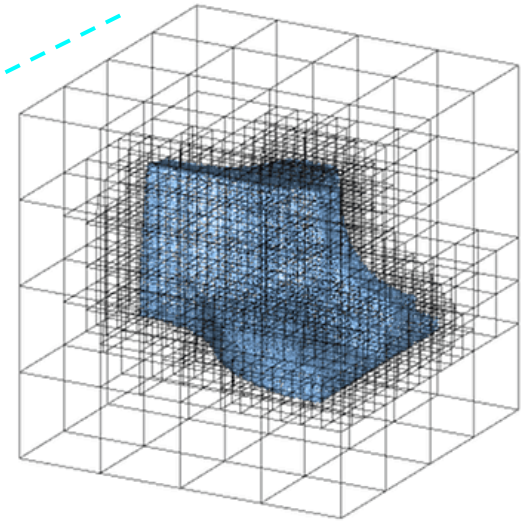
$$C(x,\omega) \approx \sum_{i=1}^{N} \left(\prod_{j=1}^{i-1} e^{-\sigma_j \Delta t_j} \right) \left(1 - e^{-\sigma_i \Delta t_i}\right) C_i$$

T_i α_i

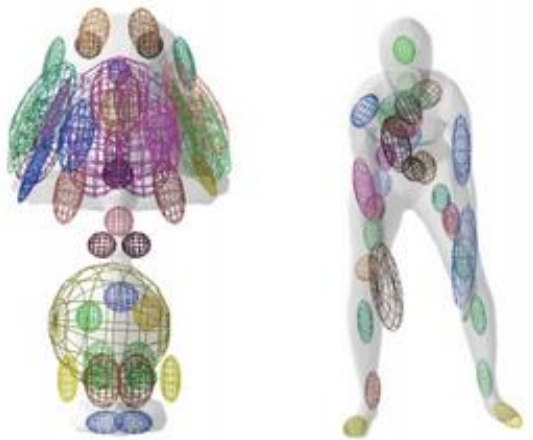
Learning objective: $\min_{\sigma_{t_i}, C_i} \| C(x,\omega) - \text{g.t.} \|$

What is the Representation of Volume?

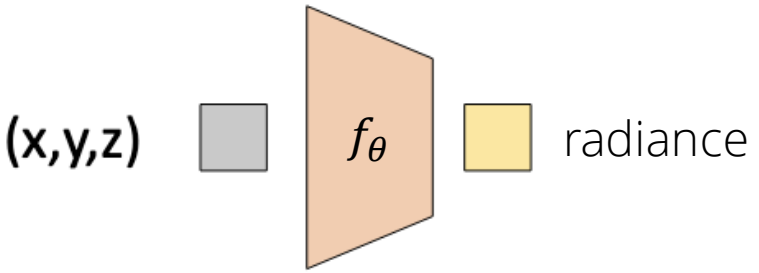




Voxel



Primitive shape

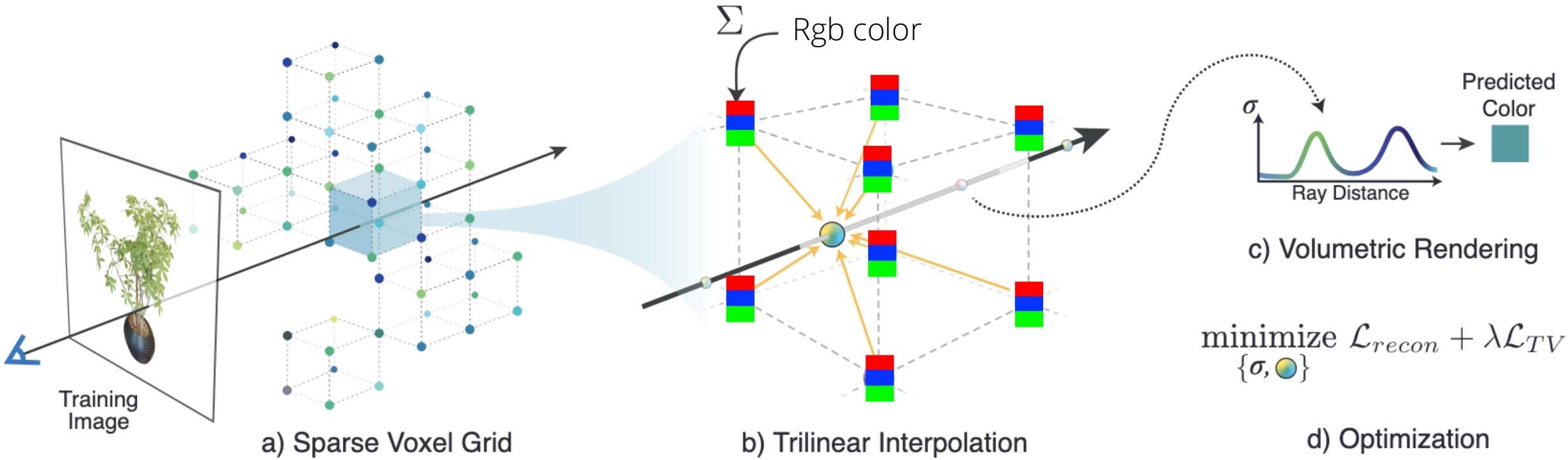


Neural network

Content

- Light Sources
- Light Measurement
- A mathematical model for image rendering
- Volume Rendering
  - Voxel
  - Nerf
  - Gaussian Splatting

Volume Rendering with Voxel Grid

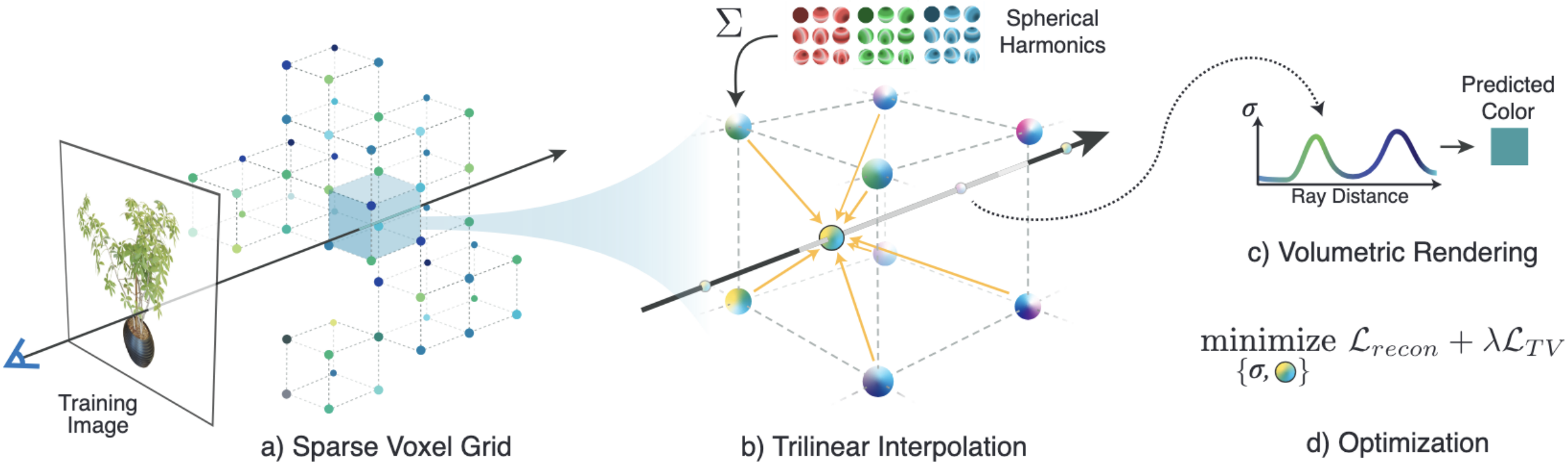


Plenoxels: Radiance Fields without Neural Networks. Yu et al.

Storing RGB Color Cannot Handle View-Dependent Effect

View-Dependent Effects for 360◦ Virtual Reality Video. Hartmann et al.

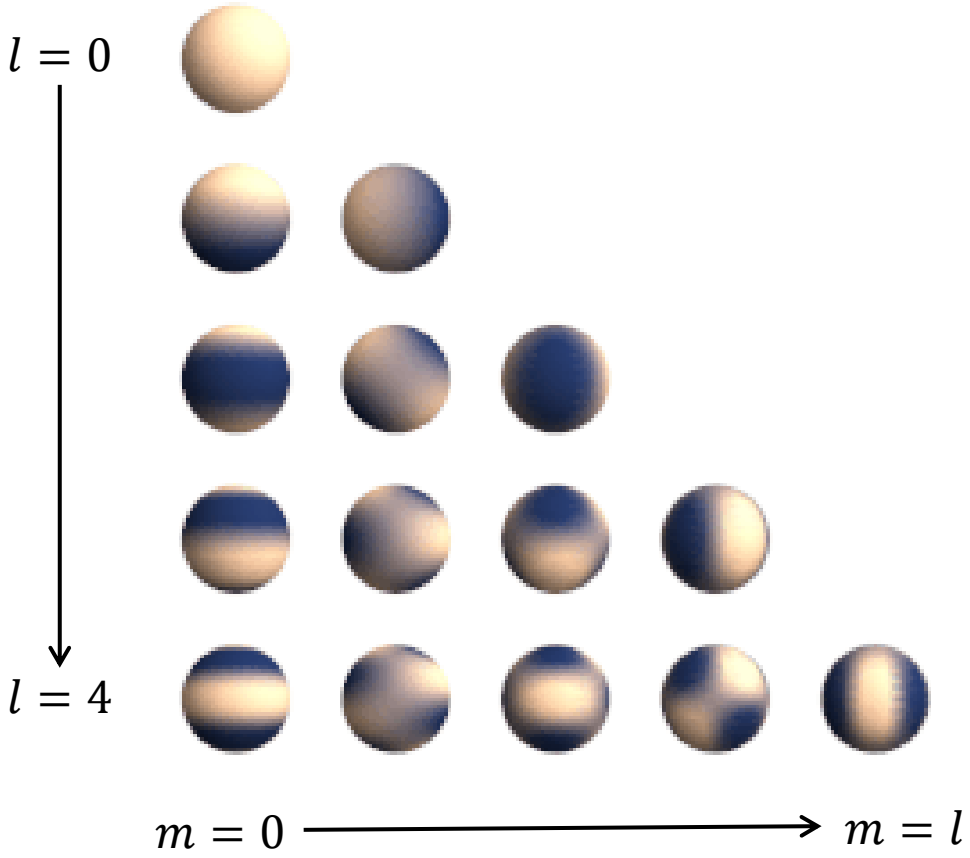
Volume Rendering with Voxel Grid



Plenoxels: Radiance Fields without Neural Networks. Yu et al.

Spherical Harmonics Model View-Dependent Lighting

$Y_m^l(\theta, \phi)$



- For each channel in RGB:
  - Predict coefficients $\boldsymbol{k} = \{k_l^m\}_{l,m}$
  - Decide the lighting

$$c(\mathbf{d}; \mathbf{k}) = S\left(\sum_{\ell=0}^{\ell_{\max}} \sum_{m=-\ell}^{\ell} k_\ell^m Y_\ell^m(\mathbf{d})\right)$$

where S denotes sigmoid function

Video source: Wikipedia

PlenOctrees for Real-time Rendering of Neural Radiance Fields. Yu et al.

Spherical Harmonics Model View-Dependent Effect

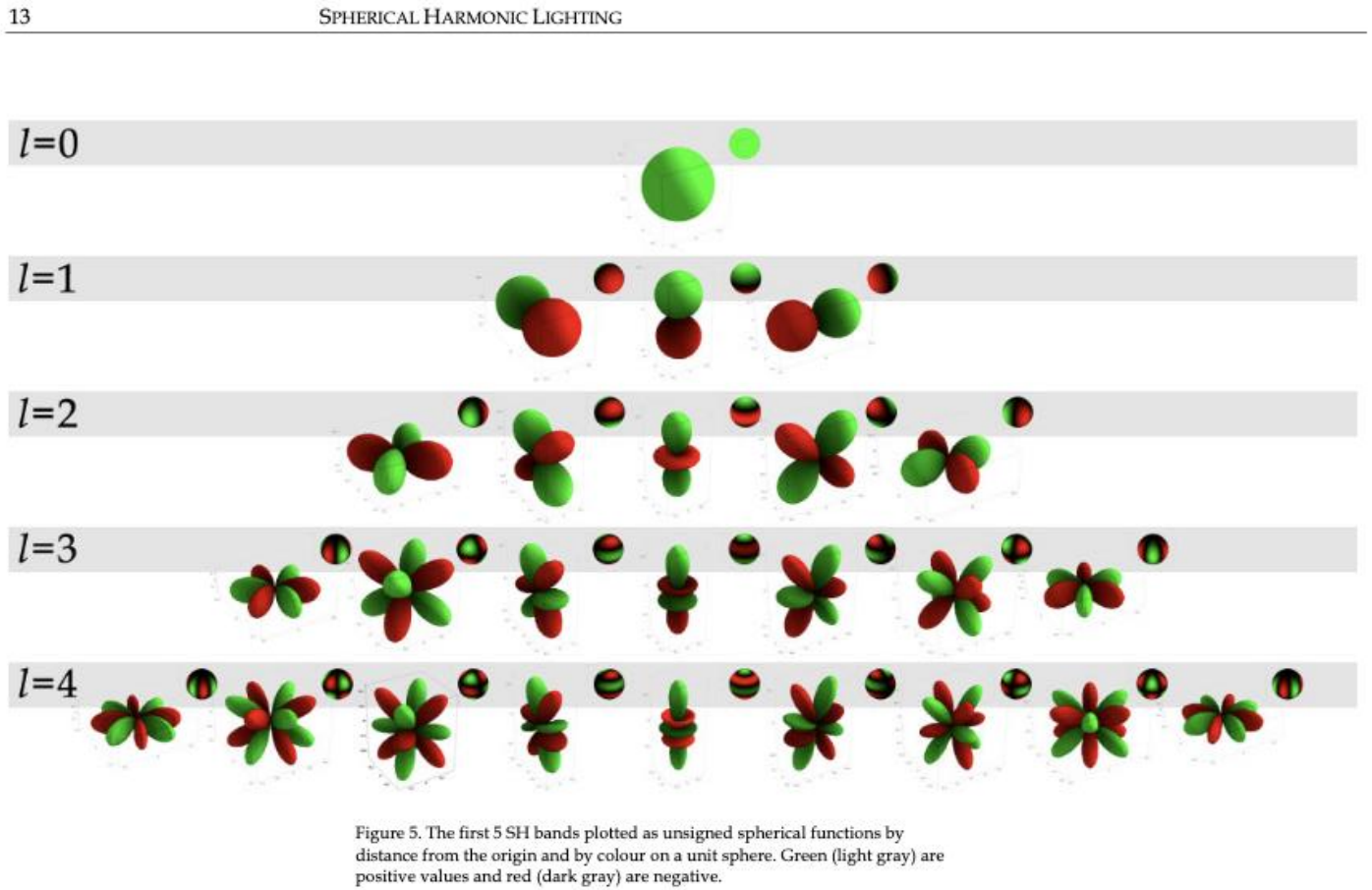


Figure 5. The first 5 SH bands plotted as unsigned spherical functions by distance from the origin and by colour on a unit sphere. Green (light gray) are positive values and red (dark gray) are negative.

$$y_l^m(\theta,\varphi)=\begin{cases}\sqrt{2}K_l^m\cos(m\varphi)P_l^m(\cos\theta), & m>0\\ \sqrt{2}K_l^m\sin(-m\varphi)P_l^{-m}(\cos\theta), & m<0\\ K_l^0P_l^0(\cos\theta), & m=0\end{cases}$$

$$K_l^m=\sqrt{\frac{(2l+1)}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}}$$

P_l^m denotes the associated Legendre polynomials

$$(l-m)P_l^m=x(2l-1)P_{l-1}^m-(l+m-1)P_{l-2}^m$$

$$P_m^m=(-1)^m(2m-1)!!(1-x^2)^{m/2}$$

$$P_{m+1}^m=x(2m+1)P_m^m$$

Spherical Harmonic Lighting: The Gritty Details. Green et al.

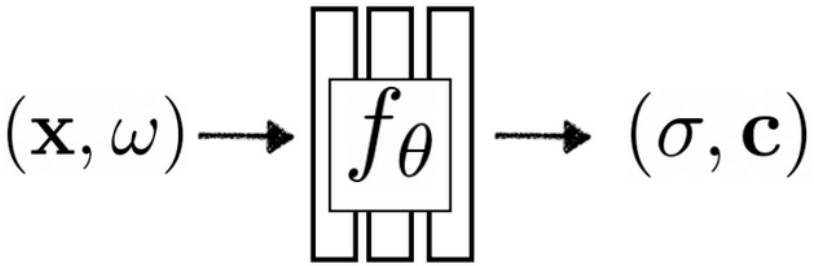
Spherical Harmonics Model View-Dependent Effect

Plenoxels: Radiance Fields without Neural Networks. Yu et al.

Content

- Light Sources
- Light Measurement
- A mathematical model for image rendering
- Volume Rendering
  - Voxel
  - Nerf
  - Gaussian Splatting

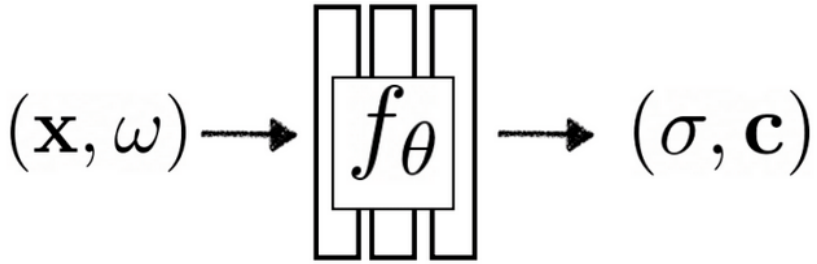
NEural Radiance Field (NERF)



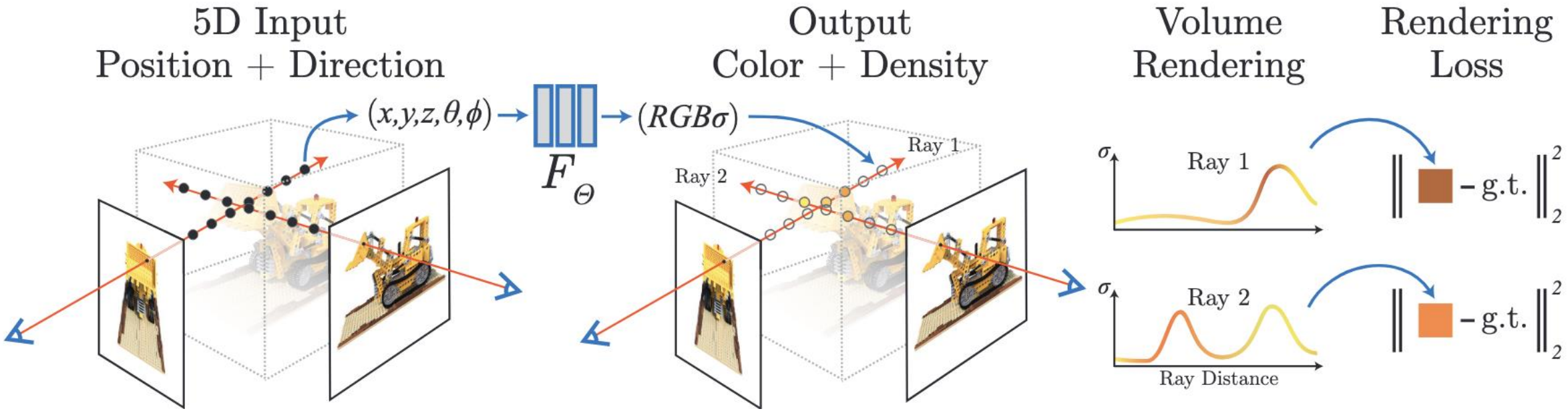
- Use a neural network f_θ to predict at each point x and view direction ω:
  - Opacity σ
  - Color c

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. Mildenhall et al.

NEural Radiance Field (NERF)

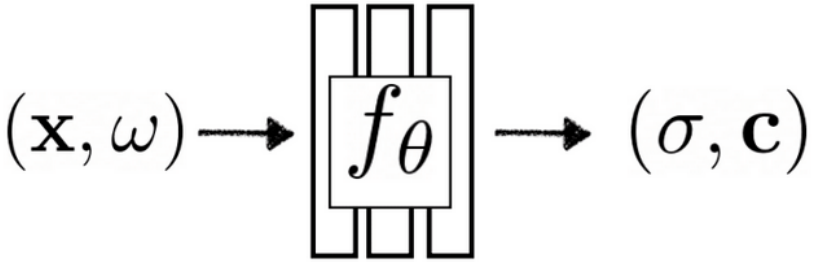


- Use a neural network f_θ to predict at each point x and view direction ω:
  - Opacity σ
  - Color$c$



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. Mildenhall et al.

Problem 1: The Network Takes Inputs as x, ω Coordinates Fails to Capture High-Frequency Features



- Use a neural network f_θ to predict at each point x and view direction ω:
  - Opacity σ
  - Color c

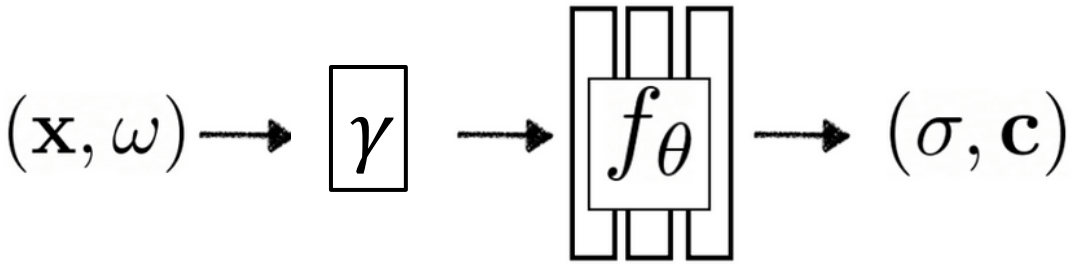
Ground Truth

No Positional Encoding

Fourier Features Let Networks Learn
High Frequency Functions in Low Dimensional Domains. Tancik et al.

Solution: Maps as x, ω Coordinates to High-Dimensional Feature Space

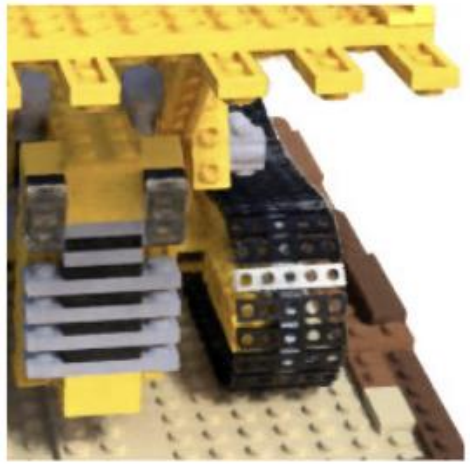
$$\gamma(v) = [\sin(v), \cos(v), \sin(2v), \cos(2v), \cdots]$$



- Use a neural network f_θ to predict at each point x and view direction ω:
  - Opacity σ
  - Color c

Ground Truth

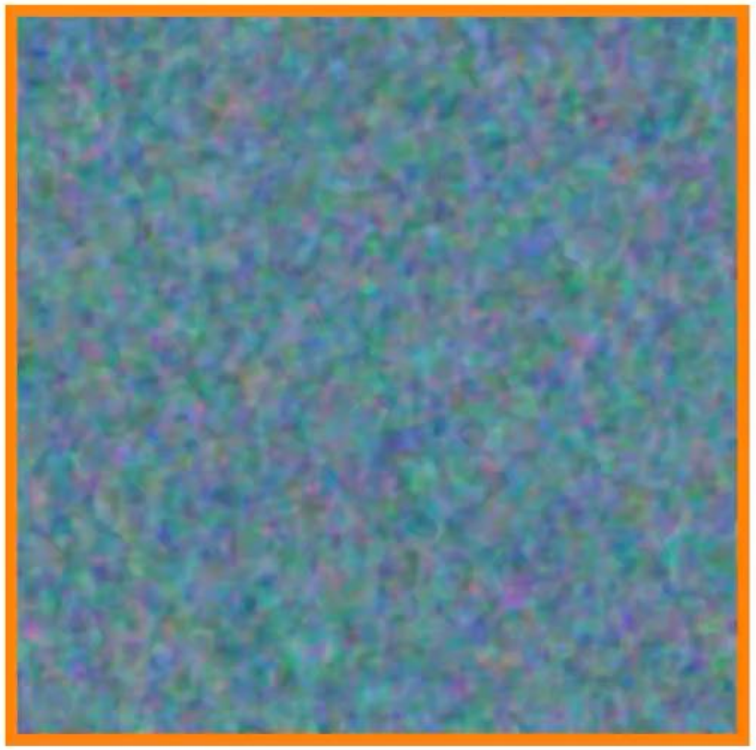
No Positional Encoding

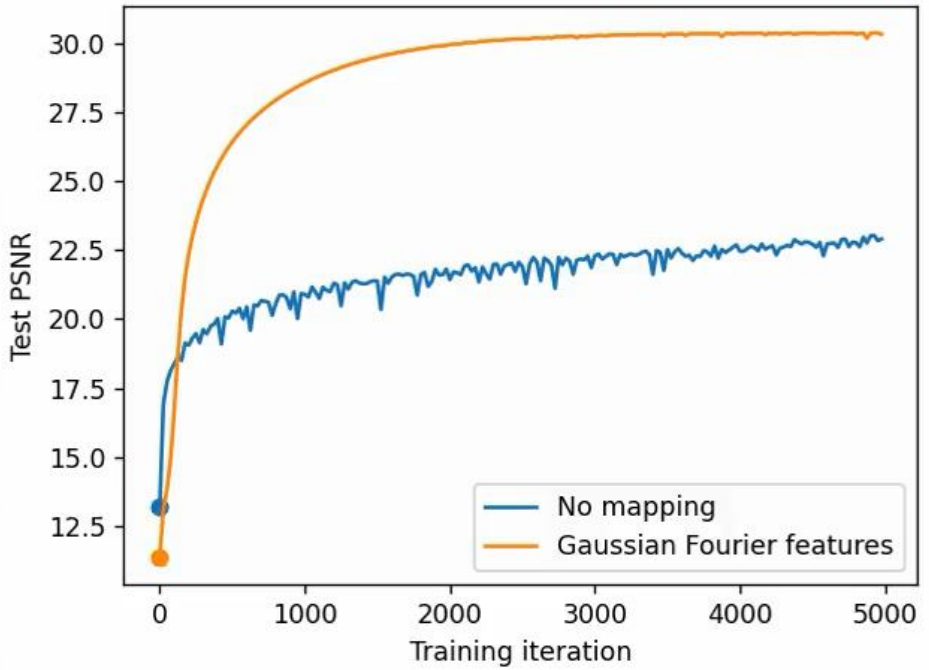


Complete Model

Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. Tancik et al.

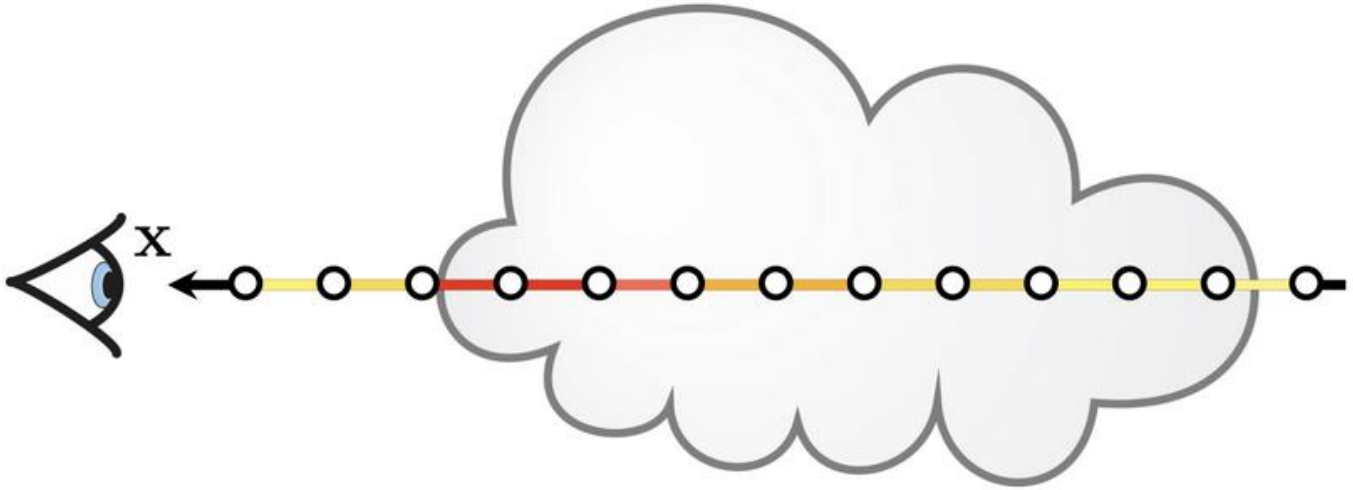
Solution: Maps as x, ω Coordinates to High-Dimensional Feature Space





Fourier Features Let Networks Learn
High Frequency Functions in Low Dimensional Domains. Tancik et al.

Problem 2: Uniform Sampling Points Along a Ray Ignores Regions with Fine Details



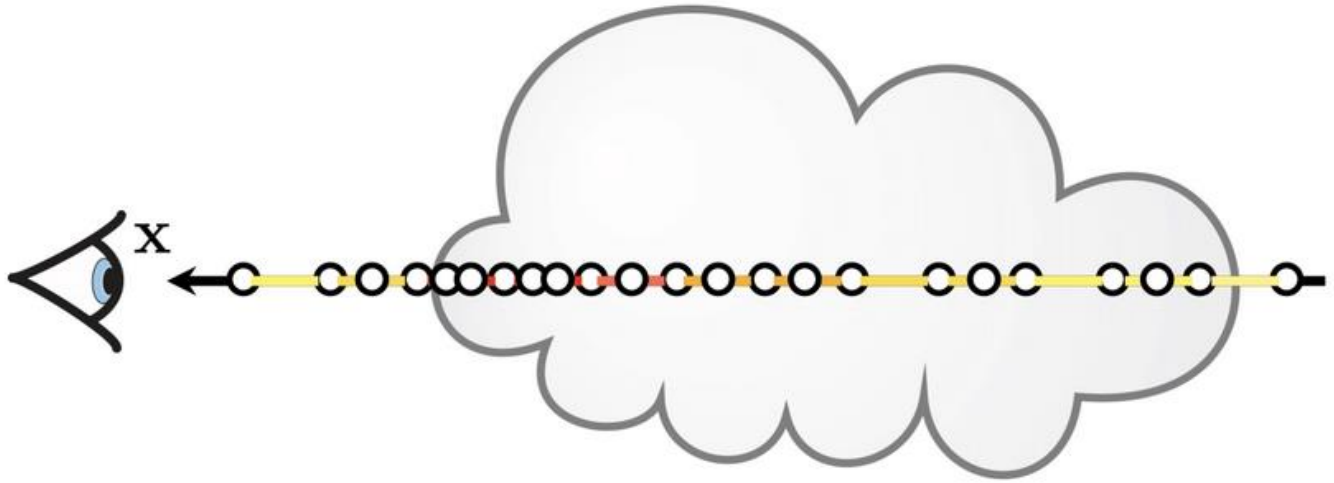
red = high, yellow = low weights

$$L(\mathbf{x}, \omega) = w_i \mathbf{c}_i; \quad w_i = T_i(1 - e^{-\sigma_{t_i} \Delta t})$$

weights indicate how much each segment contributes to the pixel value

Slide credit S. Tulsiani

Solution: Hierarchically Sampling Points Along a Ray Based on the Opacity



red = high, yellow = low weights

$$L(\mathbf{x}, \omega) = \sum_i w_i \mathbf{c}_i; \quad w_i = T_i(1 - e^{-\sigma_{t_i} \Delta t})$$

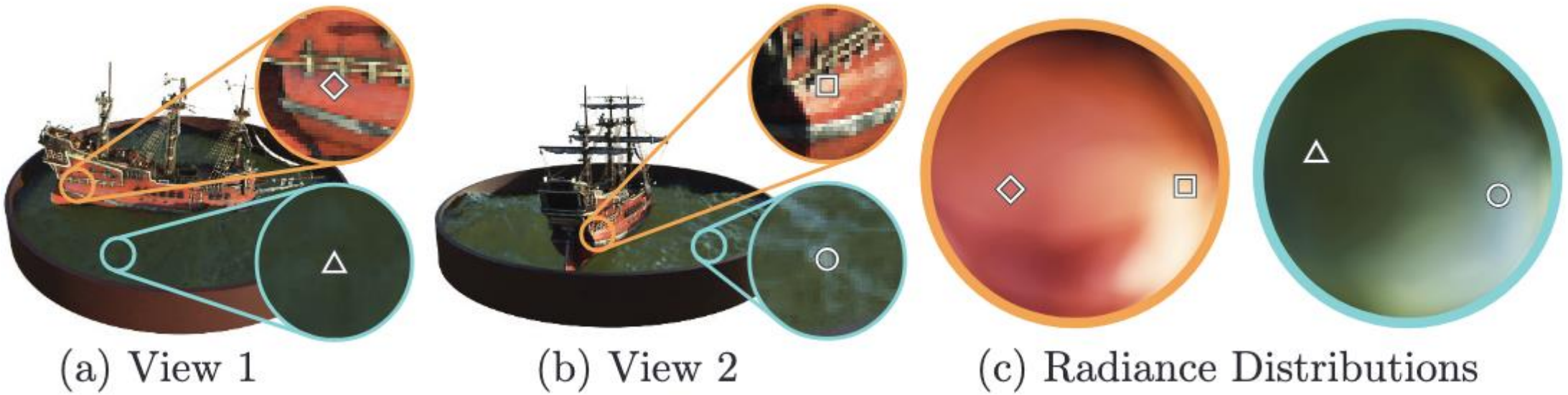
weights indicate how much each segment contributes to the pixel value

1) Sample points uniformly

2) Sample *more* points per segment depending on weight

Slide credit S. Tulsiani

NERF Captures View-Dependent Effect



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. Mildenhall et al.

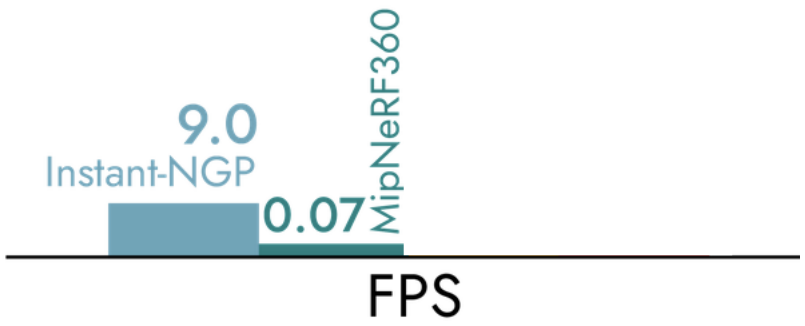
NERF Decouples View Direction and Position

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. Mildenhall et al.

Content

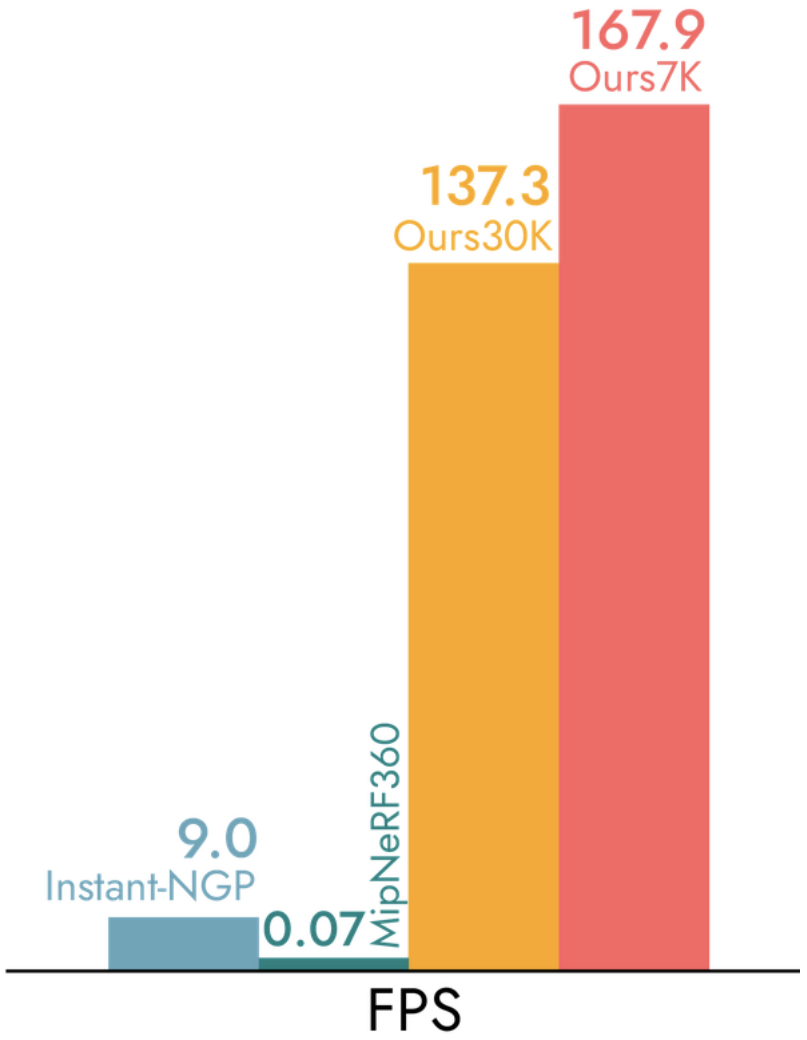
- Light Sources
- Light Measurement
- A mathematical model for image rendering
- Volume Rendering
  - Voxel
  - Nerf
  - Gaussian Splatting

NERF is Slow…



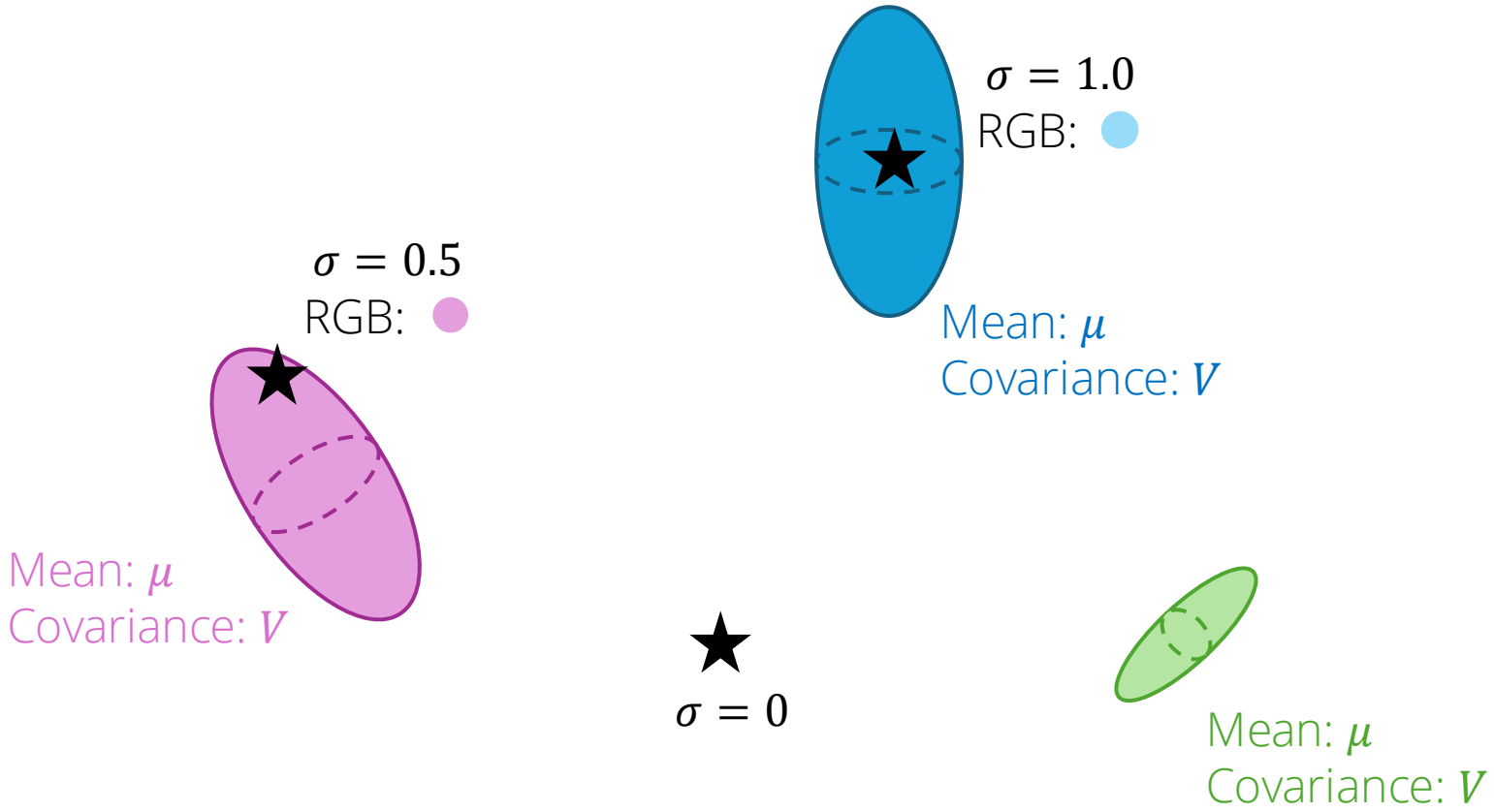
3D Gaussian Splatting for Real-Time Radiance Field Rendering. Kerbl et al.

A New Approach for Real-time Rendering



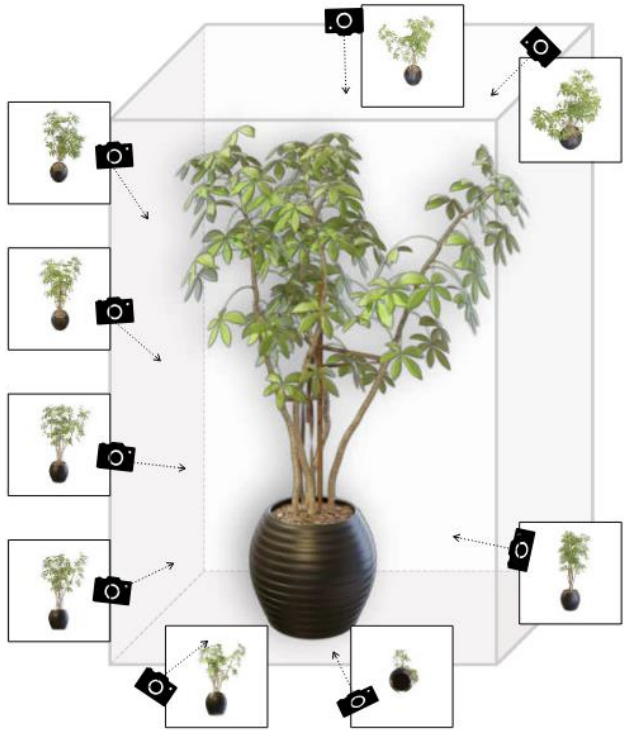
3D Gaussian Splatting for Real-Time Radiance Field Rendering. Kerbl et al.

Key Idea: Parameterize Radiance Field Sparsely, Only where Density is Nonzero



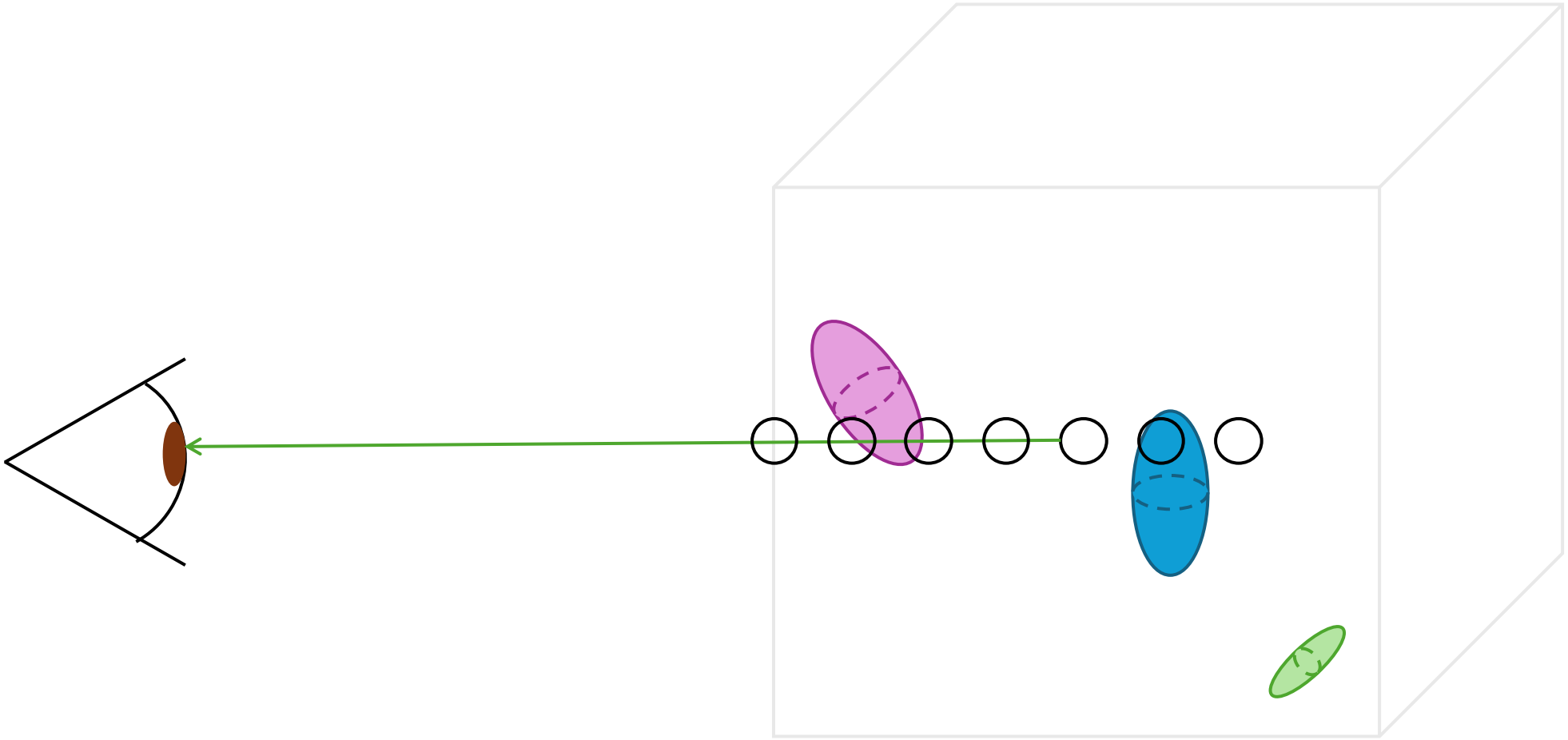
Slide adapted from V. Sitzmann

Volume Rendering with 3D Gaussian Splatting



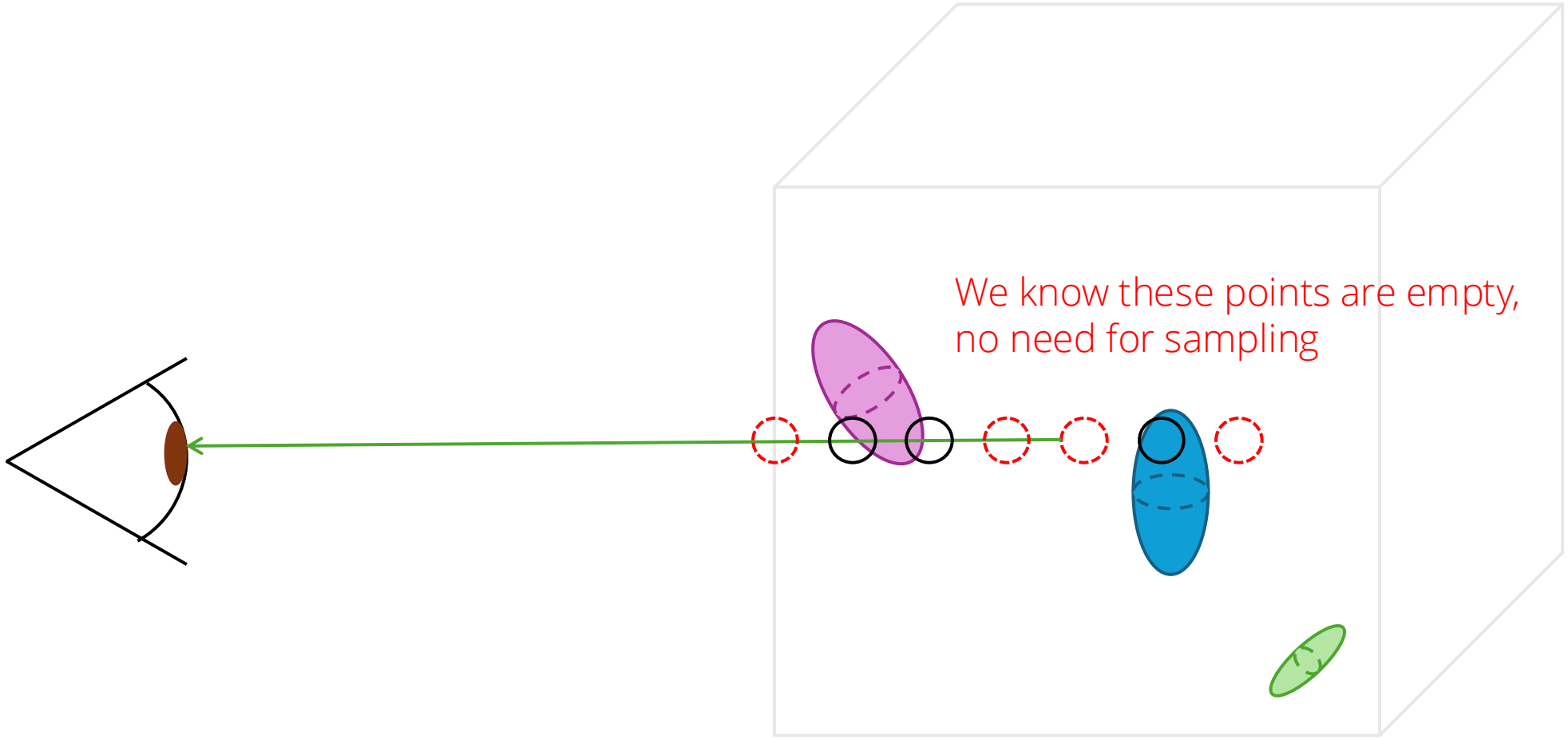
PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics. Xie et al. CVPR 2024

How to Render?
Same Volume Rendering Integral!



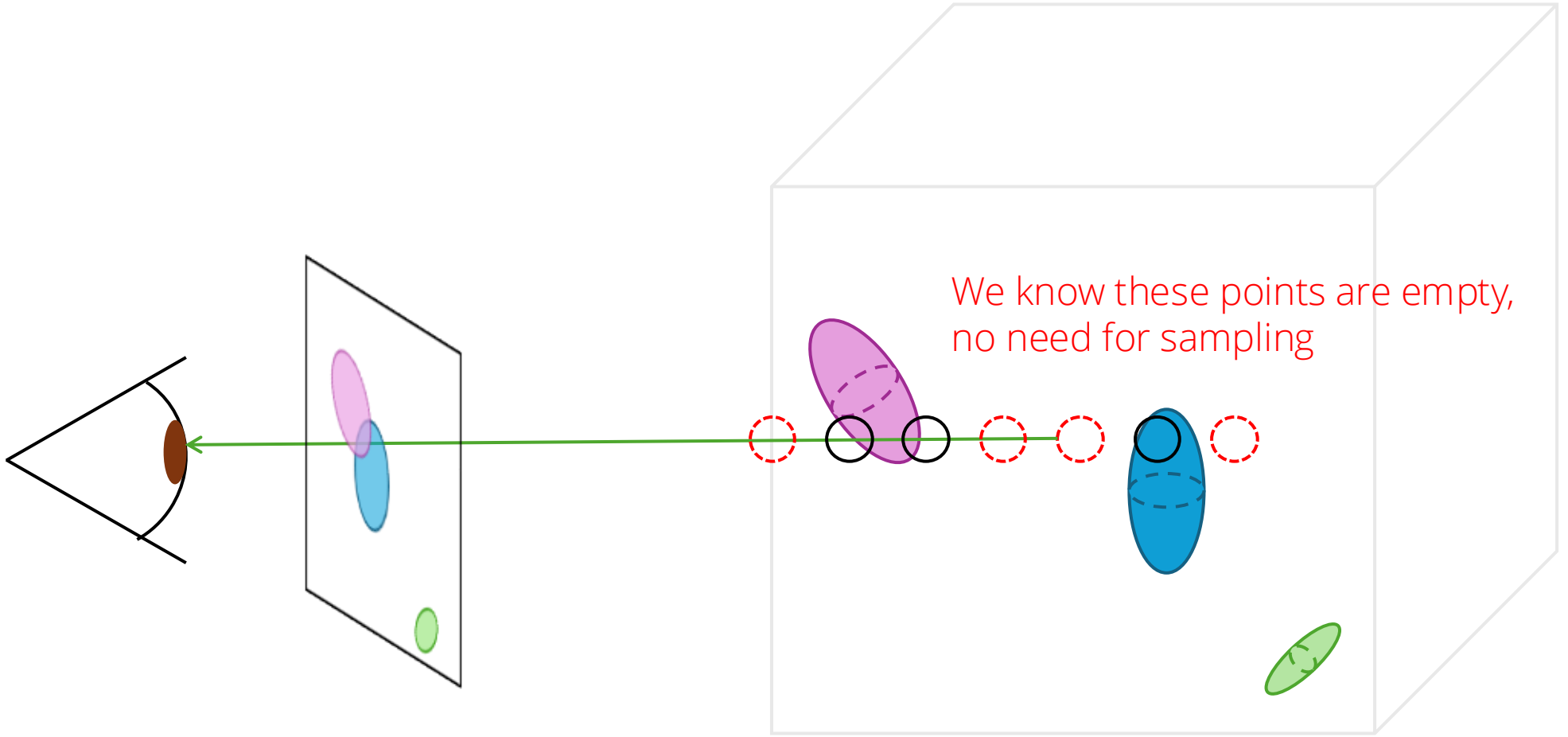
Slide adapted from V. Sitzmann

Volume Rendering Integral Samples Many Points from Empty Space, which is Slow



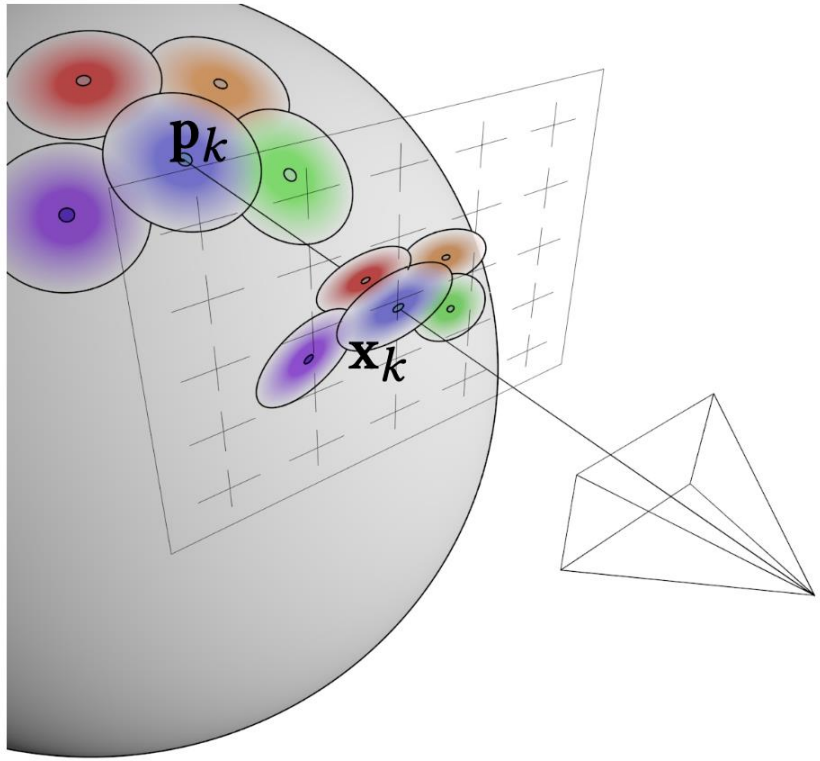
Slide adapted from V. Sitzmann

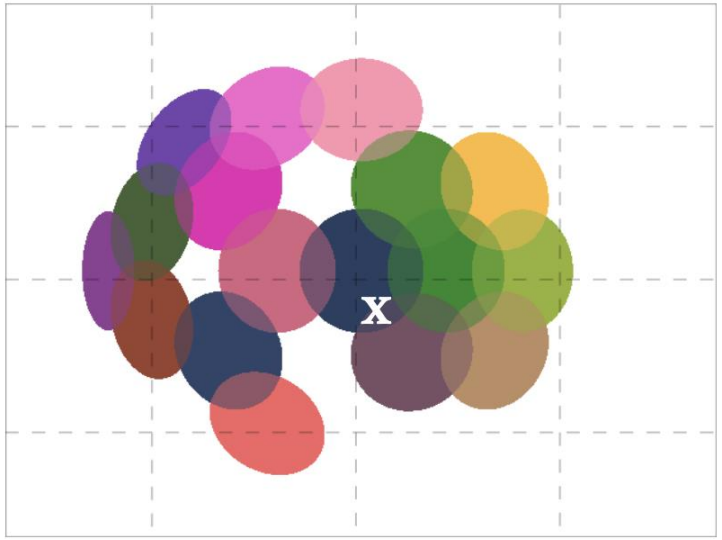
Can We Render from 3D Gaussians Smartly? Yes! Project 3D Gaussians to 2D



Slide adapted from V. Sitzmann

Idea: Solve Image Rendering by Looking Up 2D Gaussians



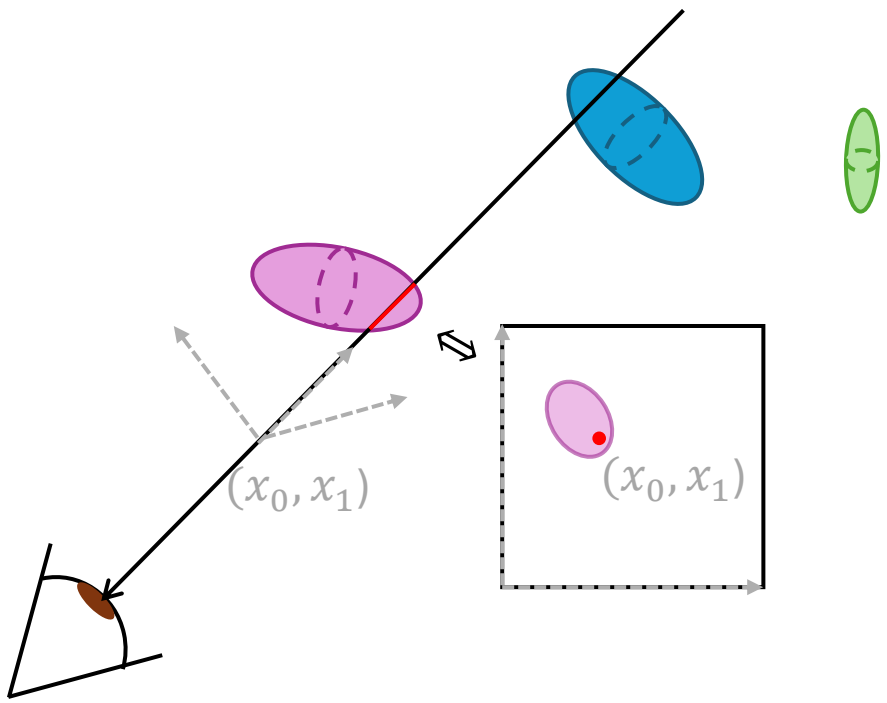


$$\mathbb{I}_{\mathbf{x}} = \frac{\sum_{k=0}^{N-1} \rho_k(\mathbf{x})\, \mathbf{w}_k}{\sum_{k=0}^{N-1} \rho_k(\mathbf{x})}.$$

$\rho_k(x)$: weightings derived from 2D Gaussian at pixel x
w_k: point attributes (e.g. opacity) at pixel x

Differentiable Surface Splatting for Point-based Geometry Processing. Yifan et al.

Gaussians are Closed Under Integrals



- Integrate along x_2 axis with fixed x_0, x_1

$$\int_{\mathbb{IR}} \mathcal{G}_V{}^3(x-p)dx_2 = \mathcal{G}_{\hat{V}}{}^2(\hat{x}-\hat{p})$$

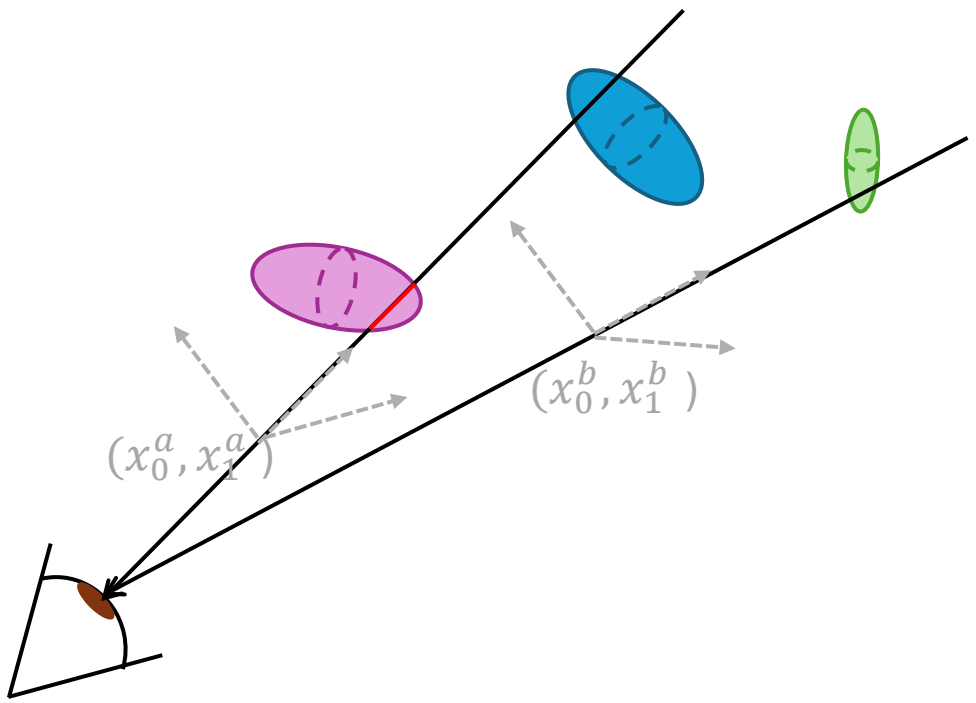
3D Gaussian 2D Gaussian

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \Leftrightarrow \hat{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

$$V = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \Leftrightarrow \hat{V} = \begin{bmatrix} a & b \\ b & d \end{bmatrix}$$

- Integrate along x_2 axis with fixed x_0, x_1

Gaussians are Closed Under Integrals



- Integrate along x_2 axis with fixed x_0, x_1

$$\int_{\mathbb{IR}} \mathcal{G}_V{}^3(x-p)dx_2 = \mathcal{G}_{\hat{V}}{}^2(\hat{x}-\hat{p})$$

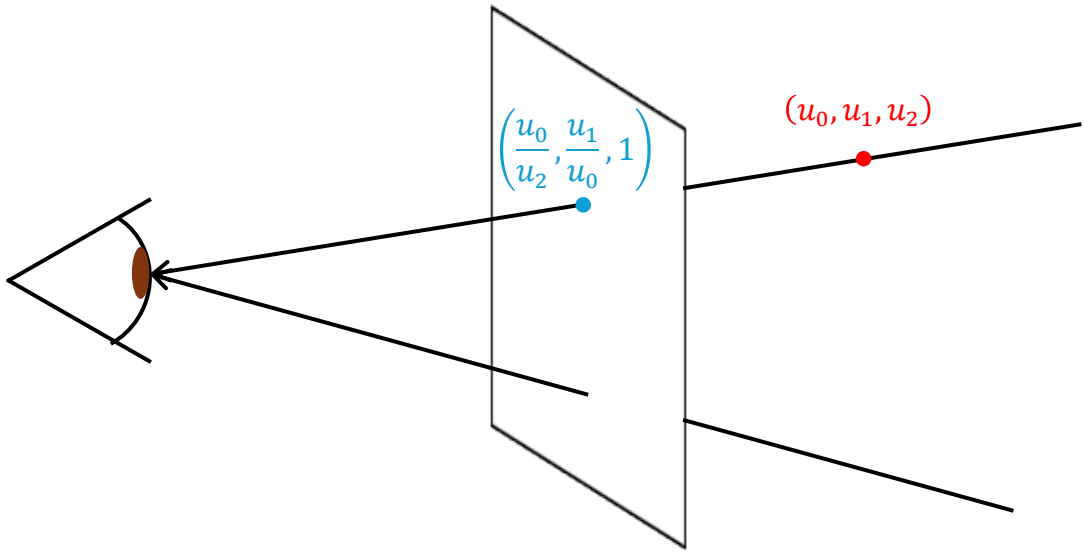
3D Gaussian 2D Gaussian

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \Leftrightarrow \hat{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

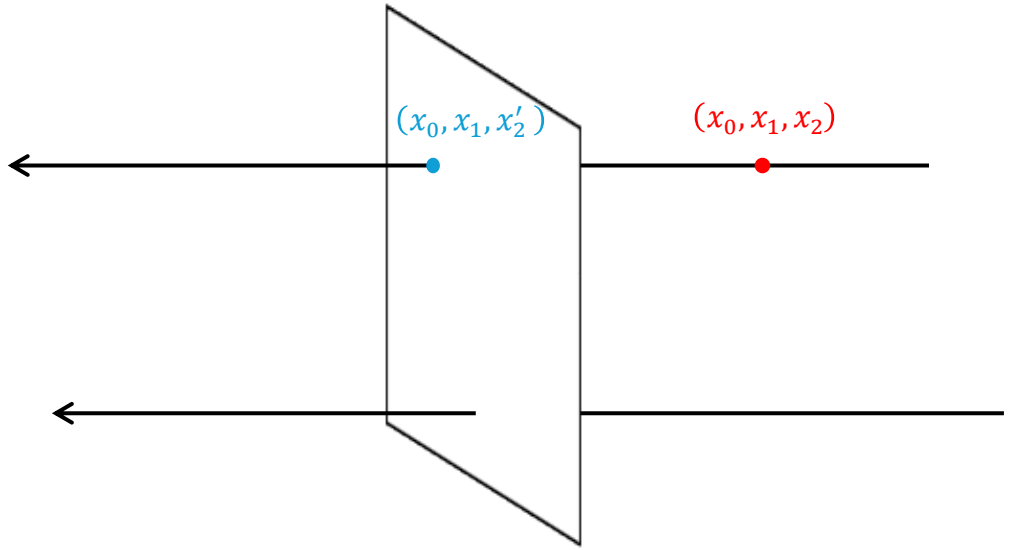
$$V = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \Leftrightarrow \hat{V} = \begin{bmatrix} a & b \\ b & d \end{bmatrix}$$

- Different rays have different axes
- How to define a space that describe each ray with x_0, x_1, x_2 ?

Camera Space vs. Ray Space



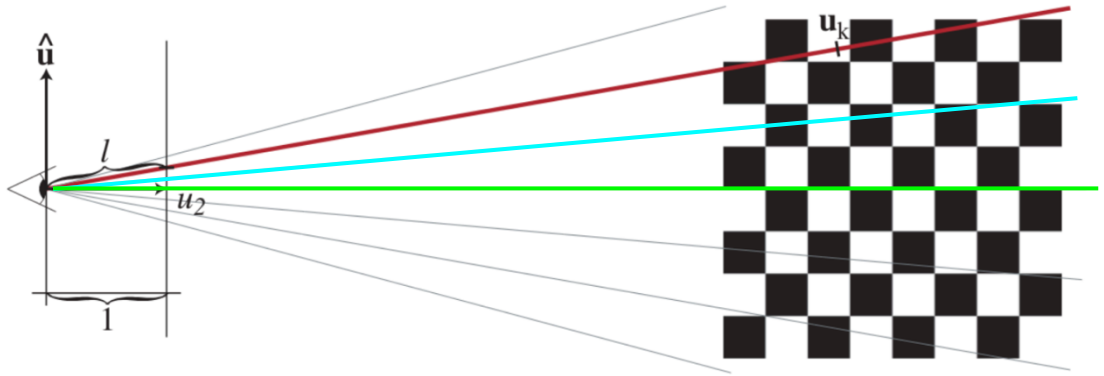
- In the camera 3D space, a point is described by (X, Y, Z) position with the camera coordinate frame
- Each point along a ray can be described as u_0, u_1, u_2
- When $u_2 = 1$, u_0, u_1 denotes the intersecting point on the camera plane

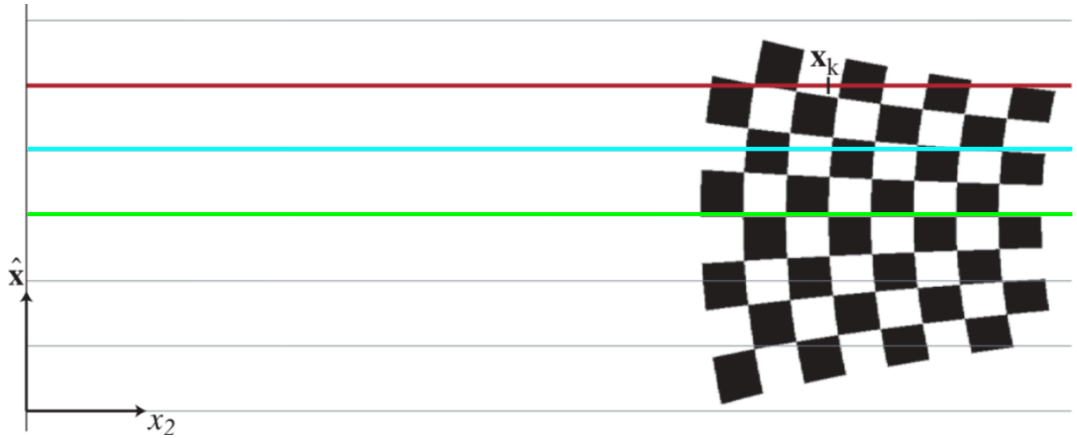


- In the ray space space, a ray is described by (x_0, x_1, x_2)
- x_0, x_1 denotes the pixel location
- x_2 describes the extent of the ray

Distorting the Camera 3D Space into Ray Space

Different rays have different levels of distortion

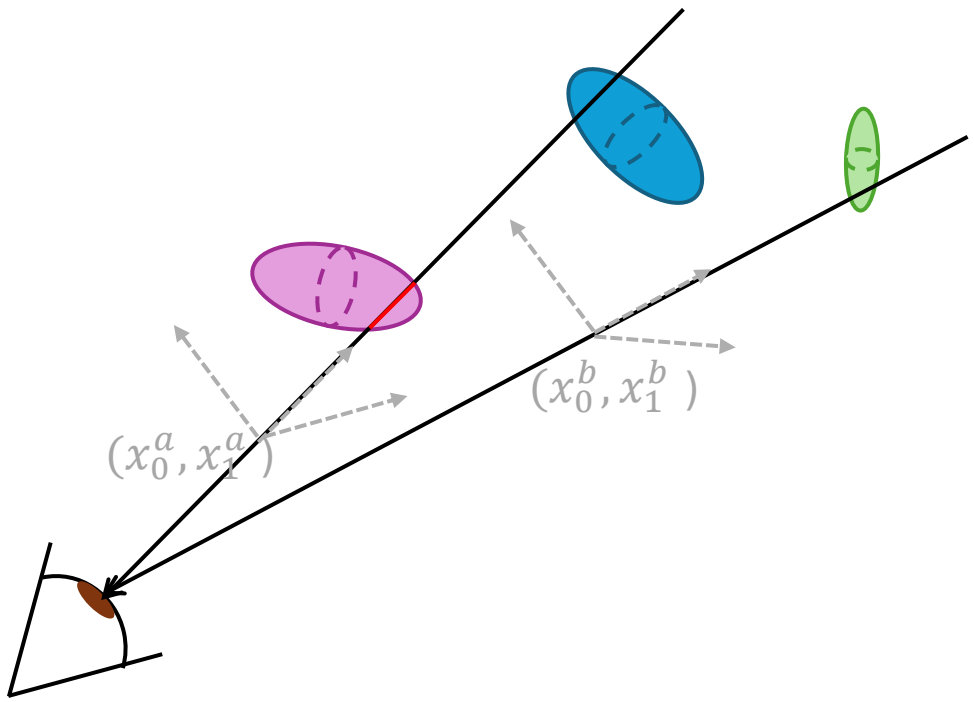




- In the camera 3D space, a point is described by (X, Y, Z) position with the camera coordinate frame
- Each point along a ray can be described as u_0, u_1, u_2
- When $u_2 = 1$, u_0, u_1 denotes the intersecting point on the camera plane

- In the ray space space, a ray is described by (x_0, x_1, x_2)
- x_0, x_1 denotes the pixel location
- x_2 describes the extent of the ray

We Can Compute 2D Gaussians on the Image to Obtain Integrals of 3D Gaussians in the Ray Space



- Integrate along x_2 axis with fixed x_0, x_1

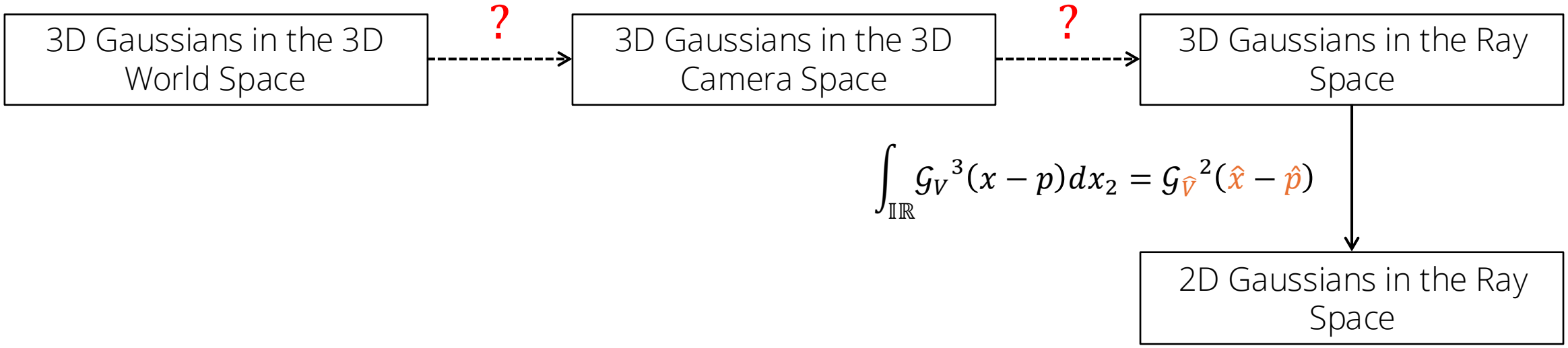
$$\int_{\mathbb{IR}} \mathcal{G}_V{}^3(x-p)dx_2 = \mathcal{G}_{\hat{V}}{}^2(\hat{x}-\hat{p})$$

3D Gaussian 2D Gaussian

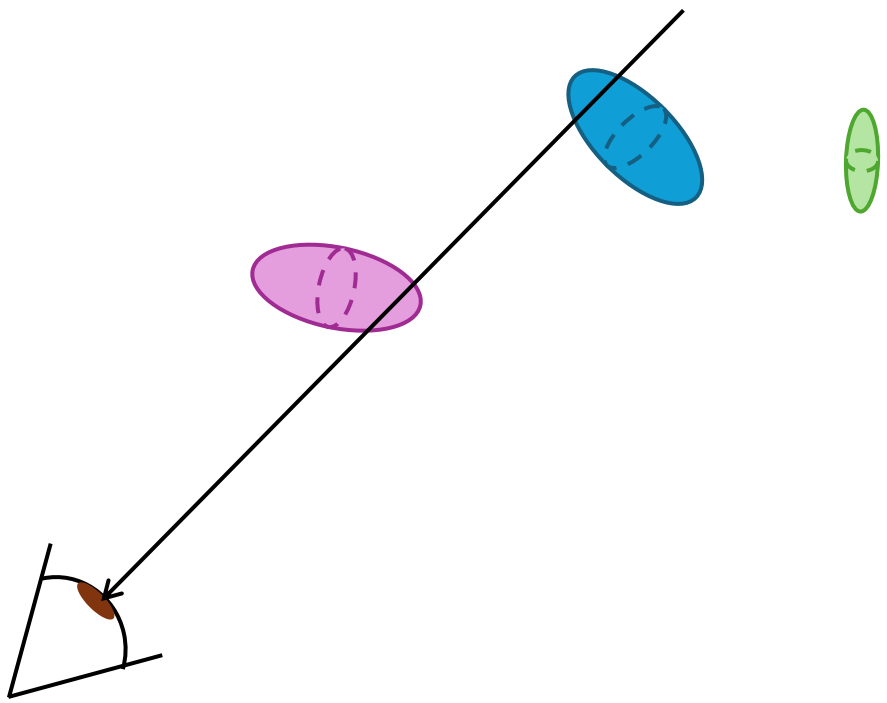
- Integrate along x_2 axis in the Ray space: we have the same x_0, x_1 along the ray!
- Integrals of 3D Gaussians along the x_2 axis in the Ray Space are Equivalent to 2D Gaussians at x_0, x_1

However, 3D Gaussians are in the World Frame

- What is missing: describe 3D Gaussians from different 3D space



Gaussians are Closed Under Affine Transforms



$$\mathcal{G}_V(x-p) = \frac{1}{2\pi|V|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-p)^\intercal V^{-1}(x-p)}$$

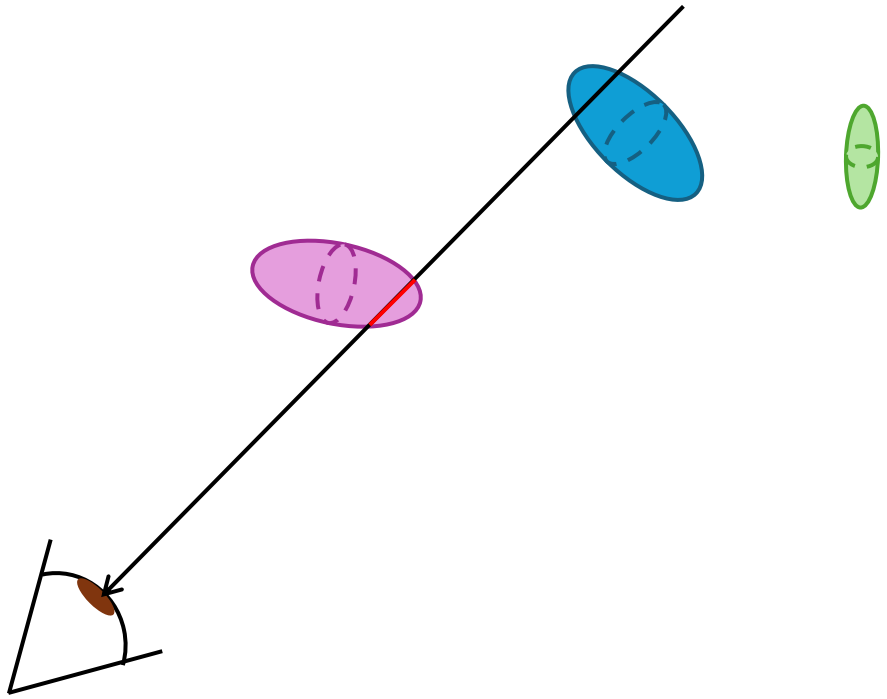
3D covariance

- With affine mapping $\phi = Mx + p$ of coordinates (such as cam2world matrix)

$$\mathcal{G}_V(\phi^{-1}(u) - p) = \frac{1}{|M^{-1}|}\mathcal{G}_{MVM^\intercal}\big(u - \phi(p)\big)$$

Slide adapted from V. Sitzmann

Step 1: Transform 3D Gaussians from the World Coordinates to the Camera Coordinates

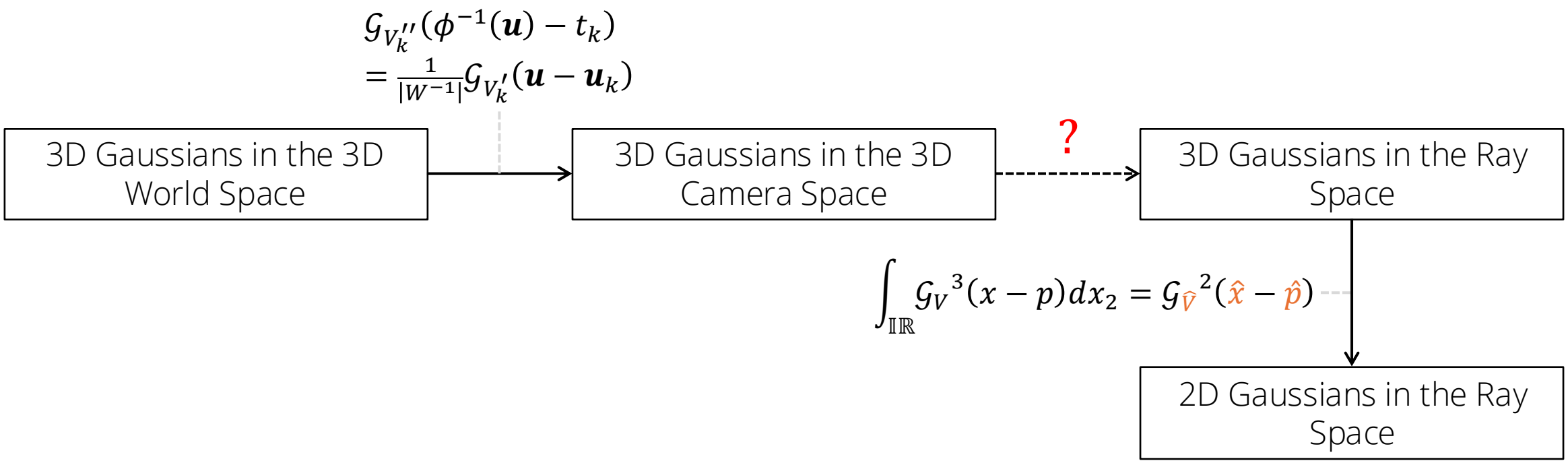


- Cam-to-World transformation is affine mapping $\phi = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{p}$:

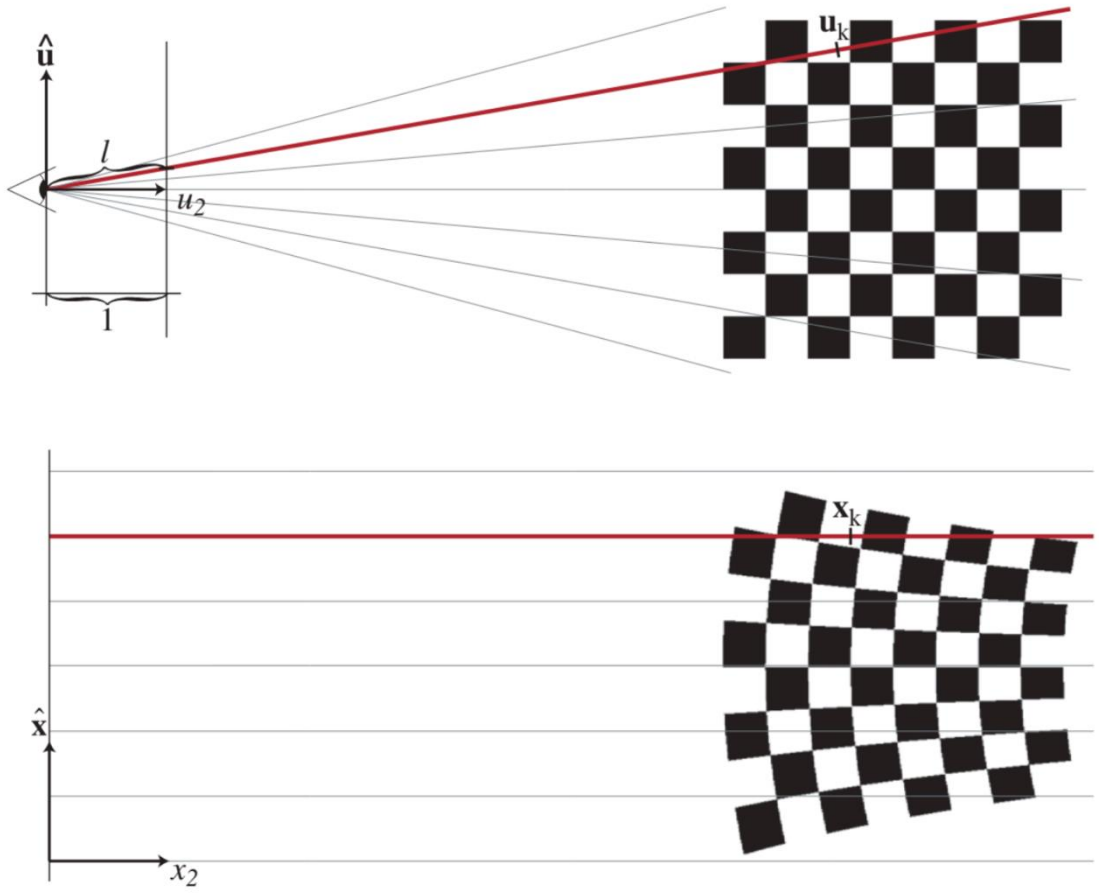
$$\mathcal{G}_{V_k''}(\phi^{-1}(\boldsymbol{u}) - t_k) = \frac{1}{|W^{-1}|}\mathcal{G}_{V_k'}(\boldsymbol{u} - \boldsymbol{u}_k)$$

Slide adapted from V. Sitzmann

However, 3D Gaussians are in the World Frame



Step 2: Convert 3D Camera Space to Ray Space



- Ray space $\boldsymbol{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$: Given a center of projection and a projection plane, x_0, x_1 specifies the intersecting points of a ray on the plane; x_2 specifies the Euclidean distance from the center of projection to a point on the viewing ray
- Camera space $\boldsymbol{u} = \begin{pmatrix} u_0 \\ u_1 \\ u_2 \end{pmatrix}$: 3D points along the same ray in the camera coordinate frame
- The mapping of ray space and camera space is:

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = m(\boldsymbol{u}) = \begin{pmatrix} u_0/u_2 \\ u_1/u_2 \\ \|\boldsymbol{u}\| \end{pmatrix}$$

EWA Splattering. Zwicker et al.

Approximate the Mapping Function with Jacobian $\boldsymbol{J_{u_k}}$

- Projection is not affine mapping

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = m(\boldsymbol{u}) = \begin{pmatrix} u_0/u_2 \\ u_1/u_2 \\ \|\boldsymbol{u}\| \end{pmatrix}$$

- We can approximate with 1st order Taylor Expansion at mean $\boldsymbol{u_k}$ and $\boldsymbol{x_k}$ of two spaces as:

$$m_{\boldsymbol{u_k}}(\boldsymbol{u}) = \underbrace{\boldsymbol{x_k}}_{m(\boldsymbol{u_k})} + J_{\boldsymbol{u_k}} \cdot (\boldsymbol{u} - \boldsymbol{u_k}), \qquad \text{where } J_{\boldsymbol{u_k}} = \frac{\partial m}{\partial \boldsymbol{u}}(\boldsymbol{u_k})$$

Slide adapted from V. Sitzmann

Approximate the Mapping Function with Jacobian J_{u_k}

- We can approximate with 1st order Taylor Expansion at mean $\boldsymbol{u_k}$ and $\boldsymbol{x_k}$ of two spaces as:

$$m_{u_k}(\boldsymbol{u}) = \boldsymbol{x_k} + J_{u_k} \cdot (\boldsymbol{u} - \boldsymbol{u_k}), \qquad \text{where } J_{u_k} = \frac{\partial m}{\partial \boldsymbol{u}}(\boldsymbol{u_k})$$

- 3D Gaussians in the Ray space:

$$\frac{1}{|W^{-1}||J^{-1}|}\mathcal{G}_{V_k}(\boldsymbol{x} - \boldsymbol{x_k})$$

$$V_k = JV_k'J^{\mathsf{T}} = JWV_k''W^{\mathsf{T}}J^{\mathsf{T}}$$

V_k'' : covariance matrix in the 3D world space
V_k' : covariance matrix in the 3D camera space
V_k : covariance matrix in the 3D ray space
W: world to camera transformation matrix
J : camera space to ray space Jacobian matrix

Slide adapted from V. Sitzmann

Putting Everything Together

- We can approximate with 1st order Taylor Expansion at mean $\boldsymbol{u_k}$ and $\boldsymbol{x_k}$ of two spaces as:

$$m_{\boldsymbol{u_k}}(\boldsymbol{u}) = \boldsymbol{x_k} + J_{\boldsymbol{u_k}} \cdot (\boldsymbol{u} - \boldsymbol{u_k}), \qquad \text{where } J_{\boldsymbol{u_k}} = \frac{\partial m}{\partial \boldsymbol{u}}(\boldsymbol{u_k})$$

- 3D Gaussians in the Ray space:

$$\frac{1}{|W^{-1}||J^{-1}|}\mathcal{G}_{V_k}(\boldsymbol{x} - \boldsymbol{x_k})$$

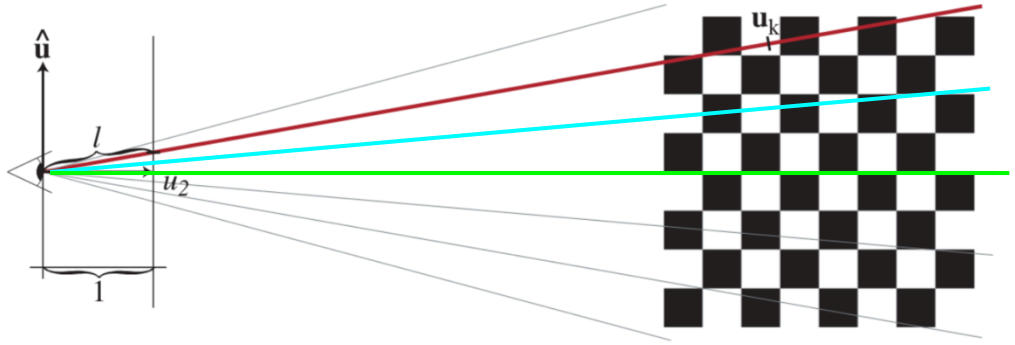
$$V_k = JV_k'J^\intercal = JWV_k''W^\intercal J^\intercal$$

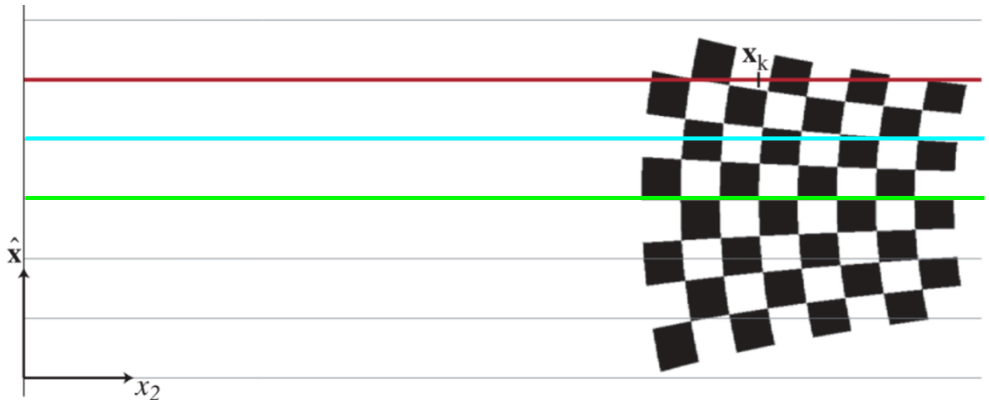
- Integrate along rays:

$$q_k(\widehat{\boldsymbol{x}}) = \int_{\mathbb{R}} \frac{1}{|W^{-1}||J^{-1}|}\mathcal{G}_{V_k}(\widehat{\boldsymbol{x}} - \widehat{\boldsymbol{x_k}}, x_2 - x_{k2})dx_2 = \frac{1}{|W^{-1}||J^{-1}|}\mathcal{G}_{\widehat{V}_k}(\widehat{\boldsymbol{x}} - \widehat{\boldsymbol{x_k}})$$

Slide adapted from V. Sitzmann

Putting Everything Together





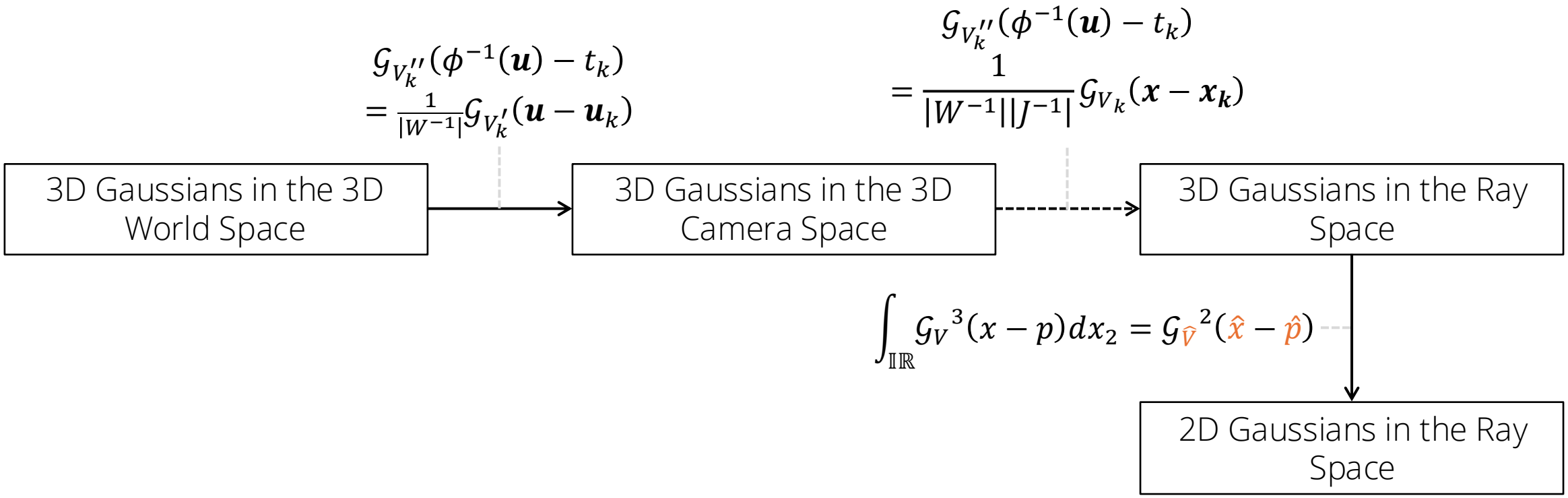
For each 3D Gaussian k:

- Compute $\boldsymbol{x_k}$ and $\boldsymbol{u_k}$
- Compute Jacobian $J_{\boldsymbol{u_k}} = \frac{\partial m}{\partial \boldsymbol{u}}(\boldsymbol{u_k})$
- Compute new covariance matrix $V_k = JWV_k''W^\intercal J^\intercal$
- Obtain 2D Gaussian $\frac{1}{|W^{-1}||J^{-1}|}\mathcal{G}_{\widehat{V}_k}(\widehat{\boldsymbol{x}} - \widehat{\boldsymbol{x_k}})$

For $\boldsymbol{x_0}, \boldsymbol{x_1}$ in all pixel locations:

- Find the set of 3D Gaussian intersected with the ray
- Sort the 3D Gaussians by depth
- For each 3D Gaussian k:
  - Retrieve the corresponding 2D Gaussian
  - Compute weightings $\rho_k(x_0, x_1)$ based on the 2D Gaussian
  - Compute the opacity $\rho_k(x_0, x_1)\sigma_k$

However, 3D Gaussians are in the World Frame



How to Obtain Jacobian?

- Jacobian $\boldsymbol{J_{u_k}}$ is given by the derivative of $\boldsymbol{m}$ at the point $\boldsymbol{u_k}$:

$$J_{\boldsymbol{u_k}} = \frac{\partial m}{\partial \boldsymbol{u}}(\boldsymbol{u_k}) = \begin{bmatrix} \frac{1}{u_{k,2}} & 0 & -\frac{u_{k,0}}{u_{k,2}^2} \\ 0 & \frac{1}{u_{k,2}} & -\frac{u_{k,1}}{u_{k,2}^2} \\ \frac{u_{k,0}}{\|\boldsymbol{u_k}\|} & \frac{u_{k,1}}{\|\boldsymbol{u_k}\|} & \frac{u_{k,2}}{\|\boldsymbol{u_k}\|} \end{bmatrix}$$

EWA Splattering. Zwicker et al.

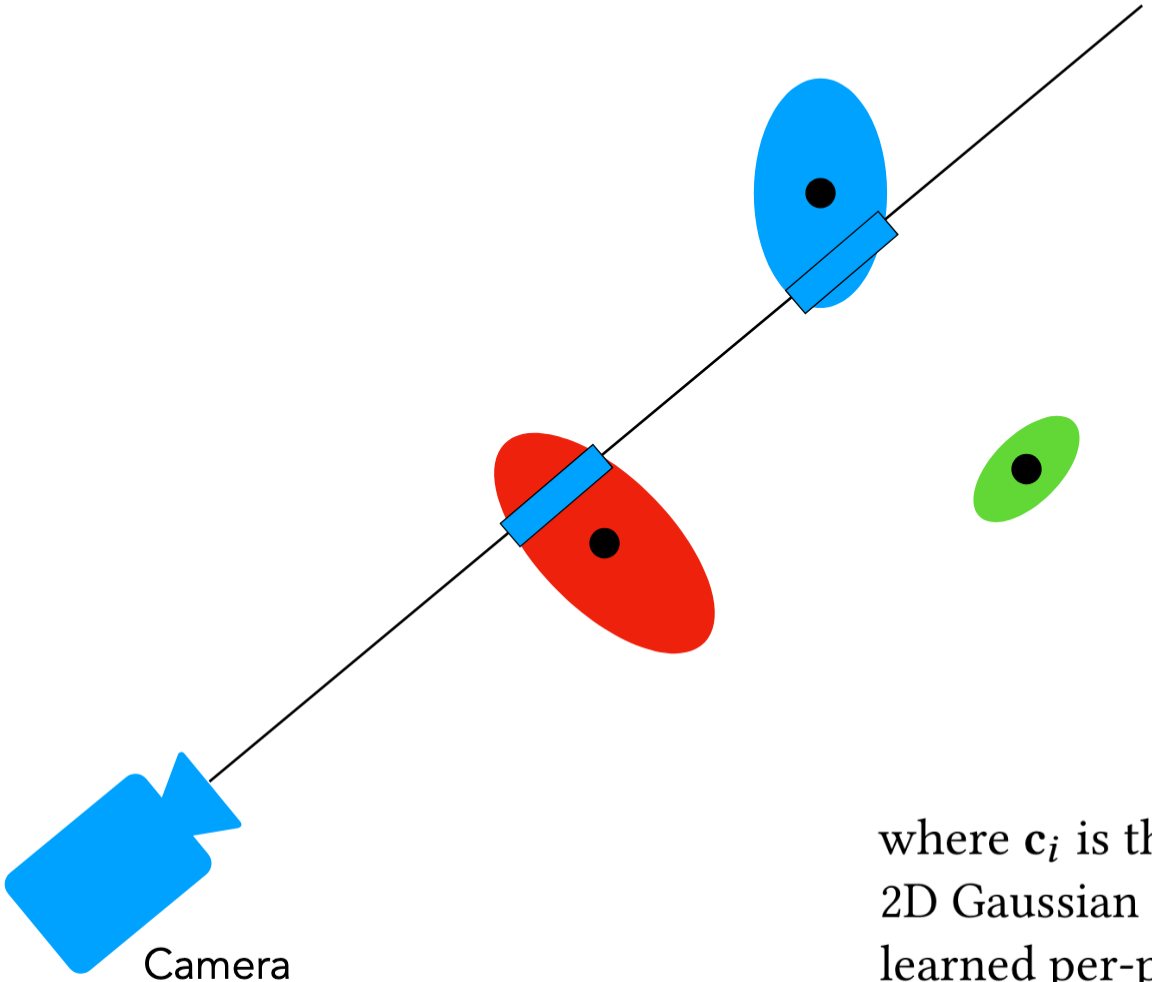
How to Obtain Jacobian?

- Jacobian $\boldsymbol{J_{u_k}}$ is given by the derivative of $\boldsymbol{m}$ at the point $\boldsymbol{u_k}$ with focal length f_x, f_y:

$$J_{\boldsymbol{u_k}} = \frac{\partial m}{\partial \boldsymbol{u}}(\boldsymbol{u_k}) = \begin{bmatrix} \frac{1}{u_{k,2}} f_x & 0 & -\frac{u_{k,0}}{u_{k,2}^2} f_x \\ 0 & \frac{1}{u_{k,2}} f_y & -\frac{u_{k,1}}{u_{k,2}^2} f_y \\ \frac{u_{k,0}}{\|\boldsymbol{u_k}\|} & \frac{u_{k,1}}{\|\boldsymbol{u_k}\|} & \frac{u_{k,2}}{\|\boldsymbol{u_k}\|} \end{bmatrix}$$

EWA Splattering. Zwicker et al.

Can compute volume rendering integral without ever sampling a single 3D point in space!



For each ray, ignore the non-overlapping 3D Gaussians, and sort the rest by ascending depth. Render with the known formulation:

$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{3}$$

where $\mathbf{c}_i$ is the color of each point and α_i is given by evaluating a 2D Gaussian with covariance Σ [Yifan et al. 2019] multiplied with a learned per-point opacity.

Slide credit V. Sitzmann

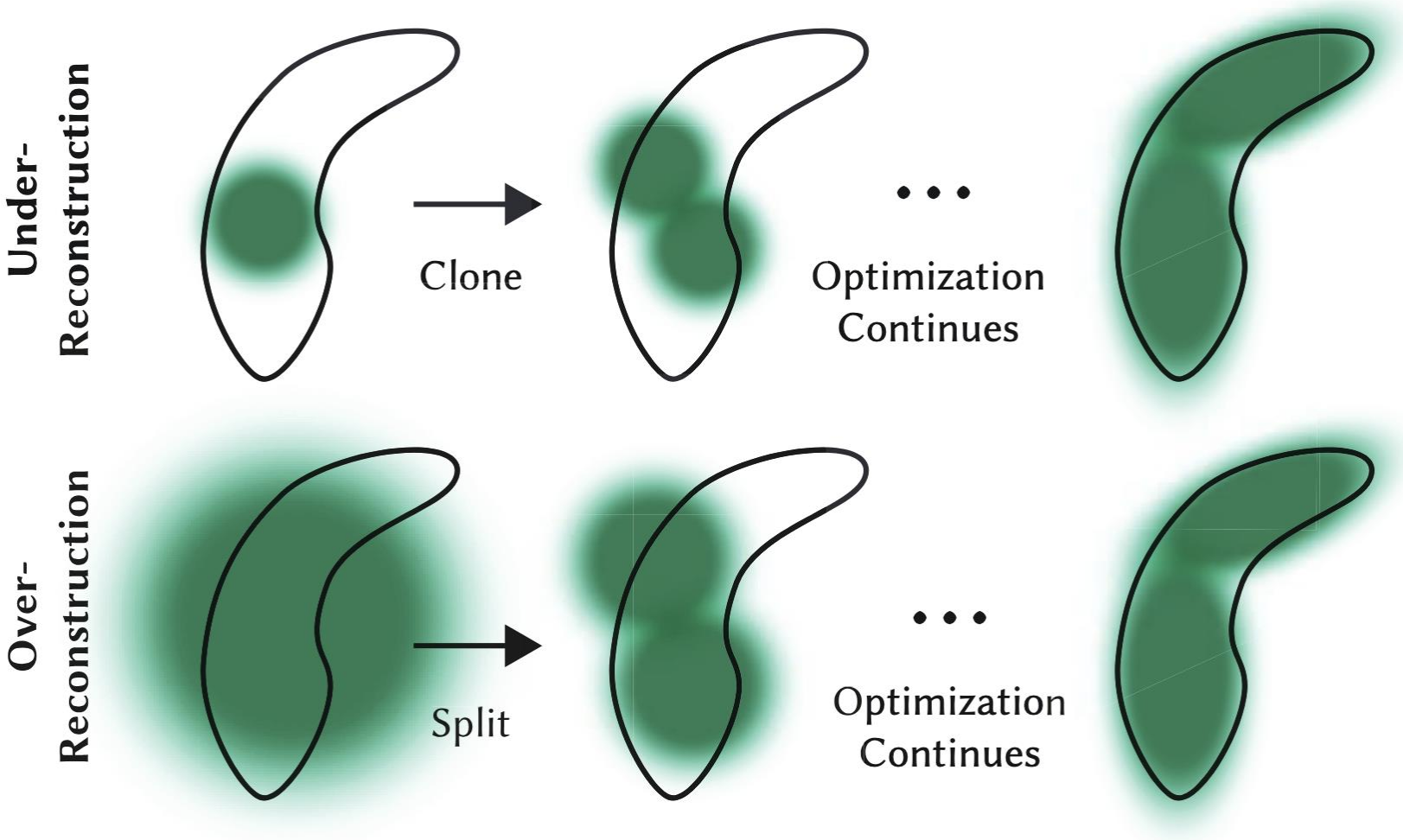
Problem 1: Local Minima

Initialize the mean of each 3D Gaussian with a random position

Initialize the mean of each 3D Gaussian with a 3D position reconstructed by Structure from Motion

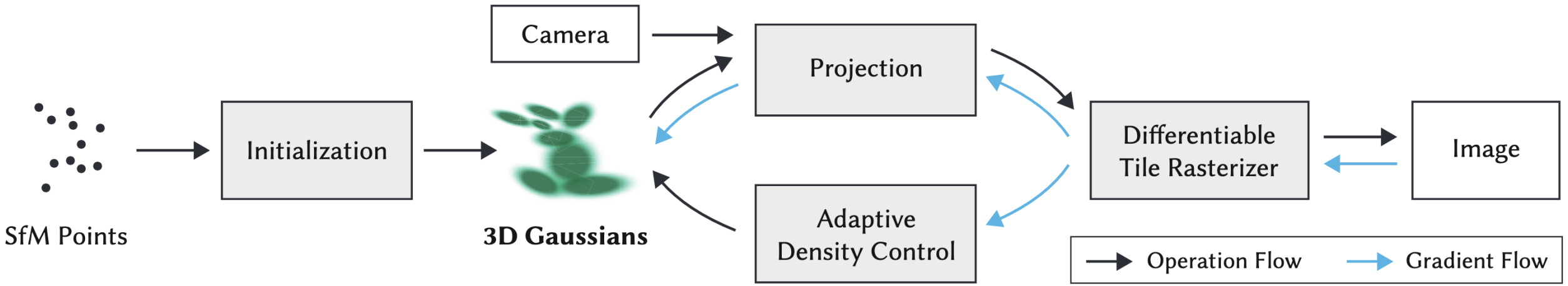
3D Gaussian Splatting for Real-Time Radiance Field Rendering. Kerbl et al.

Problem 2: Unevenly Distributed 3D Gaussians



3D Gaussian Splatting for Real-Time Radiance Field Rendering. Kerbl et al.

The Overall Framework of 3D Gaussian Splatting



3D Gaussian Splatting for Real-Time Radiance Field Rendering. Kerbl et al.

Render Scenes with 3D Gaussian Splatting

3D Gaussian Splatting for Real-Time Radiance Field Rendering. Kerbl et al.

Render Scenes with 3D Gaussian Splatting

3D Gaussian Splatting for Real-Time Radiance Field Rendering. Kerbl et al.

3D Gaussians are Explicit, Facilitating Tracking

Dynamic 3D Gaussians:
Tracking by Persistent Dynamic View Synthesis. Luiten et al.

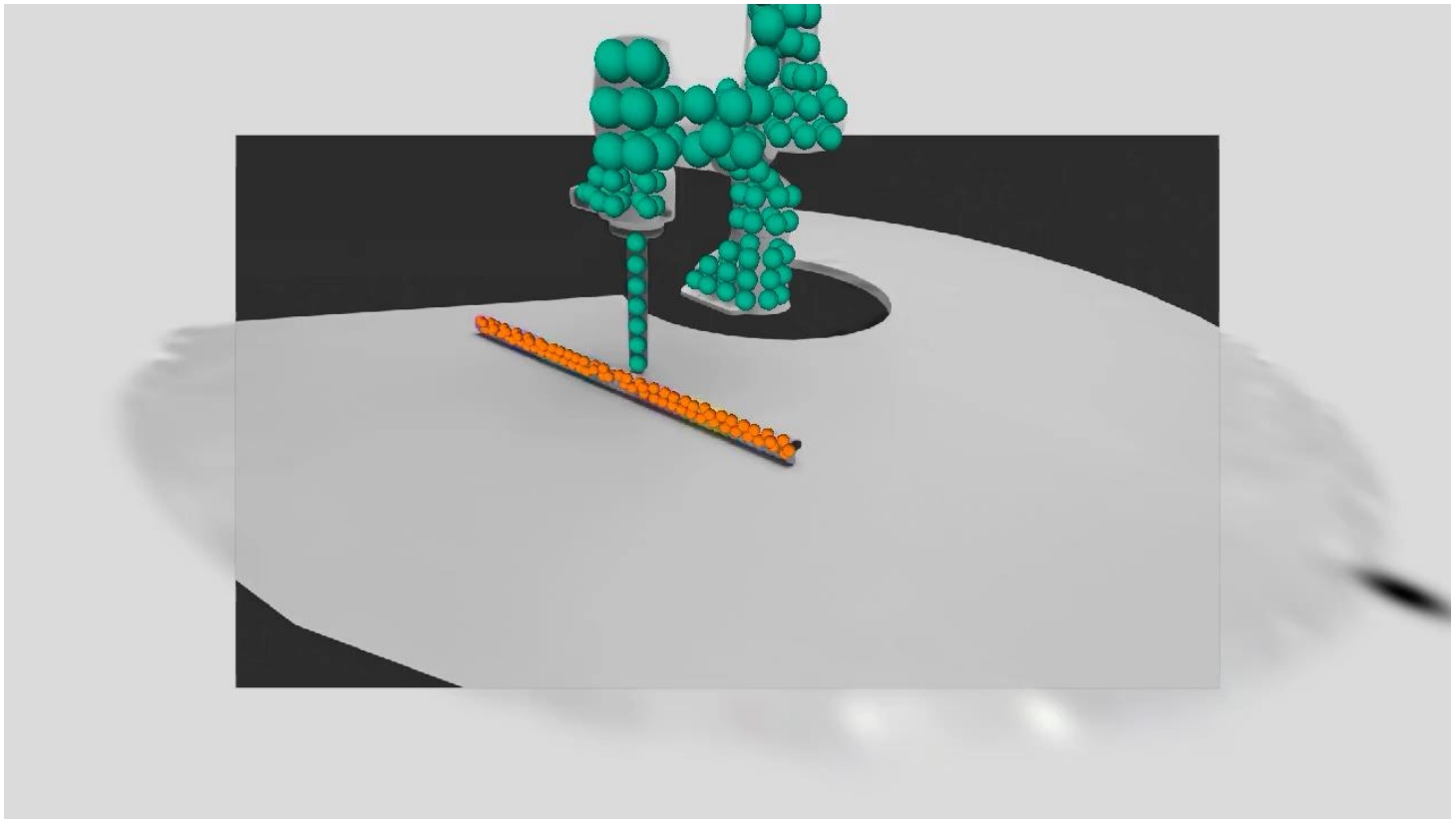
3D Gaussians are Explicit, Facilitating Tracking

Dynamic 3D Gaussians:
Tracking by Persistent Dynamic View Synthesis. Luiten et al.

3D Gaussians are Explicit, Easy to Compose

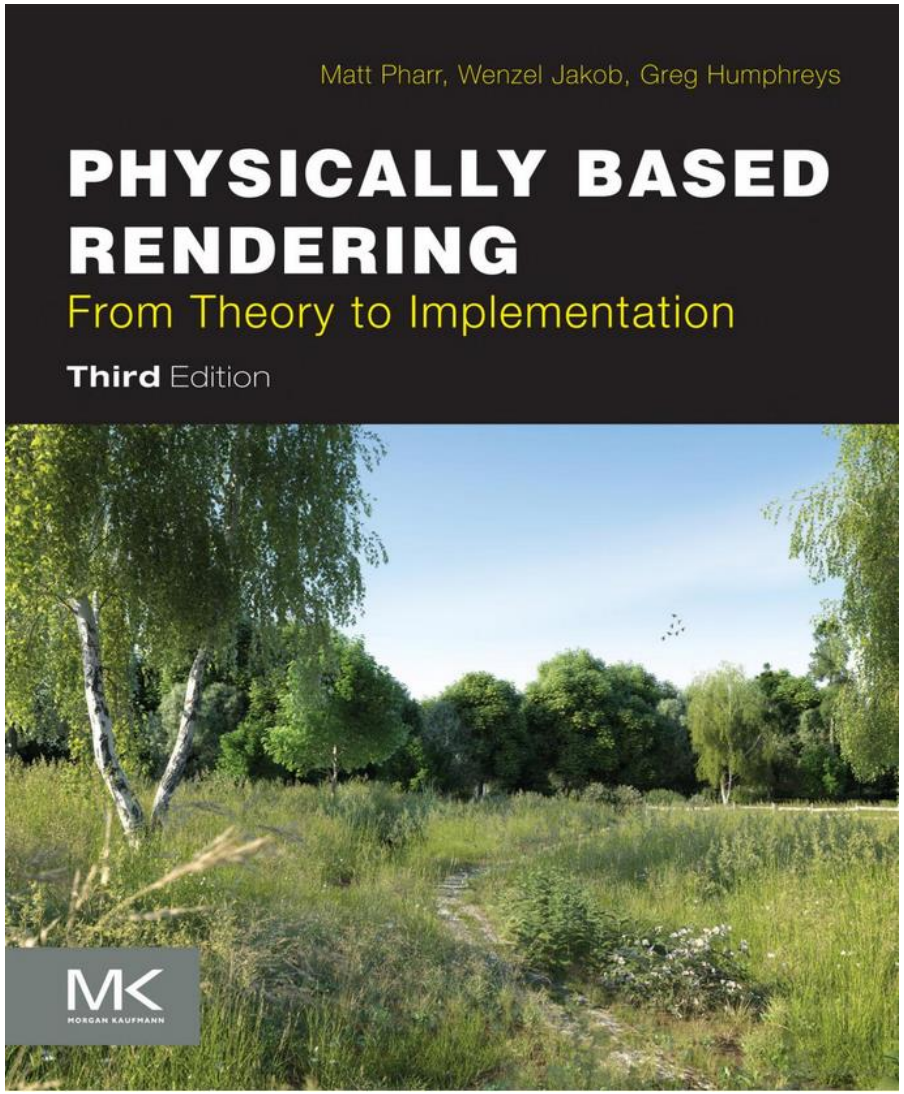
Dynamic 3D Gaussians:
Tracking by Persistent Dynamic View Synthesis. Luiten et al.

3D Gaussians are Explicit, Facilitating Physical Modeling



Physically Embodied Gaussian Splatting: A Visually Learnt and Physically Grounded 3D Representation for Robotics. Abou-Chakra et al.

Learn More About Rendering



What We Will Cover Next Week

- Modeling in 4D