CIS 5800

Machine Perception

Instructor: Lingjie Liu
Lec 25: April 30, 2025

Administrivia

Homework 5 (optional, 6 points for grade compensation) has been released. It and the small projects are due on May 14.

Final exam coming up

- Date reminder: Wednesday May 7, 3-5pm in DRLB A1. (Info is on courses.upenn.edu)
- Syllabus: Mainly the material covered in class after Wed March 19 (not covered by mid-term exam).
- Review lecture in the last class on Wed April 30.
- If you are unable to attend the midterm exam in person on May 7, please complete the form by April 30: https://forms.gle/JwaAxrKGfBzoo6z77
- Also, you need to contact the Weingarten Office for academic accommodations and send me the paperwork or approval from the Weingarten Office.

Putting the class in perspective in the context of computer vision and a quick overview of “visual recognition”

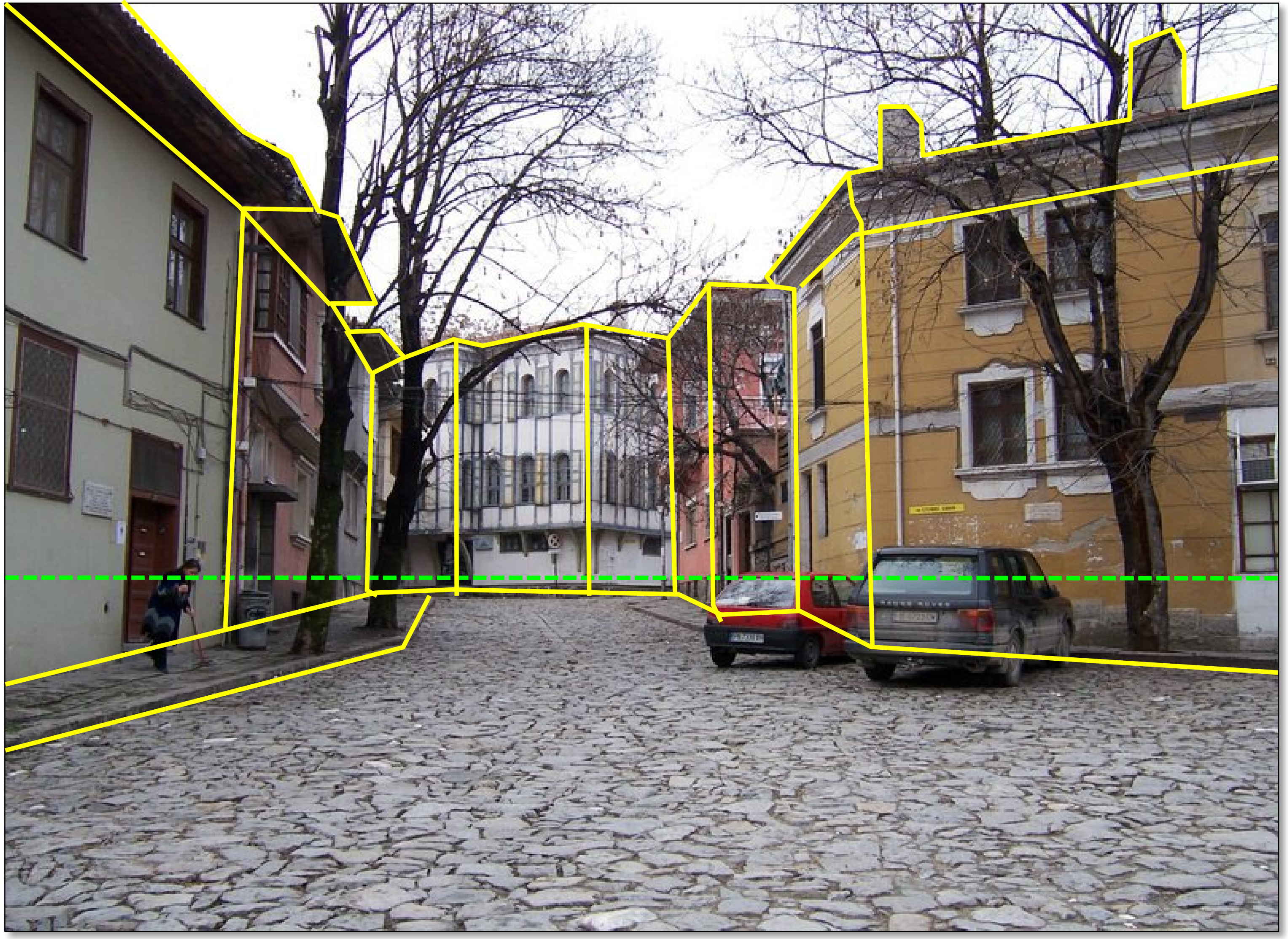
What Info can be Extracted from Images?

Source: S. Lazebnik

What Info can be Extracted from Images?

This class!

geometric information

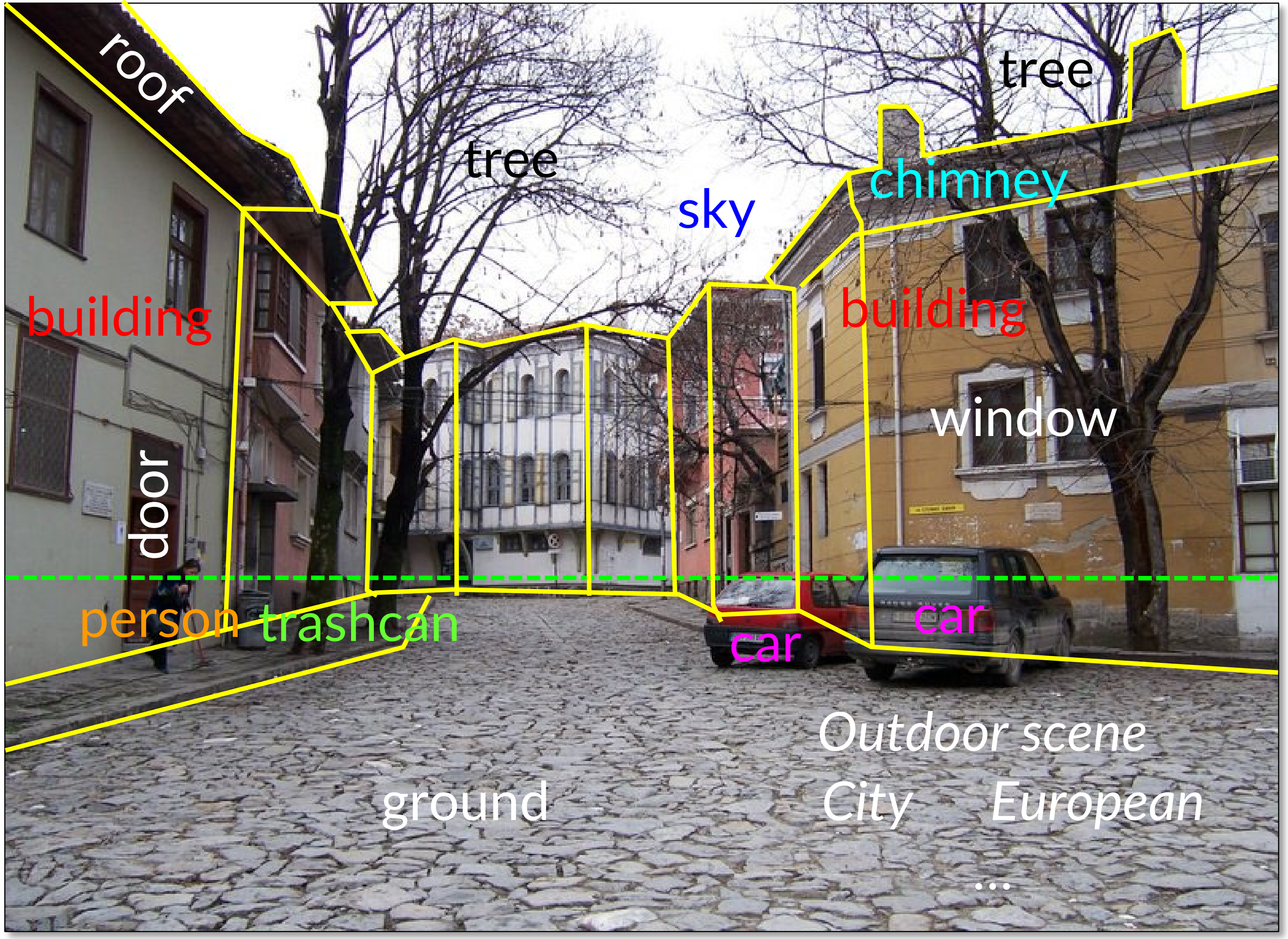


Source: S. Lazebnik

What Info can be Extracted from Images?

This class!

geometric information

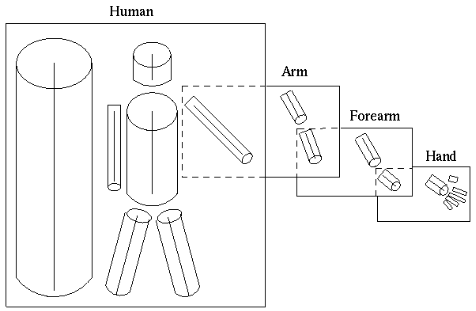


"visual recognition"

semantic information

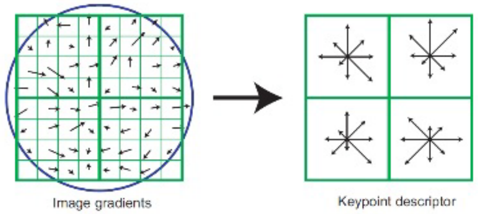
Source: S. Lazebnik

ML in Computer Vision



The *very* old: 1960's - Mid 1990's

Image → hand-def. features → hand-def. classifier

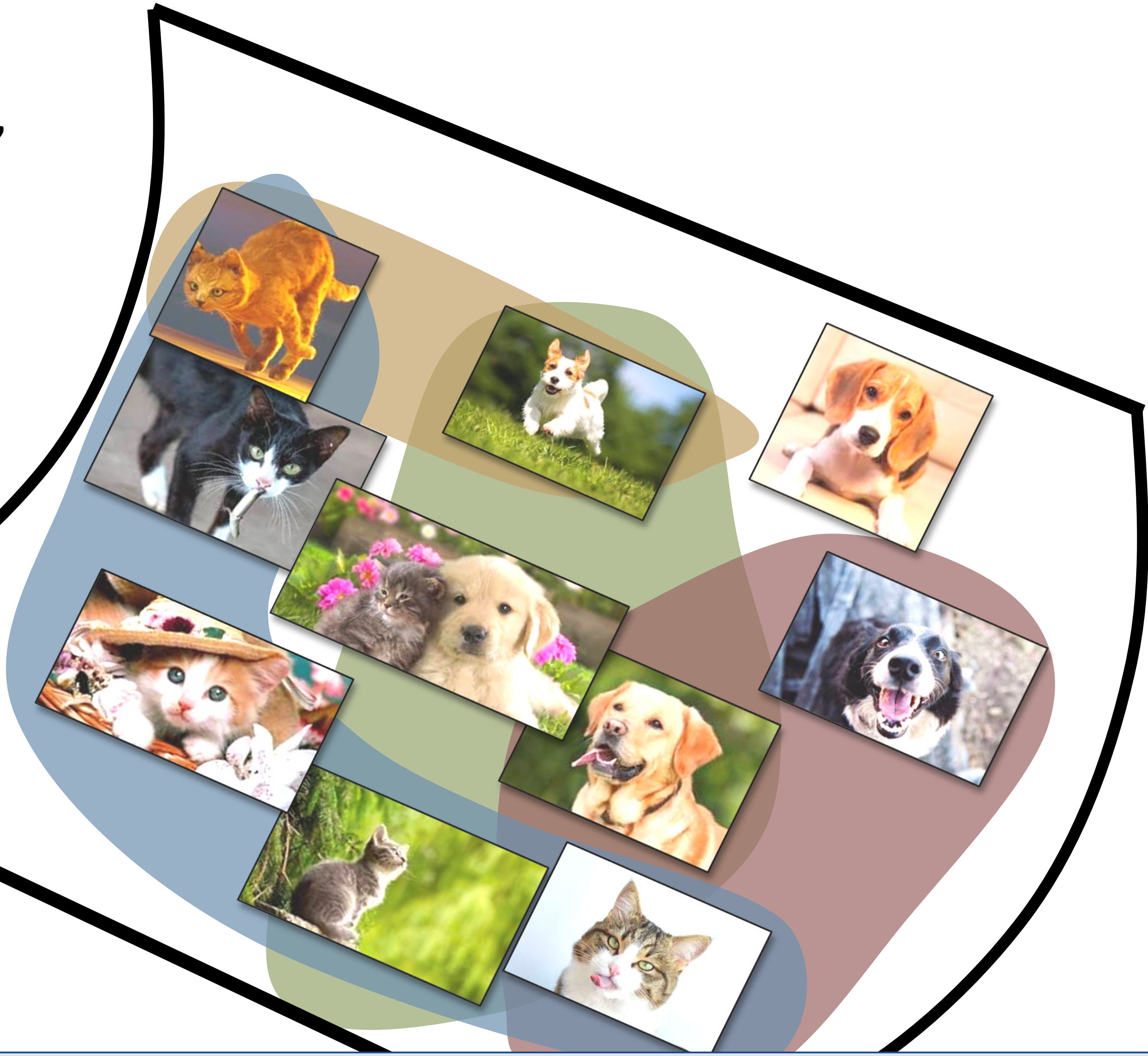


The old: Mid 1990's – 2012

Image → hand-def. features → learned classifier

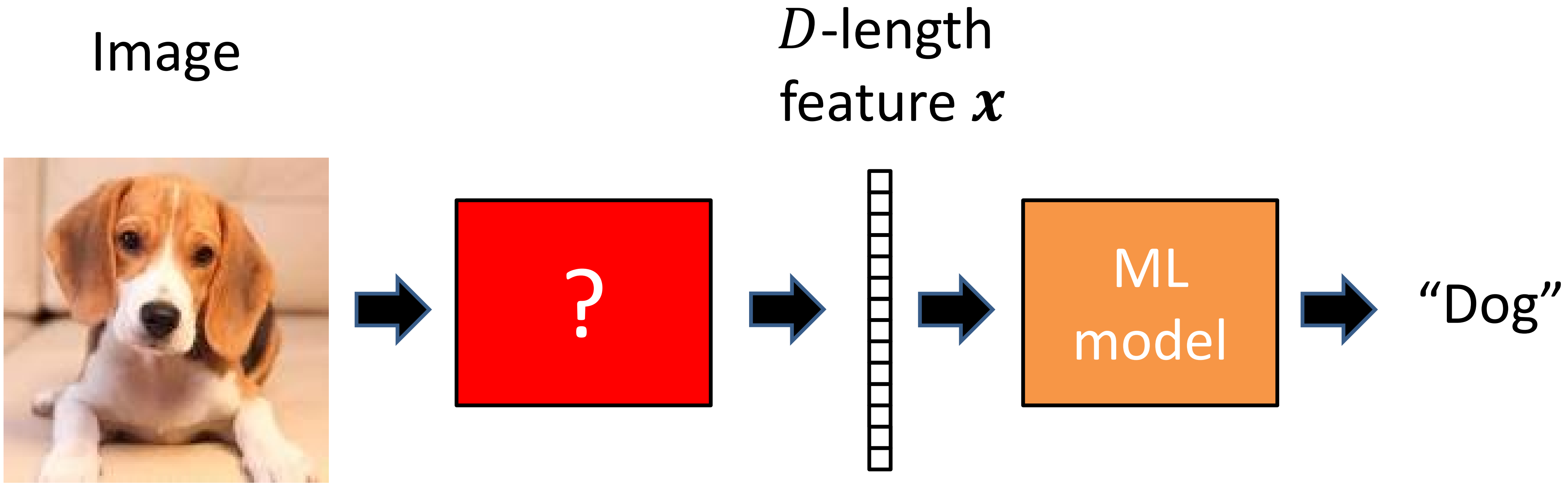
What Should Good Visual Features Do?

What is a "good" feature space?



Good features make useful tasks easy to perform.

What Should Good Visual Features Do?



How should we produce such good features?

Most Feature Extraction Frameworks Pre-2012

Step 1: Focus on "interest points" rather than all pixels

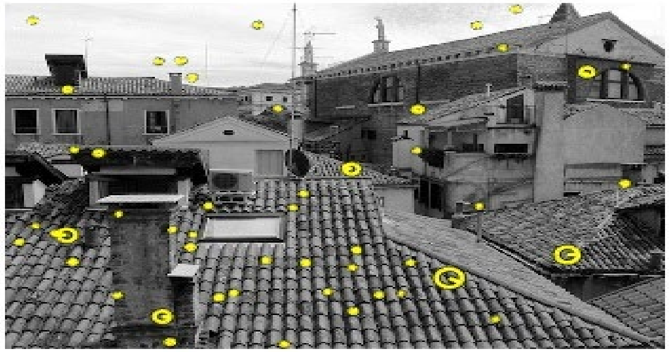
E.g. corner points, "difference of gaussians", or even a uniform grid

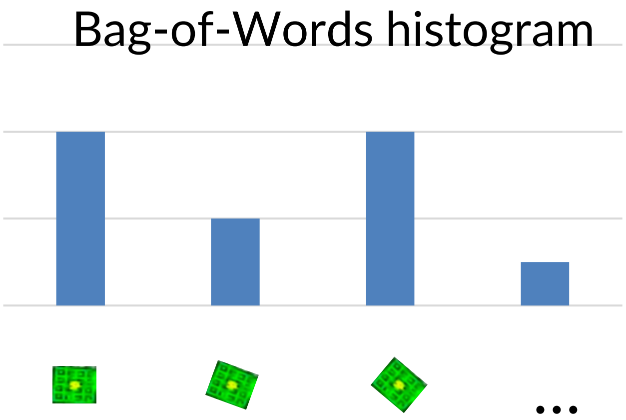
Step 2: Compute features at interest points.

E.g. "SIFT", "HOG", "SURF", "GIST", etc.

Step 3: Convert to fixed-dimensional feature vector by measuring statistics of the features such as histograms

E.g. "Bag of Words", "Spatial Pyramids", etc.

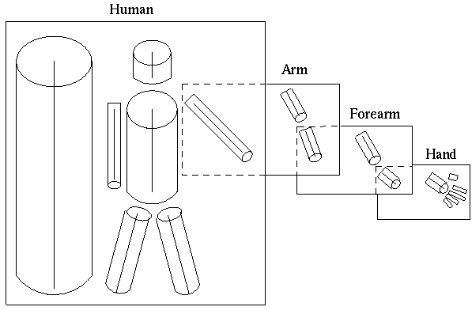




See libraries like VLFeat and OpenCV

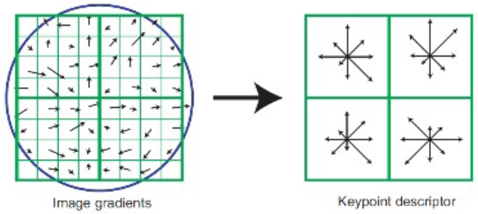
Use your favorite ML model now!

Machine Learning for *Semantic* Computer Vision



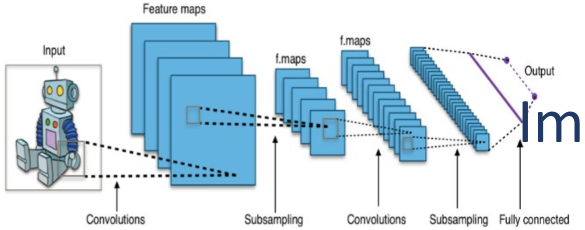
The *very* old: 1960's - Mid 1990's

Image → hand-def. features → hand-def. classifier



The old: Mid 1990's – 2012

Image → hand-def. features → learned classifier

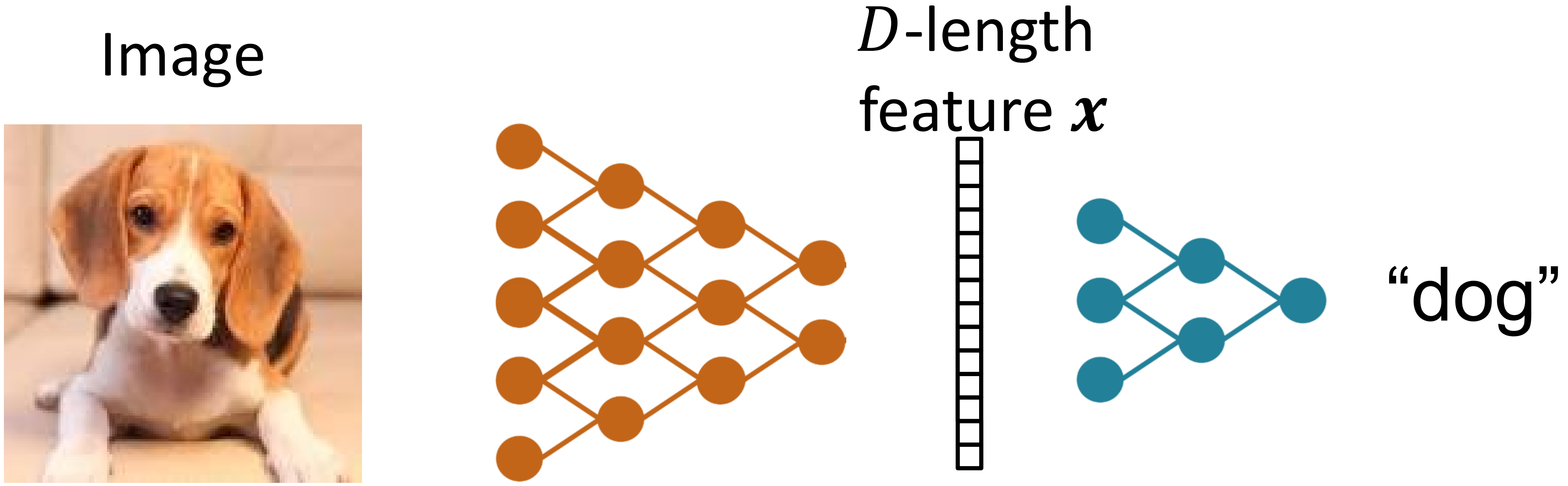


The new: 2012 – ?

Image → jointly learned features + classifier with "deep" multi-layer neural networks

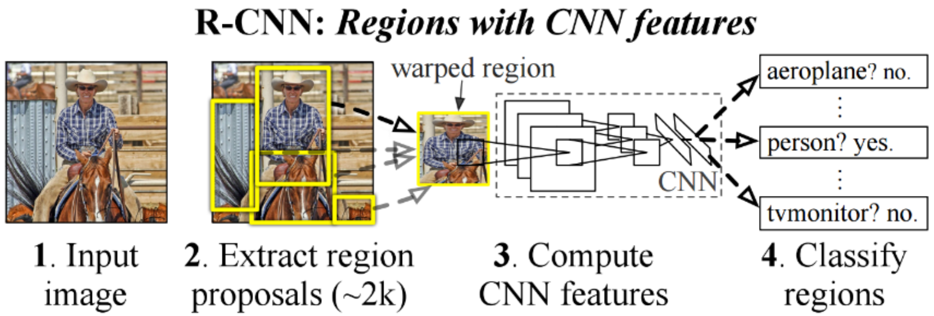
“Deep” Learning

“Deep” multi-layer neural networks are **representation learners**.
Every layer improves upon its preceding layer, tailoring the representation to the task.



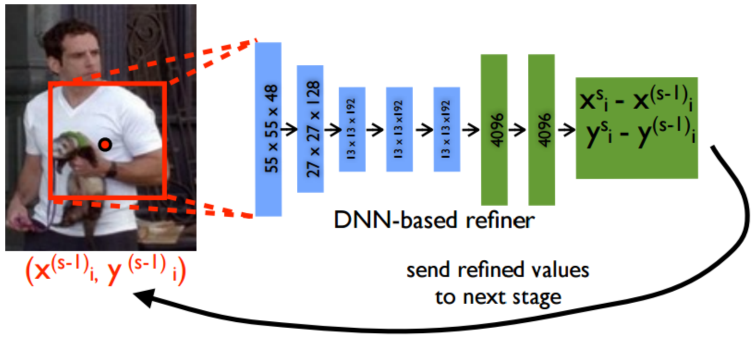
Some sample applications of semantic vision

[Girshick et al. CVPR14]



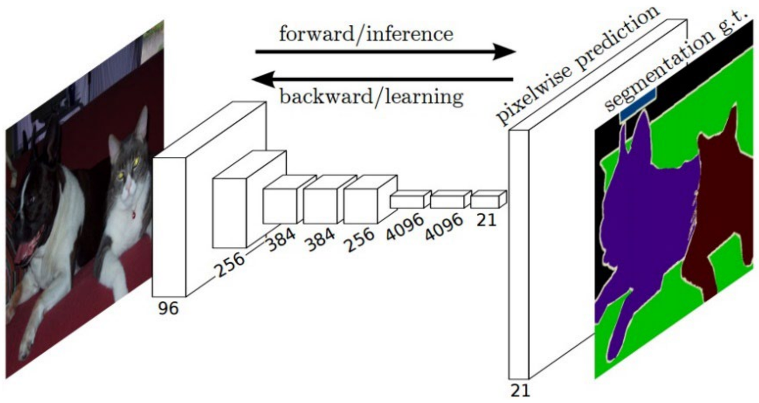
Object detection

[Toshev et al. CVPR14]



Pose detection (regression)

[Long et al. CVPR15]

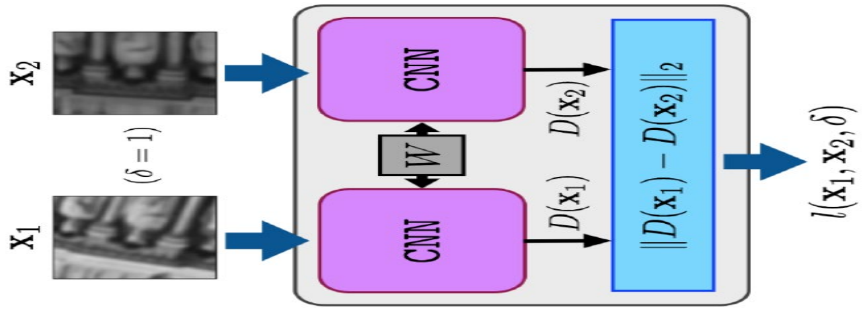


Semantic segmentation

Examples courtesy Jia-Bin Huang

Some sample applications of semantic vision

[Chopra et al. CVPR05]



Picture source

Similarity metric learning

[Dosovitskiy et al. CVPR15]

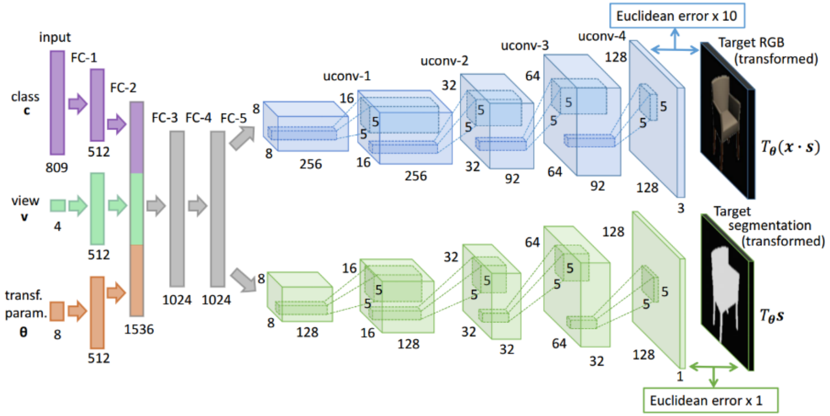
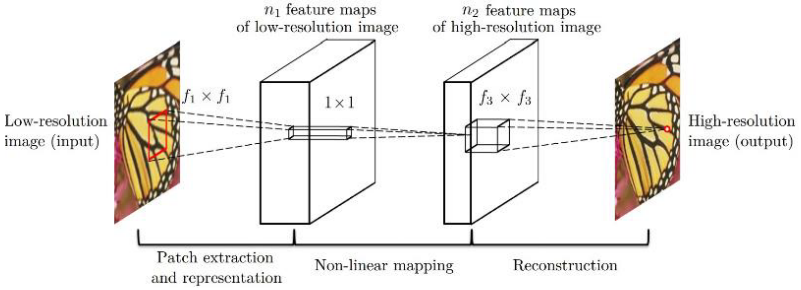


Image generation

[Dong et al. ECCV 2014]



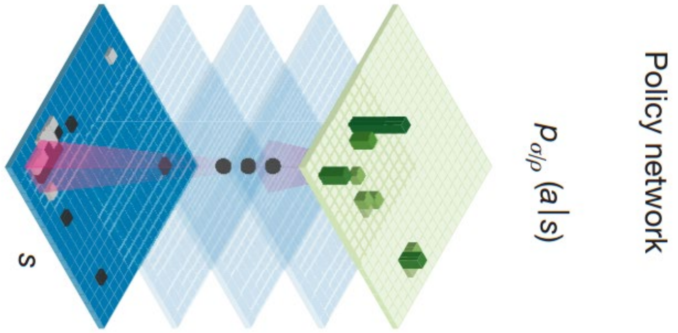
Low-level image processing: (superresolution, deblurring, image quality etc.)

Examples courtesy Jia-Bin Huang

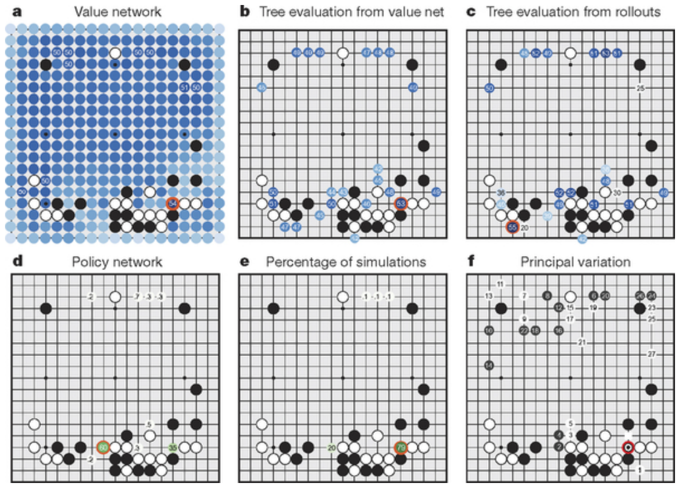
Game playing from visual inputs

CNN + Reinforcement learning

[Mnih et al, Nature' 15]



[Silver et al, Nature '16]



Generating art

Paper: Gatys et al, “Neural ... Style”, arXiv ‘15
Code (torch): https://github.com/jcjohnson/neural-style

See if you can tell artists’ originals from machine style imitations at: http://turing.deepart.io/

Where to learn more about ML and semantic vision?

Machine learning courses:

CIS 519, 520, 522 usually cover semantic computer vision briefly, as an application domain for machine learning techniques

CIS 581 Computer Vision & Computational Photography

The basics of image processing and semantic computer vision.

CIS 680 Advanced Machine Perception

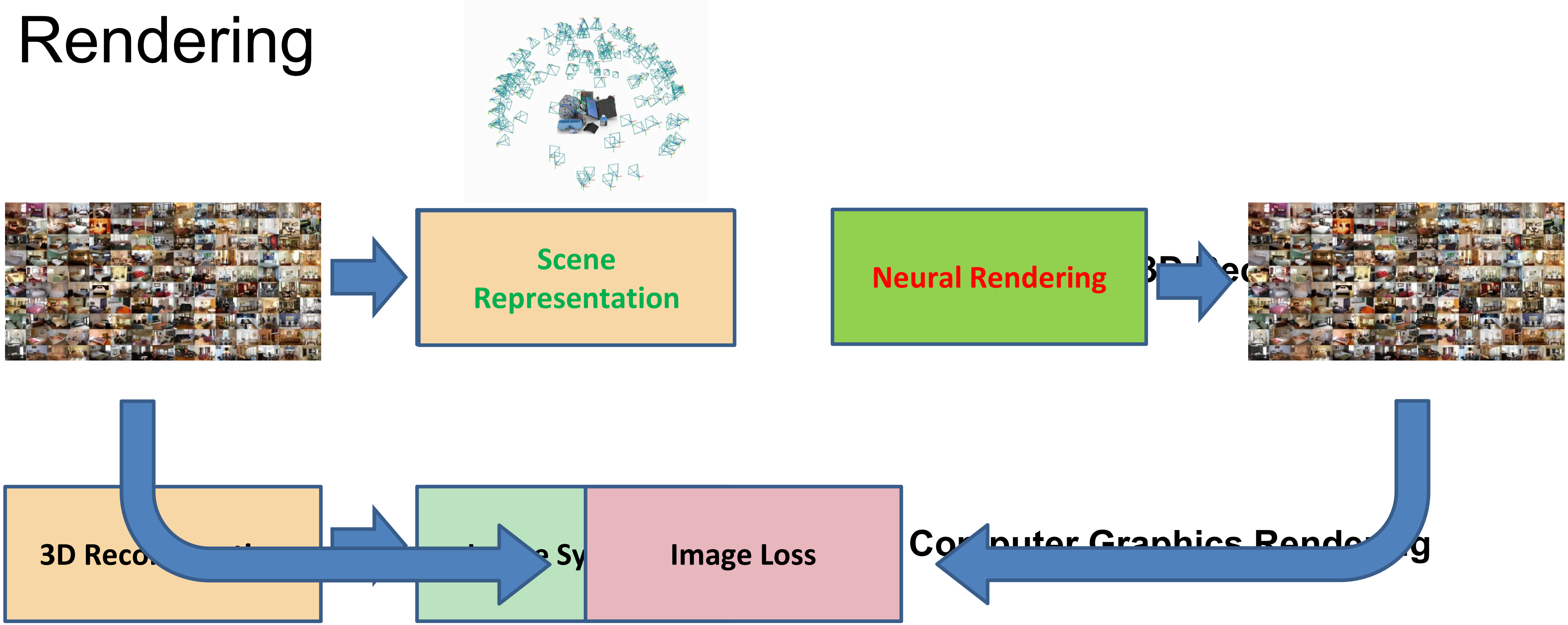
Cutting-edge techniques in semantic (largely) computer vision, best taken after some introduction to ML.

CIS 7000 Advanced Topics

There is lots of ML in geometric vision too!

(The following slides are based on materials from Ben Mildenhall, Vincent Sitzmann and Stephen Lombardi)

Neural Scene Representation and Neural Rendering



Neural Radiance Fields (NeRF)

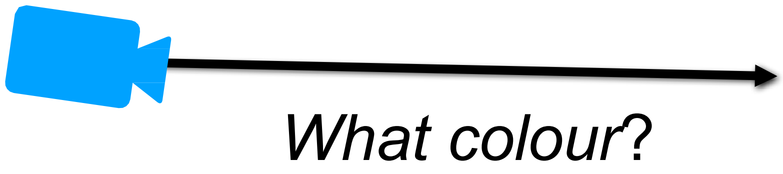
Mildenhall, Srinivasan, Tancik, Barron, Ramamoorthi, Ng

ECCV 2020

Neural Volumetric Rendering

Neural Volumetric Rendering

querying the radiance value
along rays through 3D space



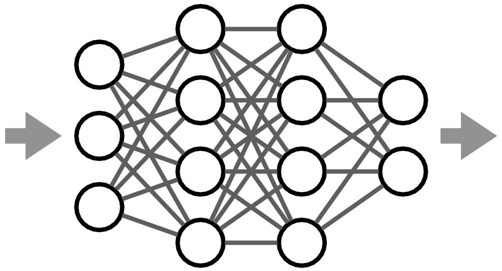
Neural Volumetric Rendering

continuous, differentiable
rendering model without
concrete ray/surface intersections

Neural Volumetric Rendering

using a neural network as a
scene representation, rather
than a voxel grid of data

(x, y, z)



Scene properties

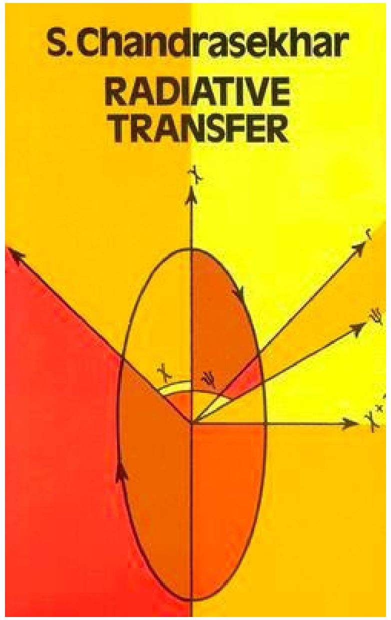
Inputs: sparse, unstructured photographs of a scene

Outputs: representation allowing us to render *new* views of that scene

Overview

- Volumetric rendering math
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

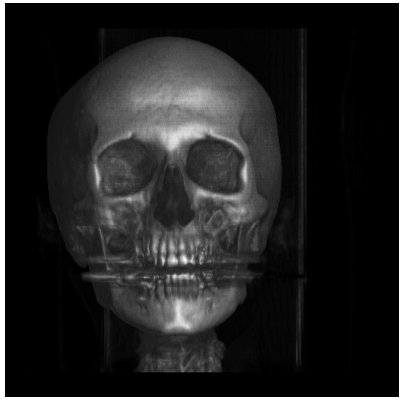
Traditional volumetric rendering



- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Adapted for visualising medical data and linked with alpha compositing
- Modern path tracers use sophisticated Monte Carlo methods to render volumetric effects

Chandrasekhar 1950, *Radiative Transfer*
Kajia 1984, *Ray Tracing Volume Densities*
Levoy 1988, *Display of Surfaces from Volume Data*
Max 1995, *Optical Models for Direct Volume Rendering*
Porter and Duff 1984, *Compositing Digital Images*
Novak et al 2018, *Monte Carlo methods for physically based volume rendering*

Traditional volumetric rendering



Medical data visualisation [Levoy]

- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Adapted for visualising medical data and linked with alpha compositing
- Modern path tracers use sophisticated Monte Carlo methods to render volumetric effects

Alpha compositing [Porter and Duff]

Chandrasekhar 1950, *Radiative Transfer*
Kajia 1984, *Ray Tracing Volume Densities*
Levoy 1988, *Display of Surfaces from Volume Data*
Max 1995, *Optical Models for Direct Volume Rendering*
Porter and Duff 1984, *Compositing Digital Images*
Novak et al 2018, *Monte Carlo methods for physically based volume rendering*

Traditional volumetric rendering

Physically-based Monte Carlo rendering [Novak et al]

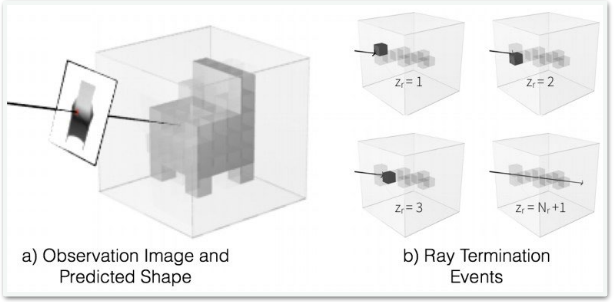
- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Adapted for visualising medical data and linked with alpha compositing
- Modern path tracers use sophisticated Monte Carlo methods to render volumetric effects

Chandrasekhar 1950, *Radiative Transfer*
Kajia 1984, *Ray Tracing Volume Densities*
Levoy 1988, *Display of Surfaces from Volume Data*
Max 1995, *Optical Models for Direct Volume Rendering*
Porter and Duff 1984, *Compositing Digital Images*
Novak et al 2018, *Monte Carlo methods for physically based volume rendering*

Volumetric rendering and machine learning

- Various volume-rendering-esque methods devised for 3D shape reconstruction methods

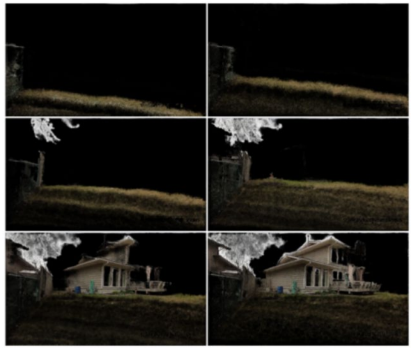
- Scaled up to higher resolution volumes to achieve excellent view synthesis results



"Probabilistic" voxel grid rendering [Tulsiani et al]

Tulsiani et al 2017, *Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency*
Henzler et al 2019, *Escaping Plato's Cave: 3D Shape From Adversarial Rendering*
Zhou et al 2018, *Stereo Magnification: Learning View Synthesis using Multiplane Images*
Lombardi et al 2019, *Neural Volumes: Learning Dynamic Renderable Volumes from Images*

Volumetric rendering and machine learning



Slices from a volumetric scene representation [Zhou et al]

View synthesis from a dynamic voxel grid [Lombardi et al]

- Various volume-rendering-esque methods devised for 3D shape reconstruction methods
- Scaled up to higher resolution voxel grids, ML methods can achieve excellent view synthesis results

Tulsiani et al 2017, *Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency*

Henzler et al 2019, *Escaping Plato's Cave: 3D Shape From Adversarial Rendering*

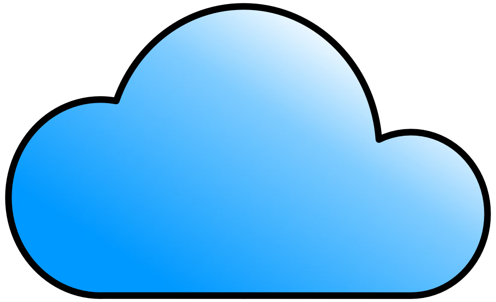
Zhou et al 2018, *Stereo Magnification: Learning View Synthesis using Multiplane Images*

Lombardi et al 2019, *Neural Volumes: Learning Dynamic Renderable Volumes from Images*

Volumetric formulation for NeRF

Max and Chen 2010, *Local and Global Illumination in the Volume Rendering Integral*

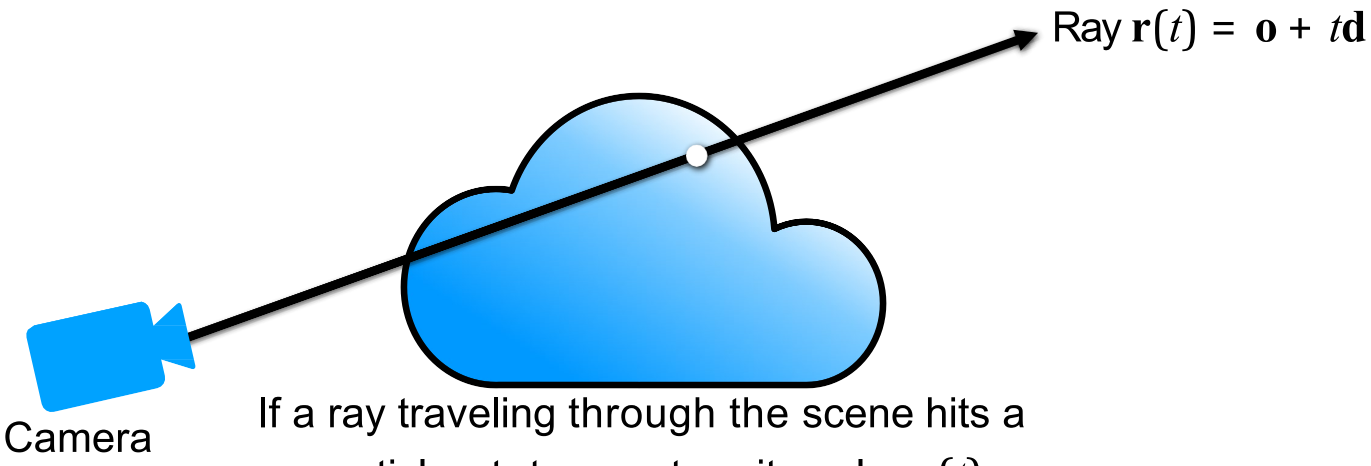
Volumetric formulation for NeRF



Scene is a cloud of tiny colored particles

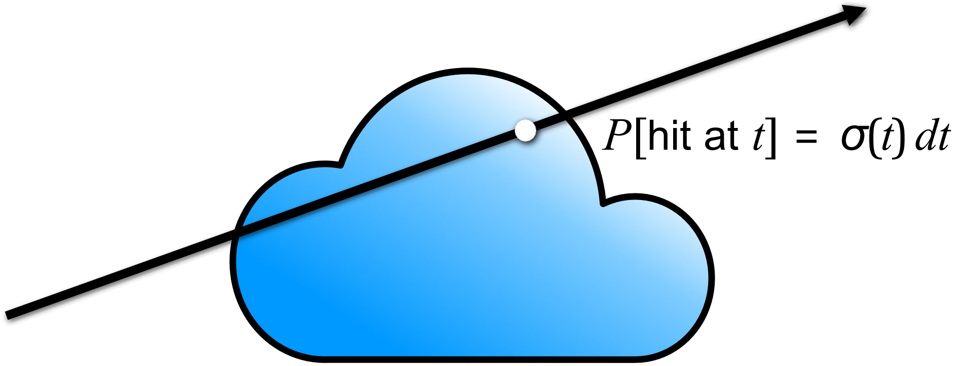
Max and Chen 2010, *Local and Global Illumination in the Volume Rendering Integral*

Volumetric formulation for NeRF



If a ray traveling through the scene hits a particle at t, we return its color $\mathbf{c}(t)$

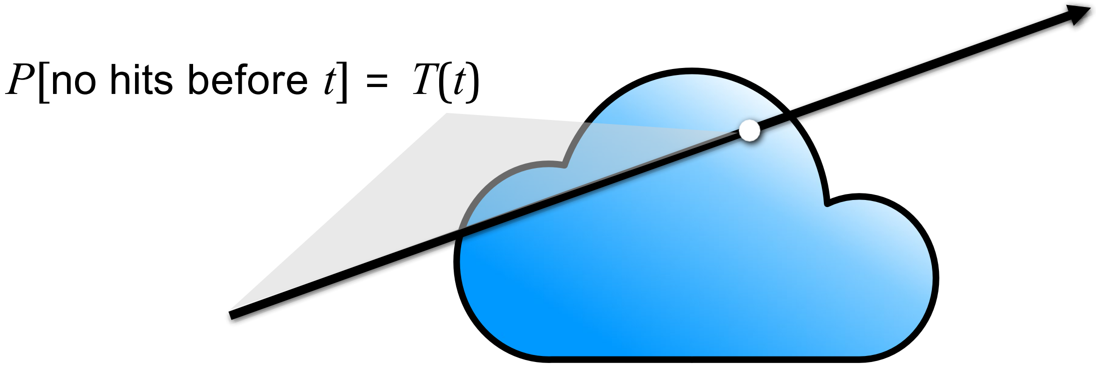
Volumetric formulation for NeRF



This notion is *probabilistic:* chance that ray stops in a small interval around t is $\sigma(t)\, dt$.

$\sigma(t)$ is known as the “volume density”

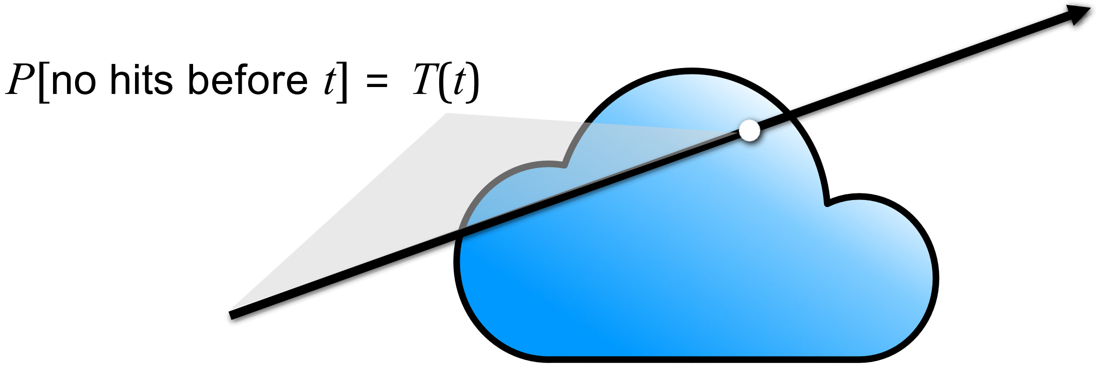
Volumetric formulation for NeRF



To determine if t is the *first* hit, need to know $T(t)$: probability that the ray didn't hit any particles earlier.

$T(t)$ is called "transmittance"

Volumetric formulation for NeRF



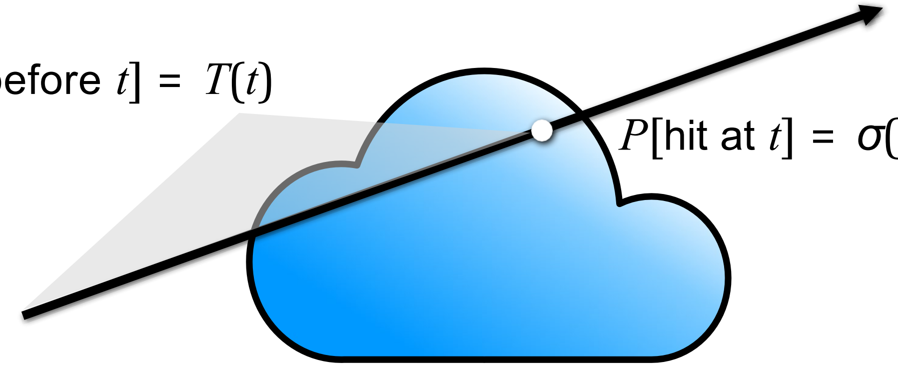
To determine if t is the *first* hit, need to know $T(t)$:

probability that the ray didn't hit any particles earlier.

$T(t)$ is called "transmittance"

We assume σ is known and want to use it to calculate $T(t)$

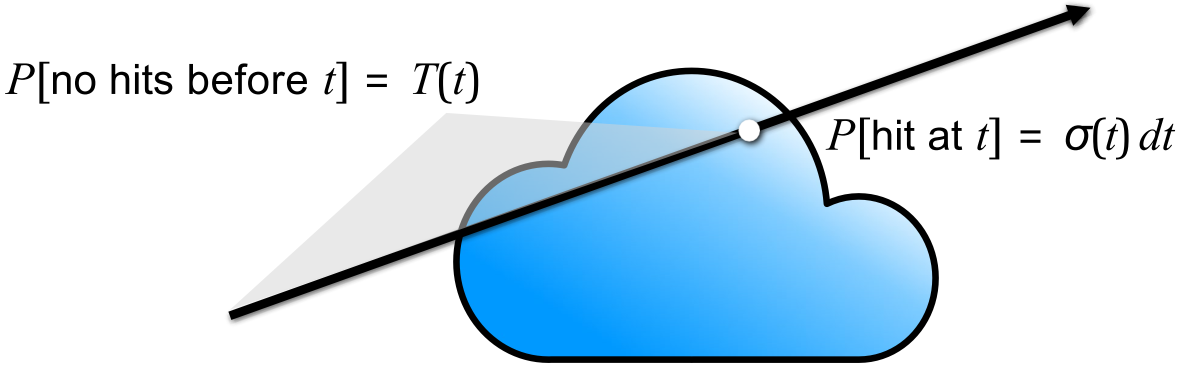
Volumetric formulation for NeRF



σ and T are related by the probability fact that

P[no hits before $t + dt$] = P[no hits before t] $\times P$[no hit at t]

Volumetric formulation for NeRF



These are related by the probability fact that

$$T(t + dt) = T(t) \times (1 - \sigma(t)dt)$$

Volumetric formulation for NeRF

$$T(t + dt) = T(t)(1 - \sigma(t)dt)$$

Volumetric formulation for NeRF

$$T(t + dt) = T(t)(1 - \sigma(t)dt)$$

Split up differential $\Rightarrow$ $$T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$$

Volumetric formulation for NeRF

$$T(t + dt) = T(t)(1 - \sigma(t)dt)$$

Split up differential $\Rightarrow$ $T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$

Rearrange $\Rightarrow$ $\dfrac{T'(t)}{T(t)}dt = -\sigma(t)dt$

Volumetric formulation for NeRF

$$T(t + dt) = T(t)(1 - \sigma(t)dt)$$

Split up differential $\Rightarrow$ $T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$

Rearrange $\Rightarrow$ $\dfrac{T'(t)}{T(t)}dt = -\sigma(t)dt$

Integrate $\Rightarrow$ $\log T(t) = -\int_{t_0}^{t} \sigma(s)ds$

$\Rightarrow$ $T(t) = \exp\left(-\int_{t_0}^{t} \sigma(t)\right)$

Volumetric formulation for NeRF

Thus, the probability that a ray first hits a particle at t is

$$T(t)\sigma(t)\,dt = \exp\left(-\int_{t_0}^{t}\sigma(t)\right)\sigma(t)\,dt$$

Volumetric formulation for NeRF

Thus, the probability that a ray first hits a particle at t is

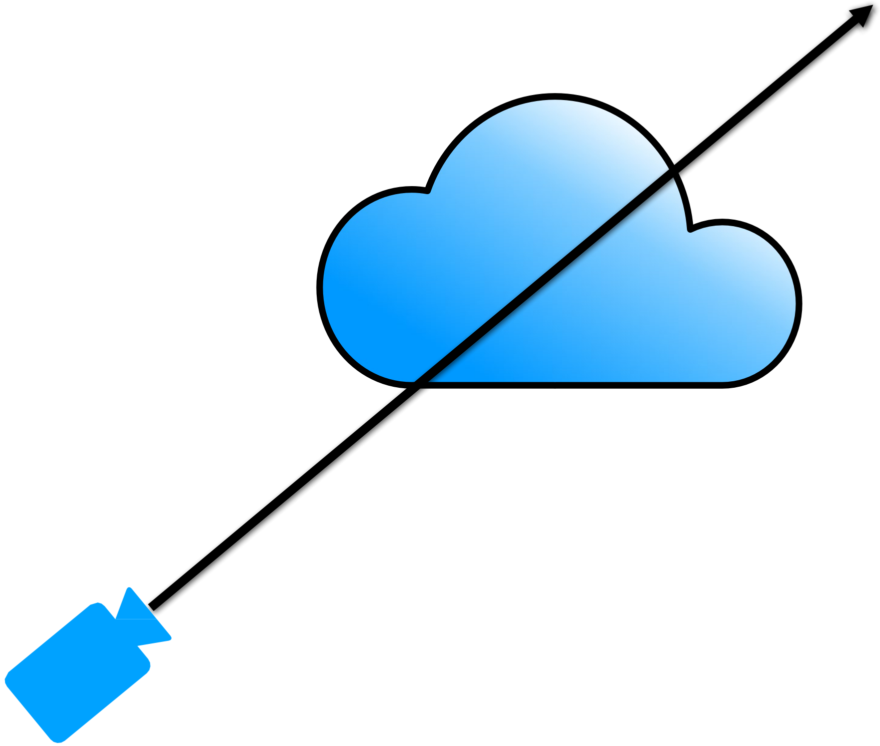
$$T(t)\sigma(t)\,dt = \exp\left(-\int_{t_0}^{t}\sigma(t)\right)\sigma(t)\,dt$$

And expected color returned by the ray will be

$$\int_{t_0}^{t_1} T(t)\sigma(t)\mathbf{c}(t)\,dt$$

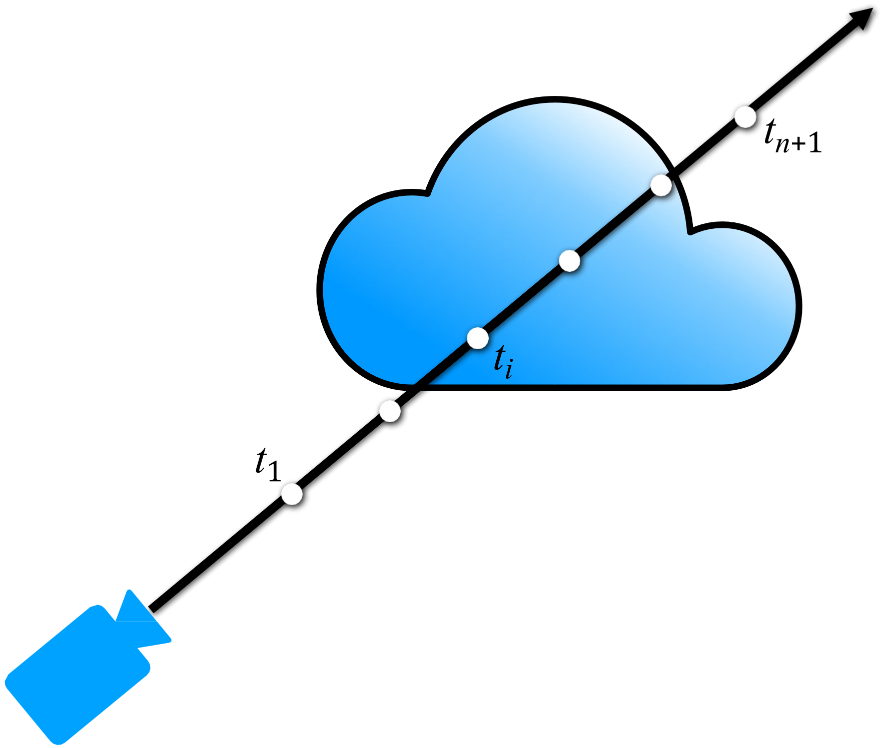
Note the nested integral!

Approximating the nested integral



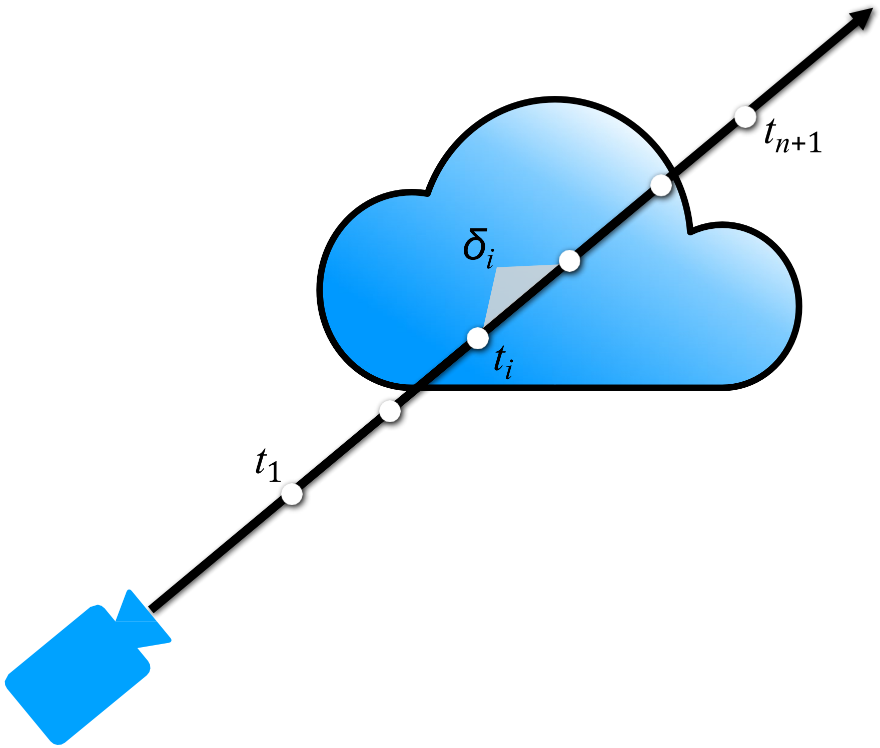
We use quadrature to approximate the nested integral,

Approximating the nested integral



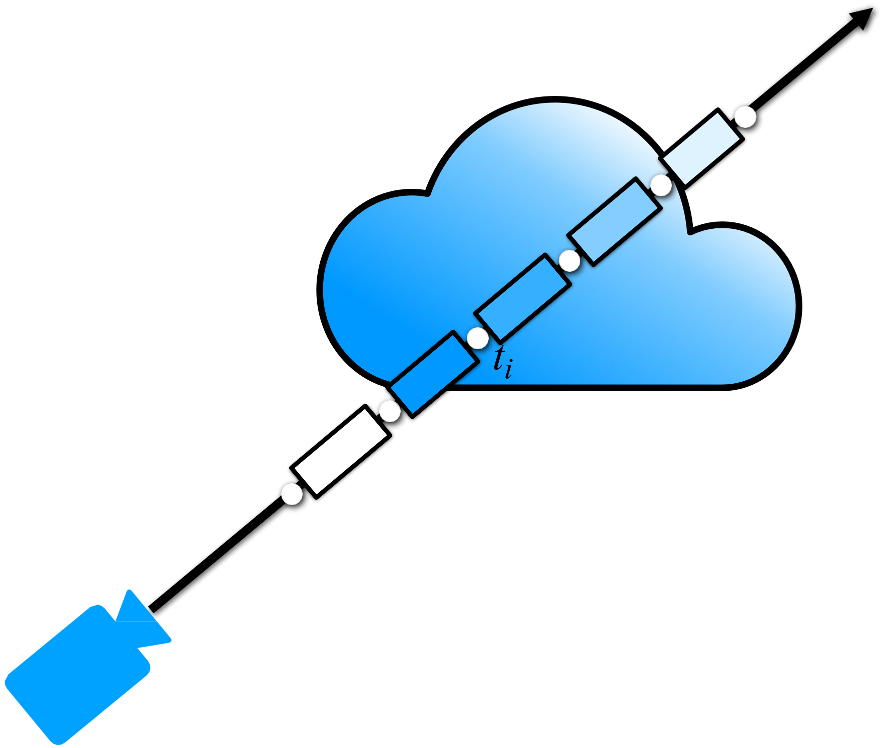
We use quadrature to approximate the nested integral, splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$

Approximating the nested integral



We use quadrature to approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$
with lengths $\delta_i = t_{i+1} - t_i$

Approximating the nested integral



We assume volume density and color are roughly constant within each interval

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt$$

This allows us to break the outer integral

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

This allows us to break the outer integral
into a sum of analytically tractable integrals

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} \boxed{T(t)}\sigma_i \mathbf{c}_i\,dt$$

Catch: piecewise constant density and color

do not imply constant transmittance!

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} \boxed{T(t)}\,\sigma_i \mathbf{c}_i\,dt$$

Catch: piecewise constant density and color **do not** imply constant transmittance!

Important to account for how early part of a segment blocks later part when σ_i is high

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

For $t \in [t_i, t_{i+1}]$, $T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i\,ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i\,ds\right)$

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n}\int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt$$

For $t \in [t_i, t_{i+1}]$, $T(t) = \exp\left(-\int_{t_1}^{t_i}\sigma_i\,ds\right)\exp\left(-\int_{t_i}^{t}\sigma_i\,ds\right)$

$$\exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right) = T_i$$

"How much is blocked by all previous segments?"

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i \,dt$$

For $t \in [t_i, t_{i+1}]$, $T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i \,ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i \,ds\right)$

"How much is blocked partway through the current segment?" $\exp\left(-\sigma_i(t - t_i)\right)$

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i \,dt$$

Substitute $$= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp\left(-\sigma_i (t - t_i)\right)\, dt$$

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

$$= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp\left(-\sigma_i(t - t_i)\right)\, dt$$

Integrate

$$= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \, \frac{\exp\left(-\sigma_i(t_{i+1} - t_i)\right) - 1}{-\sigma_i}$$

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

$$= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp\left(-\sigma_i(t - t_i)\right)\,dt$$

$$= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \frac{\exp\left(-\sigma_i(t_{i+1} - t_i)\right) - 1}{-\sigma_i}$$

Cancel σ_i

$$= \sum_{i=1}^{n} T_i \mathbf{c}_i\,(1 - \exp(-\sigma_i \delta_i))$$

Summary: volume rendering integral estimate

Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{c} \approx \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

colors

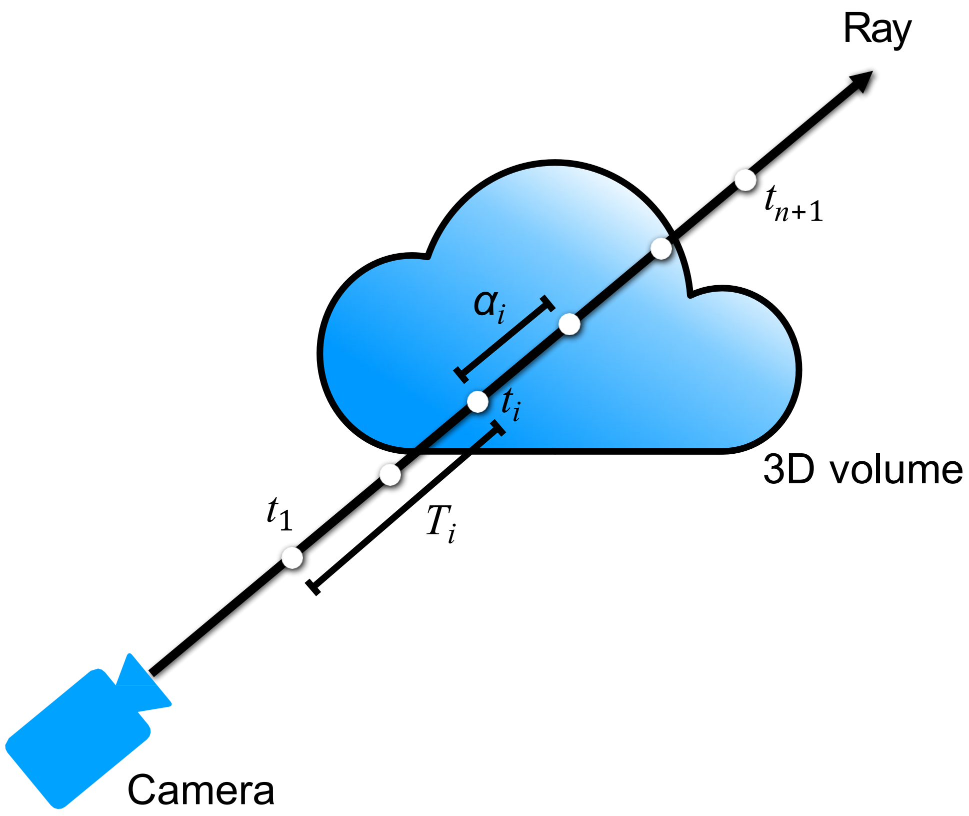
weights

How much light is blocked earlier along ray:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

How much light is contributed by ray segment i:

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$



Summary: volume rendering integral estimate

Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

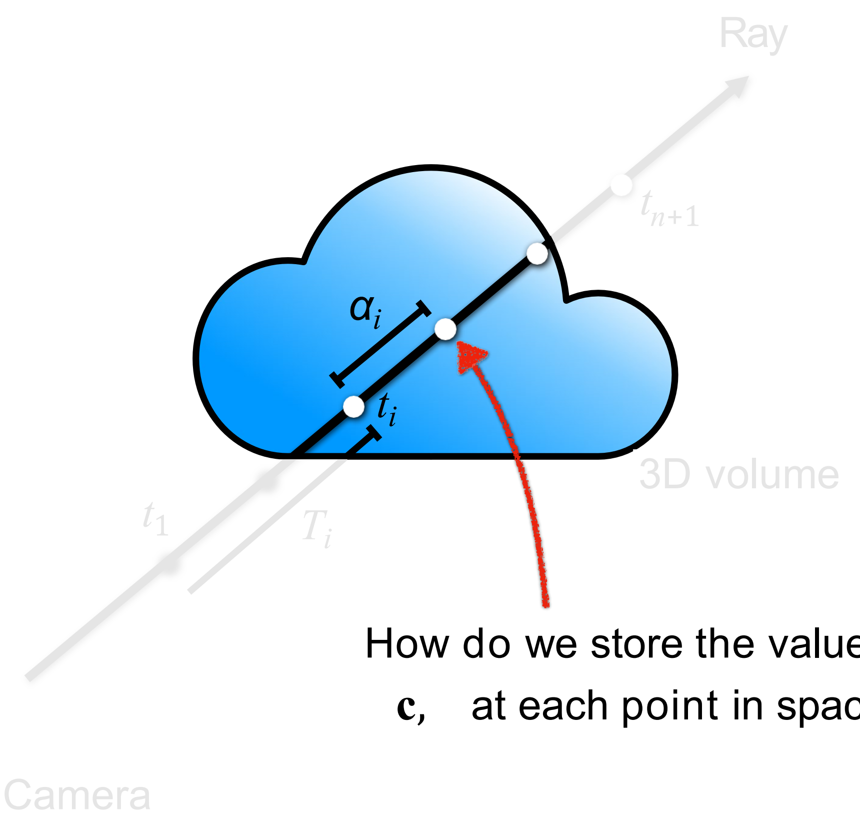
$$\mathbf{c} \approx \sum_{i=1}^{n} \underbrace{T_i \alpha_i}_{\text{weights}} \underbrace{\mathbf{c}_i}_{\text{colors}}$$

How much light is blocked earlier along ray:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

How much light is contributed by ray segment i:

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$



How do we store the values of

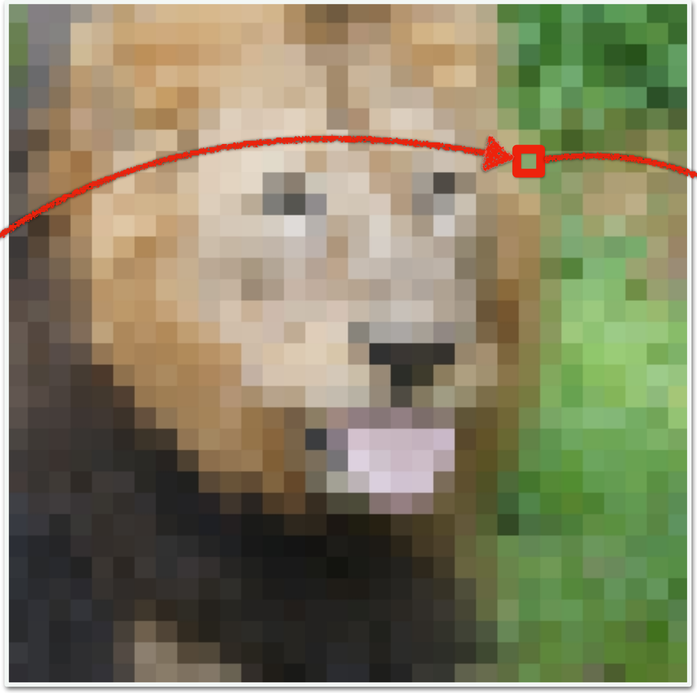
$\mathbf{c}$, at each point in space?

Overview

- Volumetric rendering math
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

Toy problem: storing 2D image data

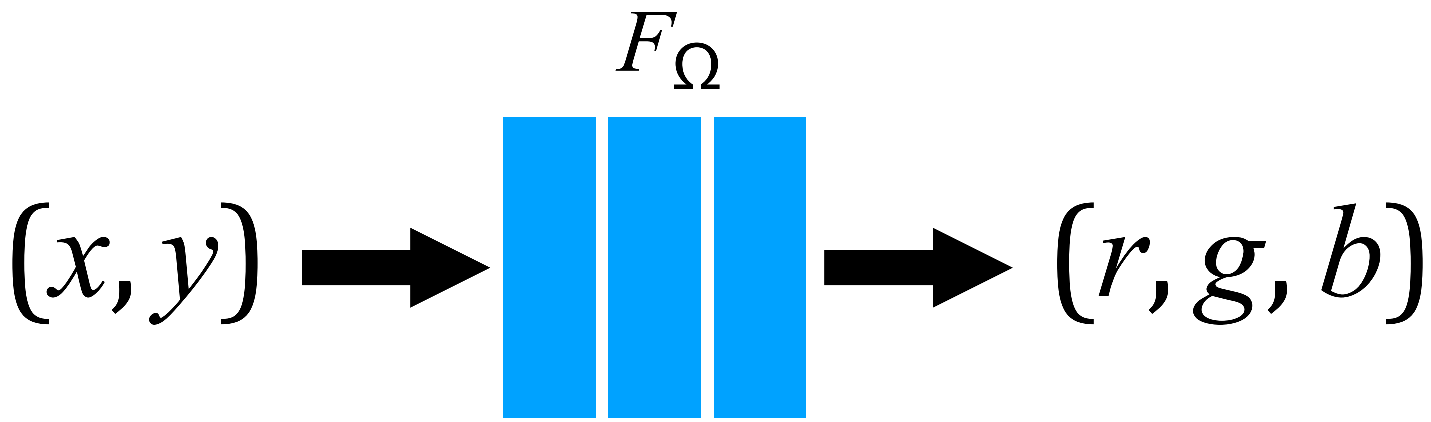
$$(x, y)$$



$$(r, g, b)$$

Usually we store an image as a
2D grid of RGB color values

Toy problem: storing 2D image data

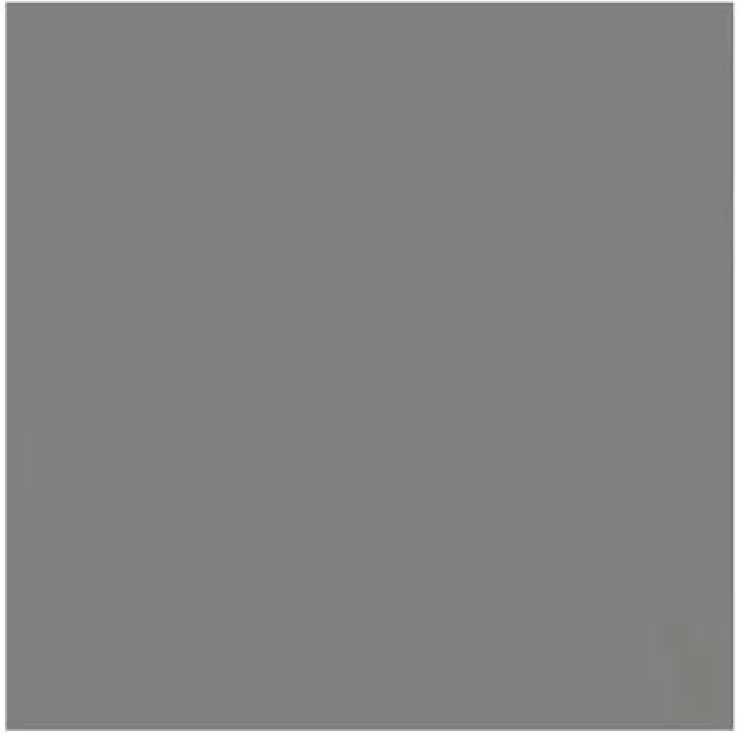


What if we train a simple fully-connected network (MLP) to do this instead?

Naive approach fails!

Ground truth image

Standard fully-connected net



Problem:

"Standard" coordinate-based MLPs cannot represent high-frequency functions

Solution:

Pass input coordinates through a high frequency mapping first

Input coordinate mapping

- Simple formula: apply a tall skinny matrix **B** to input coordinate vector **x**, then pass through *sin* and *cos*:

$$\gamma(\mathbf{x}) = (\sin(2\pi\mathbf{B}\mathbf{x}), \cos(2\pi\mathbf{B}\mathbf{x}))$$

- Passing network a subset of the Fourier basis functions. Same effect from:
  - Positional encoding
  - Fourier features
  - SIREN

Problem solved

Ground truth image

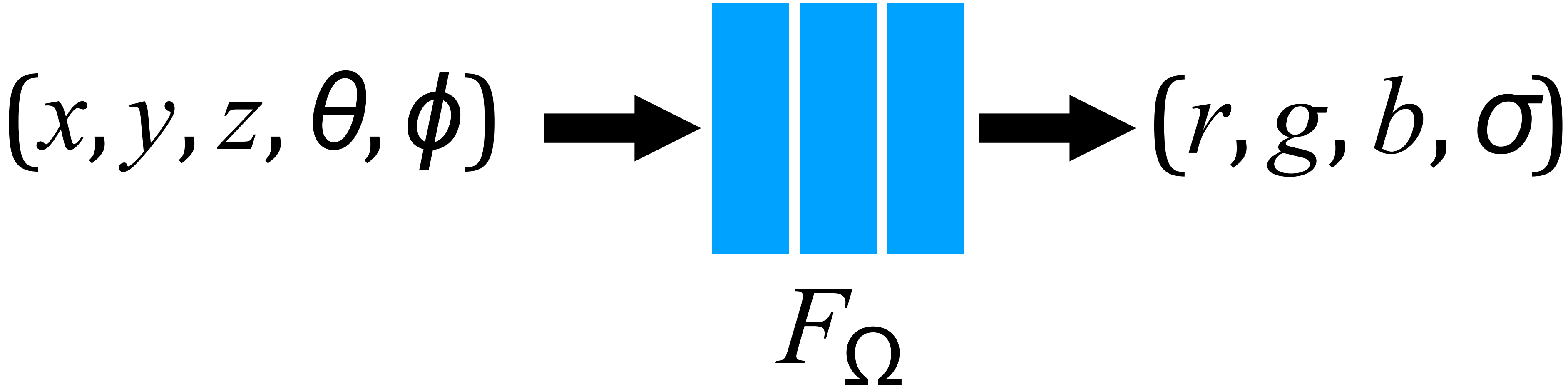
Standard fully-connected net

With "positional encoding"

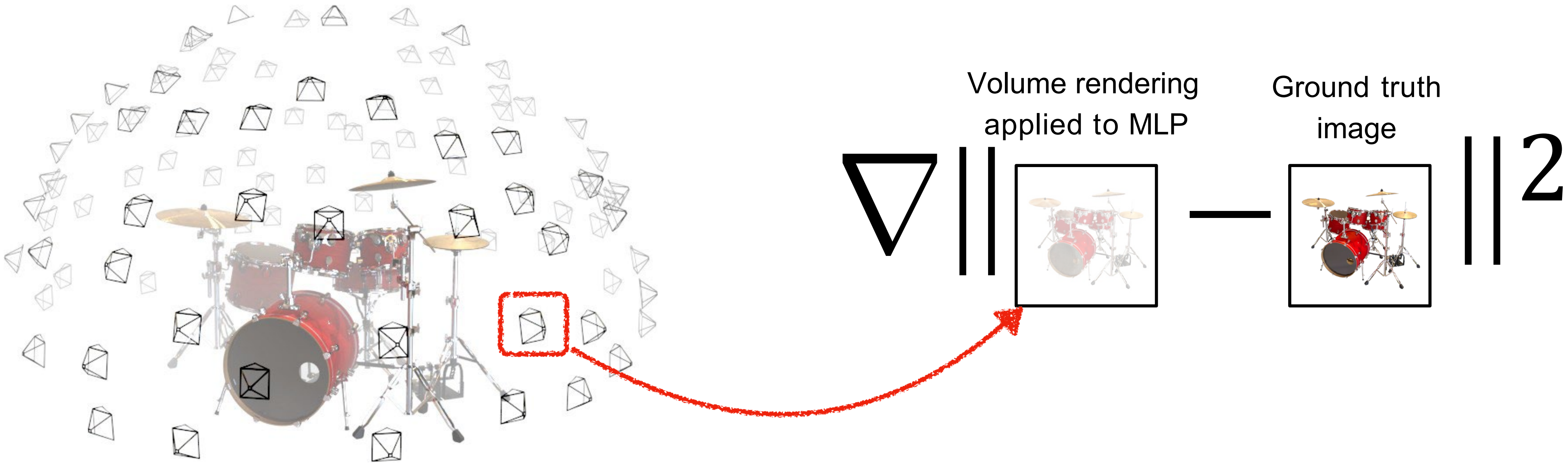
Overview

- Volumetric rendering math
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

NeRF = volume rendering + coordinate-based network


(x, y, z, θ, ϕ)
(r, g, b, σ)
F_Ω

Train network to reproduce input views of scene using gradient descent



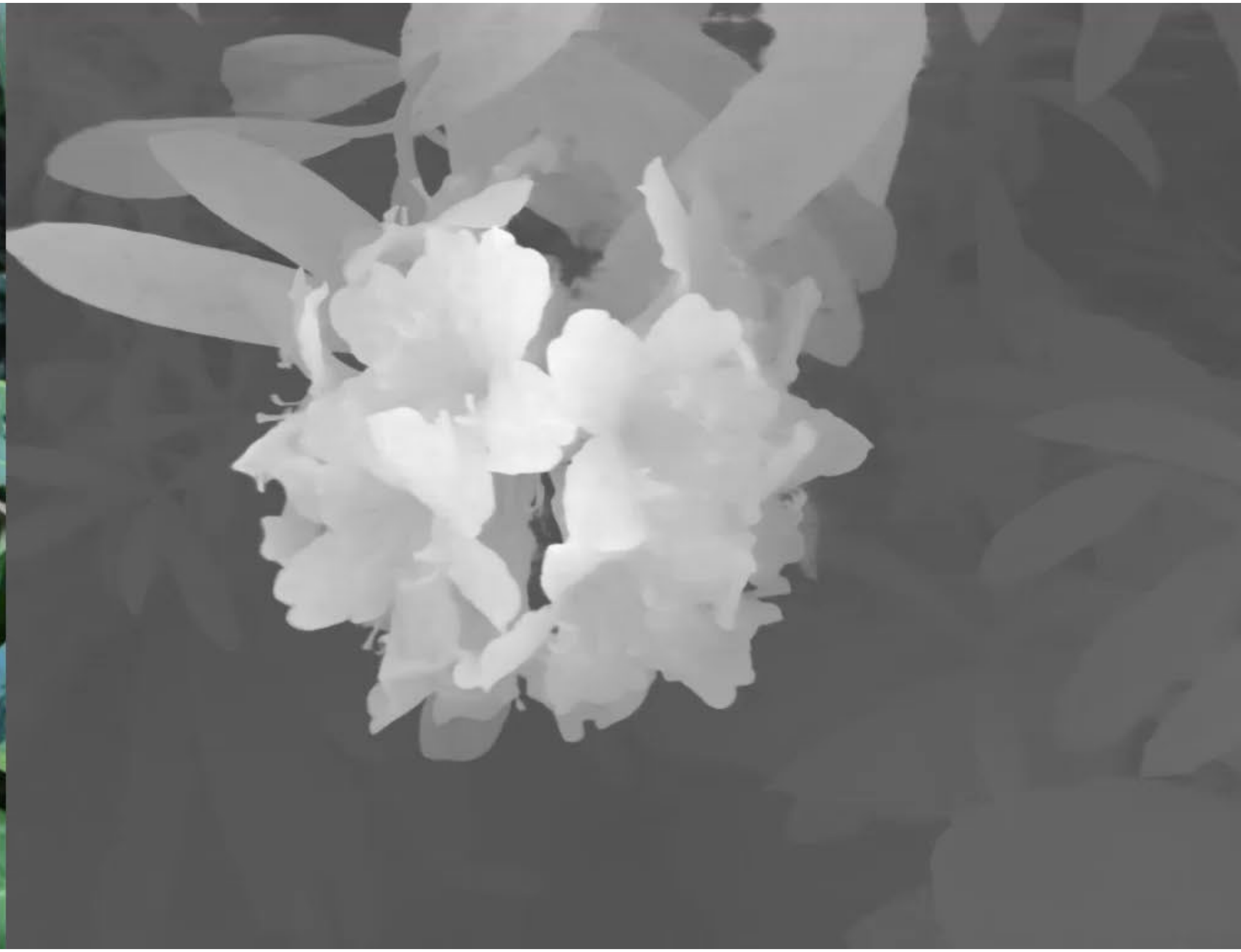
Visualizing view-dependent effects

Regular NeRF rendering

Manipulating input viewing directions

Visualizing learned density field as geometry

Regular NeRF rendering



Expected ray termination depth

Visualizing learned density field as geometry

Regular NeRF rendering

Expected ray termination depth

If you’re interested, you may take: CIS 7000-005 Introduction to Neural Scene Representation and Neural Rendering, in Fall 2025.