CS231A

Computer Vision:

From 3D Reconstruction to Recognition

Neural Radiance Fields II – Applications

Outline

- Recap: Neural Radiance Fields
- Applications
  - Pose Estimation
  - Motion-planning

The problem of novel view synthesis

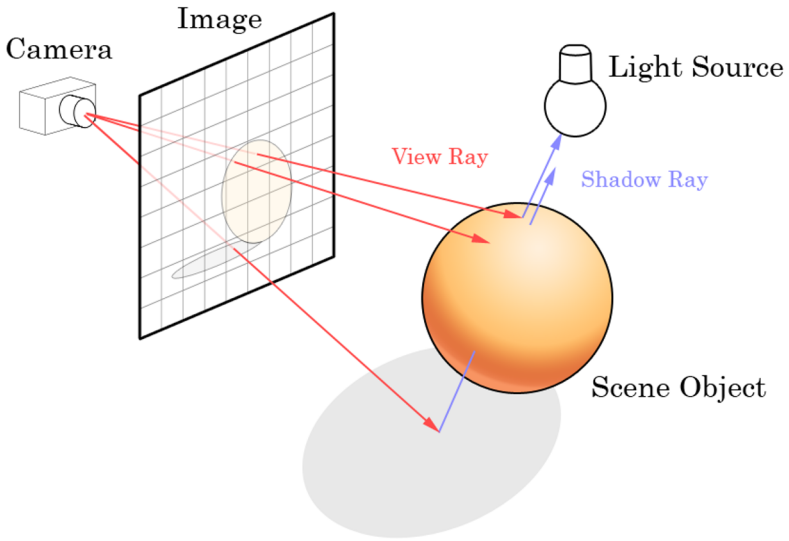
Inputs: sparsely sampled images of scene

Outputs: *new* views of same scene (rendered by our method)

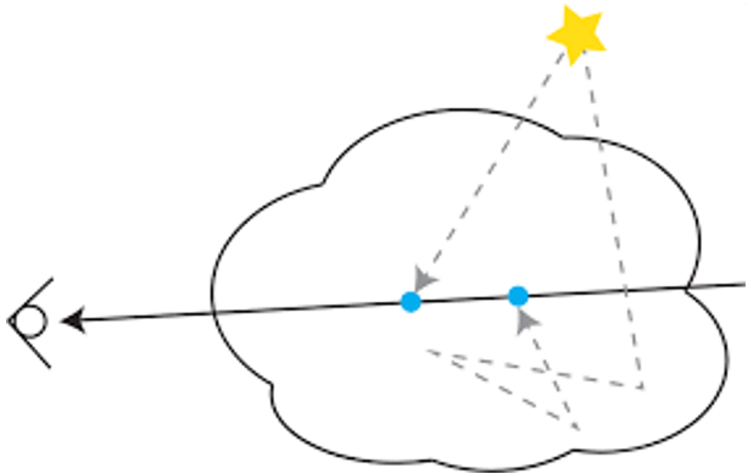
2

Mildenhall et al. ECCV 2020. https://www.matthewtancik.com/nerf

Ways to Render



Surface rendering



Volume rendering

Volume rendering equation

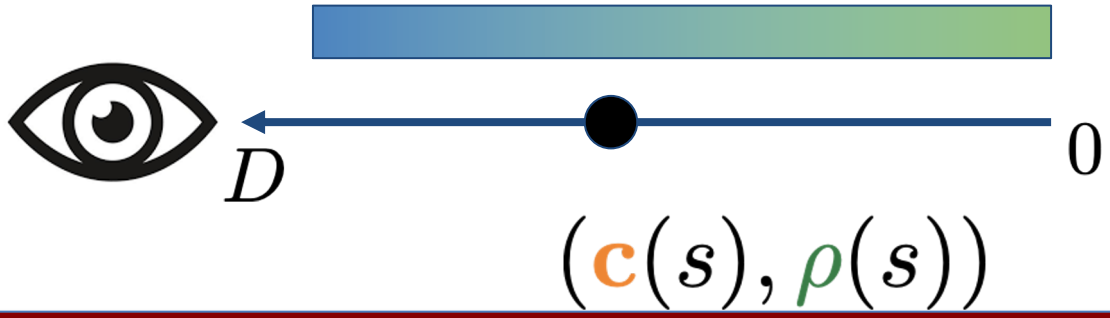
$$\mathbf{I}(D) = \mathbf{I}_0 T(0) + \int_0^D \mathbf{c}(s)\rho(s)T(s)ds$$

pixel color

radiance density

$$T(s) = \exp\left(-\int_s^D \rho(t)dt\right)$$

transparency

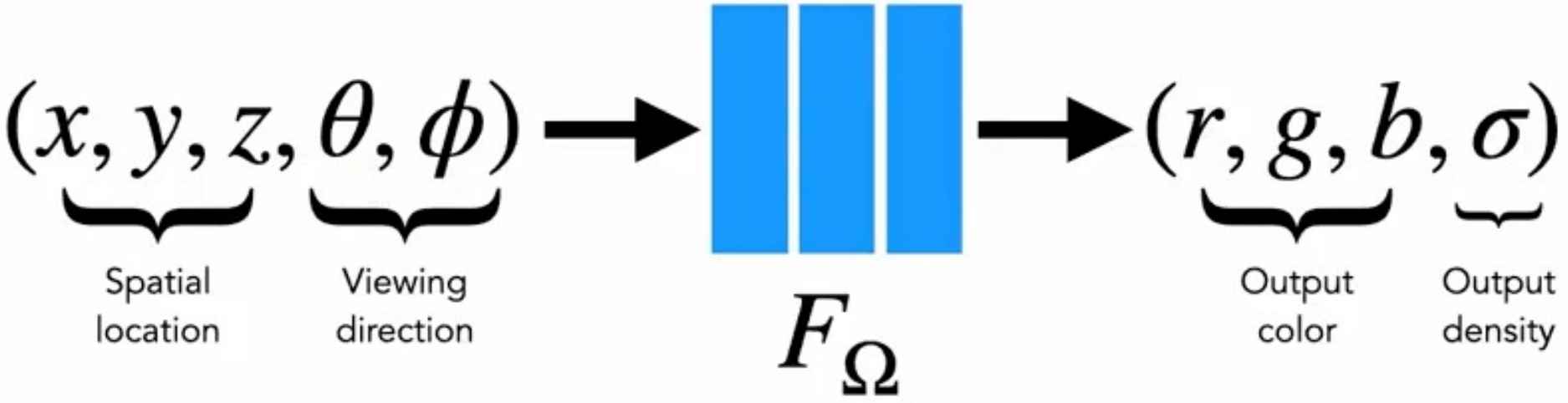


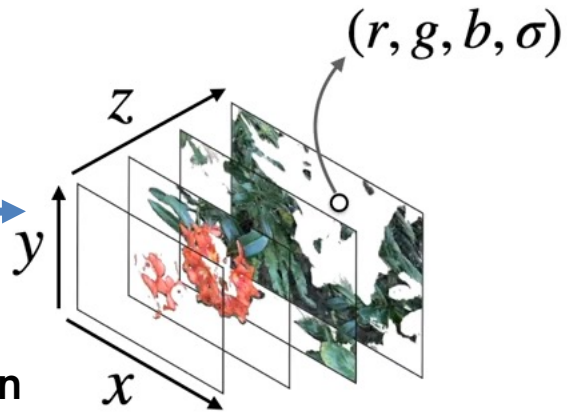
NeRF (neural radiance fields)

- Neural network as a volume representation using volume rendering to do view synthesis

- $(x, y, z, \theta, \phi) \rightarrow color, opacity$

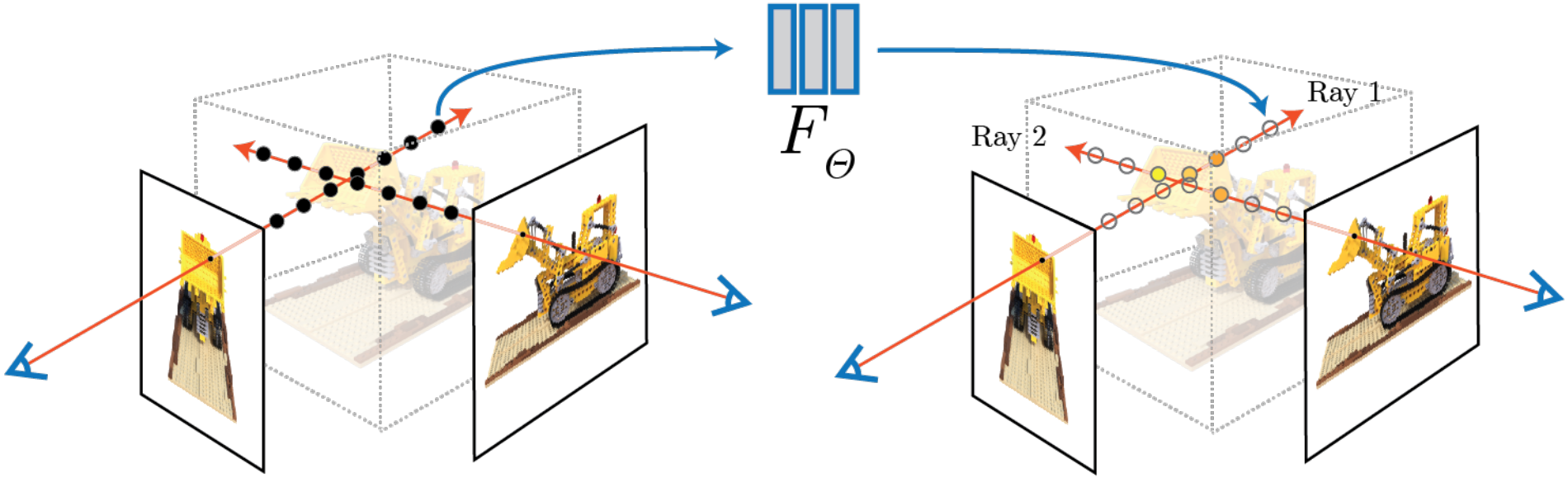
Represent a scene as a continuous 5D function





No need to instantiate Volume representation

Generate views with traditional volume rendering



Mildenhall et al. ECCV 2020. https://www.matthewtancik.com/nerf

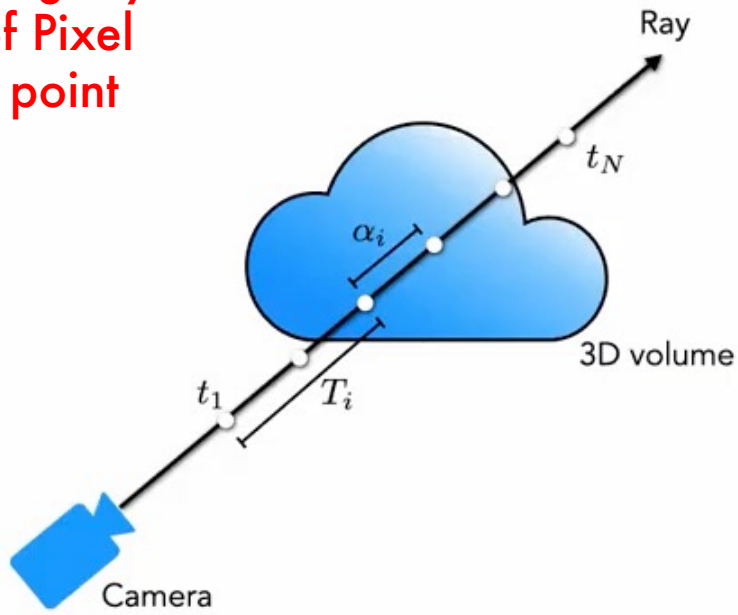
Generate views with traditional volume rendering

Rendering model for ray r(t) = o + td:

$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

weights, colors

t = point along ray
C = Color of Pixel
c = color of point



How much light is blocked earlier along ray:

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

Transparency

How much light is contributed by ray segment *i*:

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$

Function of segment length δt_i and volume density σ

From Presentation by Matthew Tancik: Neural Radiance Fields for View Synthesis. 2020.

Volume rendering is trivially differentiable

Rendering model for ray r(t) = o + td:

$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

differentiable w.r.t.

colors

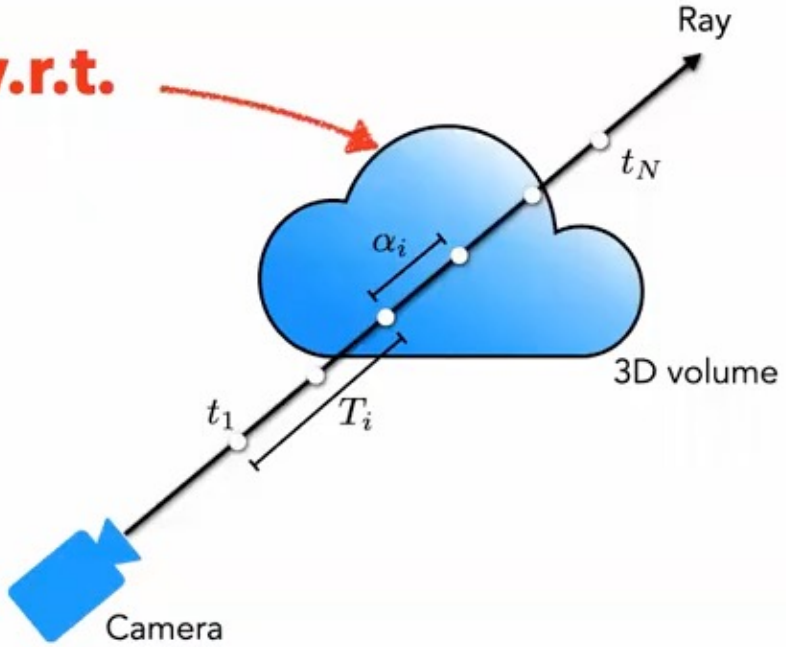
weights

How much light is blocked earlier along ray:

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

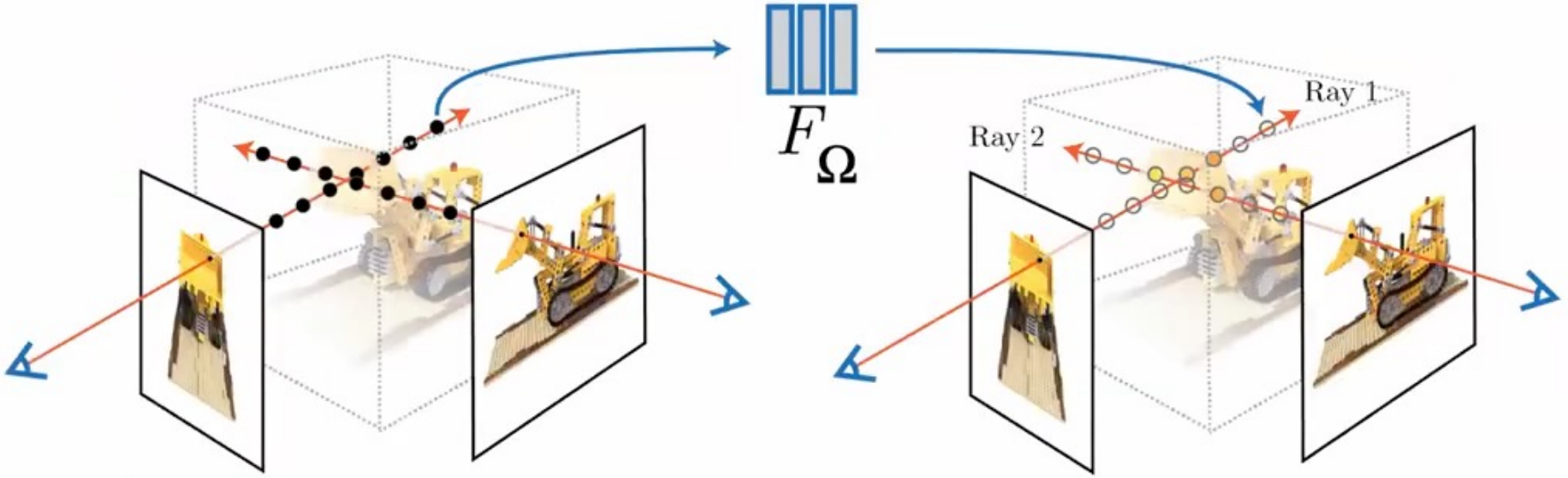
How much light is contributed by ray segment *i*:

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$

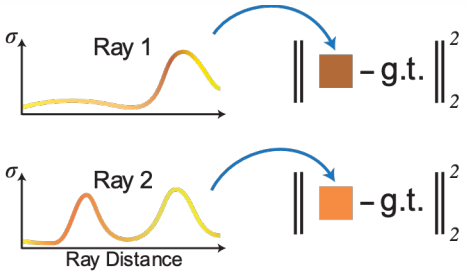


From Presentation by Matthew Tancik: Neural Radiance Fields for View Synthesis. 2020.

Optimize with gradient descent on rendering loss



$$\min_{\Omega} \sum_{i} \|\text{render}^{(i)}(F_\Omega) - I_{\text{gt}}^{(i)}\|^2$$

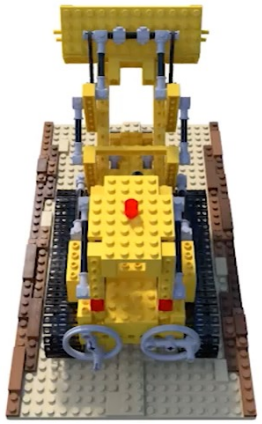


From Presentation by Matthew Tancik: Neural Radiance Fields for View Synthesis. 2020.

Training network to reproduce all input views of the scene

From Presentation by Matthew Tancik: Neural Radiance Fields for View Synthesis. 2020.

Results – Synthetic data





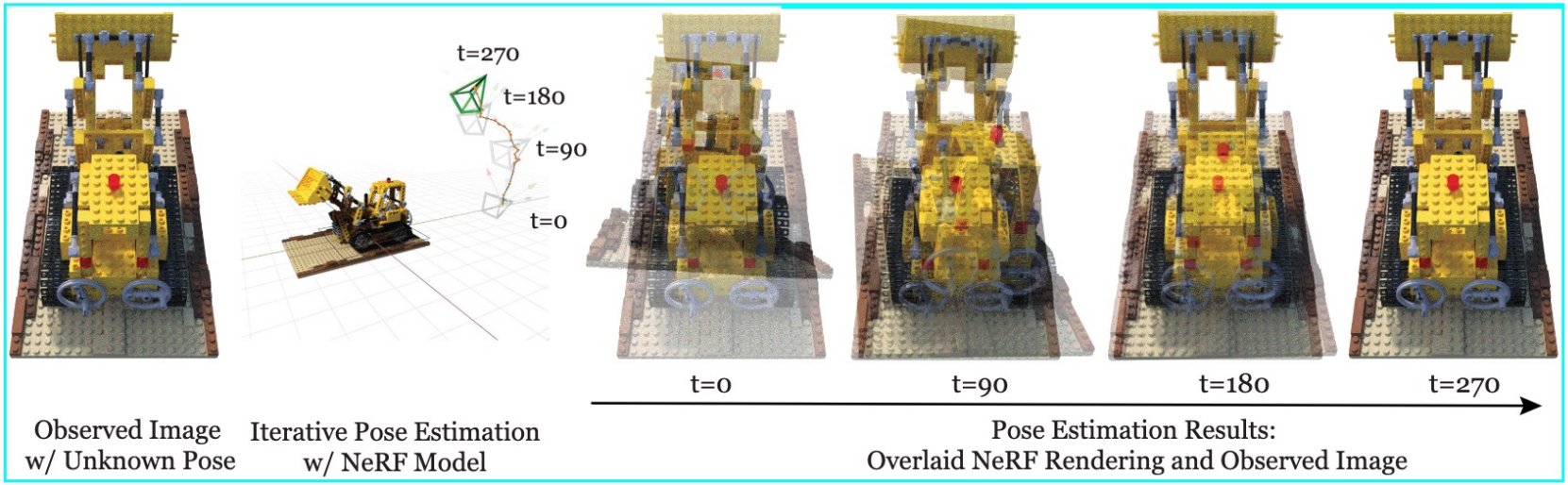
Results – Visualization Geometry

PollEverywhere

NeRF for Pose Estimation

iNeRF: Inverting Neural Radiance Fields for Pose Estimation

Lin Yen-Chen[1,2] Pete Florence[1] Jonathan T. Barron[1]
Alberto Rodriguez[2] Phillip Isola[2] Tsung-Yi Lin[1]
[1]Google Research [2]Massachusetts Institute of Technology



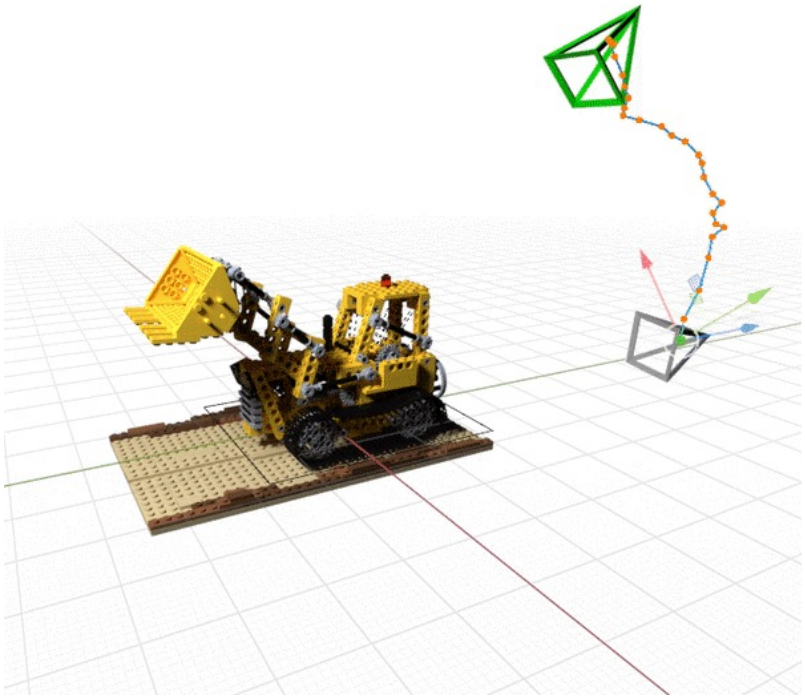
Input

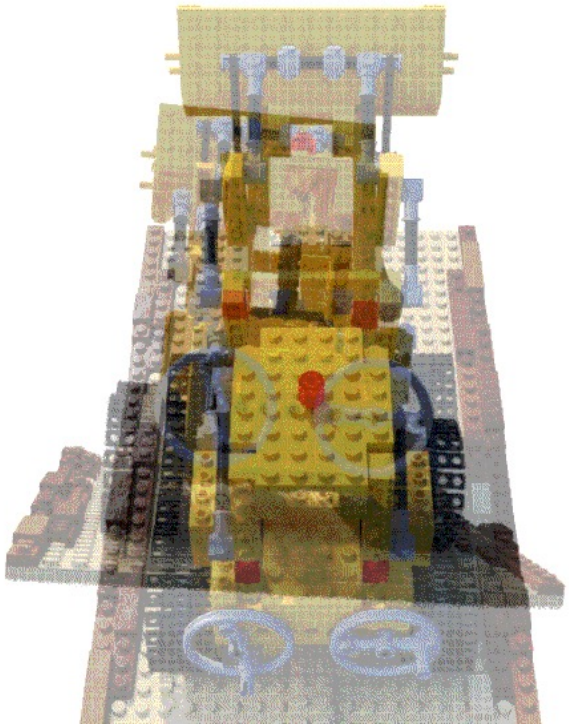
- An RGB Image
- Initial Pose
- NeRF model

- Not required
  - Depth
  - Mesh model

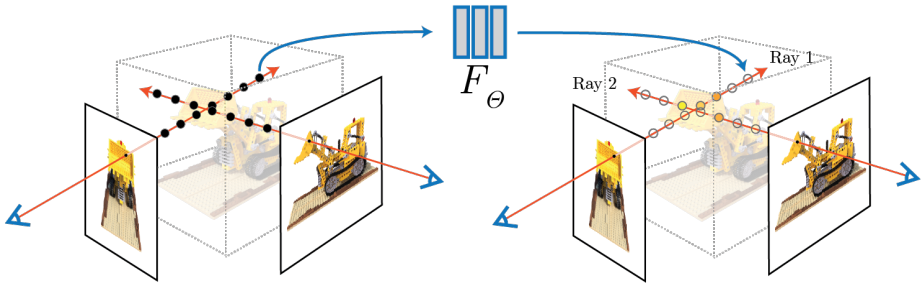
Output

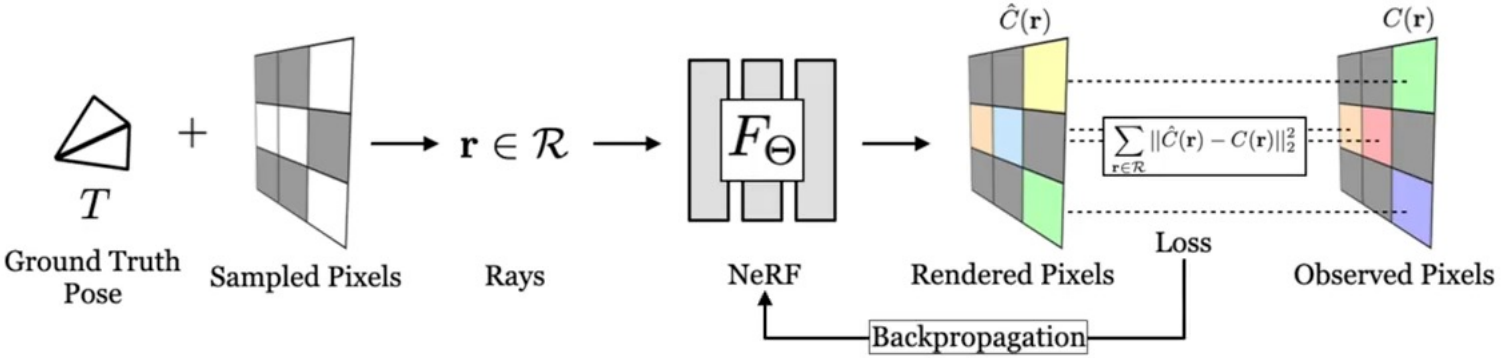
Position and orientation of Camera relative to object. (6DoF)





Recall: Training Vanilla NeRF

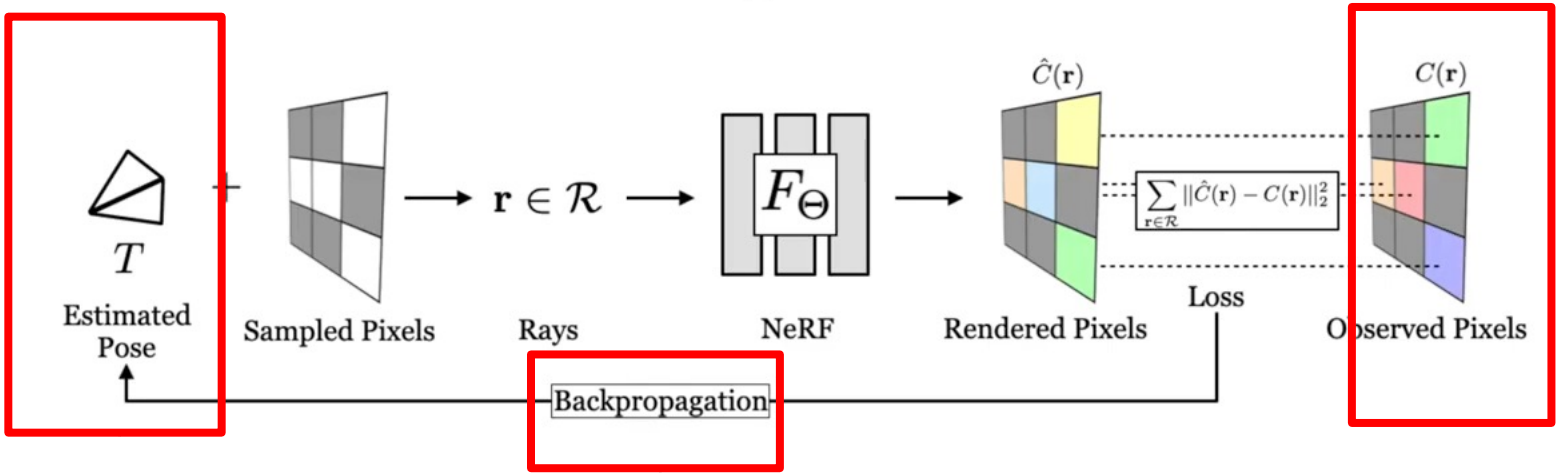




Synthesize Novel views

iNeRF: Inverting NeRF for Pose Estimation

$$\hat{T} = \underset{T \in \mathrm{SE}(3)}{\operatorname{argmin}} \mathcal{L}(T \mid I, \Theta)$$



Gradient-based SE(3) Optimization

Loss at Current Optimization Iteration $L(\hat{T}_i | I, \Theta)$

Parameterize $\hat{T}_i$ using exponential coordinates to ensure solution lies on SE(3) manifold

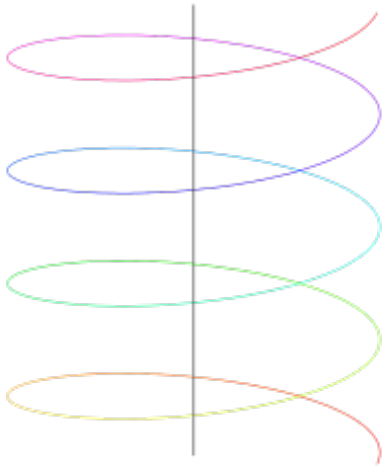
With initial pose estimate $\widehat{T_0}$

$$\hat{T}_i = e^{[\mathcal{S}_i]\theta_i} \hat{T}_0 \,,$$

$$e^{[\mathcal{S}]\theta} = \begin{bmatrix} e^{[\omega]\theta} & K(\mathcal{S}, \theta) \\ 0 & 1 \end{bmatrix}$$

Screw axis $S = [\omega, v]$

Magnitude θ

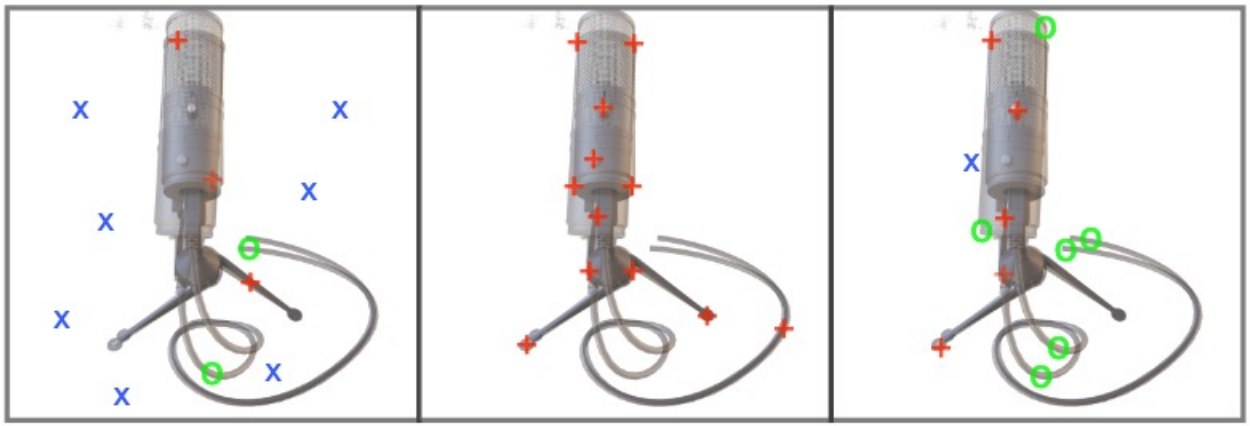


$$K(S, \theta) = (I\theta + (1 - cos\theta)[\omega] + (\theta - sin\theta)[\omega]^2)v$$

For full derivation see Modern Robotics by Lynch & Park, Section 3.3.3.1

Final Loss $\quad \widehat{\mathcal{S}\theta} = \underset{S\theta \in \mathbb{R}^6}{\text{argmin}} \; \mathcal{L}(e^{[\mathcal{S}]\theta} T_0 \mid I, \Theta)$

Sampling Rays



Random Interest Point Interest Region

Interest Region

Random

x samples in background
+ samples covered by rendered and observed images
o samples covered only by either, the sampled or observed images

Interest Point Sampling - find key points.
Interest Region Sampling – Sampling from dilated masks around interest points

Results

Rotation: 45°

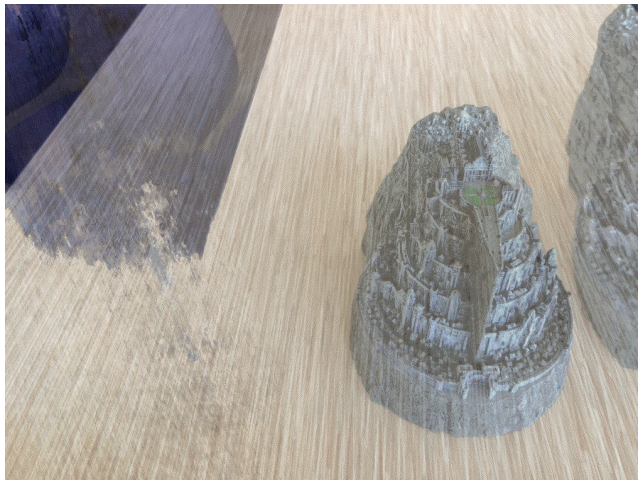
Category-Level Results

Input video.

Predicted pose & image.

Results on Real World images





NeRF versus iNeRF

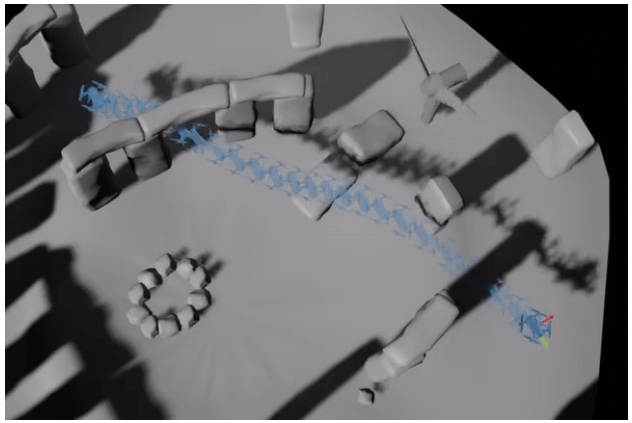
Method	NeRF	iNeRF
Input	Camera Pose	Image
Output	Image	Camera Pose

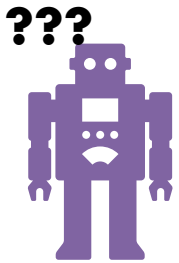
Outline

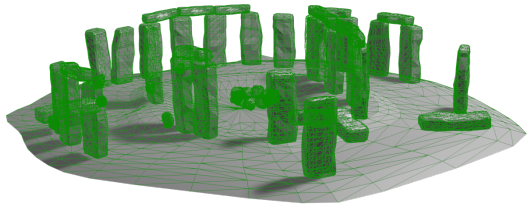
- Recap: Neural Radiance Fields
- Applications
  - Pose Estimation
  - Motion-planning

Vision-Only Robot Navigation in a Neural Radiance World

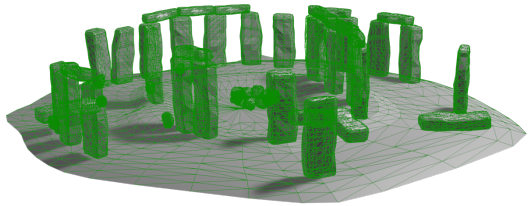
Michal Adamkiewicz*, Timothy Chen*, Adam Caccavale, Rachel Gardner,
Preston Culbertson, Jeannette Bohg, Mac Schwager
ICRA'22 + RAS Letters



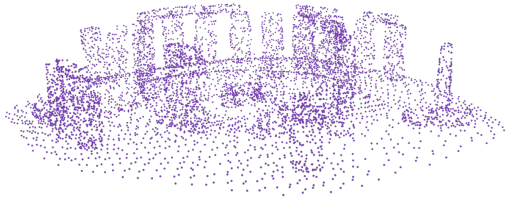




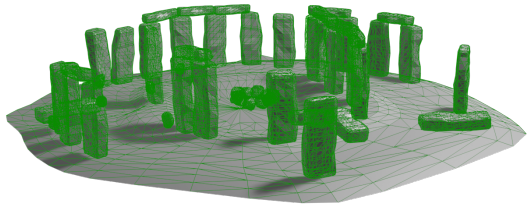
Mesh



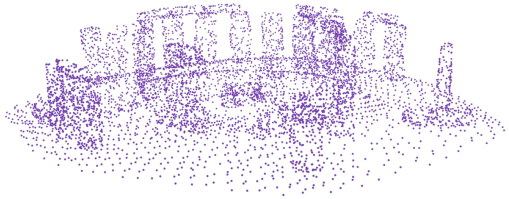
Mesh



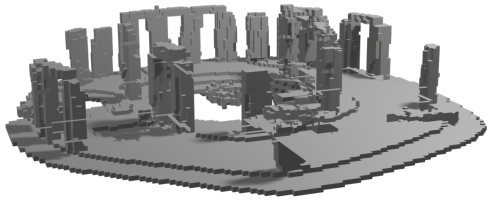
Point Cloud



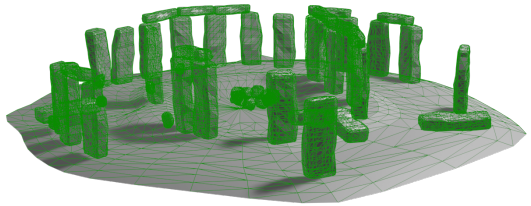
Mesh



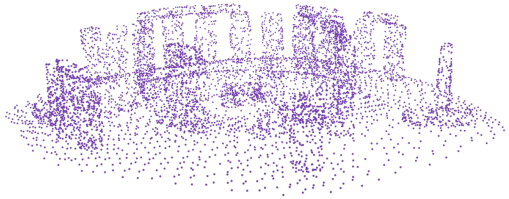
Point Cloud



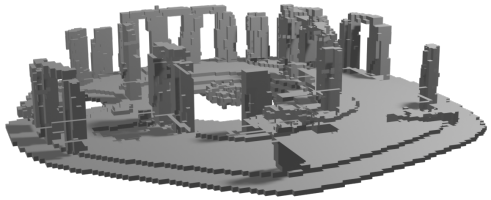
Voxels



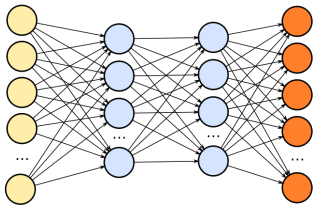
Mesh



Point Cloud



Voxels

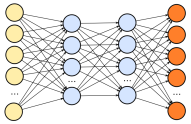


Implicit representations

Implicit representations

density

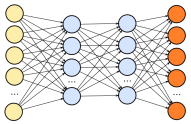
colour



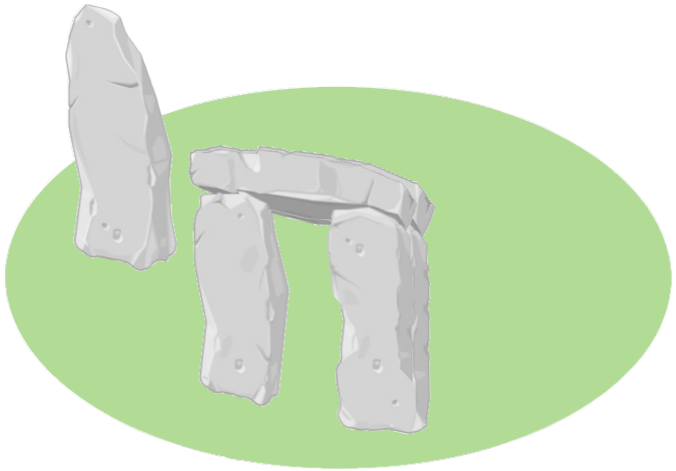
$$(x, y, z) = \sigma, c$$

position (and optionally view direction)

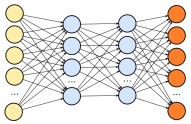
Implicit representations



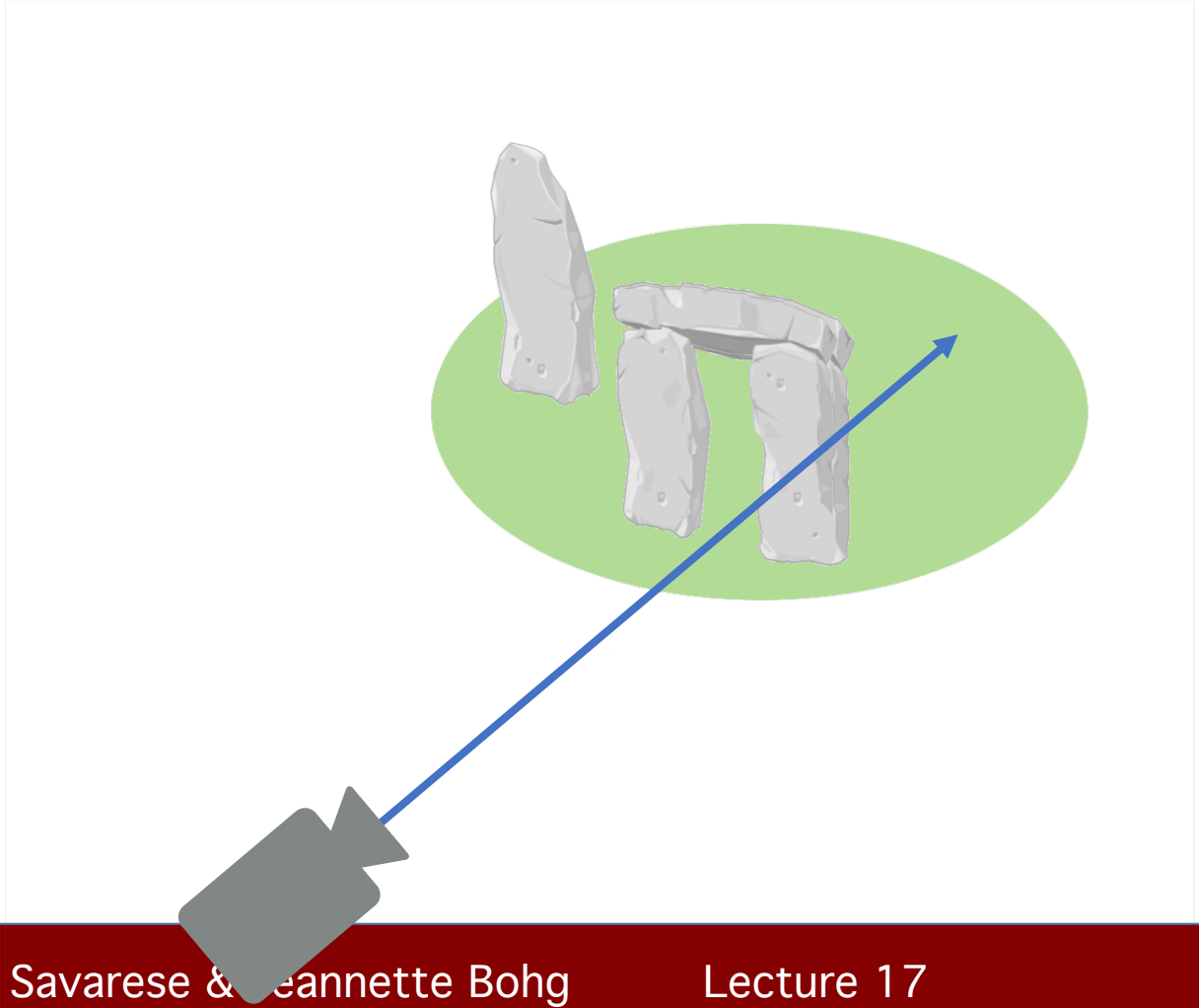
$$(x, y, z) = \sigma, c$$



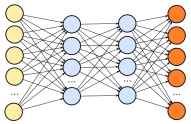
Implicit representations



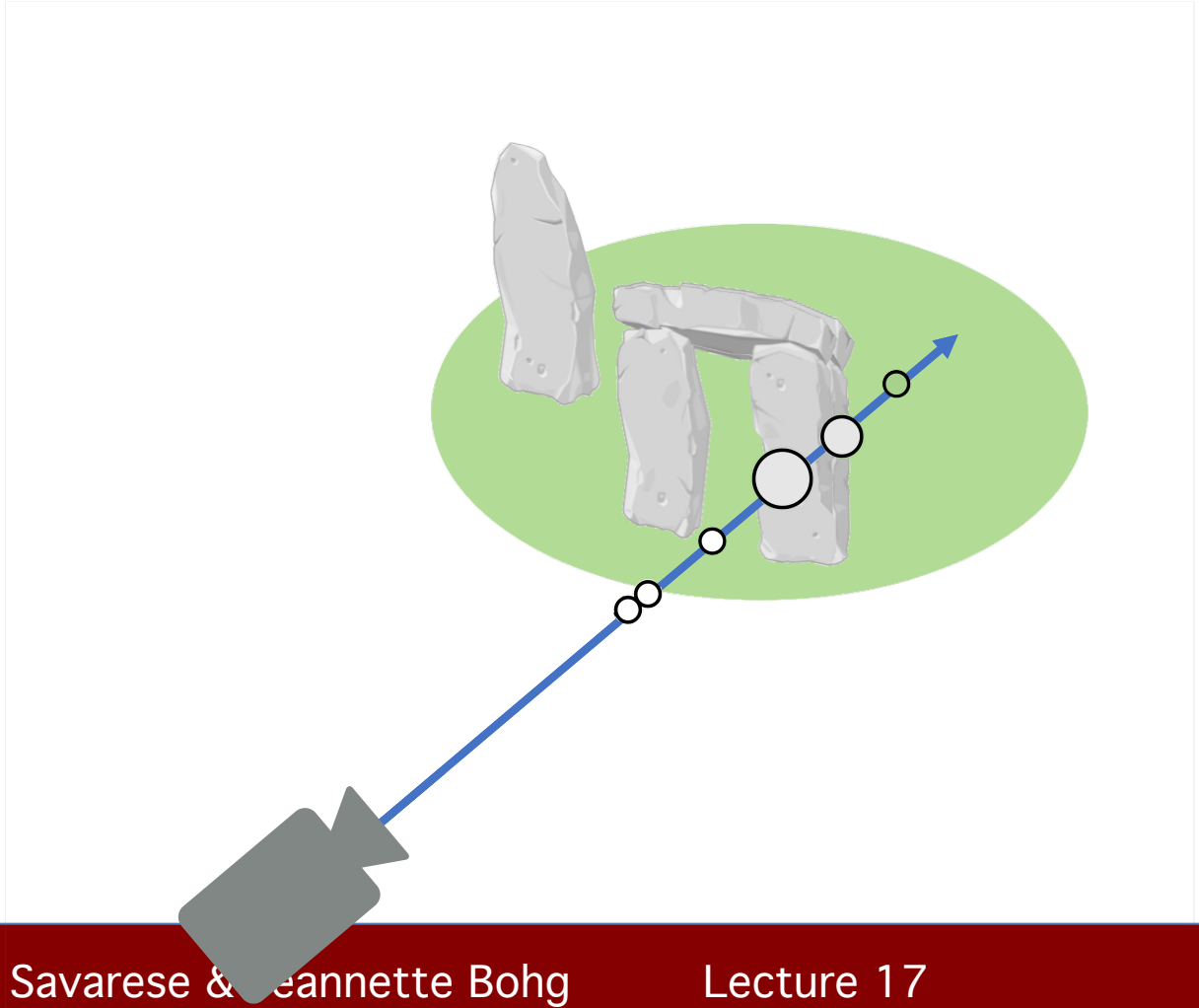
$$(x, y, z) = \sigma, c$$



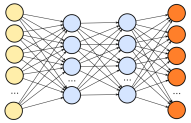
Implicit representations



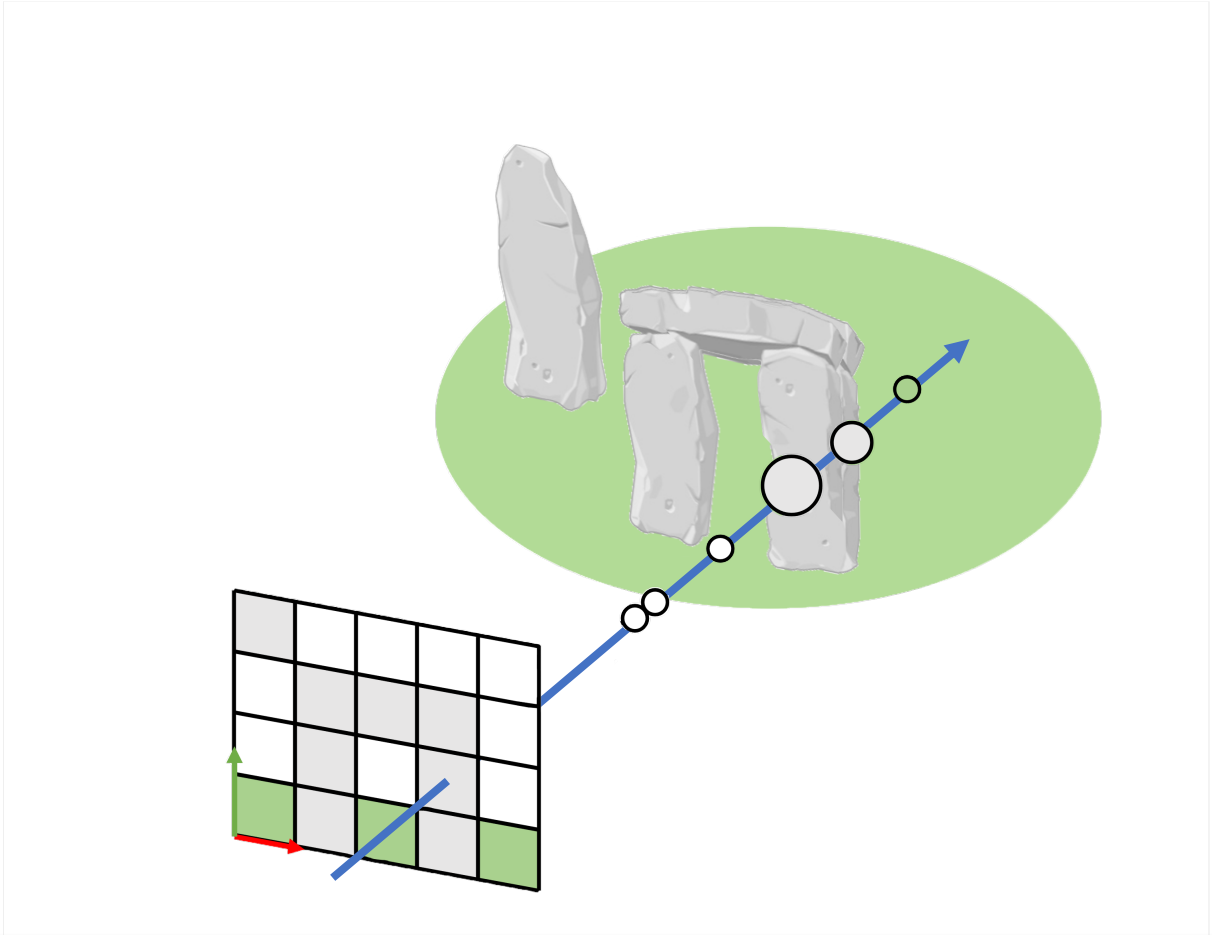
$$(x, y, z) = \sigma, c$$

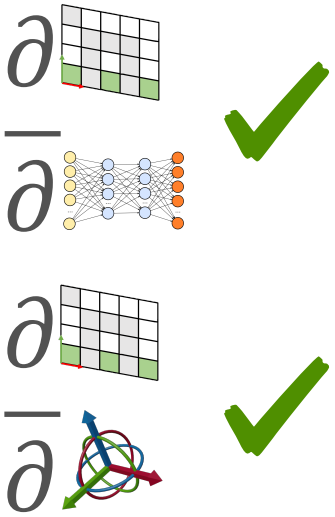


Implicit representations

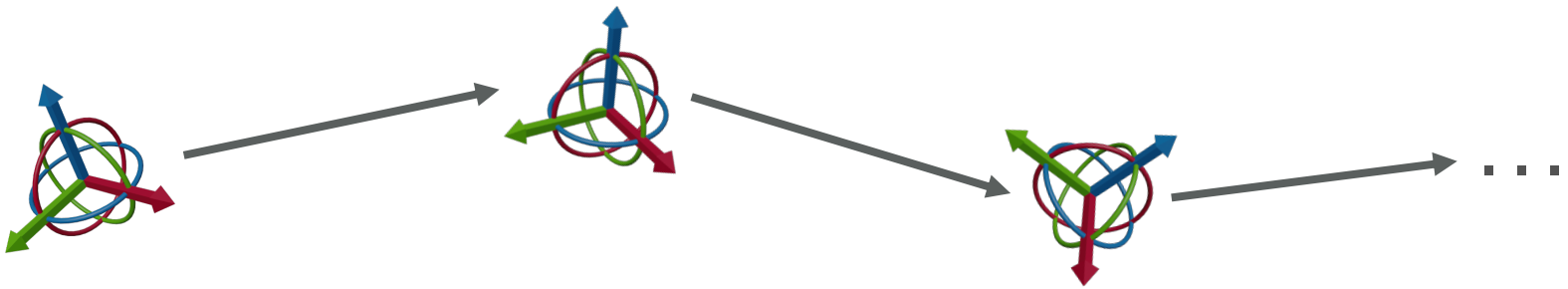


$$(x, y, z) = \sigma, c$$

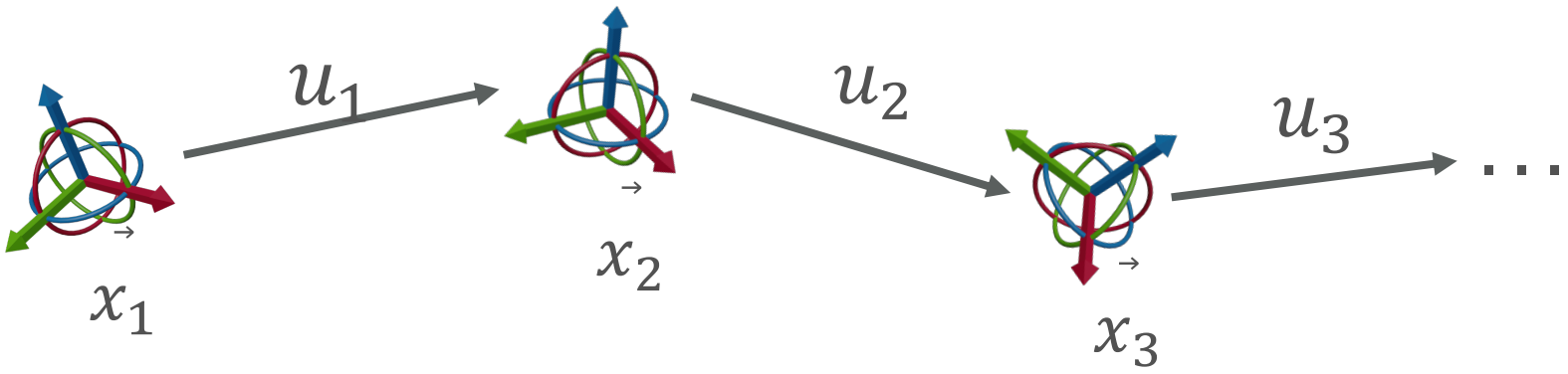




Trajectory Planning

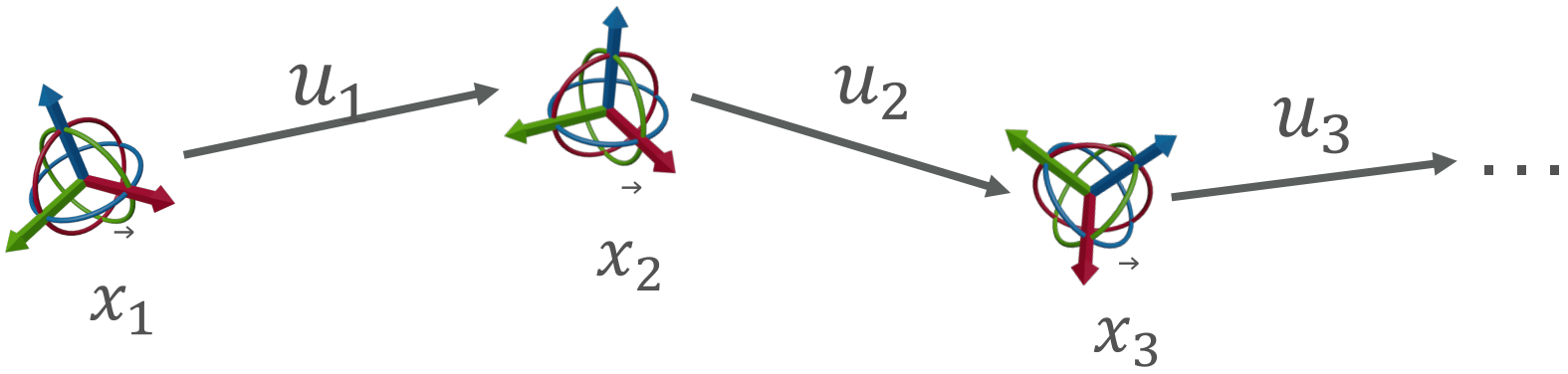


Trajectory Planning



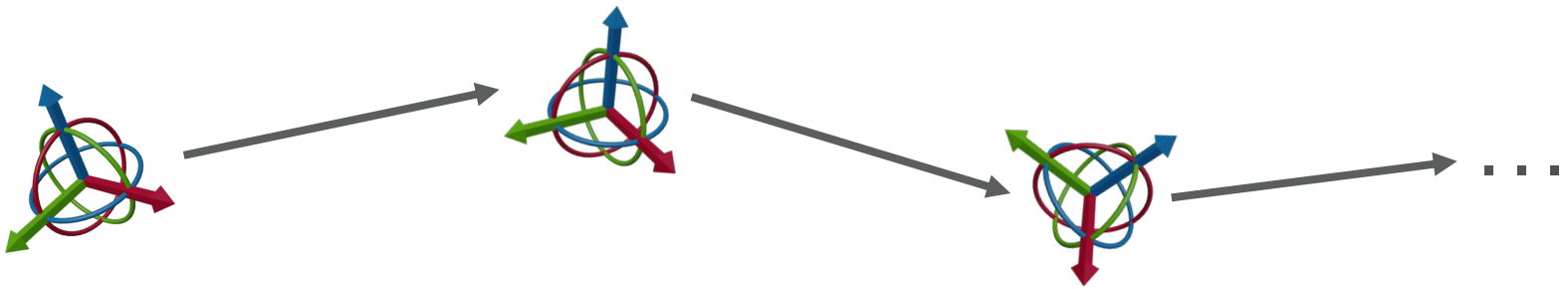
$$Loss = L_{collision} + L_{controleffort}$$

Trajectory Planning



$$Loss = L_{collision} + L_{controleffort}(u_1) + L_{controleffort}(u_2) + \dots$$

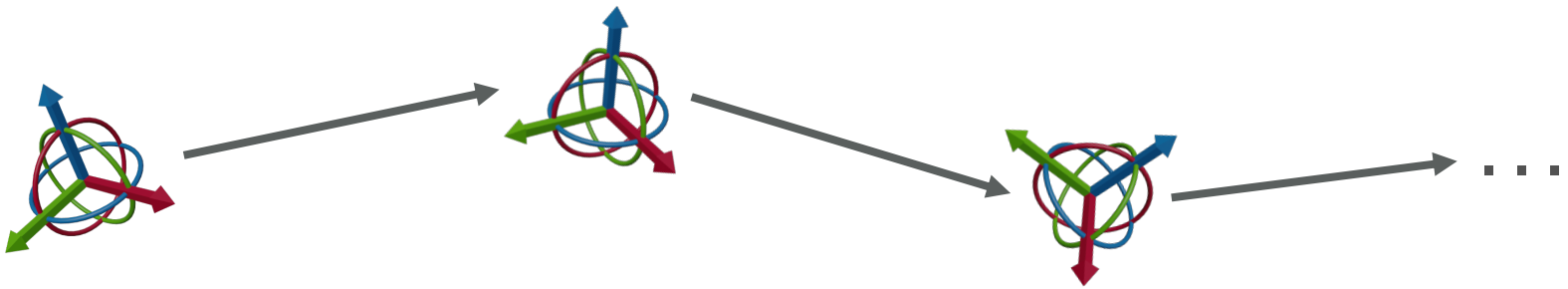
Trajectory Planning



$$Loss = \text{(NN)}(\text{pose}) + \text{(NN)}(\text{pose}) + \text{(NN)}(\text{pose}) + \ldots$$

$$+\Sigma L_{controleffort}$$

Trajectory Planning

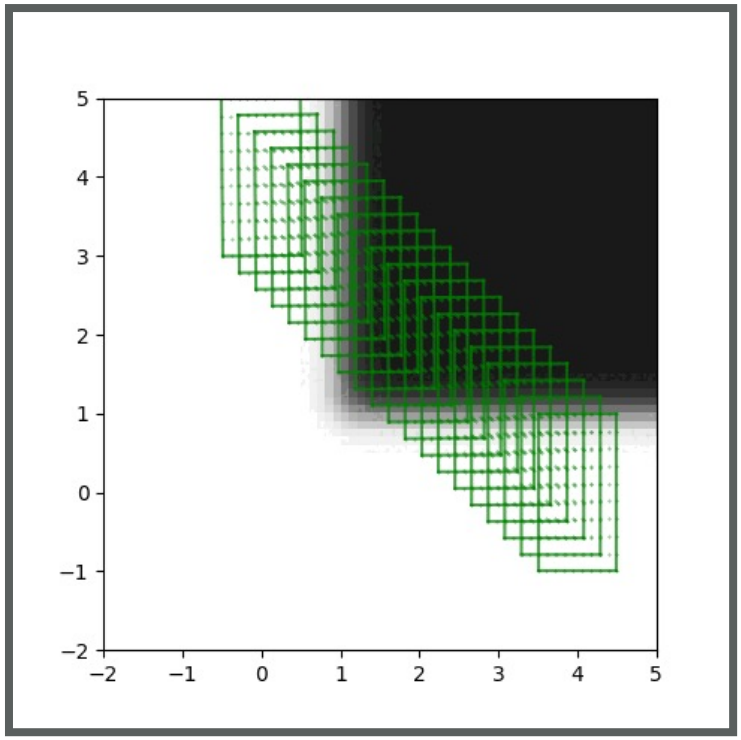


$$Loss =$$

$$|\overrightarrow{v_1}| \; \text{NN}(\;) + |\overrightarrow{v_2}| \; \text{NN}(\;) + |\overrightarrow{v_3}| \; \text{NN}(\;) + \ldots$$

$$+\Sigma L_{controleffort}$$

Trajectory Planning

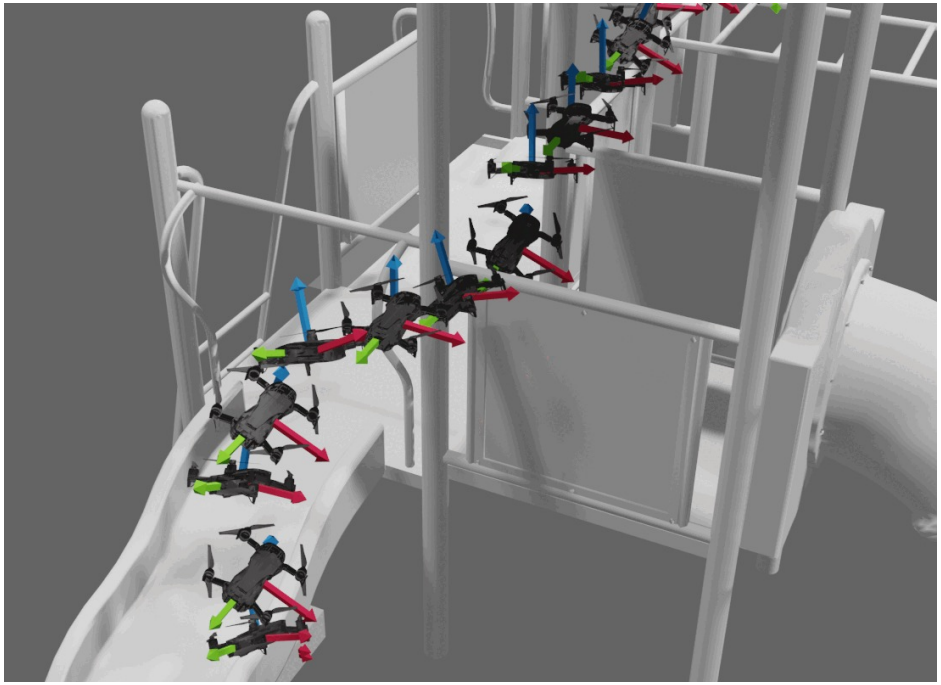


$$Loss =$$

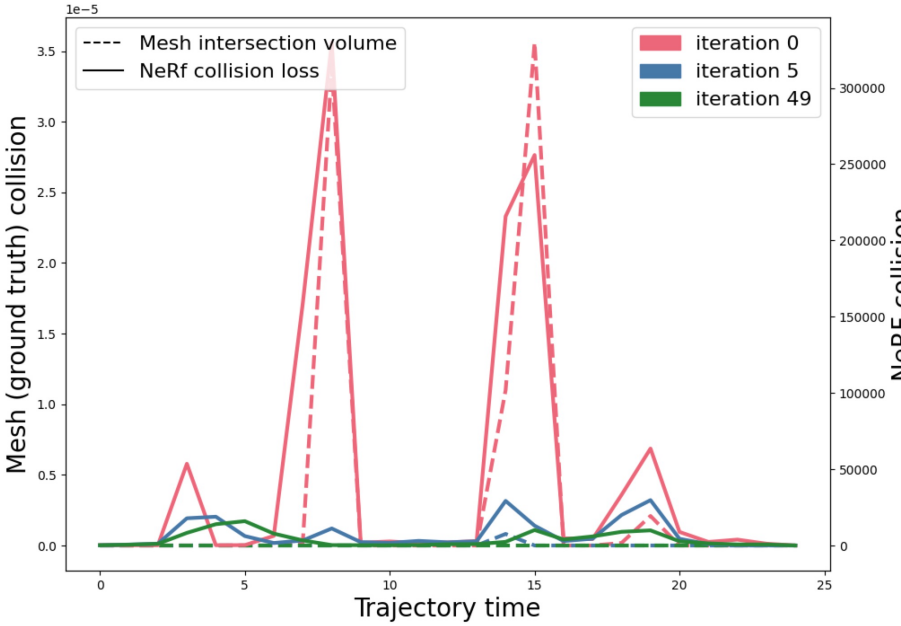
$$|\overrightarrow{v_1}| \text{ [NN] } (\text{ [robot] }) + |\overrightarrow{v_2}| \text{ [NN] } (\text{ [robot] }) + |\overrightarrow{v_3}| \text{ [NN] } (\text{ [robot] }) + \ldots$$

$$+\Sigma L_{controleffort}$$

Trajectory Planning

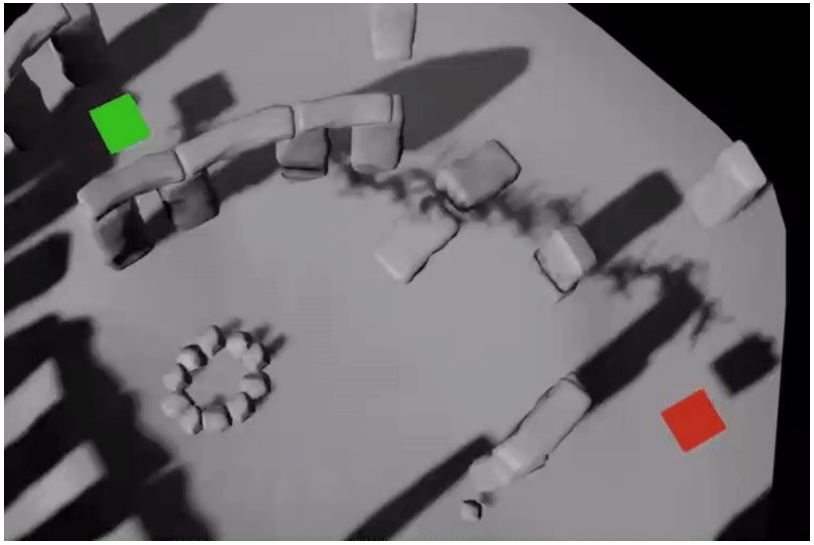


Optimisation process

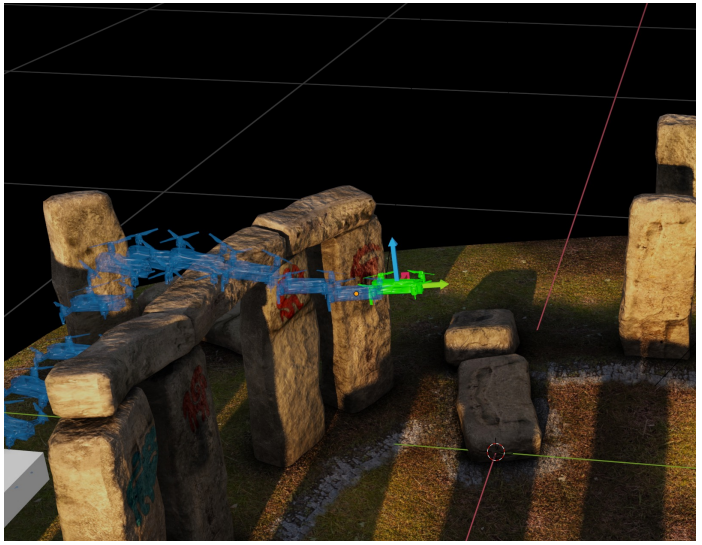


GT mesh comparison

Trajectory Planning



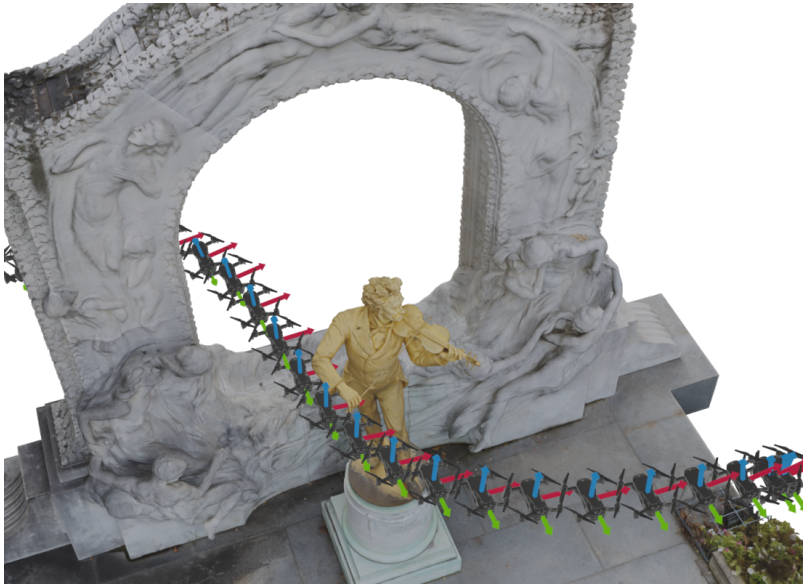
A* initialisation



Avoids obstacles

Trajectory Planning

Aware of robot geometry



Aware of scene detail

State Estimation – Nonlinear System

$$\boldsymbol{\mu}_{t|t-1} = f(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t)$$

$$\mathbf{A}_{t-1} = \frac{\partial f(\mathbf{x}, \mathbf{u}_t)}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}}$$

$$\boldsymbol{\Sigma}_{t|t-1} = \mathbf{A}_{t-1}\boldsymbol{\Sigma}_{t-1}\mathbf{A}_{t-1}^T + \mathbf{Q}_{t-1}$$

Predict Step

State Estimation

$$Loss = |\text{📷} - NeRF(\vec{x})|^2 + Loss_{Process}(\vec{x})$$

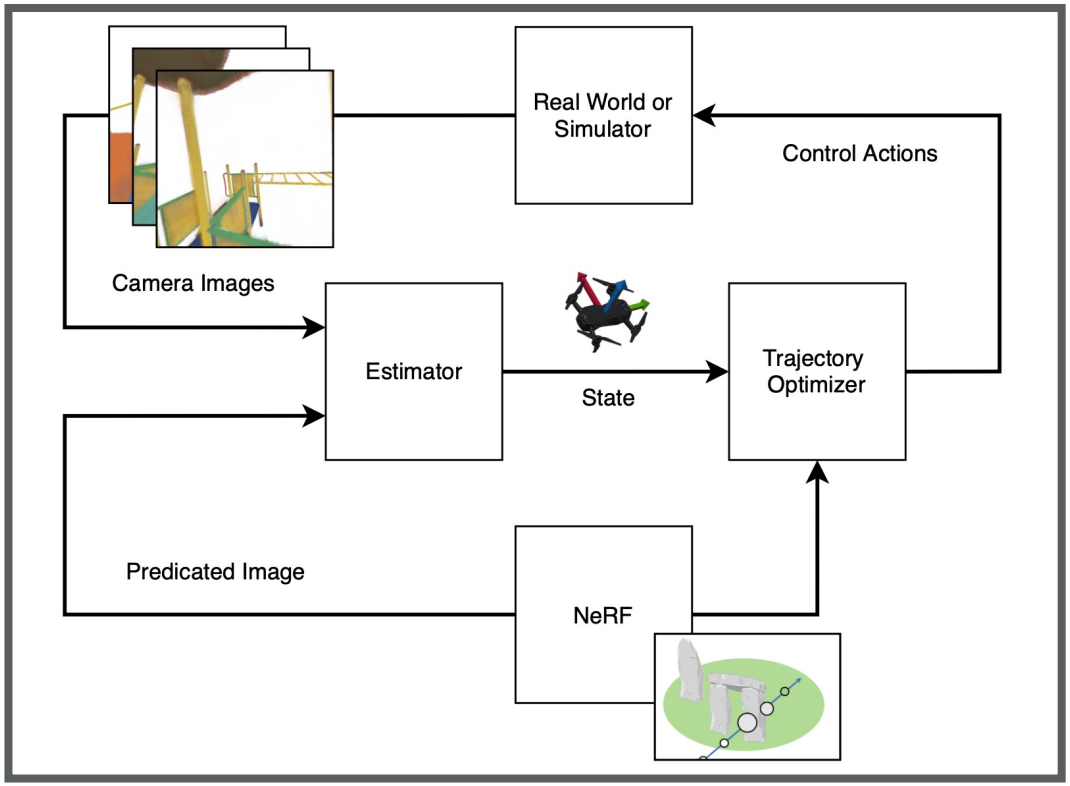
State Estimation

$$Loss = |\text{📷} - NeRF(\vec{x})|^2 + Loss_{Process}(\vec{x}, \Sigma_{t|t-1})$$

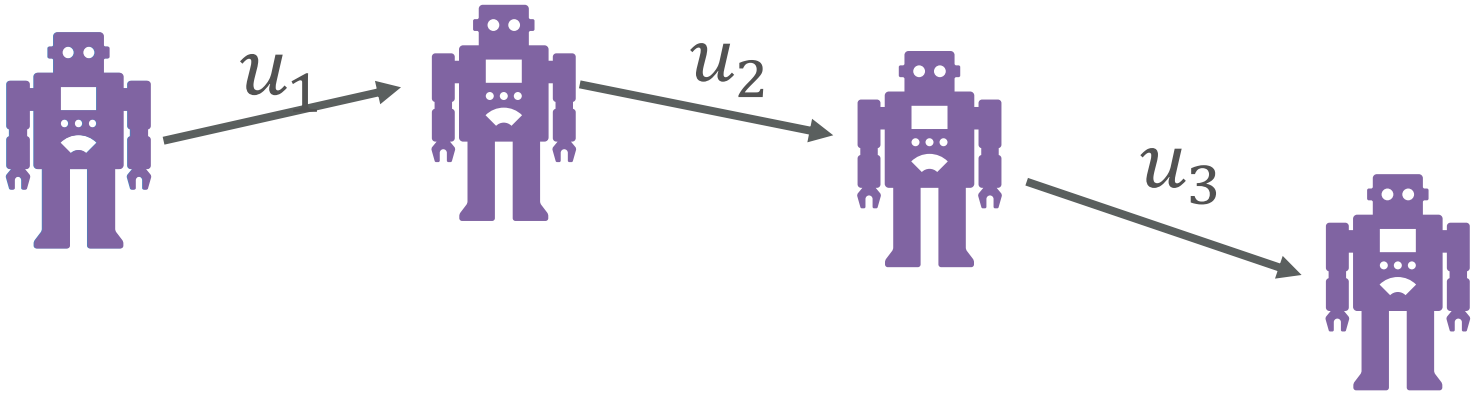
Minimizing this gives posterior state estimate

$$\Sigma_t = (\frac{\partial^2 Loss(x)}{\partial x^2}|_{x=x_{opt}})^{-1}$$

Full System Combining State Estimation and MPC controller

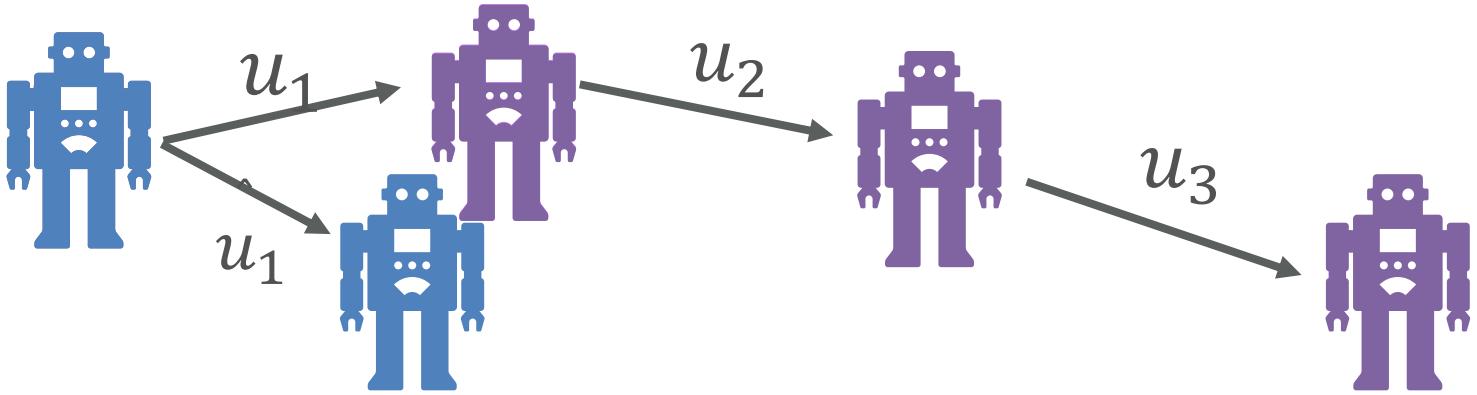


MPC controller



Plan

MPC controller

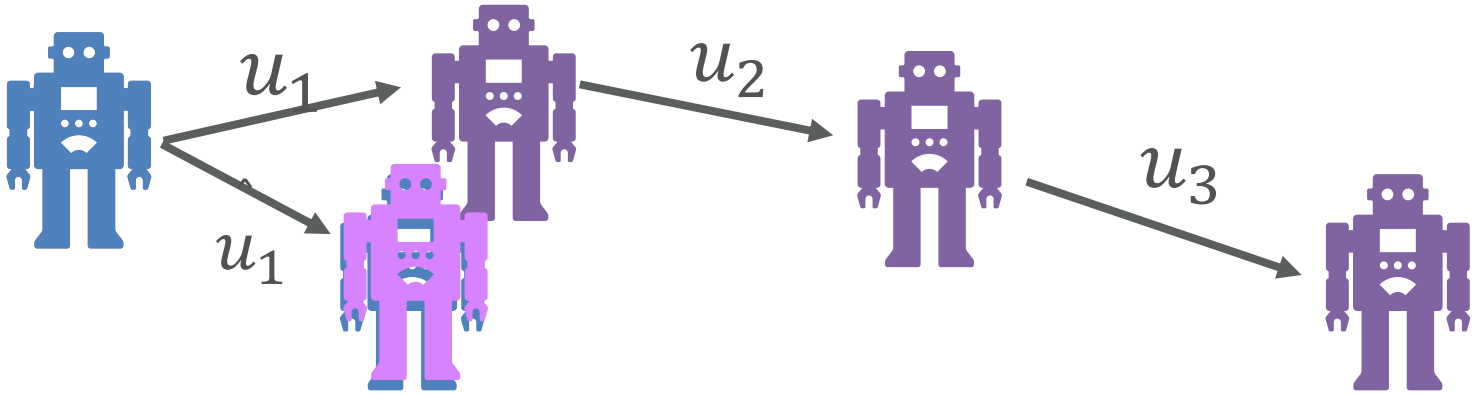


Execute Action

Plan

Ground Truth

MPC controller



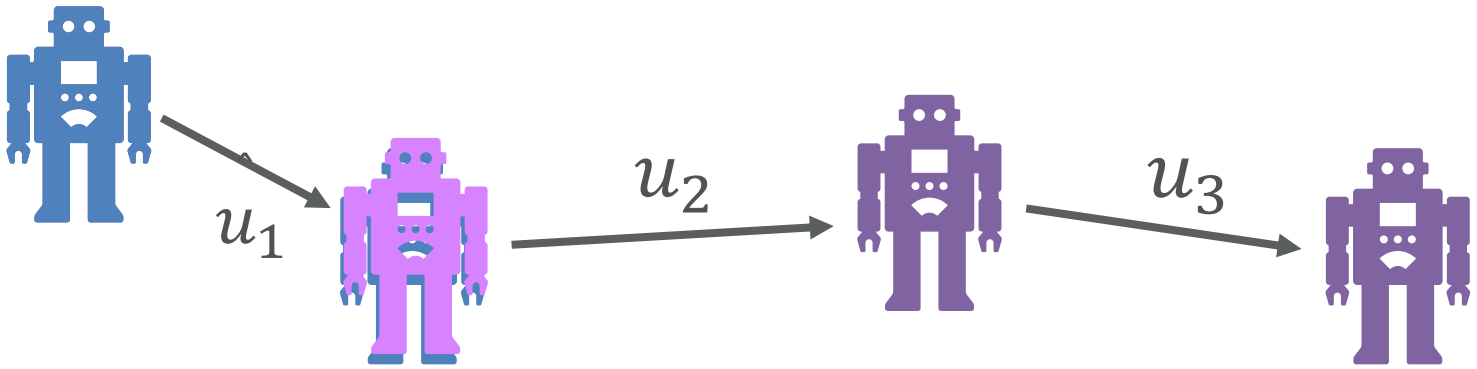
Estimate State

Plan

Ground Truth

State Estimate

MPC controller



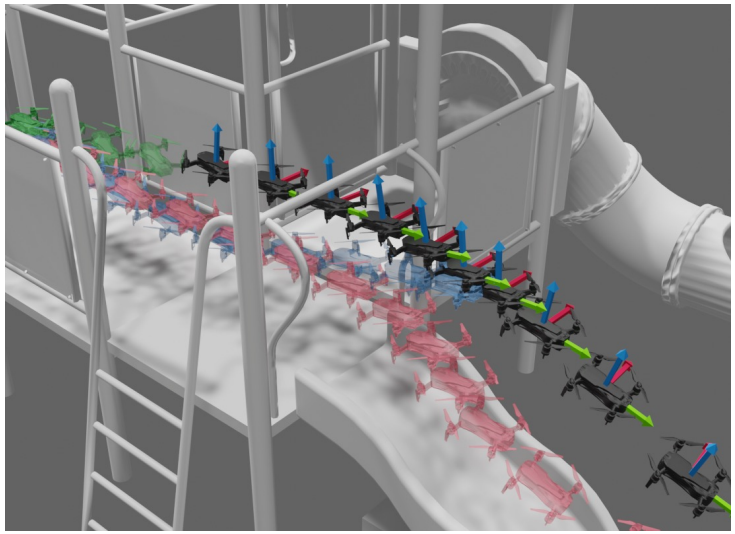
Replan

Plan

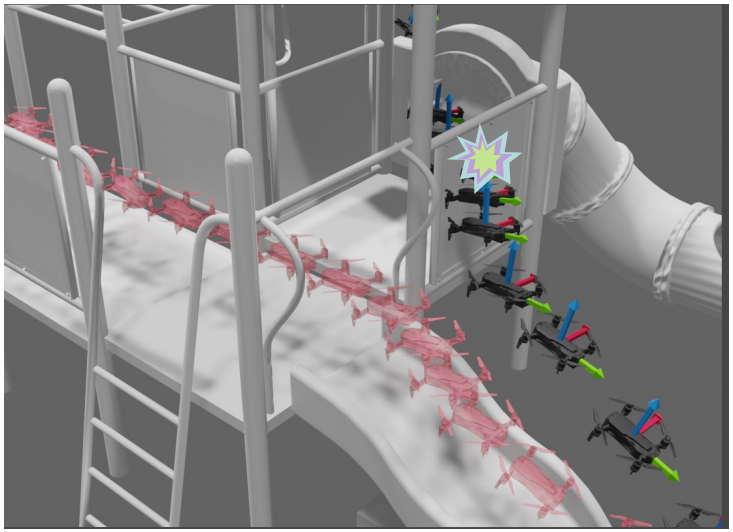
Ground Truth

State Estimate

MPC controller



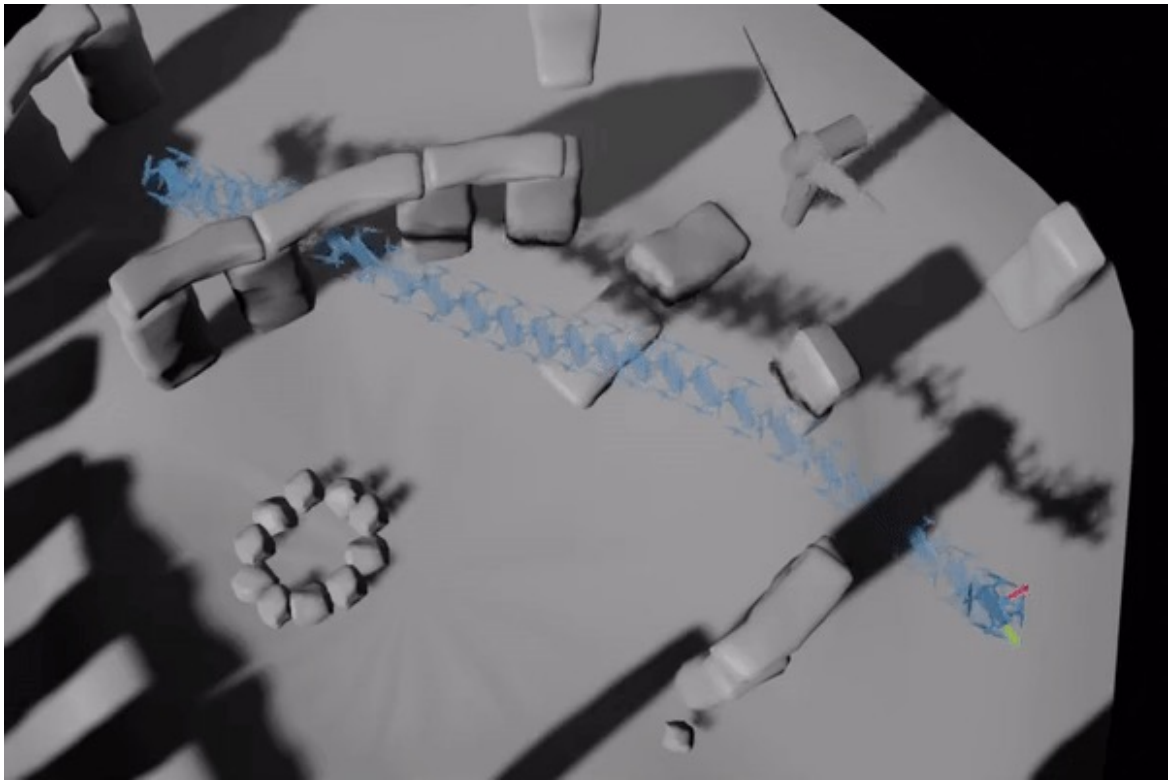
With replanning



Open loop

Red is initially planned trajectory

MPC controller



CS231A

Computer Vision:

From 3D Reconstruction to Recognition

Next lecture:

Angjoo Kanazawa from UC Berkeley