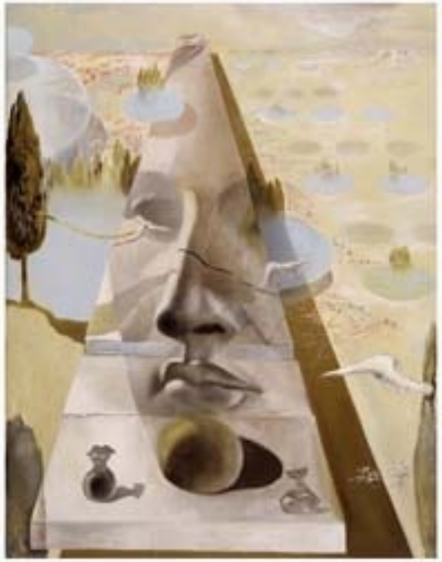
CS231A

Computer Vision: From 3D Reconstruction to Recognition

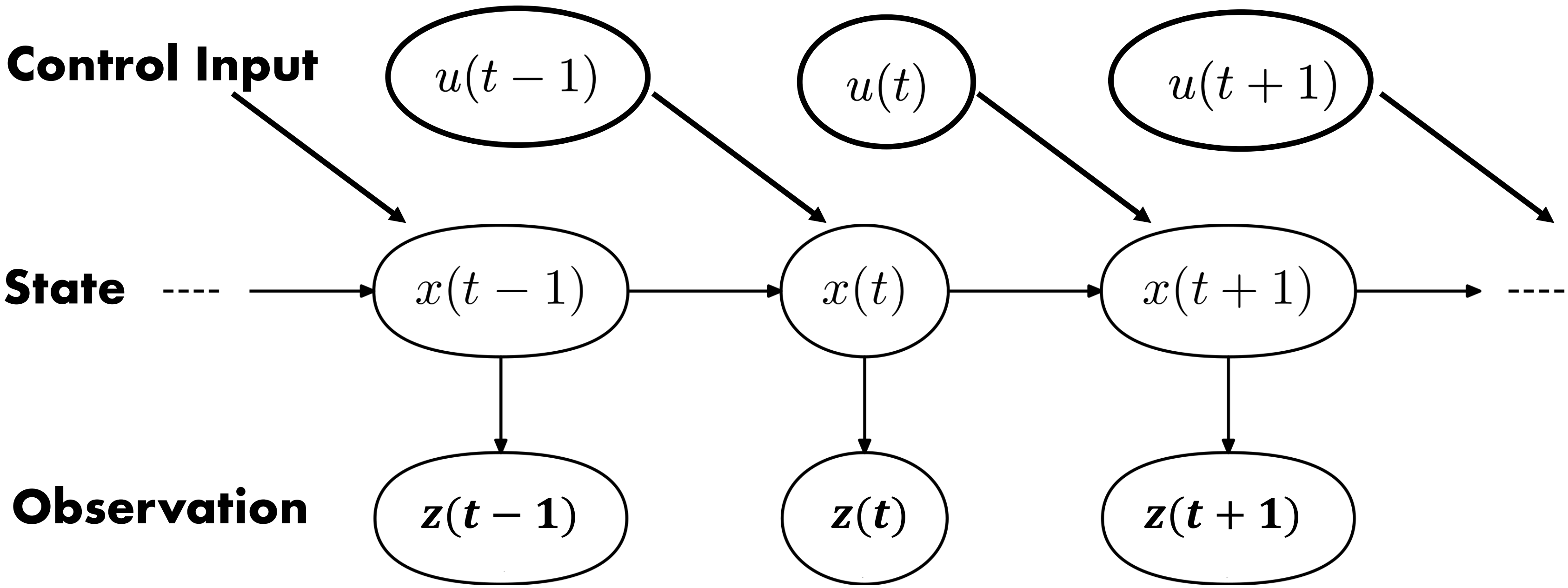


Neural Radiance Fields for Novel View Synthesis

Outline

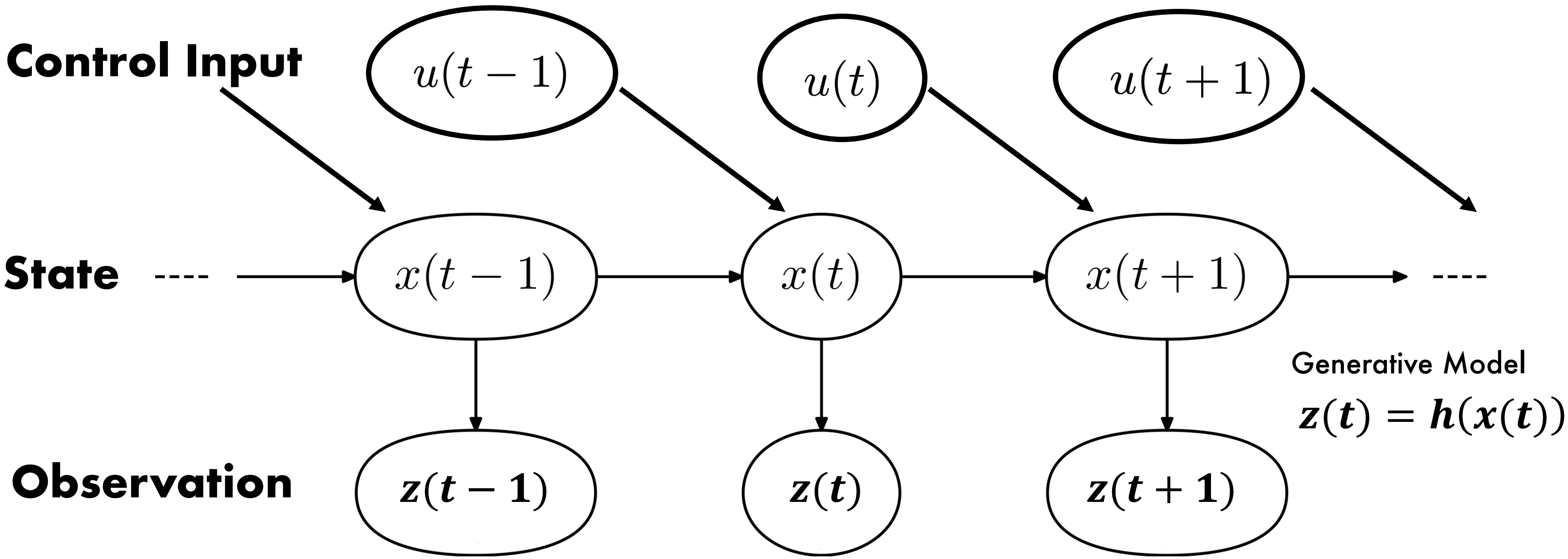
- Recap: Filtering and Generative Observation Models
- Representations for Novel View Synthesis
- Neural Radiance Fields

Graphical Model of System to Estimate



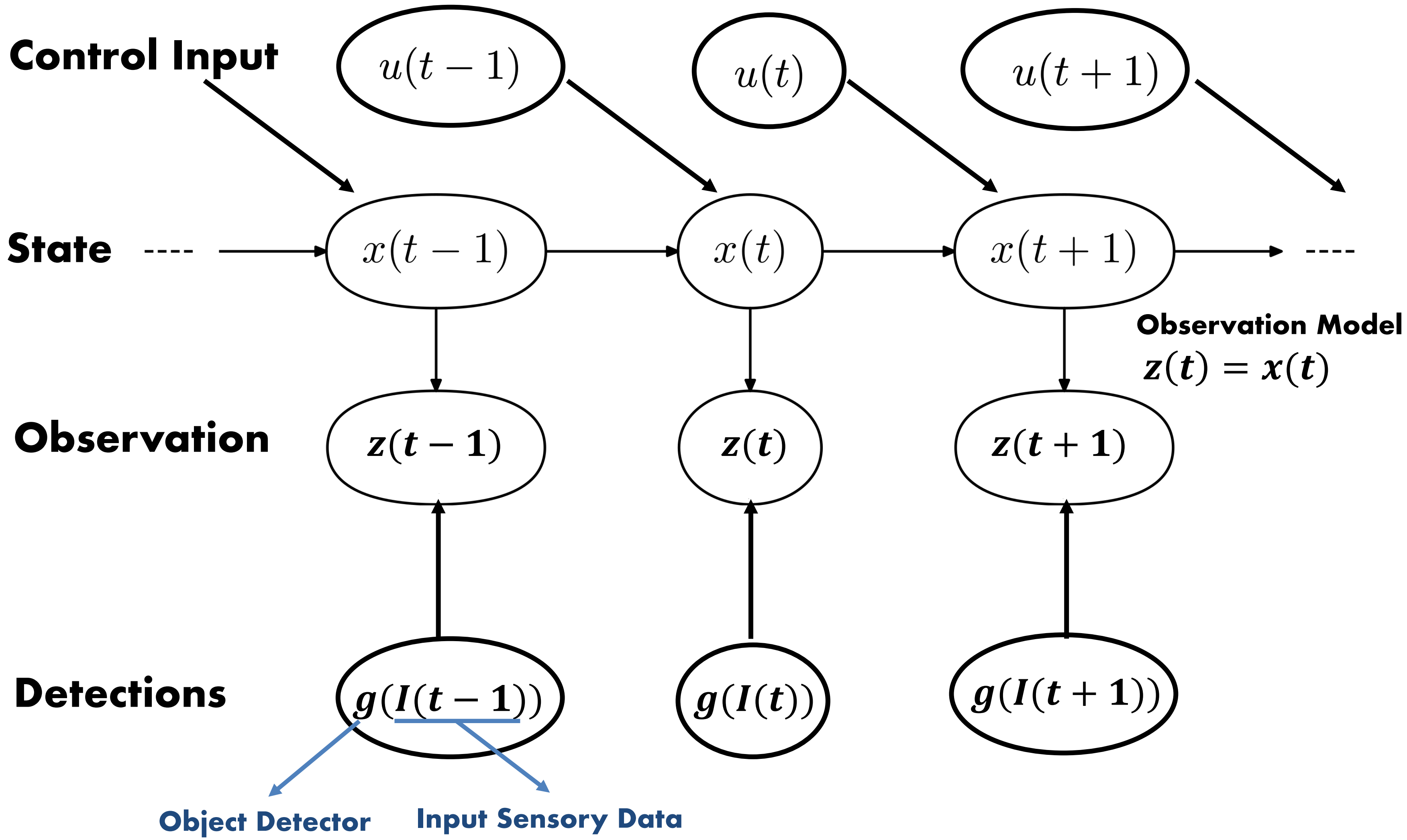
Algorithm Bayes_filter($bel(x_{t-1}), u_t, z_t$**):**
  for all x_t do
    $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx$
    $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$
  endfor
  return $bel(x_t)$

Graphical Model of System to Estimate

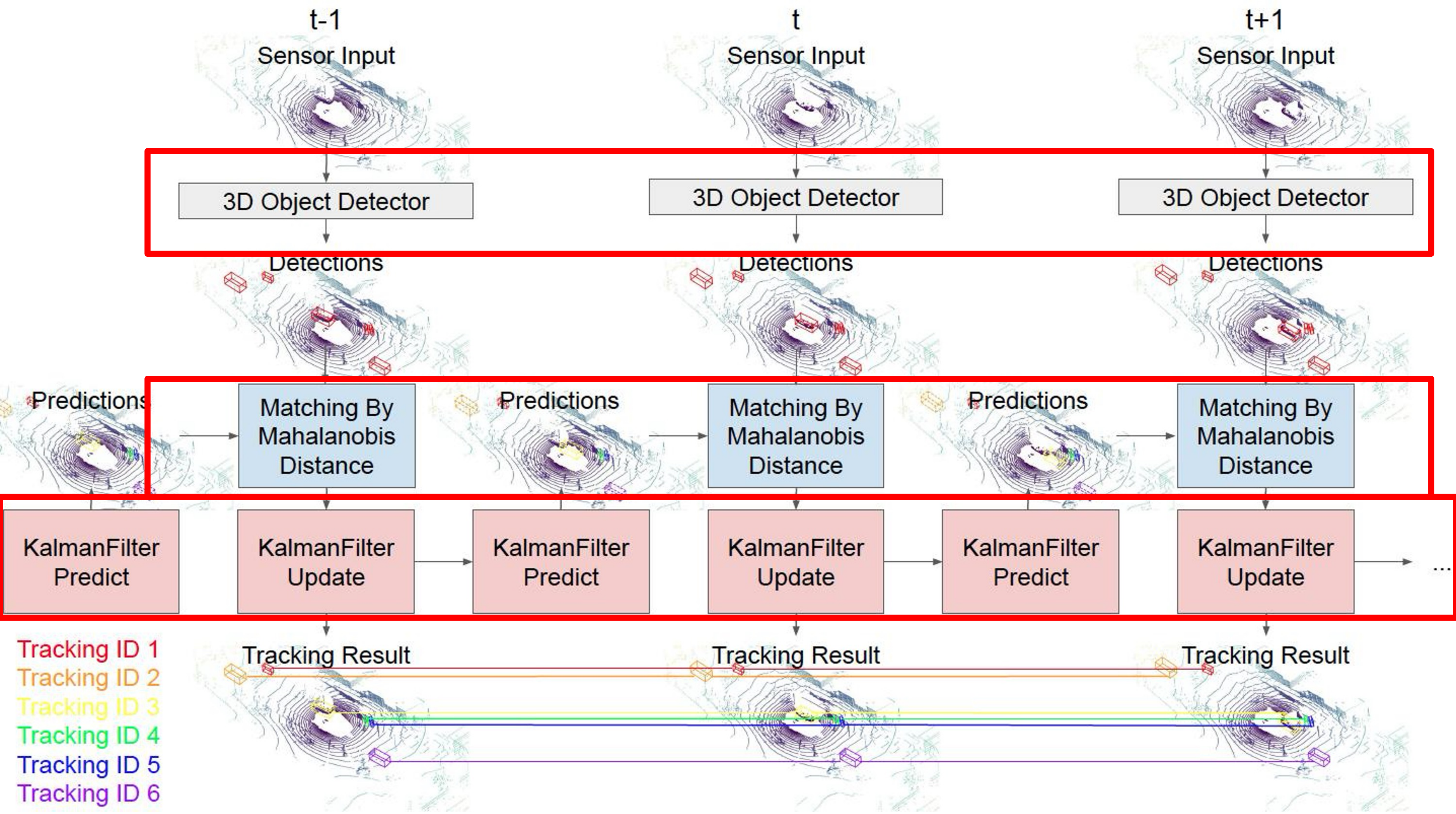


Algorithm Bayes_filter($bel(x_{t-1}), u_t, z_t$):
    for all x_t do
        $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx$
        $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$
    endfor
    return $bel(x_t)$

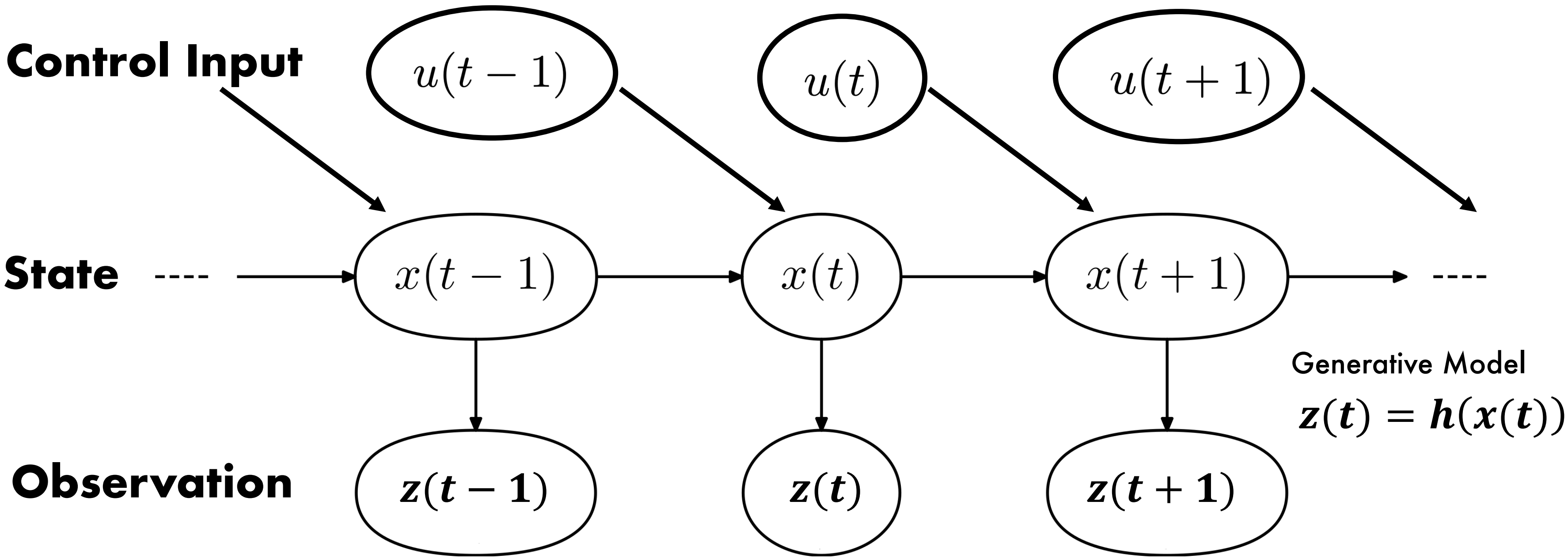
Tracking by Detection



Multi-Object Tracking by Detection

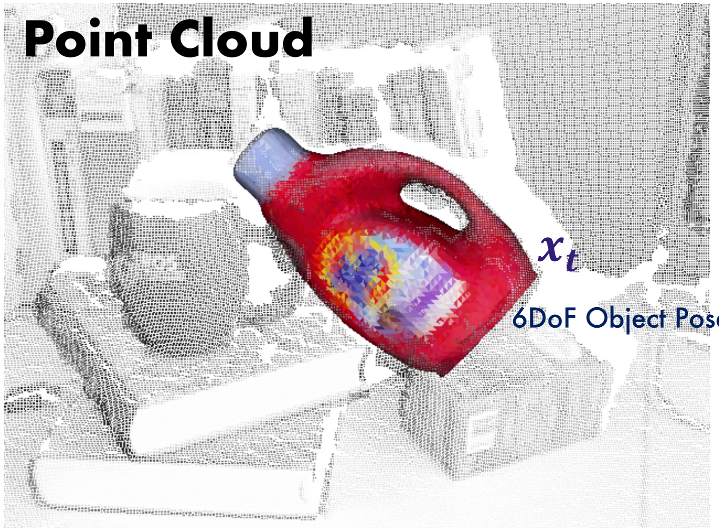


Graphical Model of System to Estimate



Algorithm Bayes_filter($bel(x_{t-1}), u_t, z_t$)**:**
  for all x_t do
    $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx$
    $bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$
  endfor
  return $bel(x_t)$

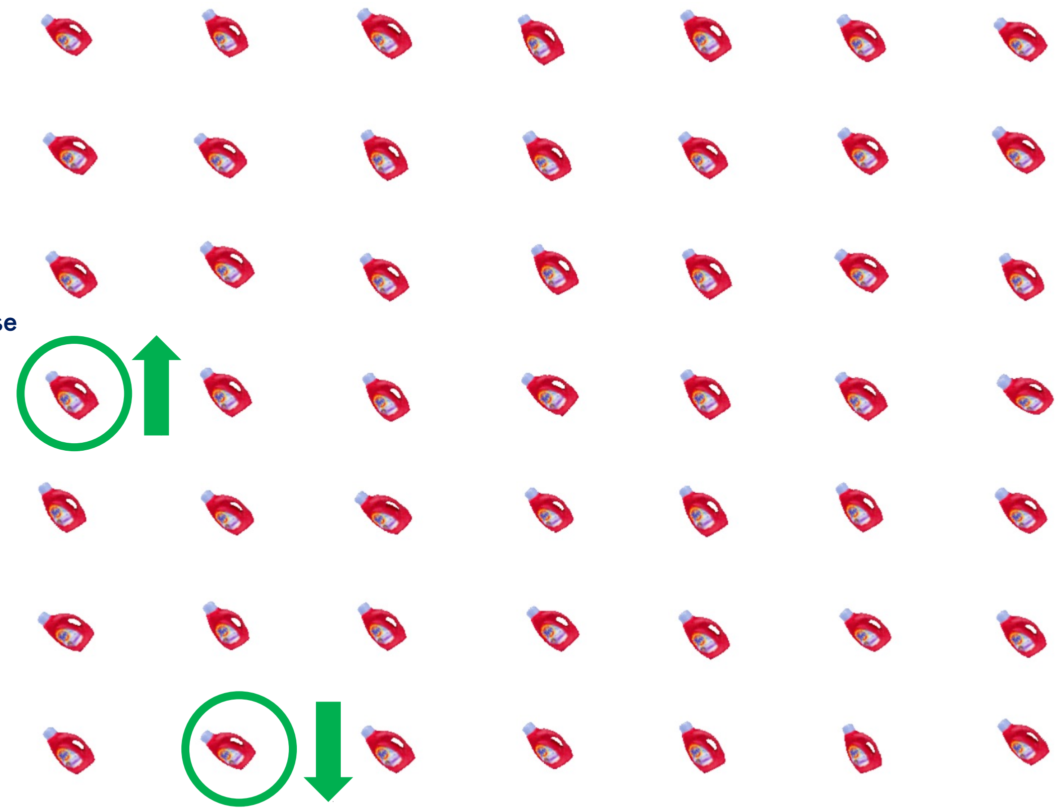
Example Observation model for 3D object



Algorithm Particle_filter$(\mathcal{X}_{t-1}, u_t, z_t)$**:**

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
for $m = 1$ to M do
  sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
  $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
endfor
for $m = 1$ to M do
  draw i with probability $\propto w_t^{[i]}$
  add $x_t^{[i]}$ to $\mathcal{X}_t$
endfor
return $\mathcal{X}_t$

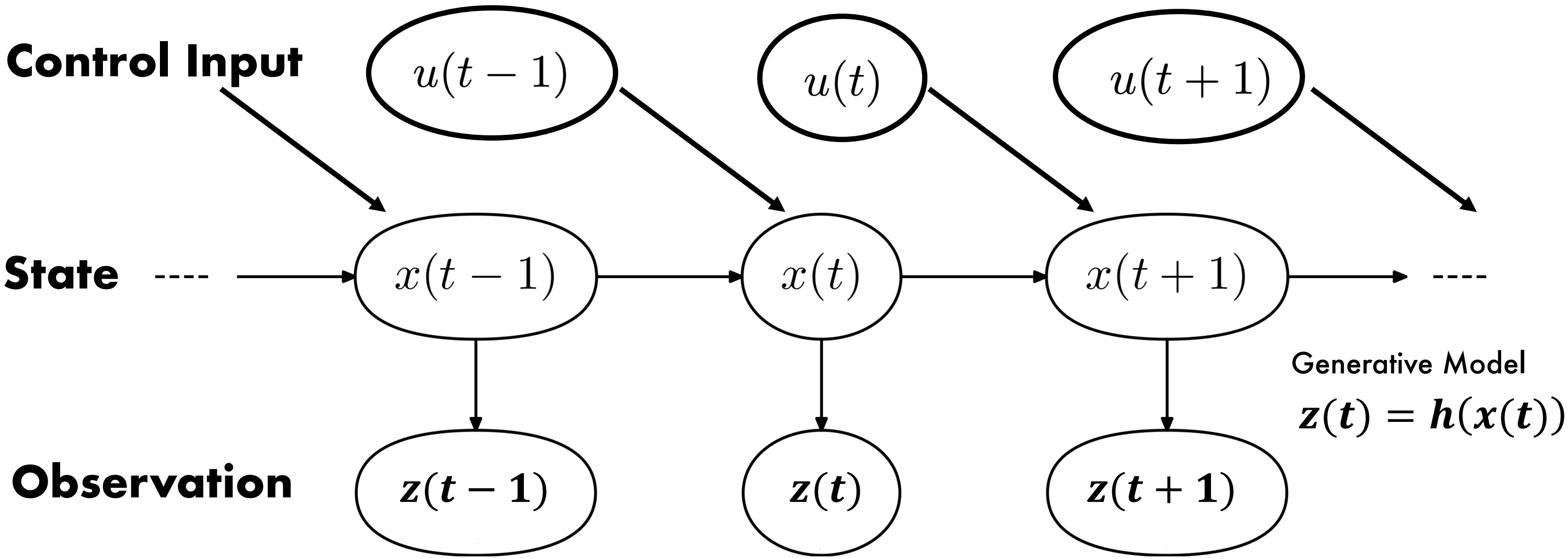
Importance Sampling



Rendered Particles

Changhyun Choi and Henrik I. Christensen. Rgb-d object tracking: A particle filter approach on gpu. In IROS, pages 1084–1091, 2013

Graphical Model of System to Estimate



Algorithm Bayes_filter($bel(x_{t-1}), u_t, z_t$)**:**
for all x_t do
$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx$
$bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$
endfor
return $bel(x_t)$

Novel view synthesis

- Can be an implementation of the generative observation model
- A scene learned from a few discrete views
  - Let's say you want to localize the camera relative to the scene in new poses
  - Track camera pose with filter

NeRF

Representing Scenes as Neural Radiance Fields for View Synthesis

ECCV 2020 Oral - Best Paper Honorable Mention

Ben Mildenhall* UC Berkeley | Pratul P. Srinivasan* UC Berkeley | Matthew Tancik* UC Berkeley | Jonathan T. Barron Google Research | Ravi Ramamoorthi UC San Diego | Ren Ng UC Berkeley

*Denotes Equal Contribution

The problem of novel view synthesis

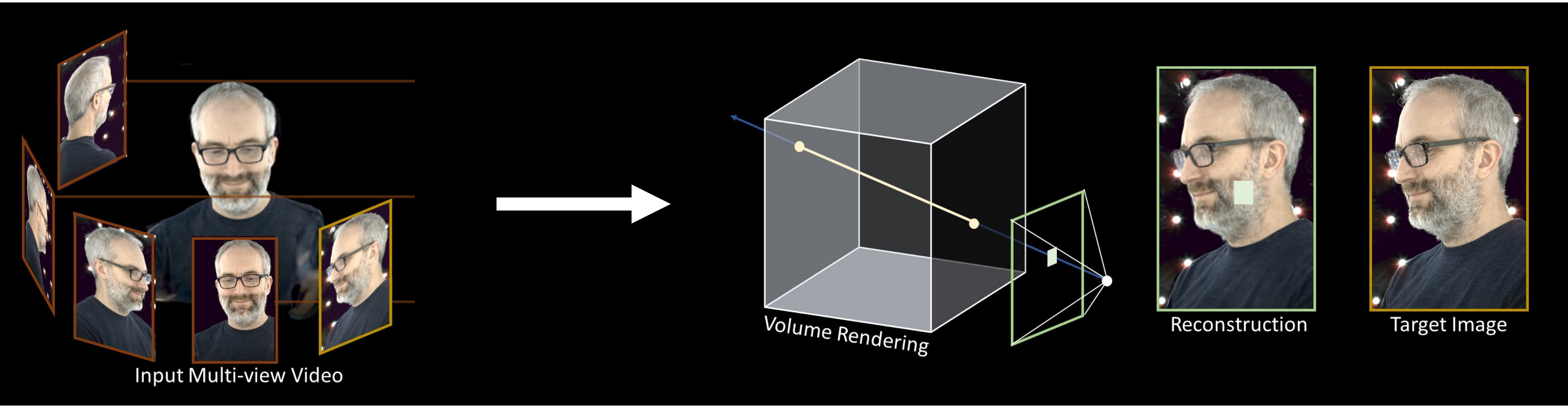
Inputs: sparsely sampled images of scene

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

2

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

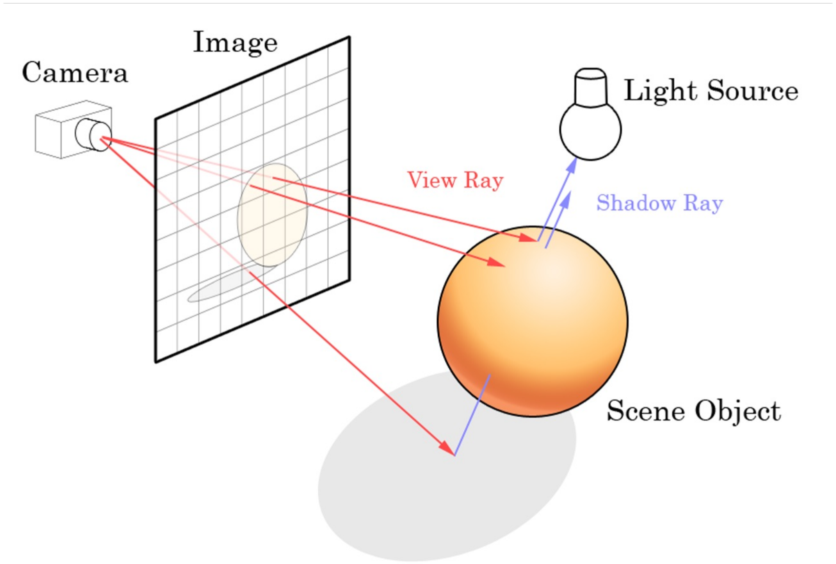
One Approach: Reconstruct 3D voxel RGB-alpha grid



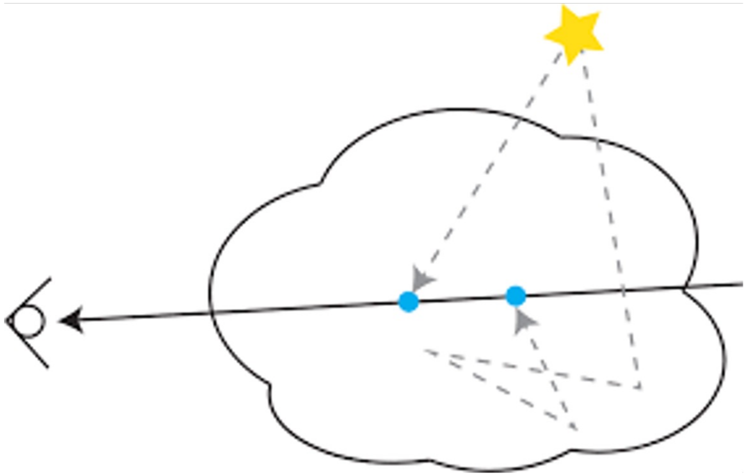
Multiview geometry for Reconstruction, Shape Carving, …

Neural Volumes, Lombardi et al. 2019

Ways to Render



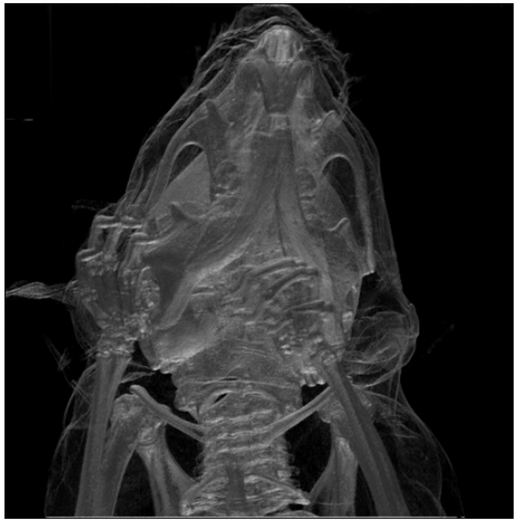
Surface rendering



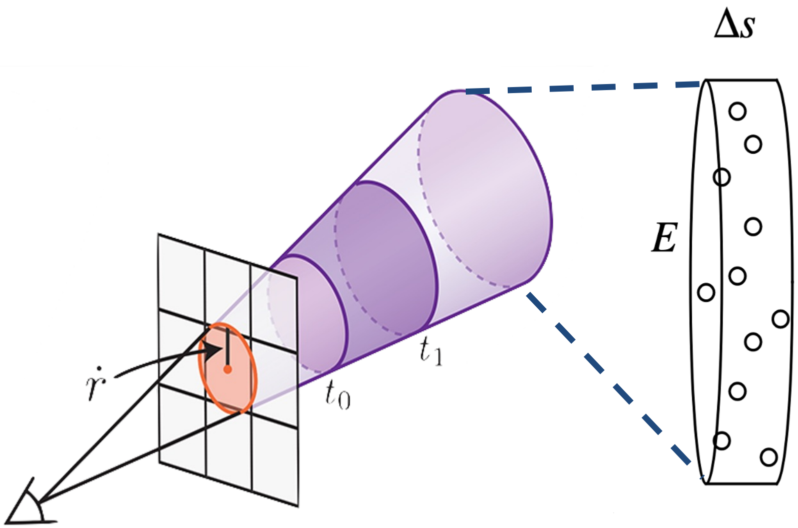
Volume rendering

Reasons to use volume rendering

- Show smoke / other diffuse effects in scene.
- Generating surfaces from 3D data can produce nasty artifacts; volume renderings are “soft.”
- Don’t need to reason about *where* surfaces are located to reflect light.



Physical model



- Ray defines a cylinder in space which contains particles.
- Particles can:
  - emit light
  - occlude light from behind them
  - reflect / scatter light from environment

Volume rendering equation

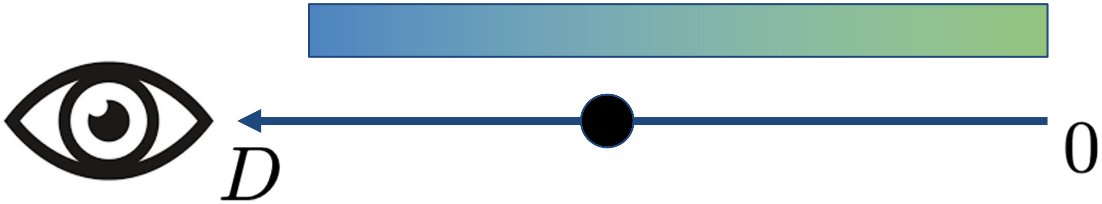
$$\mathbf{I}(D) = \mathbf{I}_0 T(0) + \int_0^D \mathbf{c}(s)\rho(s)T(s)ds$$

pixel color at coordinates D

radiance **density**

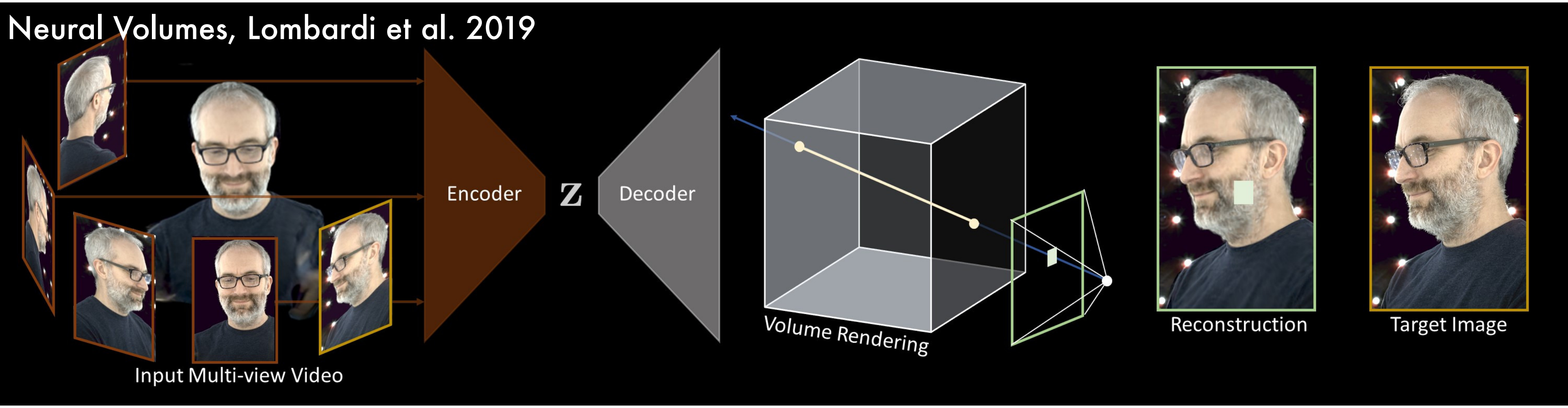
$$T(s) = \exp\left(-\int_s^D \rho(t)dt\right)$$

transparency

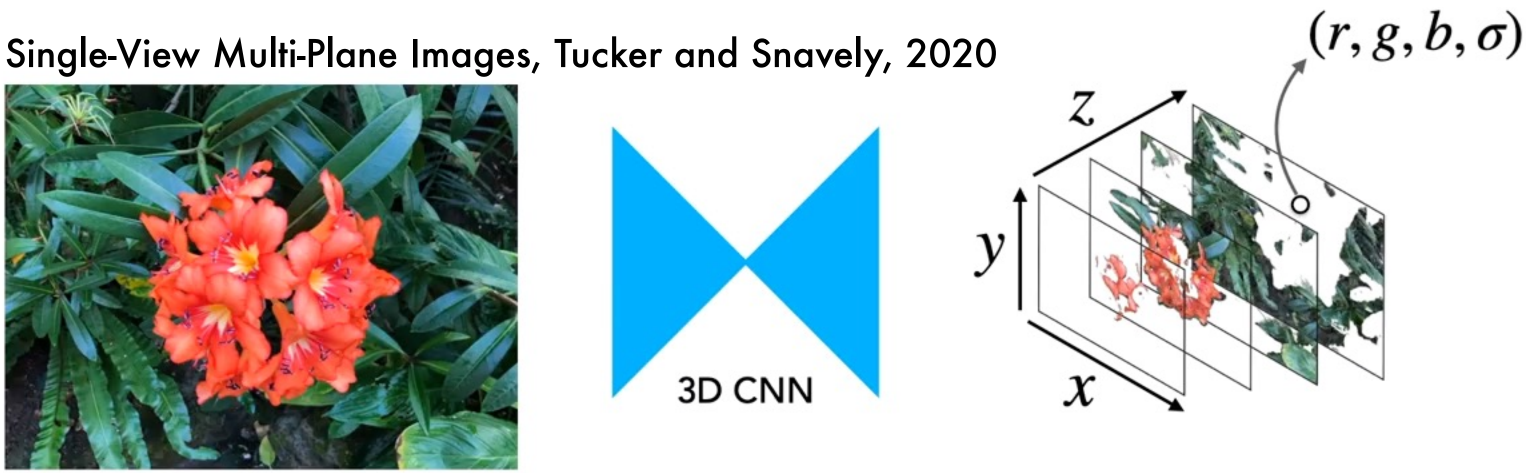


$$(\mathbf{c}(s), \rho(s))$$

Predict 3D Voxel RGB-alpha Grid



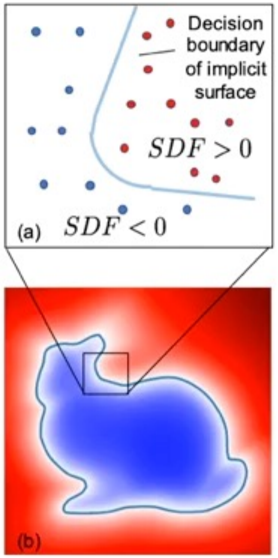
Single-View Multi-Plane Images, Tucker and Snavely, 2020



Pros and Cons of RGB-alpha volume rendering for view Synthesis

- Alpha Composition is trivially differentiable, plays nicely with gradient-based optimization
- Bad storage requirements for 3D grid

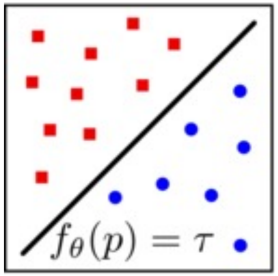
Neural networks as a shape representation



DeepSDF, Park et al. 2019

Supervised with 3D:
DeepSDF [Park et al. 2019],
Occupancy Networks [Mescheder et al. 2019],
Local Deep Implicit Functions [Genova et al. 2020],
Local Implicit Grids [Jiang et al. 2020]

Supervised with images:
Scene Representation Networks [Sitzmann et al. 2019],
Differentiable Volumetric Rendering [Niemeyer et al. 2020],
DIST [Liu et al. 2020]



Pros and Cons of Neural networks as a continuous shape representation

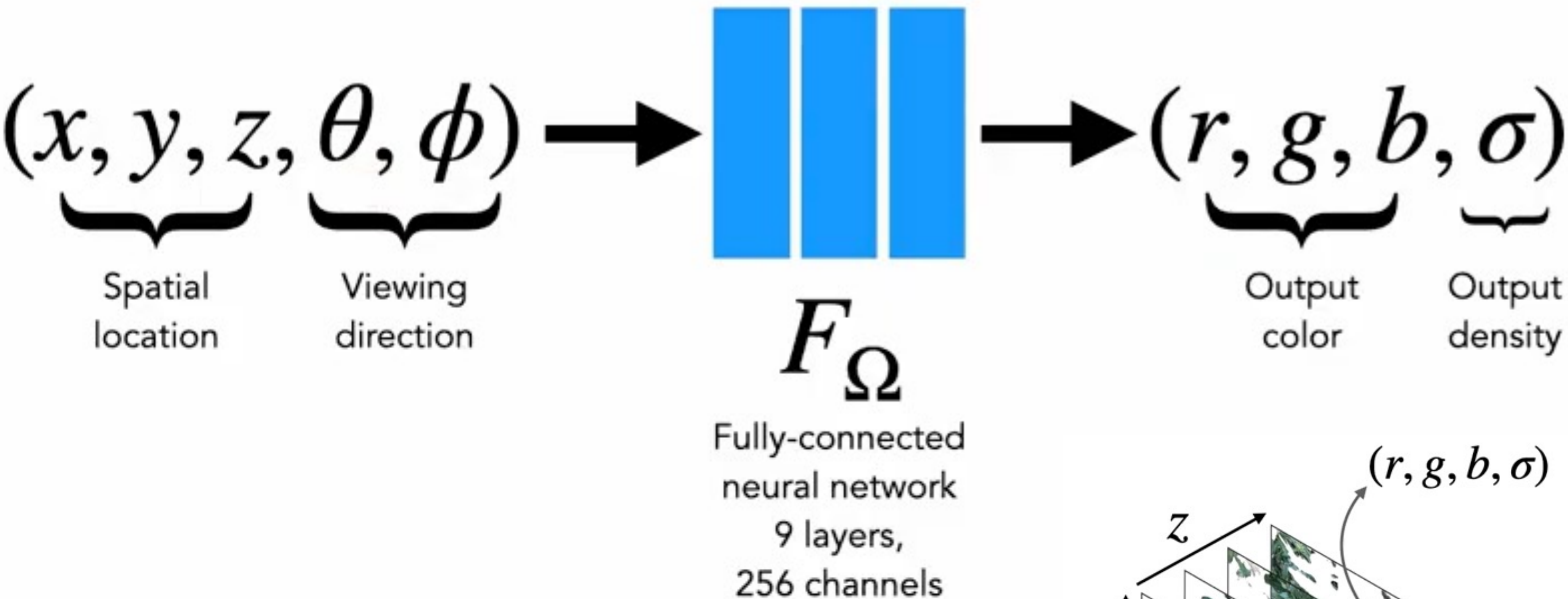
- Limited rendering model: Difficult to optimize (Shape as surface instead of volume)
- Highly compressible

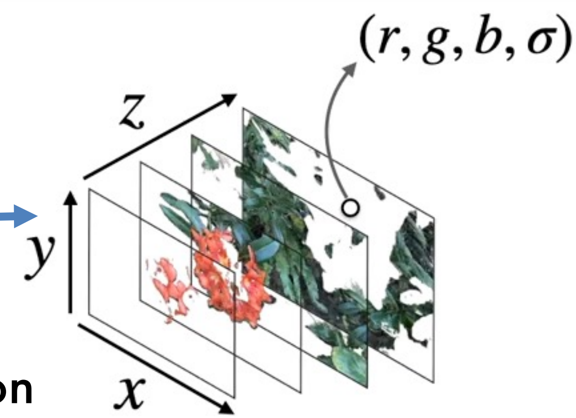
NeRF (neural radiance fields)

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

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

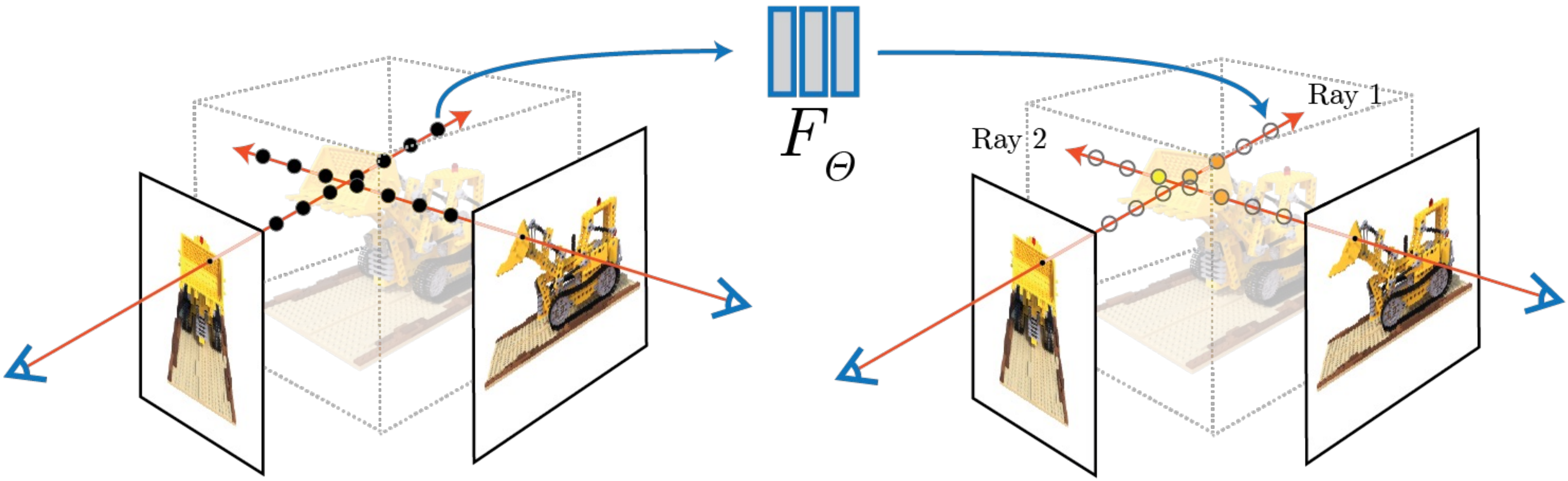
Represent a scene as a continuous 5D function





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

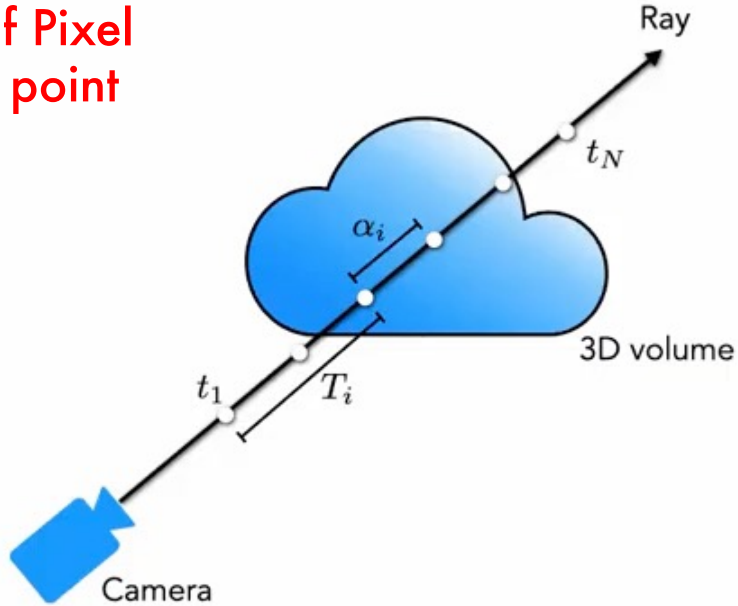
Rendering model for ray r(t) = o + td:

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

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

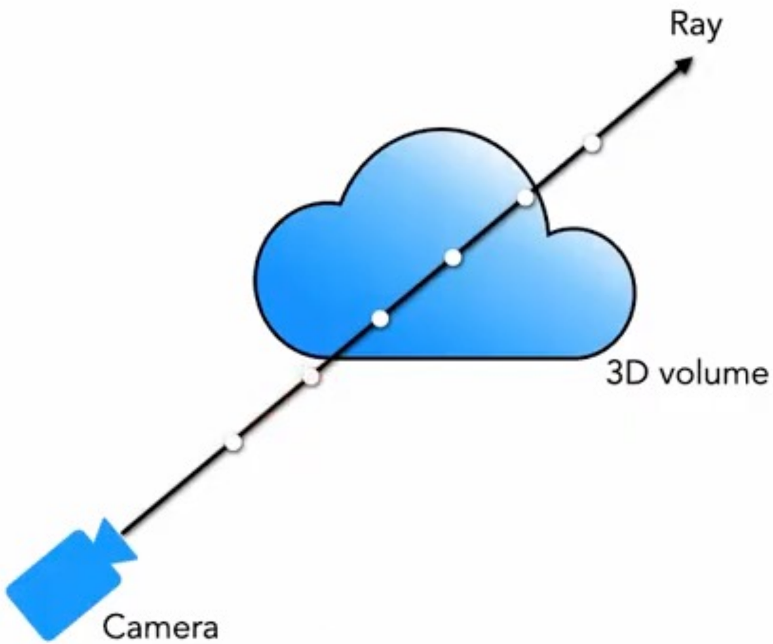
How much light is contributed by ray segment *i*:

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

Function of segment length δt_i and volume density σ

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

Can we allocate samples more efficiently? Two pass rendering

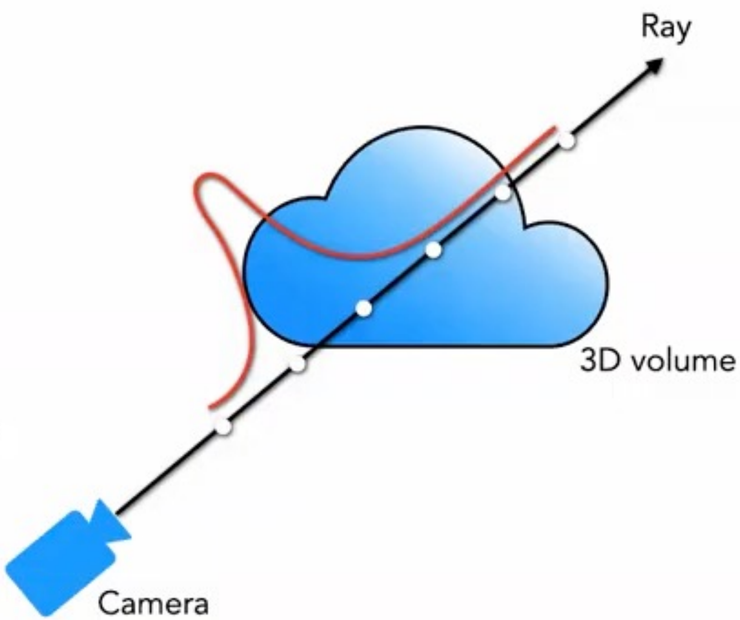


From Presentation by Matthew Tancik: Neural Radiance Fields for View Synthesis. 2020.

Two pass rendering: coarse

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

treat weights as probability
distribution for new samples

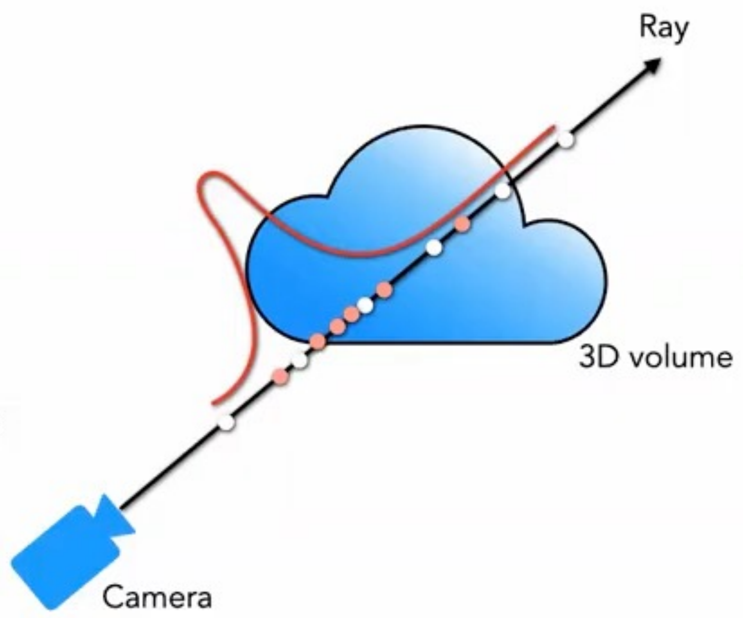


From Presentation by Matthew Tancik: Neural Radiance Fields for View Synthesis. 2020.

Two pass rendering: fine

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

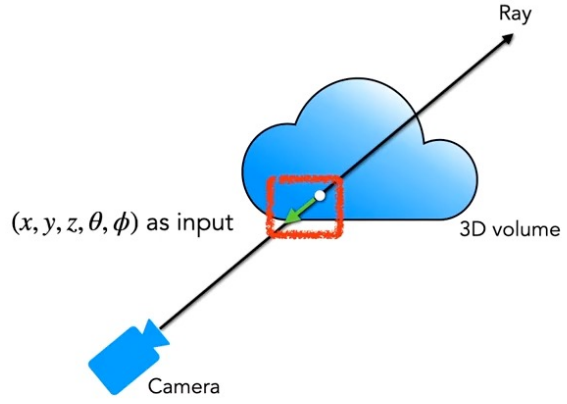
treat weights as probability distribution for new samples

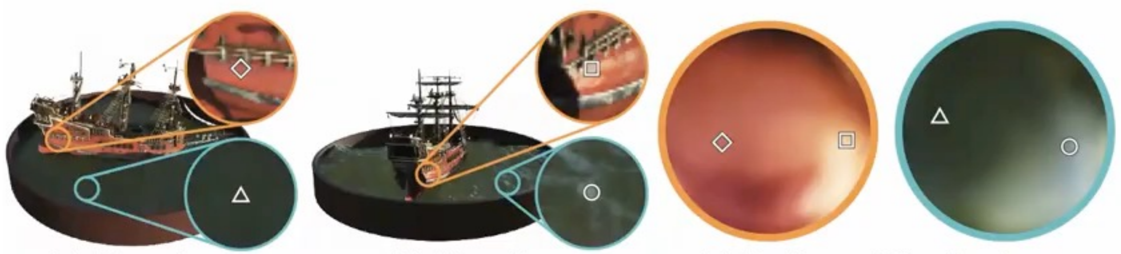


From Presentation by Matthew Tancik: Neural Radiance Fields for View Synthesis. 2020.

Viewing direction as input

- Color of any point varies as function of viewing direction, i.e. Radiance field
- If points are fixed but direction varies, the view dependent specularity comes out





(a) View 1 (b) View 2 (c) Radiance Distributions

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

Volume rendering is trivially differentiable

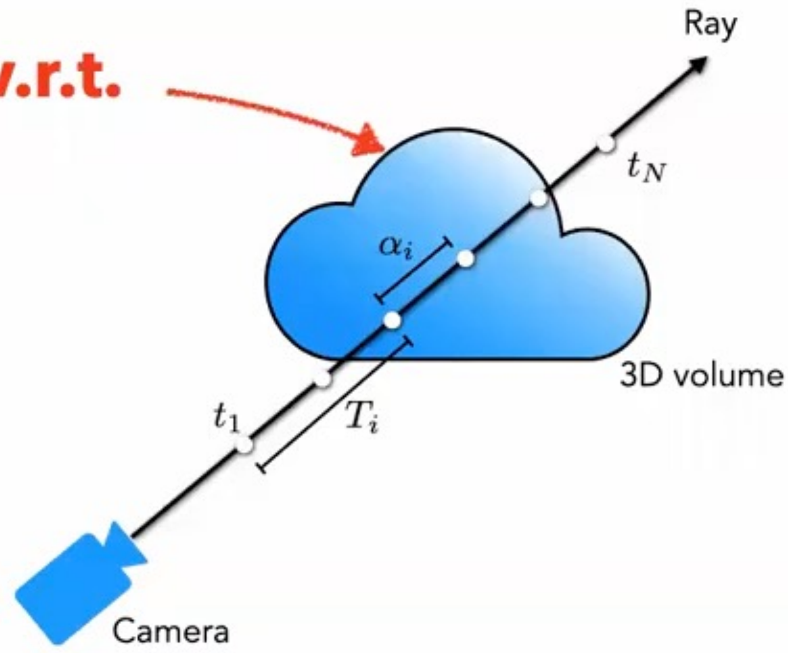
Rendering model for ray r(t) = o + td:

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

weights

colors

differentiable w.r.t.



How much light is blocked earlier along ray:

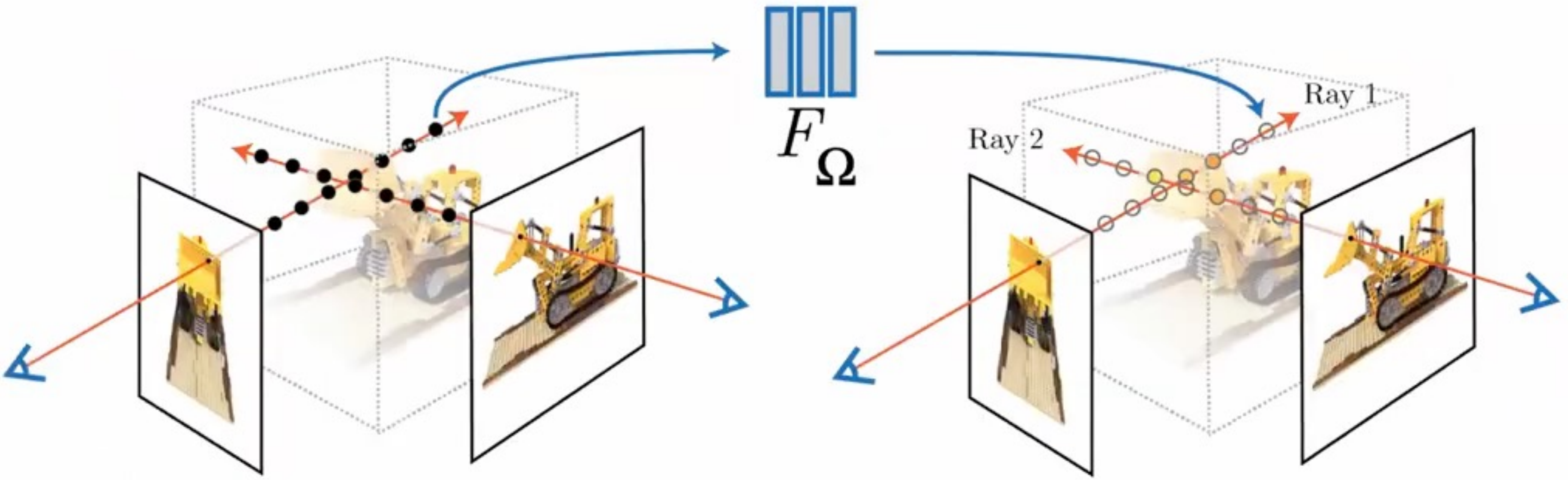
$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

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

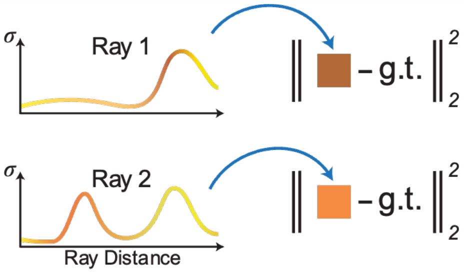
$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$

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

Optimize with gradient descent on rendering loss



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

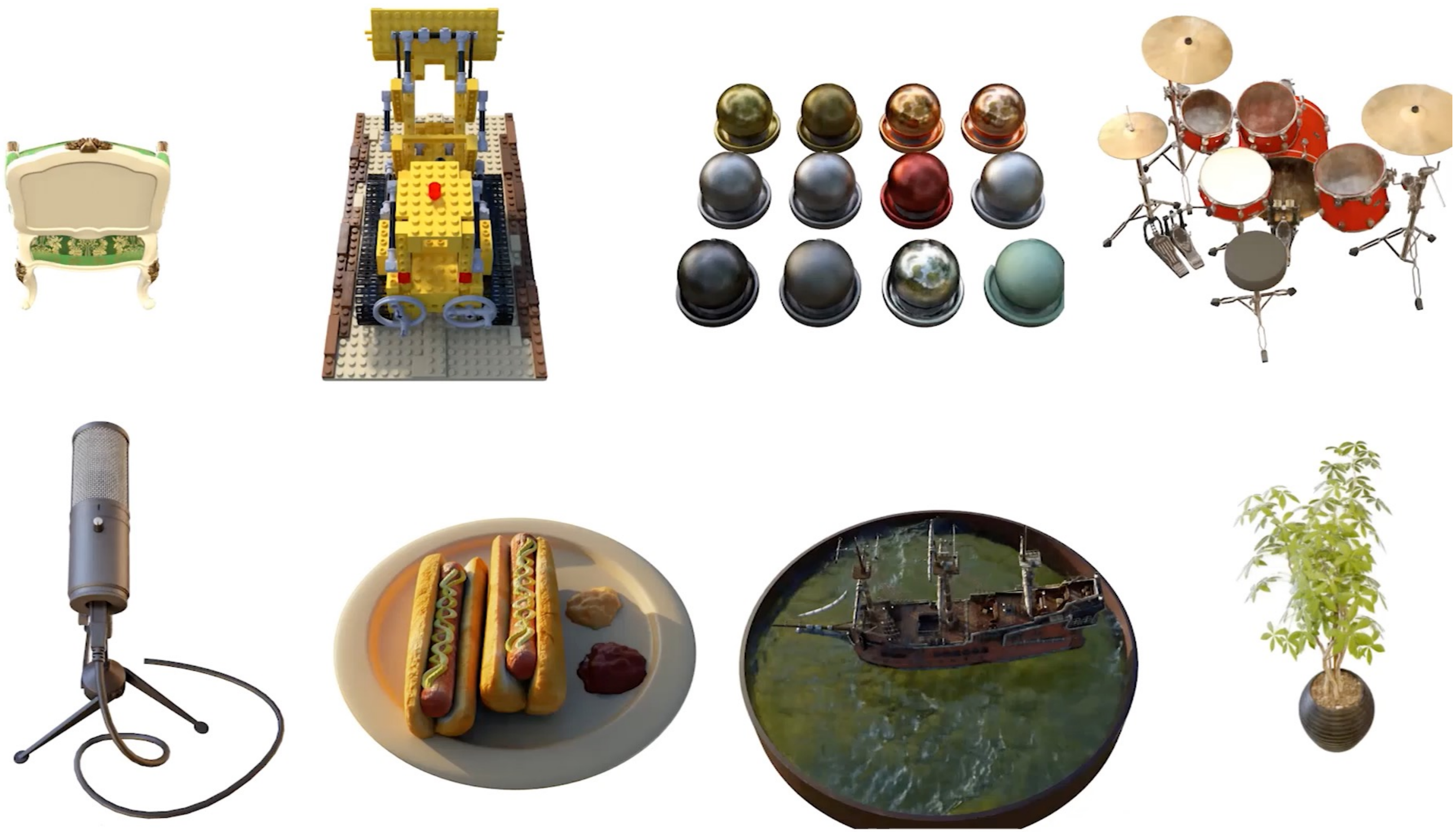


From Presentation by Matthew Tancik: Neural Radiance Fields for View Synthesis. 2020.

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 – View Dependent Appearance

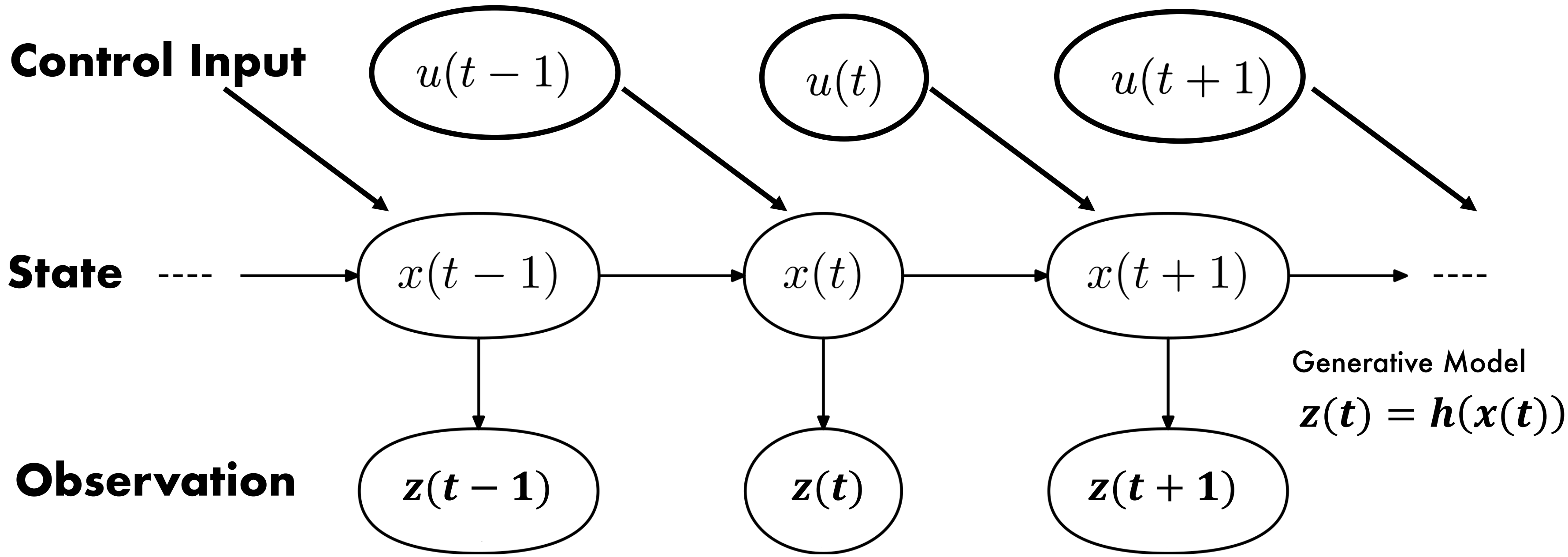
Results – View Dependent Appearance

Results – Visualization Geometry

Results – Visualization Geometry

Results on Real Scenes

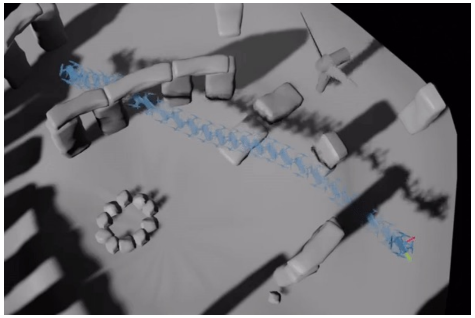
Graphical Model of System to Estimate



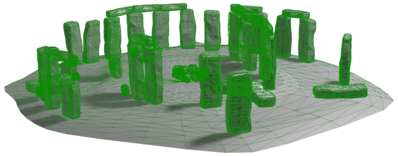
Algorithm Bayes_filter($bel(x_{t-1}), u_t, z_t$)**:**
  for all x_t do
    $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx$
    $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$
  endfor
  return $bel(x_t)$

Vision-Only Robot Navigation in a Neural Radiance World

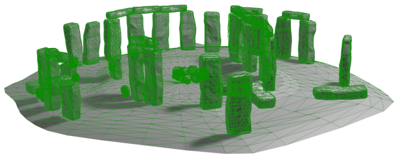
Michal Adamkiewicz*, Timothy Chen*, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, Mac Schwager



*denotes equal contribution

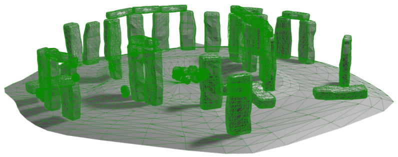


Mesh



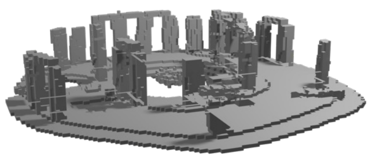
Mesh

Point Cloud

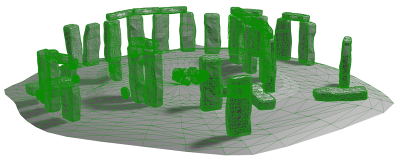


Mesh

Point Cloud

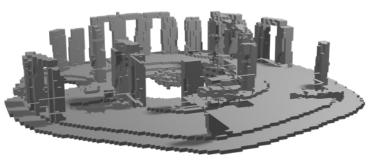


Voxels

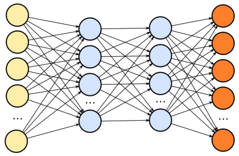


Mesh

Point Cloud



Voxels

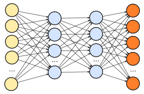


Implicit representations

Implicit Representations

density

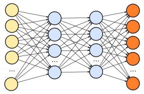
colour



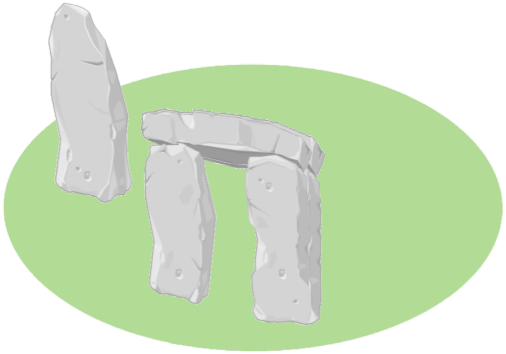
$$(x, y, z) = \sigma, c$$

position (and view direction)

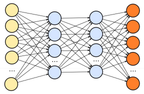
Implicit Representations



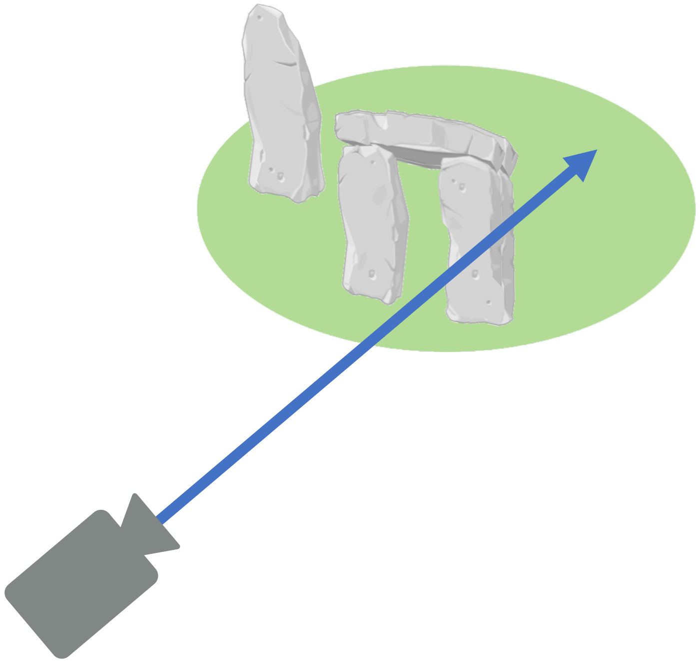
$(x, y, z) = \sigma, c$



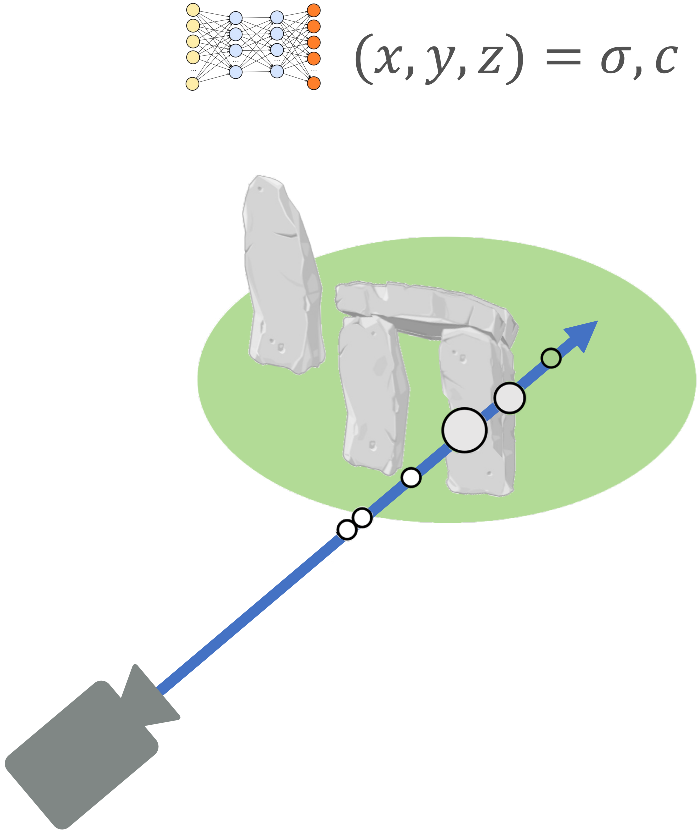
Implicit Representations



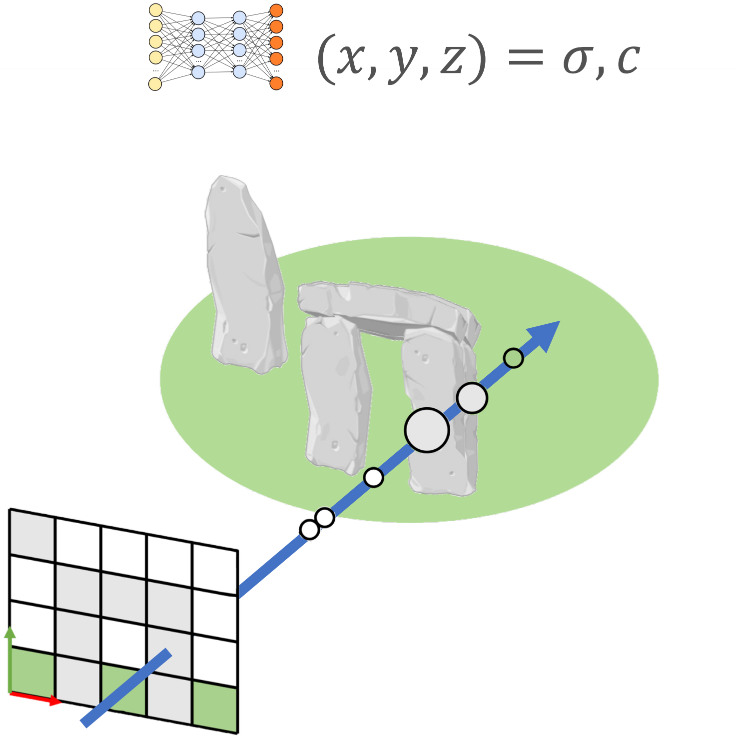
$$(x, y, z) = \sigma, c$$

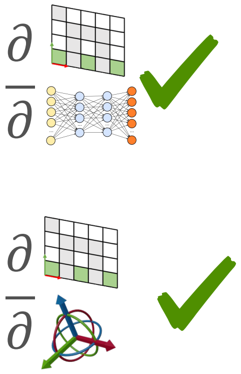


Implicit Representations

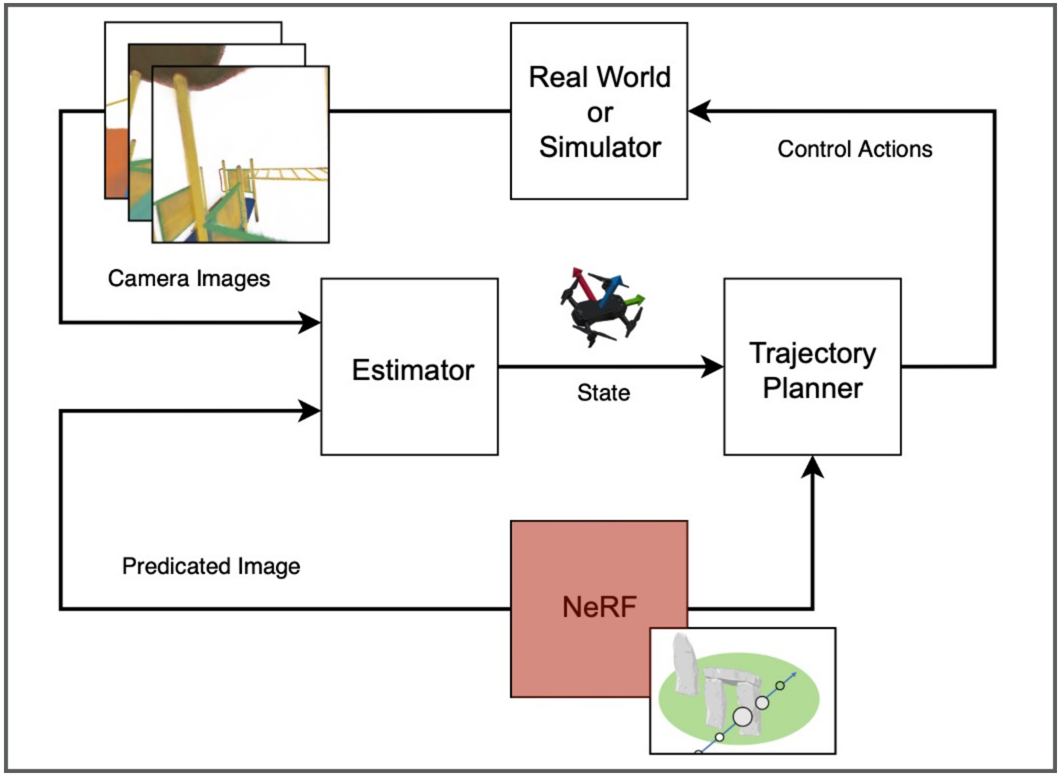


Implicit Representations

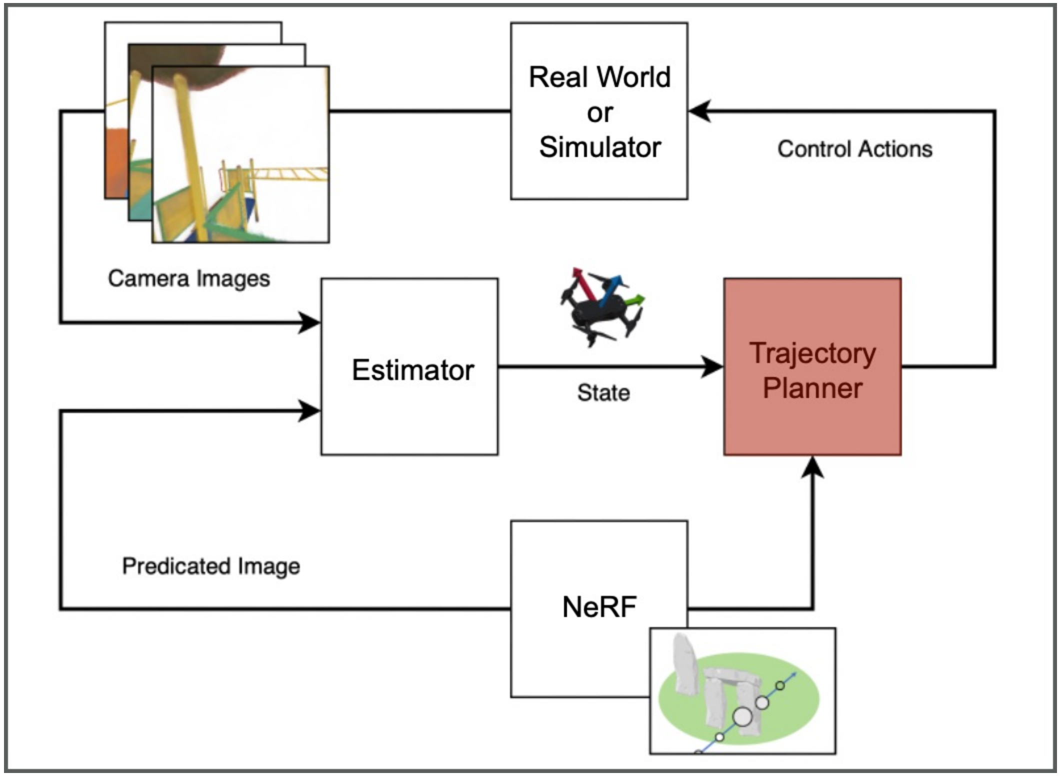




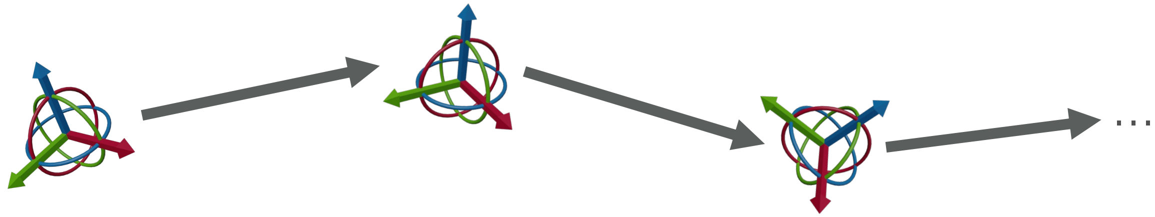
Vision-Only Navigation



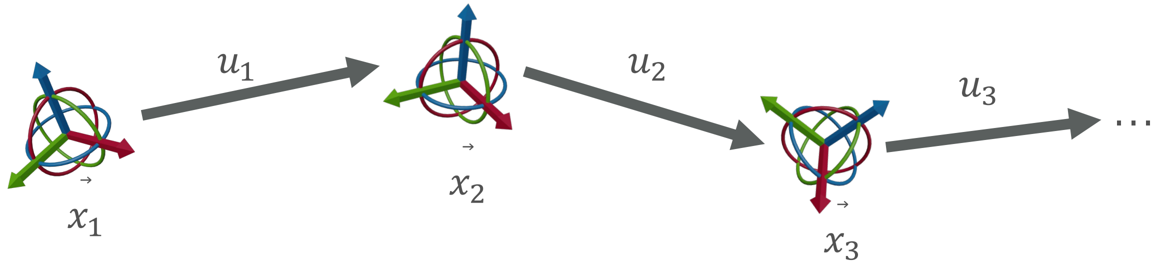
Vision-Only Navigation



Trajectory Optimization

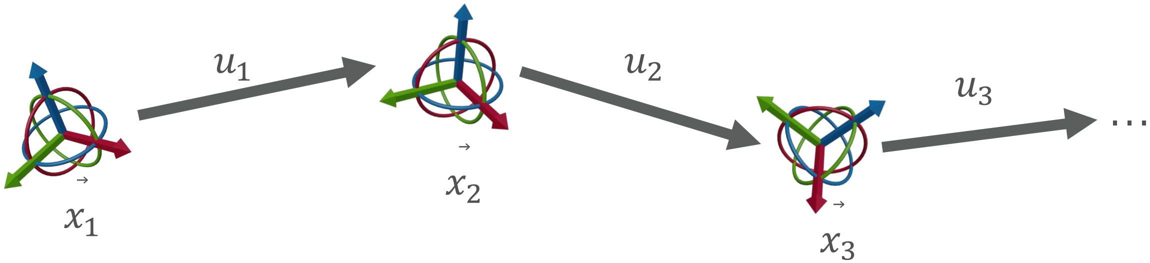


Trajectory Optimization



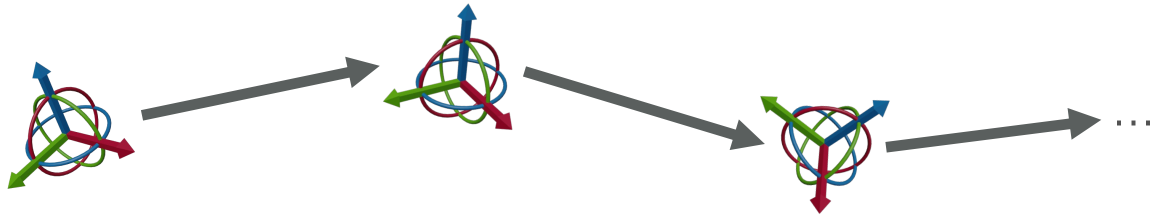
$$Loss = L_{collision} + L_{controleffort}$$

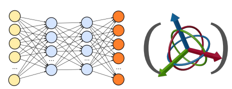
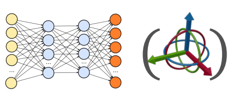
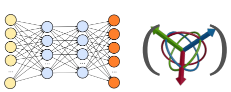
Trajectory Optimization



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

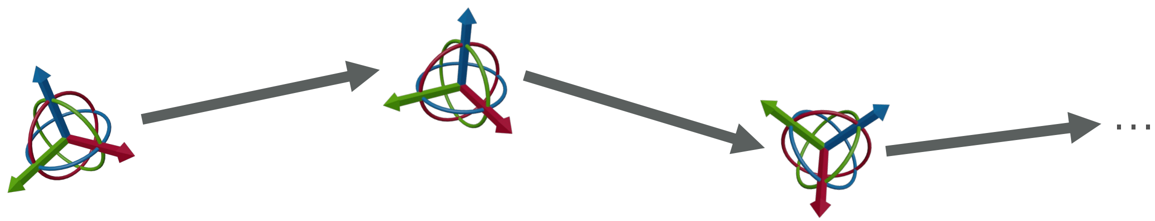
Trajectory Optimization

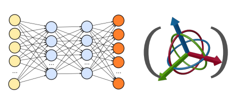
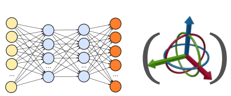
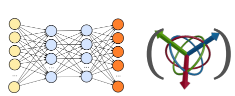


$Loss =$  $+$  $+$  $+\ldots$

$$+\Sigma L_{controleffort}$$

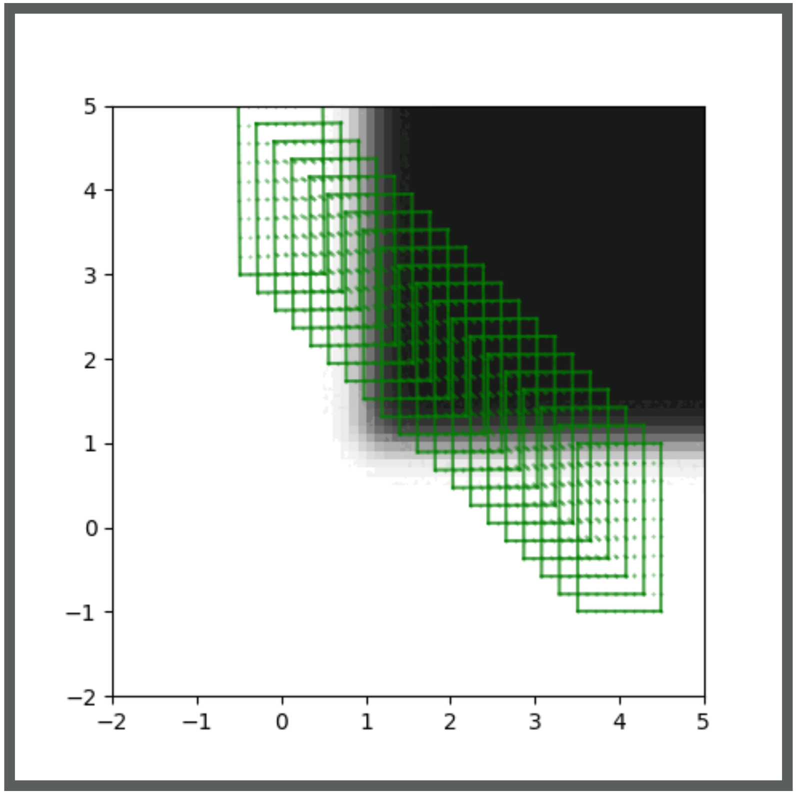
Trajectory Optimization

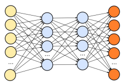
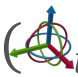


$Loss = |\vec{v}_1|$  $+ |\vec{v}_2|$  $+ |\vec{v}_3|$  $+ \dots$

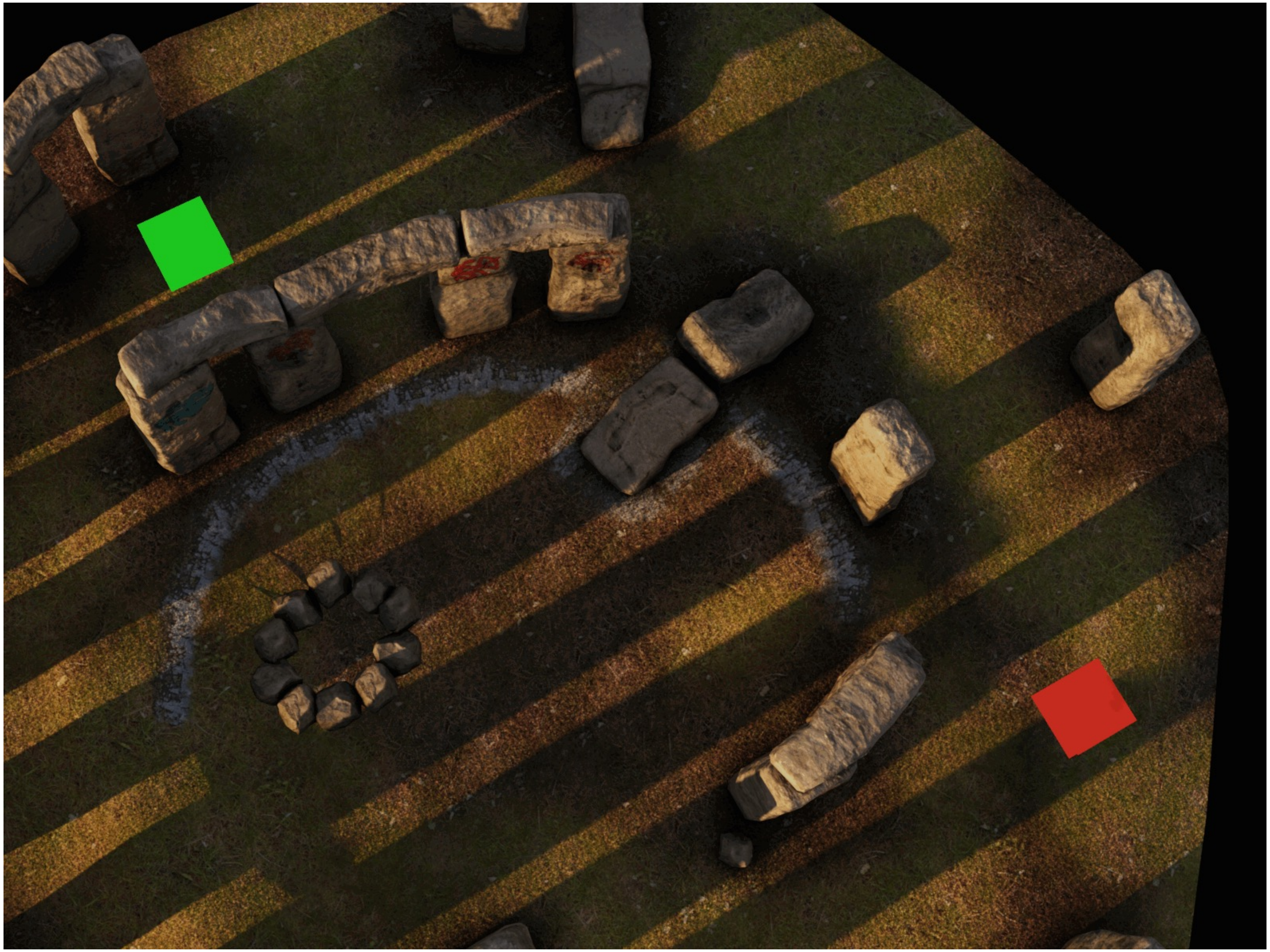
$+ \Sigma L_{controleffort}$

Trajectory Optimization

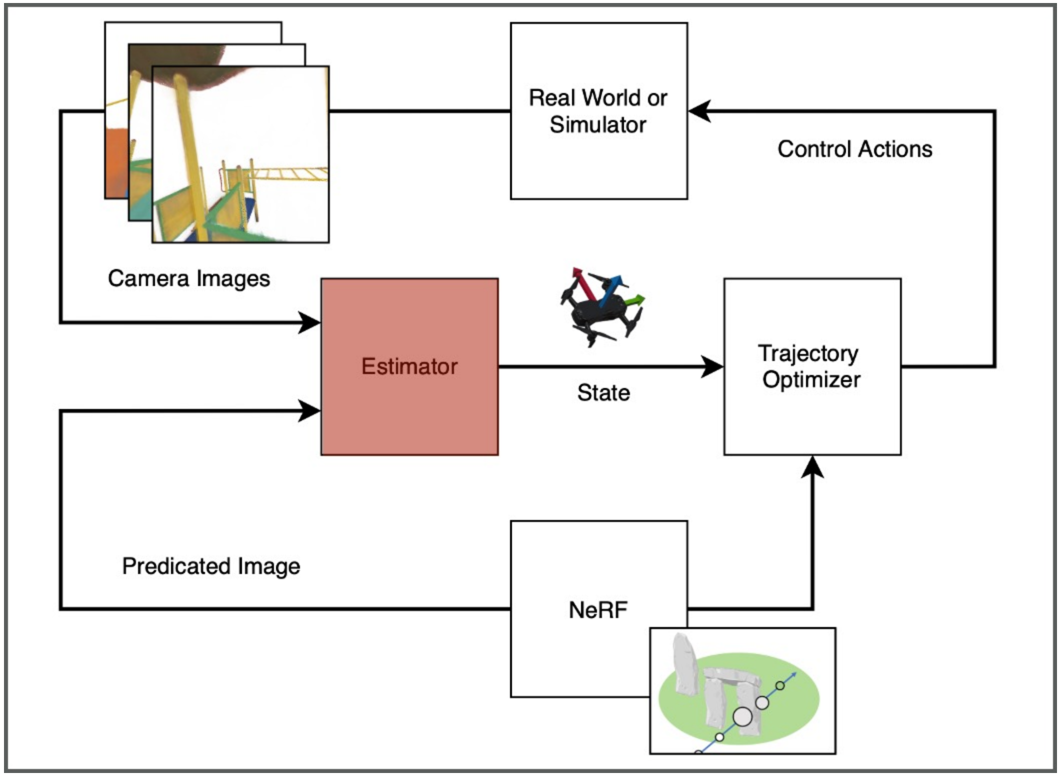


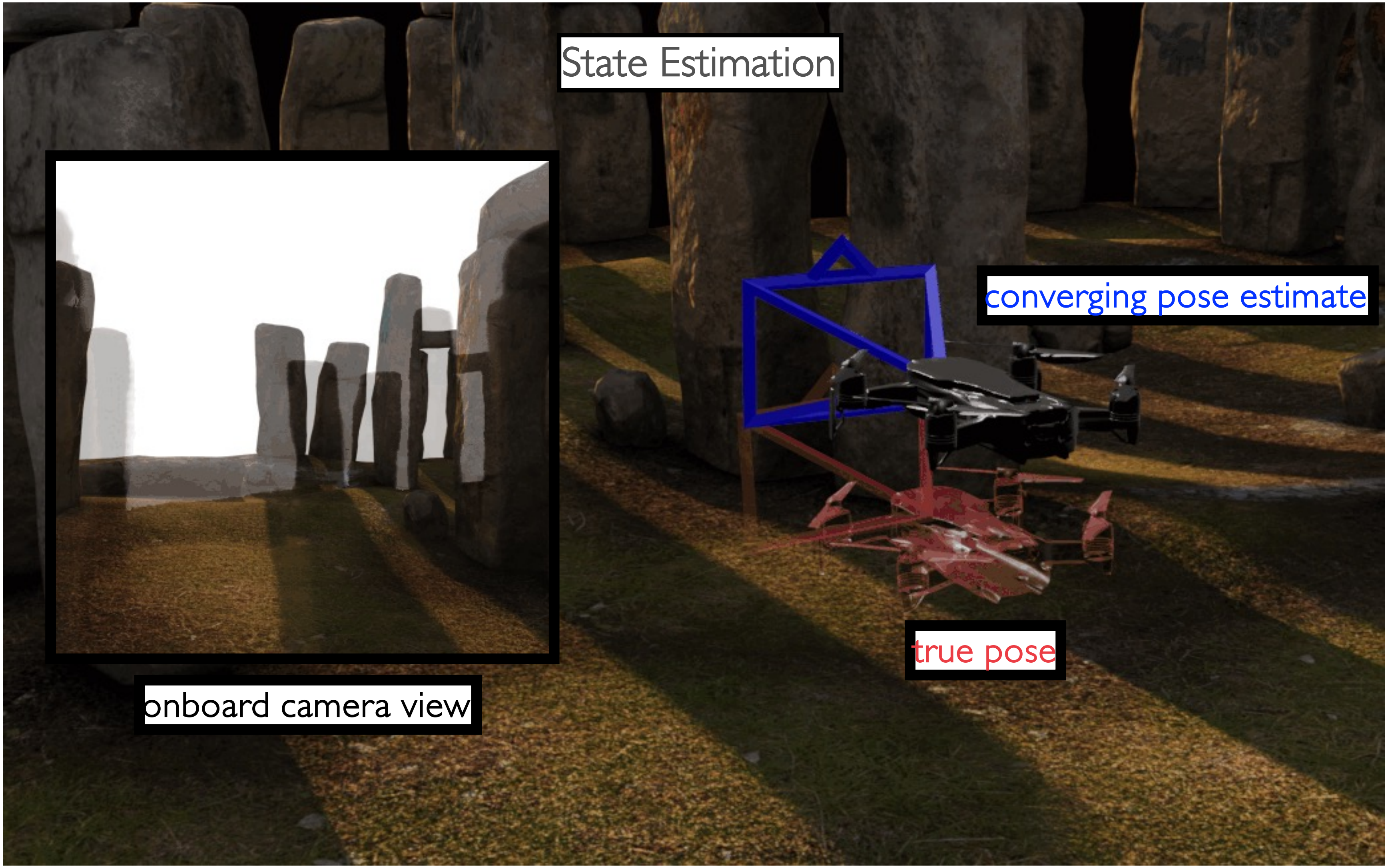
$Loss = \sum_{timestep} \sum_{bodypoint} |v_{i,j}|$  ( $_{i,j}$)

$$+ \sum_{timestep} L_{Controleffort}$$



Vision-Only Navigation

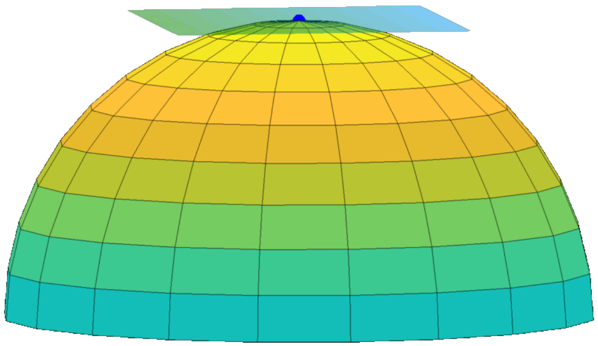



State Estimation
converging pose estimate
true pose
onboard camera view

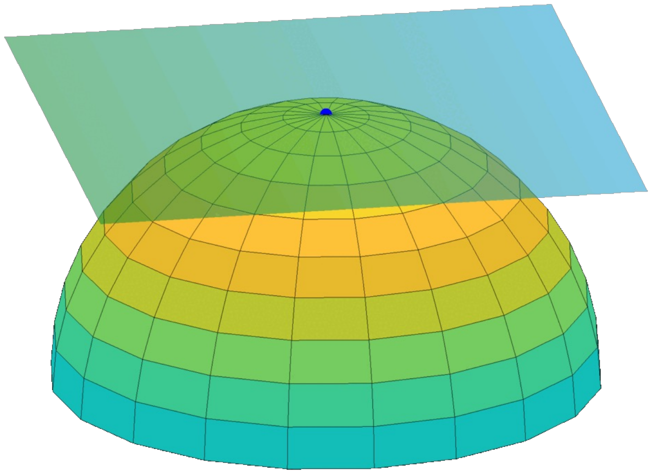
State Estimation

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

Optimization on SE(3)



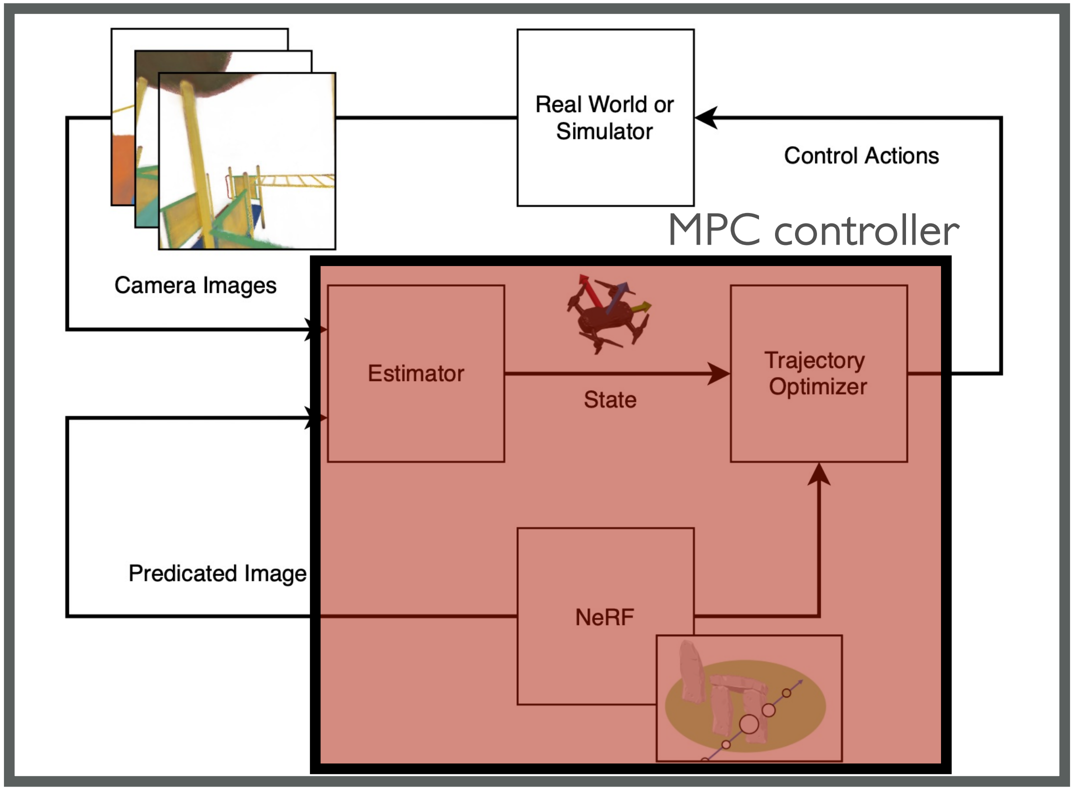
vs



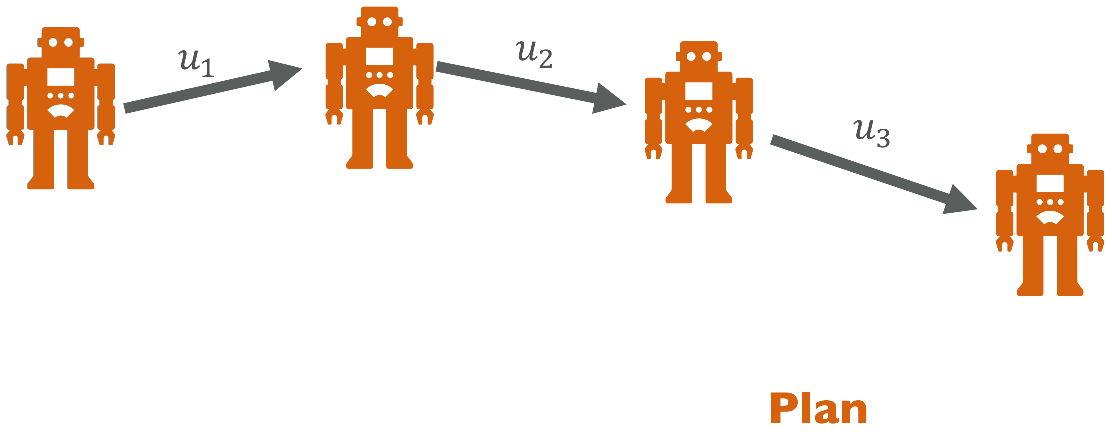
SE(3) Space

SE(3) Tangent Space

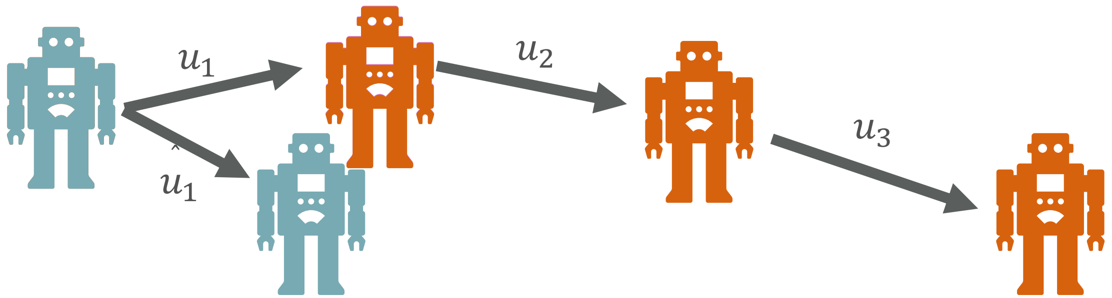
Vision-Only Navigation



MPC Controller



MPC Controller

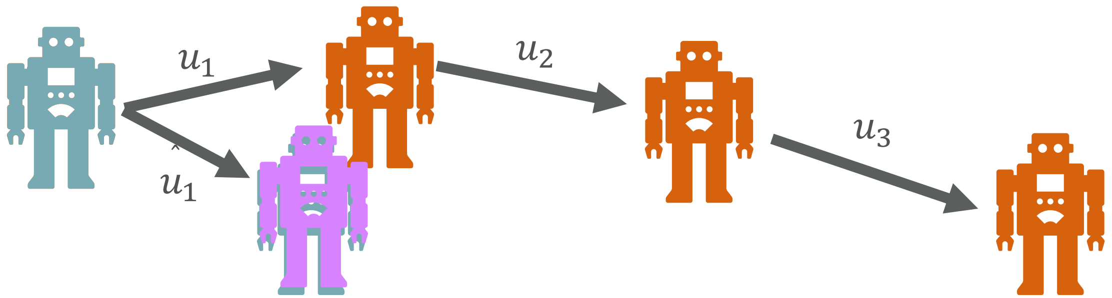


Execute Action

Plan

Ground Truth

MPC Controller



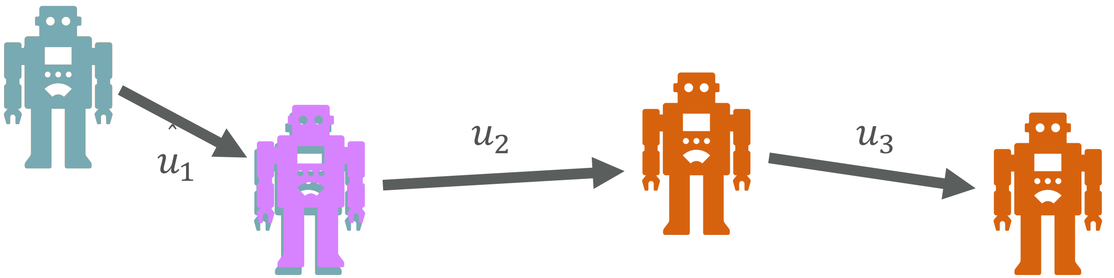
Estimate State

Plan

Ground Truth

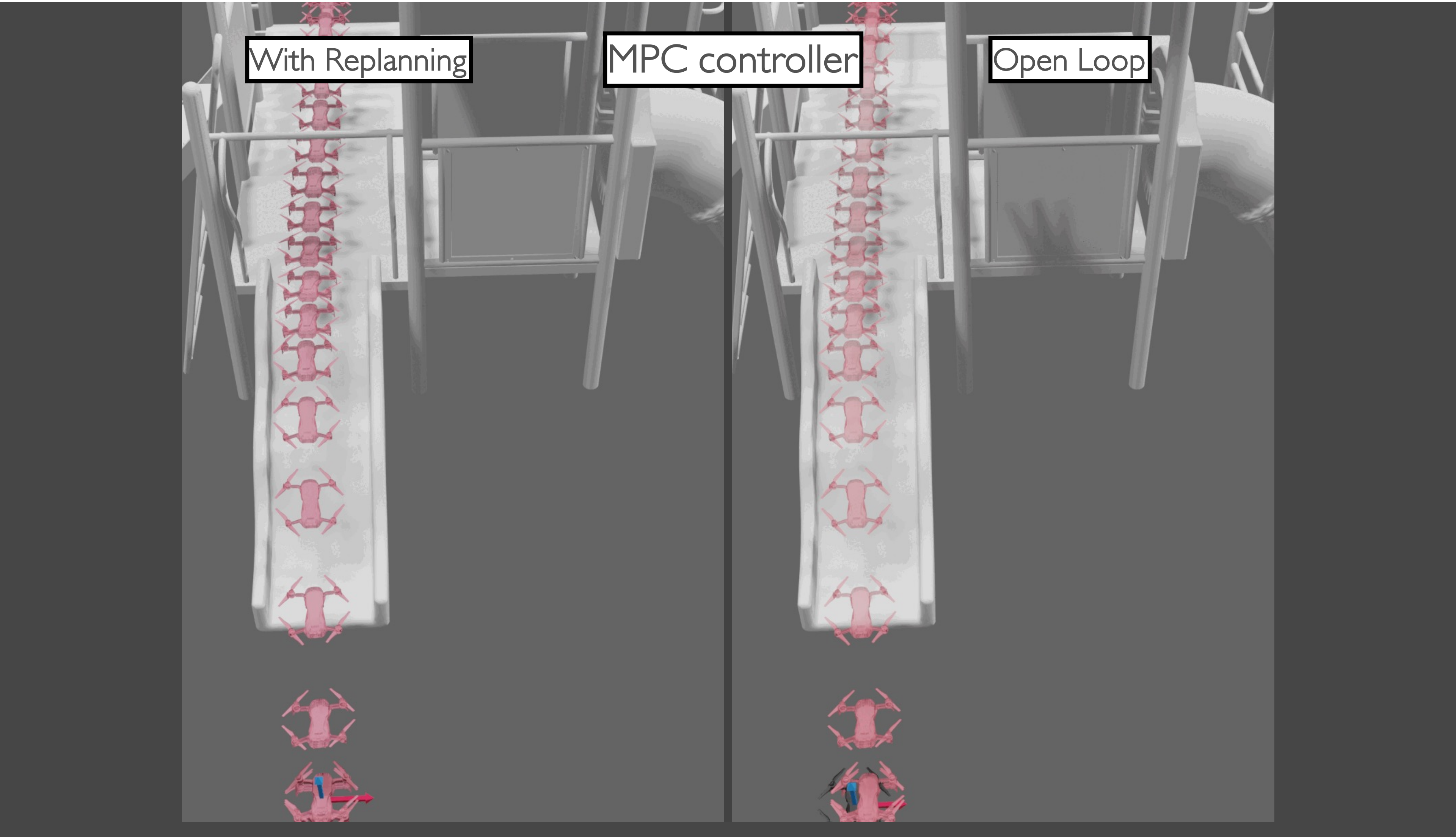
State Estimate

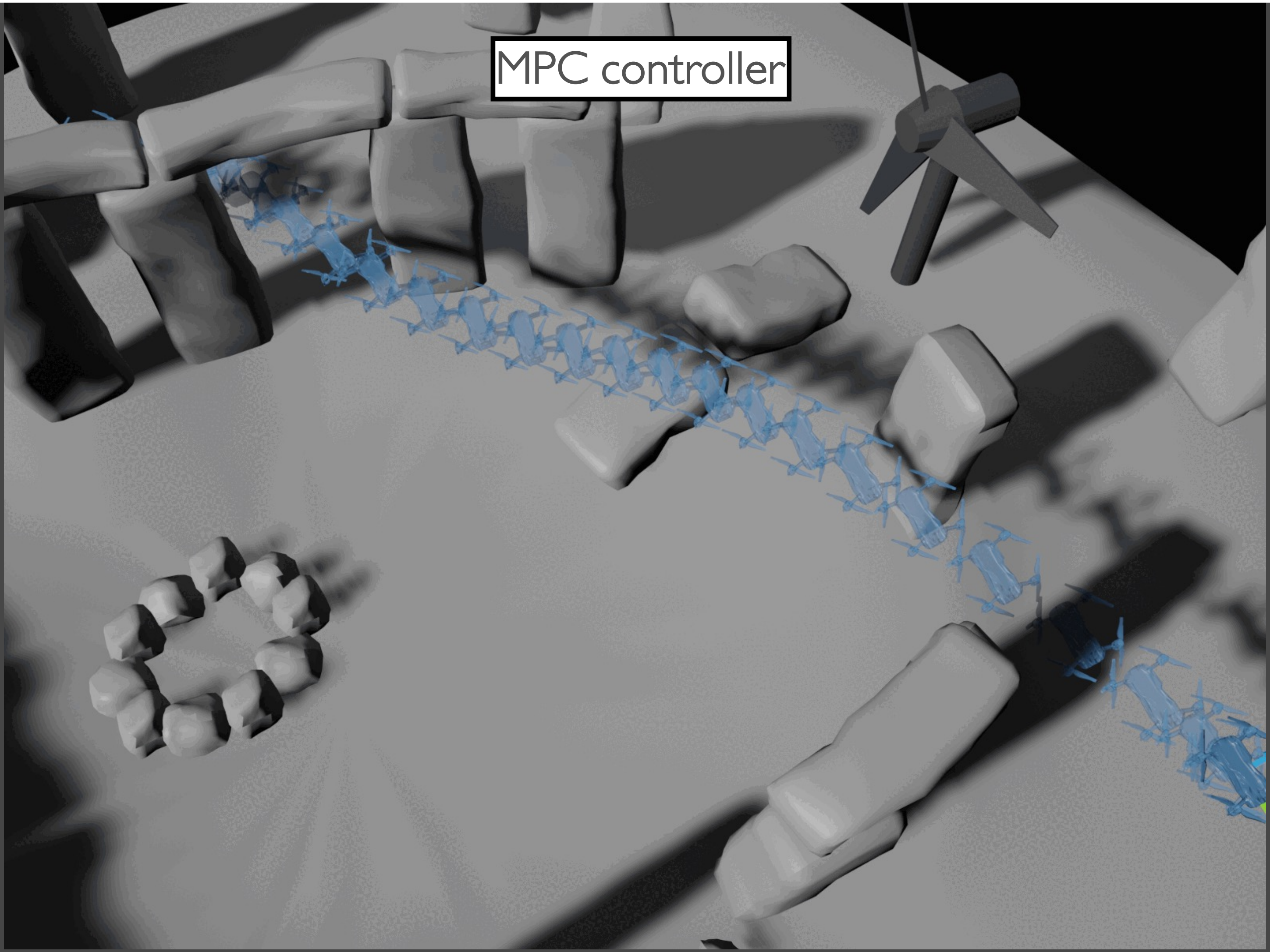
MPC Controller



Replan

Plan
Ground Truth
State Estimate

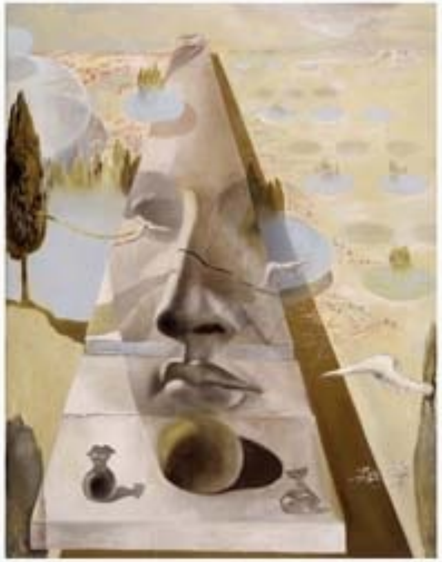

With Replanning
MPC controller
Open Loop


MPC controller

CS231A

Computer Vision:

From 3D Reconstruction to Recognition



Next lecture:

Gaussian Splatting