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)

Ways to Render

Surface rendering

Volume rendering

Camera → Image
Light Source → View Ray
Scene Object

Shadow Ray
Volume rendering equation

\[ I(D) = I_0 T(0) + \int_0^D c(s) \rho(s) T(s) \, ds \]

- **Pixel color**
- **Radiance**
- **Density**
- **Transparency**

\[ T(s) = \exp \left( - \int_s^D \rho(t) \, dt \right) \]
NeRF (neural radiance fields)

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

- \((x, y, z, \theta, \phi) \rightarrow \text{color, opacity}\)
Represent a scene as a continuous 5D function

\[(x, y, z, \theta, \phi) \rightarrow F_{\Omega} \rightarrow (r, g, b, \sigma)\]

Spatial location
Viewing direction

Fully-connected neural network
9 layers, 256 channels

No need to instantiate Volume representation
Generate views with traditional volume rendering

Generate views with traditional volume rendering

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

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

- $t = \text{point along ray}$
- $C = \text{Color of Pixel}$
- $c = \text{color of point}$

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 $\delta t_i$ and volume density $\sigma$

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.

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 $i$:

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$
Optimize with gradient descent on rendering loss

$$\min_{\Omega} \sum_i \| \text{render}^{(i)}(F_\Omega) - I^{(i)}_{gt} \|^2$$

Training network to reproduce all input views of the scene

Results – Synthetic data
Results – Visualization Geometry
PollEverywhere
NeRF for Pose Estimation

iNeRF: Inverting Neural Radiance Fields for Pose Estimation

Lin Yen-Chen\textsuperscript{1,2}, Alberto Rodriguez\textsuperscript{2}
 Petro Florence\textsuperscript{1}, Phillip Isola\textsuperscript{2}, Jonathan T. Barron\textsuperscript{1}, Tsung-Yi Lin\textsuperscript{1}
\textsuperscript{1}Google Research, \textsuperscript{2}Massachusetts Institute of Technology

Observed Image w/ Unknown Pose
Iterative Pose Estimation w/ NeRF Model
Pose Estimation Results:
Overlaid NeRF Rendering and Observed Image
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
iNeRF: Inverting NeRF for Pose Estimation

\[ \hat{T} = \underset{T \in \text{SE}(3)}{\text{argmin}} \mathcal{L}(T \mid I, \Theta) \]
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 $\hat{T}_0$

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

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

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

Magnitude $\theta$

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

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

Final Loss $\hat{S} \theta = \arg\min_{S \theta \in \mathbb{R}^6} \mathcal{L}(e^{[S] \theta T_0} | I, \Theta)$
Sampling Rays

- Random
- Interest Point
- Interest Region

- 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**

<table>
<thead>
<tr>
<th>Method</th>
<th>NeRF</th>
<th>iNeRF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Camera Pose</td>
<td>Image</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Image</td>
<td>Camera Pose</td>
</tr>
</tbody>
</table>
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

\[(x, y, z) = \sigma, c\]
Implicit representations

\[(x, y, z) = \sigma, c\]
Implicit representations

\[(x, y, z) = \sigma, c\]
Implicit representations

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

\[(x, y, z) = \sigma, c\]
Trajectory Planning
Trajectory Planning

\[ \text{Loss} = L_{\text{collision}} + L_{\text{control effort}} \]
Trajectory Planning

\[ \text{Loss} = L_{\text{collision}} + L_{\text{control effort}}(u_1) + L_{\text{control effort}}(u_2) + \ldots \]
Trajectory Planning

\[ \text{Loss} = (\text{term 1}) + (\text{term 2}) + (\text{term 3}) + \ldots + \sum L_{\text{control effort}} \]
Trajectory Planning

\[
\text{Loss} = |\vec{v}_1| + |\vec{v}_2| + |\vec{v}_3| + \ldots + \Sigma L_{\text{control effort}}
\]
Loss =
\[ |\vec{v}_1| + |\vec{v}_2| + |\vec{v}_3| + \ldots + \sum L_{control\_effort} \]
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

\[ \mu_{t|t-1} = f(\mu_{t-1}, u_t) \]

\[ A_{t-1} = \left. \frac{\partial f(x, u_t)}{\partial x} \right|_{x=\mu_{t-1}} \]

\[ \Sigma_{t|t-1} = A_{t-1} \Sigma_{t-1} A_{t-1}^T + Q_{t-1} \]

Predict Step
State Estimation

\[ Loss = |\text{Camera} - NeRF(\hat{x})|^2 + Loss_{\text{Process}}(\hat{x}) \]
State Estimation

\[ \text{Loss} = |\mathbf{O} - NeRF(\hat{x})|^2 + \text{Loss}_{\text{Process}}(\hat{x}, \Sigma_{t|t-1}) \]

Minimizing this gives posterior state estimate

\[ \Sigma_t = \left( \frac{\partial^2 \text{Loss}(x)}{\partial x^2} \right)_{x=x_{opt}}^{-1} \]
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