6DoF Pose Estimation from Mesh using Differentiable Rendering

Meet Gera   Abhinav Raundhal   Amey Karan

International Institute of Information Technology, Hyderabad, India

December 7, 2024

Outline

Introduction

- **Pose Estimation**:
  - Determine position and orientation (6DoF) of objects.
  - Crucial in robotics, augmented reality, computer vision.
- **Objective**:
  - Optimize pose parameters using **Differentiable Rendering**.
  - Compare different rotation representations:
    - Axis-Angle
    - Quaternions
    - Rotation Matrices
- **Approach**:
  - Use silhouette loss between rendered and target images.
  - Employ gradient-based optimization.

Differentiable Rendering

- **Definition**:
  - Rendering process where gradients can be computed w.r.t input parameters.
- **Advantages**:
  - Enables optimization of 3D scene parameters.
  - Utilize standard backpropagation for parameter updates.
- **Applications**:
  - Pose estimation
  - Shape optimization
  - Material property estimation

Problem Formulation

- **Given**:
  - 3D mesh $\mathcal{M}$.
  - Target image $\mathcal{I}_{\text{GT}}$.
- **Objective**:
  - Estimate pose $\mathbf{T} = (\mathbf{R}, \mathbf{t})$ to align $\mathcal{M}$ with $\mathcal{I}_{\text{GT}}$.
- **Rendering Function**:

$$\mathcal{I}_{\text{rendered}} = \mathcal{R}(\mathcal{M}, \mathbf{T})$$

- **Optimization Problem**:

$$\min_{\mathbf{T}} \; \mathcal{L}(\mathcal{I}_{\text{rendered}}, \mathcal{I}_{\text{GT}})$$

Silhouette Loss Function

- **Loss Function**:

$$\mathcal{L} = \sum_{i,j} \left(S_{i,j}^{\text{rendered}} - S_{i,j}^{\text{GT}} \right)^2$$

- **Where**:
  - $S_{i,j}^{\text{rendered}}$: Pixel value at (i, j) in rendered silhouette.
  - $S_{i,j}^{\text{GT}}$: Pixel value at (i, j) in ground truth silhouette.
- **Purpose**:
  - Measures alignment between rendered and target silhouettes.

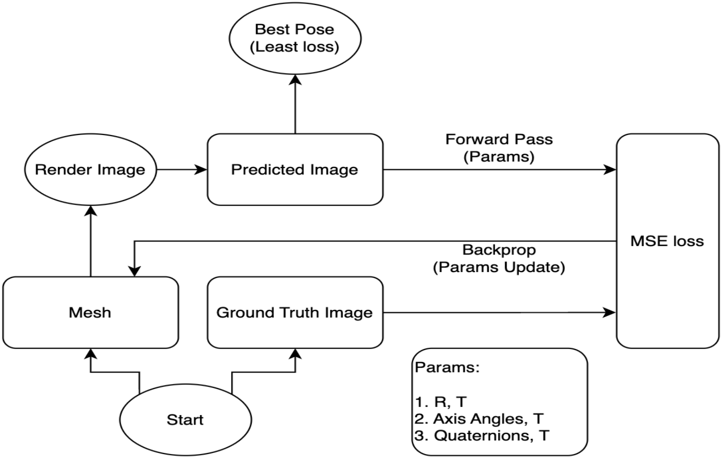
Gradient Computation

- **Goal**:
  - Compute gradient of loss w.r.t pose parameters.
- **Gradient of Loss**:

$$\frac{\partial \mathcal{L}}{\partial q_i} = \frac{\partial \mathcal{L}}{\partial I_p} \cdot \frac{\partial I_p}{\partial R} \cdot \frac{\partial R}{\partial q_i}$$

- **Where**:
  - q_i: Pose parameter (e.g., quaternion component).
  - I_p: Projected image.
  - R: Rotation matrix.

Optimization Process



Optimization Process

1. **Initialization**:
   - Set initial estimates for $\mathbf{R}$ and $\mathbf{t}$.
   - Choose rotation representation (Axis angle, rotation matrix, quaternion).
2. **Forward Pass**:
   - Render image I_p using current parameters.
   - Compute loss $\mathcal{L}$.
3. **Backward Pass**:
   - Compute gradients w.r.t parameters.
4. **Parameter Update**:
   - Update parameters using optimizer (e.g., Adam).
   - Apply constraints (normalization, orthogonality).
5. **Iterate until Convergence**.

Parameter Constraints

- **Quaternion Normalization**:

$$\boldsymbol{q} \leftarrow \frac{\boldsymbol{q}}{\|\boldsymbol{q}\| + \epsilon}$$

- **Rotation Matrix**: Should remain orthonormal at each step.
- **Purpose**:
  - Ensure parameters represent valid rotations.
  - Maintain numerical stability.

Avoiding Local Minima

- **Problem**:
  - Optimization may get stuck in local minima.
- **Solution**:
  - Introduce perturbations when progress stalls.
  - **Methods**:
    - **Axis-Angle**:
    - Add small random vector to axis-angle parameters.
    - **Quaternions**:
    - Perturb and re-normalize quaternion.
    - **Rotation Matrices**:
    - Convert to axis-angle, perturb, convert back.

Results and Observations

Results can be viewed here: Github

Results and Observations

- **Loss Convergence**:
  - All models showed convergence.
  - Rotation matrices had more fluctuations.
- **Parameter Updates**:
  - Translation and rotation parameters converged exponentially.
  - Perturbations caused spikes in updates.
- **Comparison**:
  - Axis-Angle and Quaternions converged faster.
  - Fewer parameters lead to quicker convergence.

Interpretation of Results

- **Number of Parameters**:
  - Axis-Angle: 3+3 parameters.
  - Quaternions: 4+3 parameters (normalized).
  - Rotation Matrices: 9+3 parameters (with constraints).
- **Impact on Optimization**:
  - Higher dimensionality increases computational load.
  - Constraints add complexity.
- **Recommendation**:
  - Use Axis-Angle or Quaternions for efficiency.

Conclusion

- **Differentiable Rendering**:
  - Effective for refining pose estimation.
  - Enables gradient-based optimization of pose parameters.
- **Rotation Representations**:
  - Choice impacts convergence and stability.
  - Axis-Angle and Quaternions preferred for efficiency.
- **Future Work**:
  - Explore hybrid approaches.
  - Investigate other loss functions and regularizations.

Contribution of Team Members

- **Meet Gera**:
- Implemented Differential Rendering Algorithm.
- Derived mathematical expressions for backpropagation.
- Contributed to report and documentation.
- **Abhinav Raundhal**:
- Implemented Differential Rendering Algorithm.
- Conducted experiments on different objects.
- Contributed to report and documentation.
- **Amey Karan**:
- Set up environment and dependencies.
- Implemented Differential Rendering Algorithm.
- Contributed to report and documentation.

References I

I. Shugurov, I. Pavlov, S. Zakharov, and S. Ilic, "Multi-view object pose refinement with differentiable renderer," *IEEE Robotics and Automation Letters*, vol. 6, pp. 2579–2586, Apr. 2021.

A. Palazzi, L. Bergamini, S. Calderara, and R. Cucchiara, "End-to-end 6-dof object pose estimation through differentiable rasterization," in *Computer Vision – ECCV 2018 Workshops* (L. Leal-Taixé and S. Roth, eds.), (Cham), pp. 702–715, Springer International Publishing, 2019.

N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, N. Neverova, G. Larsson, R. Girdhar, M. Singh, S. Somasundaram, A. Makadia, and M. Rohrbach, "Pytorch3d: An optimized library for 3d deep learning." `https://github.com/facebookresearch/pytorch3d`, 2020.

GitHub repository.