6DoF pose Estimation from Mesh using Differentiable rendering

Meet Gera (2022102039)*, Abhinav Raundhal (2022101089)†, and Amey Karan (2022111026)‡
meet.gera@students.iiit.ac.in, abhinav.raundhal@students.iiit.ac.in, amey.karan@research.iiit.ac.in

International Institute of Information Technology, Hyderabad, India
*Robotics Research Centre †Centre for Visual Information Technology ‡Software Engineering Research Centre

Github

***Abstract*—In 3D computer vision and graphics, pose estimation is critical for tasks such as object rendering, alignment, and scene understanding. This report investigates pose optimization using different rotation representations: axis-angle, quaternions, and direct rotation matrices. We implement models utilizing these representations and analyze their performance in optimizing object poses by minimizing silhouette loss between rendered images and ground truth silhouettes.**

***Index Terms*—Pose optimization, Axis-angle, Quaternions, Rotation matrices, 3D rendering, PyTorch3D.**

I. Introduction

Pose estimation involves determining the position and orientation of an object in space. Accurately optimizing pose parameters is vital in applications like augmented reality, robotics, and computer graphics. Different mathematical representations for rotations exist, including axis-angle, quaternions, and rotation matrices. Each has its advantages and challenges in terms of numerical stability, singularities, and computational efficiency.

This report explores the implementation of pose optimization models using these rotation representations. We employ PyTorch and PyTorch3D to render 3D meshes and optimize pose parameters by minimizing the difference between rendered silhouettes and ground truth images.

II. Background

A. Rotation Representations

1) Axis-Angle: The axis-angle representation defines a rotation by an axis vector $\boldsymbol{\omega}$ and an angle θ, representing a rotation of θ radians about $\boldsymbol{\omega}$.

2) Quaternions: Quaternions extend complex numbers and represent rotations without suffering from gimbal lock. A unit quaternion consists of a scalar part w and a vector part $\boldsymbol{v} = (x, y, z)$.

3) Rotation Matrices: Rotation matrices are 3×3 orthogonal matrices with determinant 1. They directly represent rotations but require constraints to maintain orthogonality during optimization.

III. Problem Statement

This project aims to address the challenge of pose estimation by breaking it down into two stages:

1. Multiview Object Reconstruction: Given a set of multiview images of an object, the first stage involves constructing a 3D representation of the object in the form of a Truncated Signed Distance Field (TSDF). The TSDF provides a volumetric representation of the object. This enables the application of the marching cubes algorithm to extract a high-resolution mesh from the TSDF. This mesh captures the object's detailed shape and structure, which serves as the input for pose estimation.

2. 6-DoF Pose Estimation via Differentiable Rendering: The second stage, which is the primary focus of this work, leverages differentiable rendering techniques to accurately estimate the object's 6-DoF pose from the generated mesh. Differentiable rendering using gradient based optimization allows us to optimize pose parameters by creating synthetic images of the mesh and comparing them with real images. This approach enables us to adjust the object's position and orientation iteratively for a precise match. The pose is defined by six parameters: three for position (x, y, z) and three for rotation (roll, pitch, yaw).

This project's objective is to explore and implement an effective differentiable rendering framework, accurate 6-DoF pose estimation, providing a scalable solution for various mobile robotics applications.

IV. Differentiable Rendering for Pose Estimation

Differential Rendering allows to use Image based Loss function for optimizing a complex 3D scene to match with the input image. Using Differential Chain equations to calculate optimal position of vertices in scene, the material parameters like specularity, roughness etc., lighting parameters like intensity and source location, and also camera parameters like location, focal length etc. We Can fix the some parameters depending on our setup like light source or focal length.

A. Problem Formulation

Given an object with known geometry represented as a 3D mesh $\mathcal{M}$, the goal of pose estimation is to determine the object's 3D pose $\mathbf{T} = (\mathbf{R}, \mathbf{t})$ where $\mathbf{R} \in SO(3)$ is the rotation matrix, and $\mathbf{t} \in \mathbb{R}^3$ is the translation vector. The object's appearance from a camera's viewpoint, captured in an image $\mathcal{I}$, is defined by the rendering function $\mathcal{R}(\mathcal{M}, \mathbf{T})$:

$$\mathcal{I} = \mathcal{R}(\mathcal{M}, \mathbf{T}).$$

The pose estimation problem can be formulated as minimizing the photometric error between the rendered image $\mathcal{I}$ and a target image $\mathcal{I}_t$:

$$L = \min_{\mathbf{T}} \|\mathcal{R}(\mathcal{M}, \mathbf{T}) - \mathcal{I}_t\|^2.$$

In this equation only camara relative position is optimised.

B. Differentiable Rendering Pipeline

To enable gradient-based optimization, the rendering process must be differentiable with respect to the pose parameters. Differentiable rendering techniques compute partial derivatives of the image with respect to the pose parameters $\frac{\partial \mathcal{I}}{\partial \mathbf{T}}$, which requires differentiating through several steps, including projection, shading, and rasterization.

a) Projection and Transformation: Using perspective projection, the transformation of a point $\mathbf{p} = (x, y, z)^T$ on the object in 3D space to image space (u, v) can be written as:

$$\mathbf{p}_{\text{proj}} = \mathbf{K} \cdot (\mathbf{R} \cdot \mathbf{p} + \mathbf{t}),$$

where $\mathbf{K}$ is the intrinsic camera matrix. The gradients $\frac{\partial \mathbf{p}_{\text{proj}}}{\partial \mathbf{R}}$ and $\frac{\partial \mathbf{p}_{\text{proj}}}{\partial \mathbf{t}}$ are computed to adjust pose during optimization.

C. Pose Optimization with Gradient Descent

1) **Forward Pass:**
   - Define initial parameters for geometry, material, lighting, and camera.
   - Render an image I_{pred} from the 3D scene with the current parameters.
   - Compute a loss $L = \|I_{\text{pred}} - I_{\text{target}}\|^2$ by comparing with the target image.
2) **Backward Pass:**
   - Use automatic differentiation to compute the gradient of the loss L with respect to all parameters.
   - Update each parameter by taking a small step in the direction that reduces the loss:
     - Camera: $c \leftarrow c - \alpha \frac{\partial L}{\partial c}$

This is optimisation for all possible parameters such that 2D projection of mesh from a particular viewpoint has minimum Loss function value with given input image.

$$\mathbf{T}_{k+1} = \mathbf{T}_k - \eta \frac{\partial \|\mathcal{R}(\mathcal{M}, \mathbf{T}_k) - \mathcal{I}_t\|^2}{\partial \mathbf{T}},$$

where η is the learning rate.

V. Related Work

The idea for this project was inspired from this paper: [1]"End-to-end 6-DoF Object Pose Estimation through Differentiable Rasterization" which introduces a novel approach for refining 6-DoF object pose estimation using a differentiable renderer. The method involves a two-branch convolutional encoder network to predict both object class and pose. A differentiable raster-based renderer re-projects 3D objects into 2D silhouettes based on the predicted pose, enabling alignment error measurement. This error is back-propagated to refine pose predictions during test time. Unlike prior voxel-based or ray-tracing methods, the approach uses lightweight mesh representations for efficiency. Experimental results demonstrate significant improvements in pose accuracy and alignment, even with surrogate 3D models, showcasing the method's robustness. [2] Here, we do not have object classification as a part of the pipeline, but we assume that we have been given the image and the corresponding mesh to estimate the pose.

VI. Methodology

A. Silhouette Loss Function

The loss function used for optimization is the mean squared error between the rendered silhouette and the ground truth silhouette:

$$\mathcal{L} = \sum_{i,j} (S_{i,j}^{\text{rendered}} - S_{i,j}^{\text{gt}})^2 \quad (1)$$

where $S_{i,j}$ denotes the pixel value at position (i, j).

B. Models

1) Axis-Angle Model: The axis-angle model parameterizes rotations using an axis-angle vector $\boldsymbol{\alpha}$. The rotation matrix R is obtained via:

$$R = \text{axis\_angle\_to\_matrix}(\boldsymbol{\alpha}) \quad (2)$$

2) Quaternion Model: The quaternion model uses a quaternion $\boldsymbol{q} = (w, x, y, z)$ for rotation. To ensure valid rotations, the quaternion is normalized:

$$\boldsymbol{q}_{\text{normalized}} = \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|} \quad (3)$$

The rotation matrix is then derived from the normalized quaternion.

3) Rotation-Translation Model: This model directly optimizes the rotation matrix R and the translation vector T. To maintain R as a valid rotation matrix during optimization, we apply orthogonalization techniques.

VII. Implementation Details

A. Optimization Process

For each model, we optimize the pose parameters using the Adam optimizer with appropriate learning rates. During each iteration:

1) Compute the rendered image using the current pose parameters. [3]

2) Calculate the silhouette loss.
3) Backpropagate to compute gradients.
4) Update the parameters using the optimizer.
5) Apply necessary constraints (e.g., normalization).

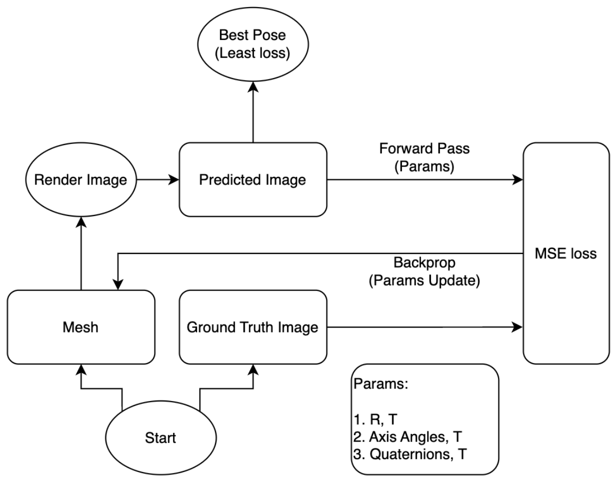

Fig. 1. Differentiable Rendering Pipeline

B. Parameter Constraints

1) Quaternion Normalization: After each optimizer step, we normalize the quaternion to maintain it as a unit quaternion:

$$\boldsymbol{q} \leftarrow \frac{\boldsymbol{q}}{\|\boldsymbol{q}\| + \epsilon} \tag{4}$$

where ϵ is a small constant for numerical stability.

C. Avoiding Local Minima

Since we use differential methods to find minima, we may eventually reach a local minimum. To overcome this, we perturb the parameters as soon as we recognize a local minimum through a differential threshold.

1) Quaternion: After each perturbation, we normalize the quaternion vector to maintain a valid rotation matrix representation.

2) Axis-Angle: Perturbations do not need special handling here.

3) Rotation Matrix: Rotation matrices are converted to axis-angle representation, where the perturbation is applied. They are then converted back to rotation matrices. As Direct perturbation will lead to leaving SO(3) space.

VIII. Results

A. Plots

Results for different object meshes and different ground truth poses have been obtained. The results are quite dependent on the initializations. We have summarized the results for a constant initial guess across the 3 different representations to compare their performance.

1) The initial guess and ground truth is plotted.
2) The loss across the epochs has been plotted.
3) The final best fitting pose and the ground has been overlaid and compared.
4) The updates in Translation components has been shown over epochs
5) The updates in Rotation components has been shown over epochs

B. Inferences

1) **Loss Convergence**
All models showed convergence in the loss function over iterations, with minor differences in the convergence rates due to the nature of the rotation representations. A notable observation here is that the loss in case of Rotation Matrix fluctuates more as it decays. This is due to the requirement of more number of parameters (9+3) to represent a rotation matrix as compared to other approaches (3+3) in axis angles and (4+3) in quaternions.
2) **Update of Translation and Rotation components**
The Translation and Rotation components converge to the Ground Truth values exponentially over a number of epochs. The spikes in these graphs result due to the perturbations introduced to avoid getting stuck at local minima.
3) **Comparison between approaches** The Translation and Rotation components using Axis Angle representation and Quaternion is much closer to the ground truth as compared to rotation matrix (with same number of epochs). This is clearly due to the lesser number of parameters that need to be optimized. This can be used to conclude that the number of iterations required to converge is directly proportional to the number of parameters.

Results using Axis Angles

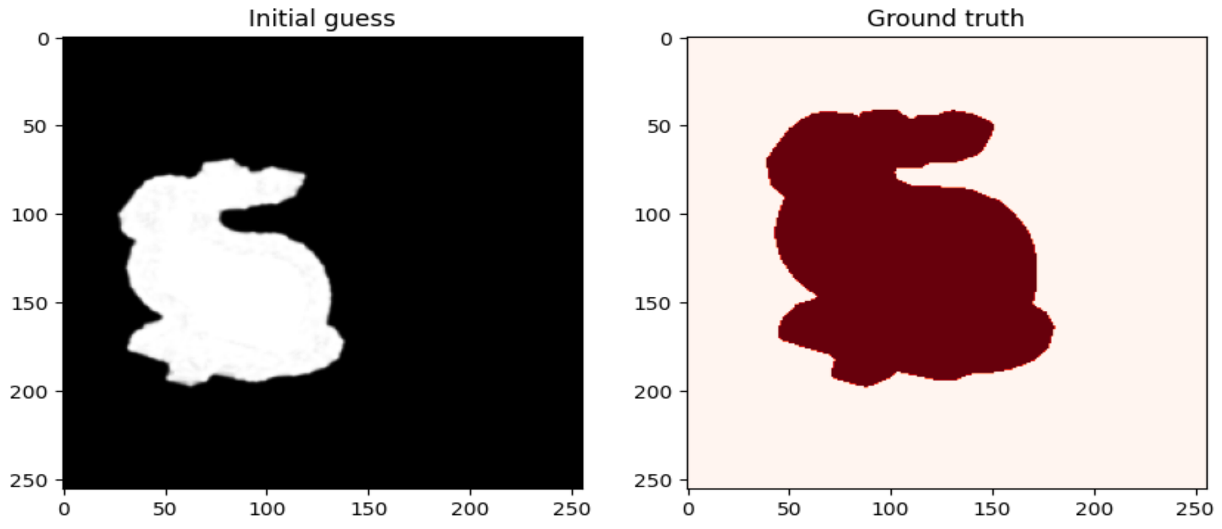

Fig. 2. The Initial Guess(left) and Ground Truth(right) Images

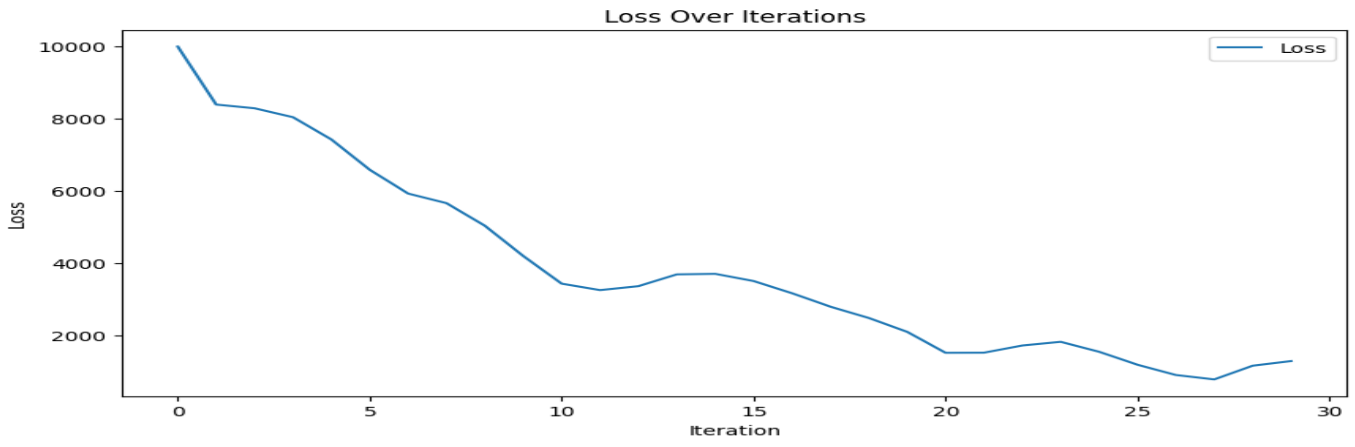

Fig. 3. MSE loss using Axis Angles over epochs

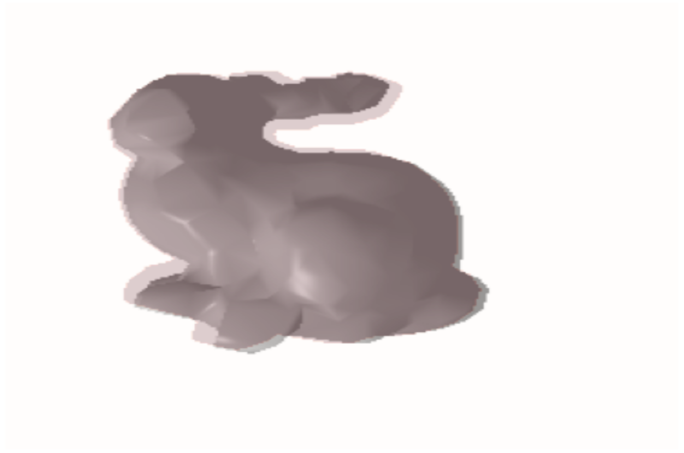

Fig. 4. Best predicted(dark) overlaid on ground truth(light) Image

Results using Axis Angles

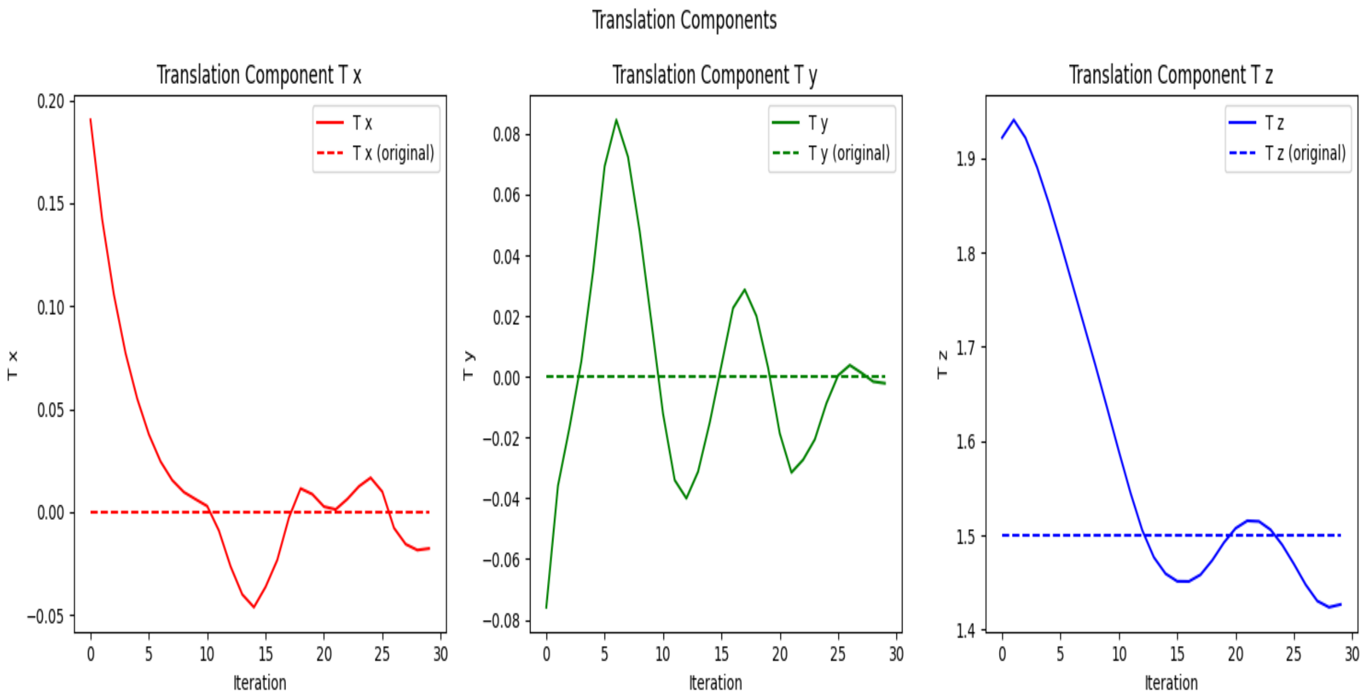

Fig. 5. Update in Translation components over epochs

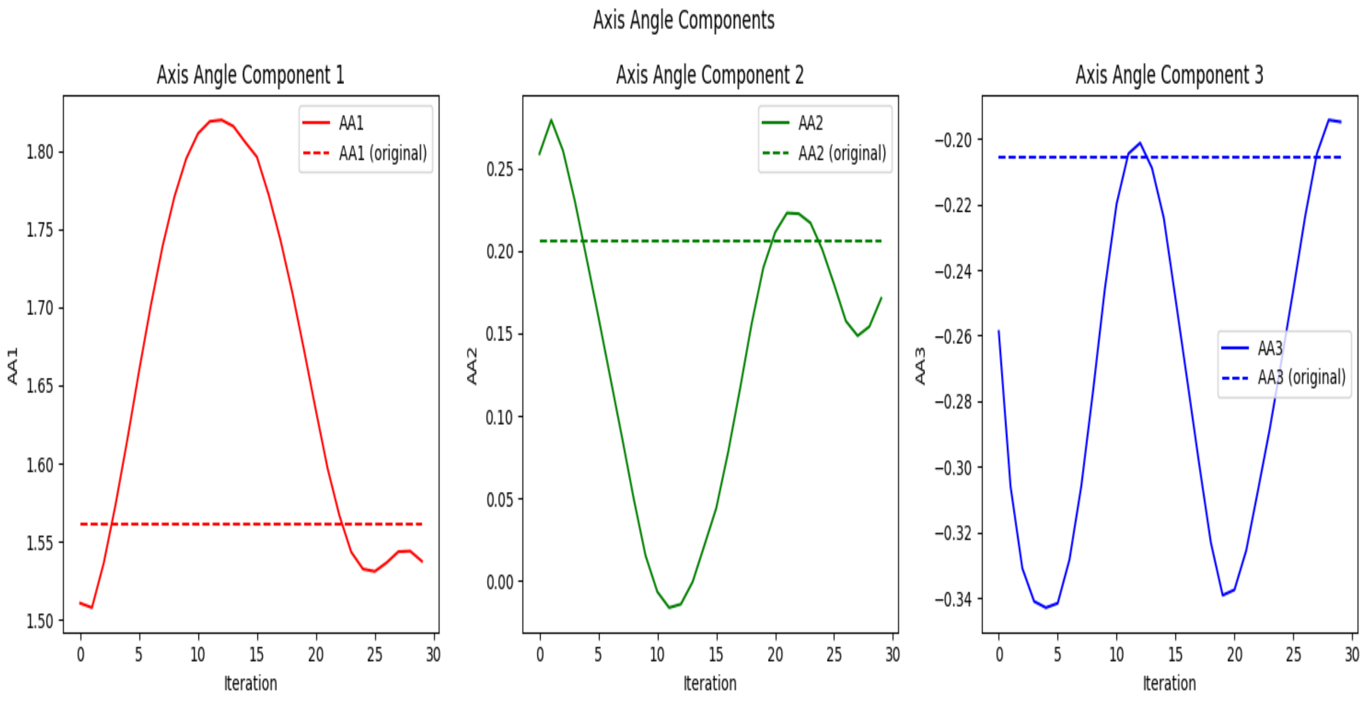

Fig. 6. Update in Rotation components (axis angles) over epochs

Results using Rotation Matrix

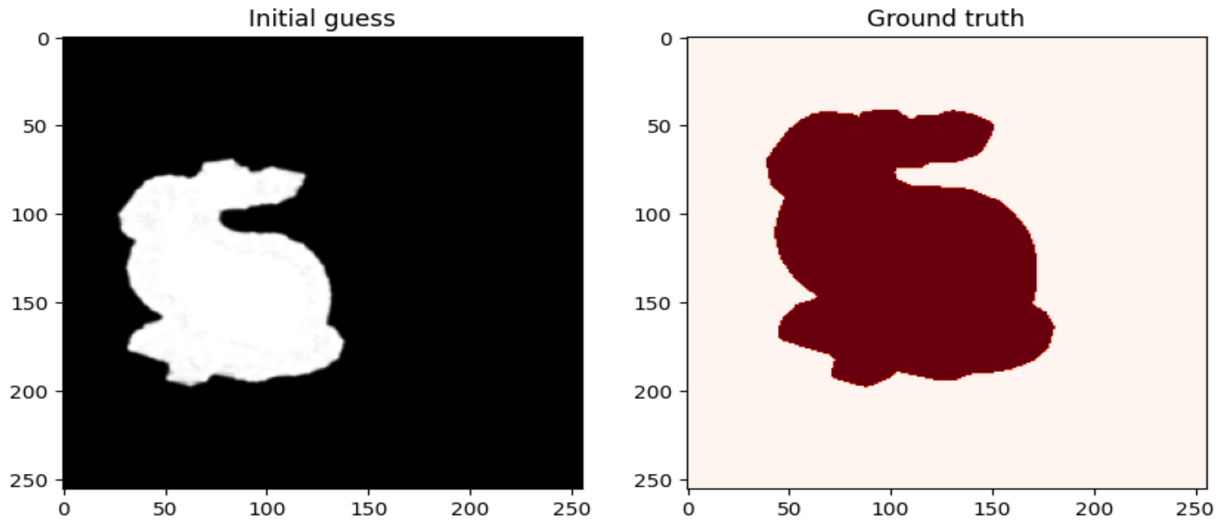

Fig. 7. The Initial Guess(left) and Ground Truth(right) Images

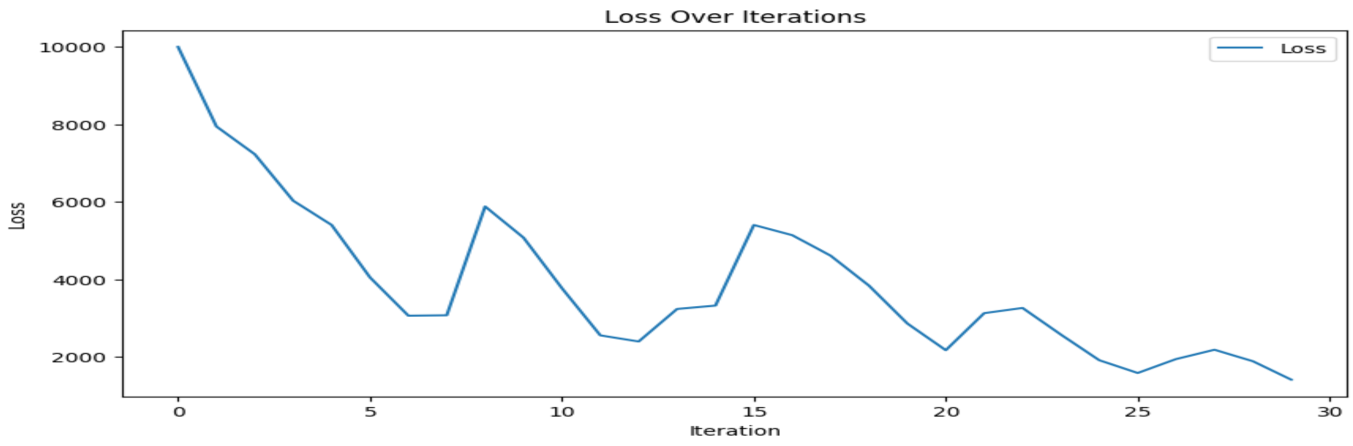

Fig. 8. MSE loss using Rotation matrix over epochs

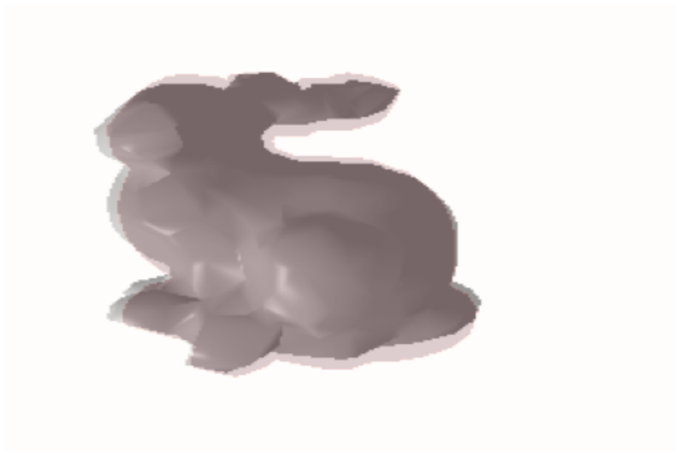

Fig. 9. Best predicted(dark) overlaid on ground truth(light) Image

Results using Rotation Matrix

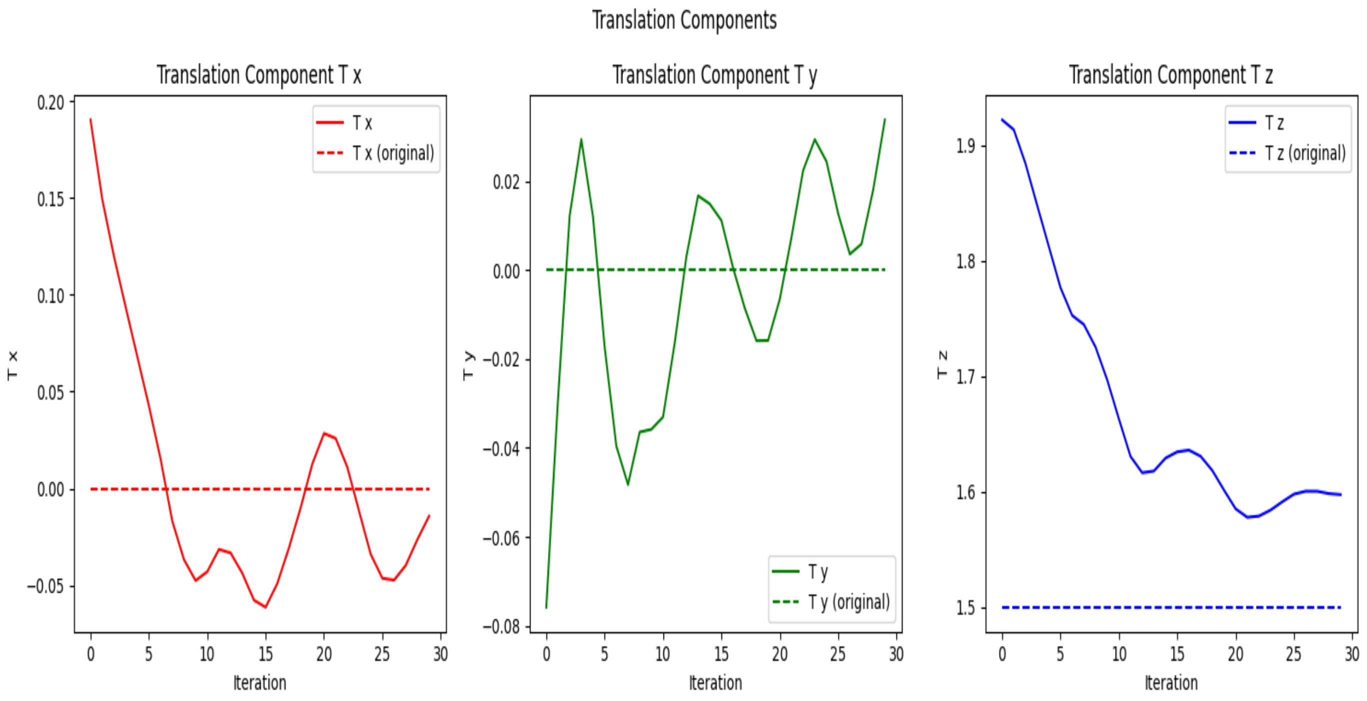

Fig. 10. Update in Translation components over epochs

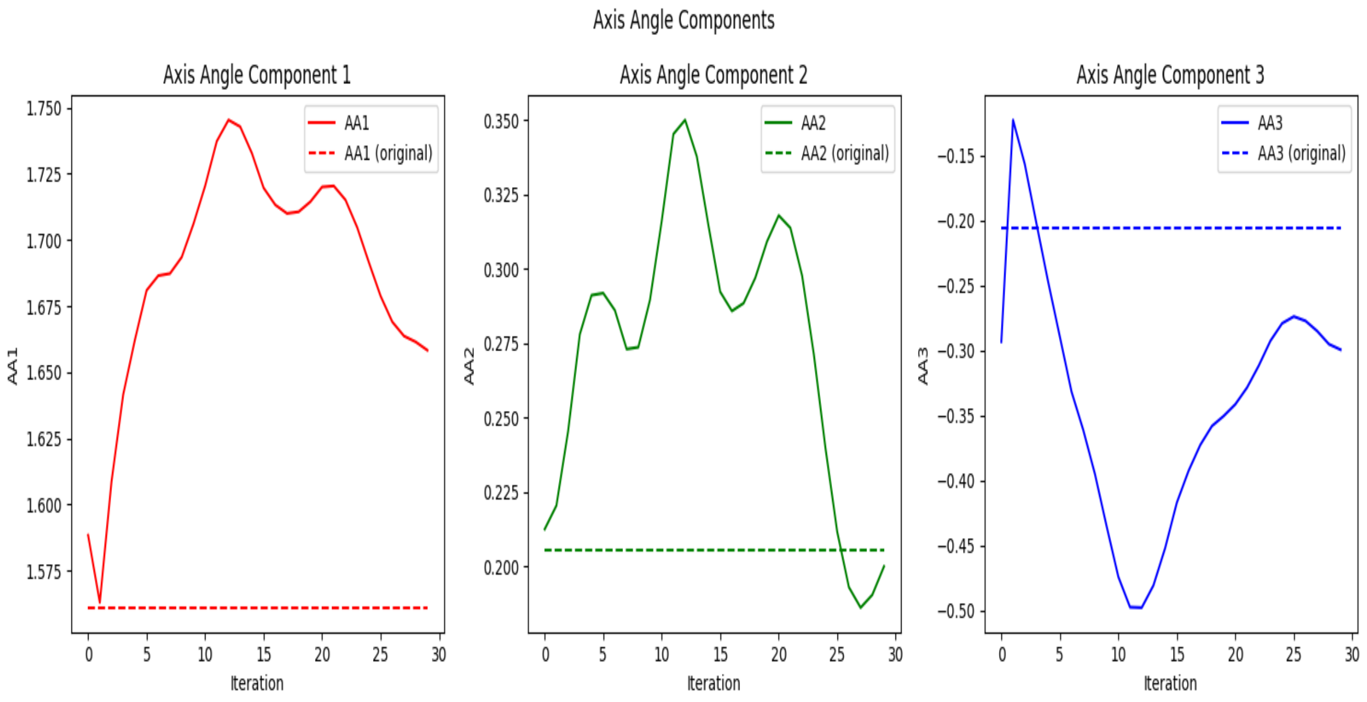

Fig. 11. Update in Rotation components (axis angles) over epochs

Results using Quaternions

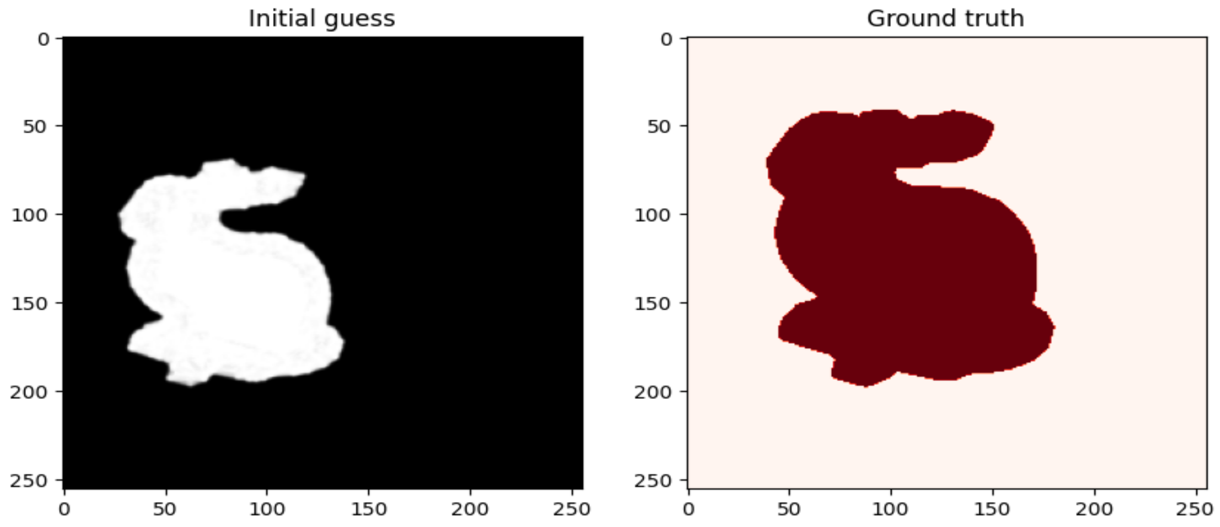

Fig. 12. The Initial Guess(left) and Ground Truth(right) Images

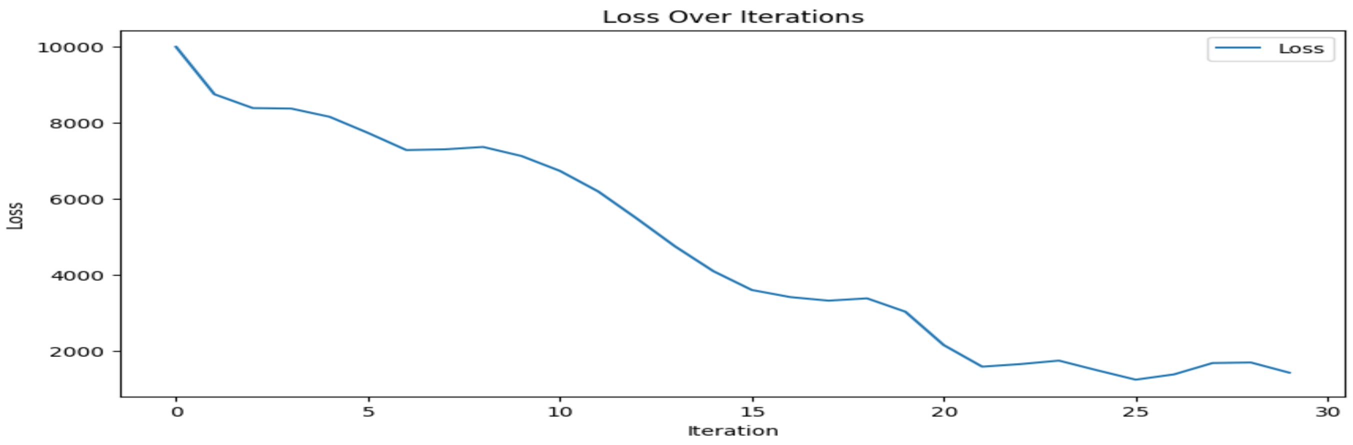

Fig. 13. MSE loss using Quaternions over epochs

Fig. 14. Best predicted(dark) overlaid on ground truth(light) Image

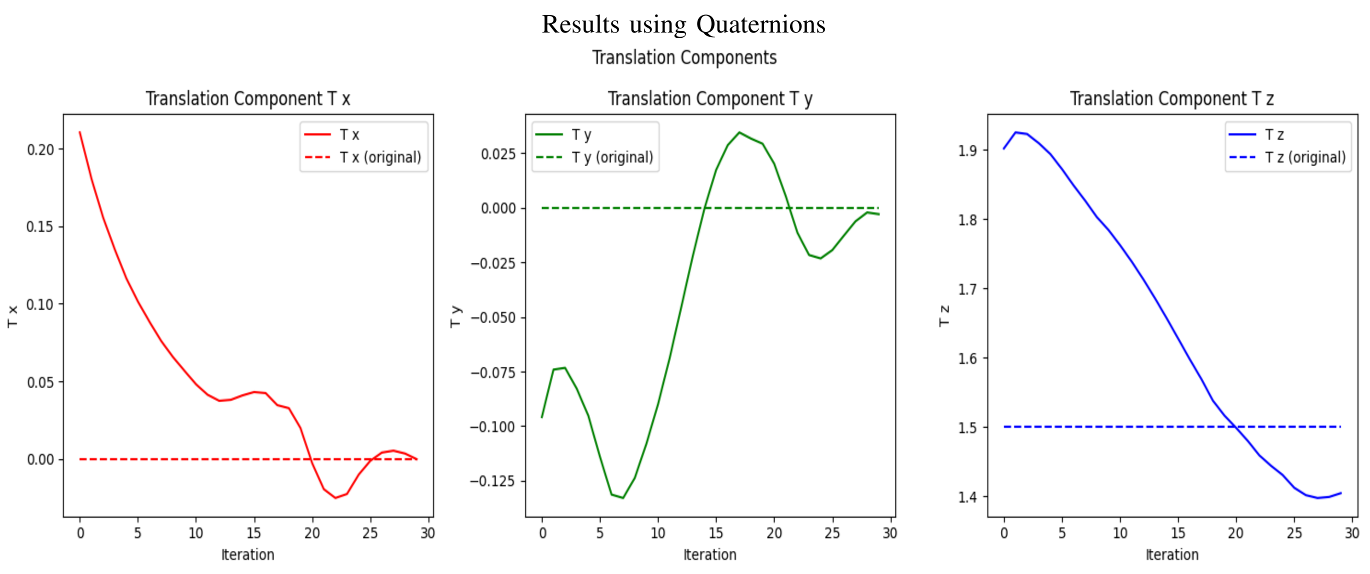

Fig. 15. Update in Translation components over epochs

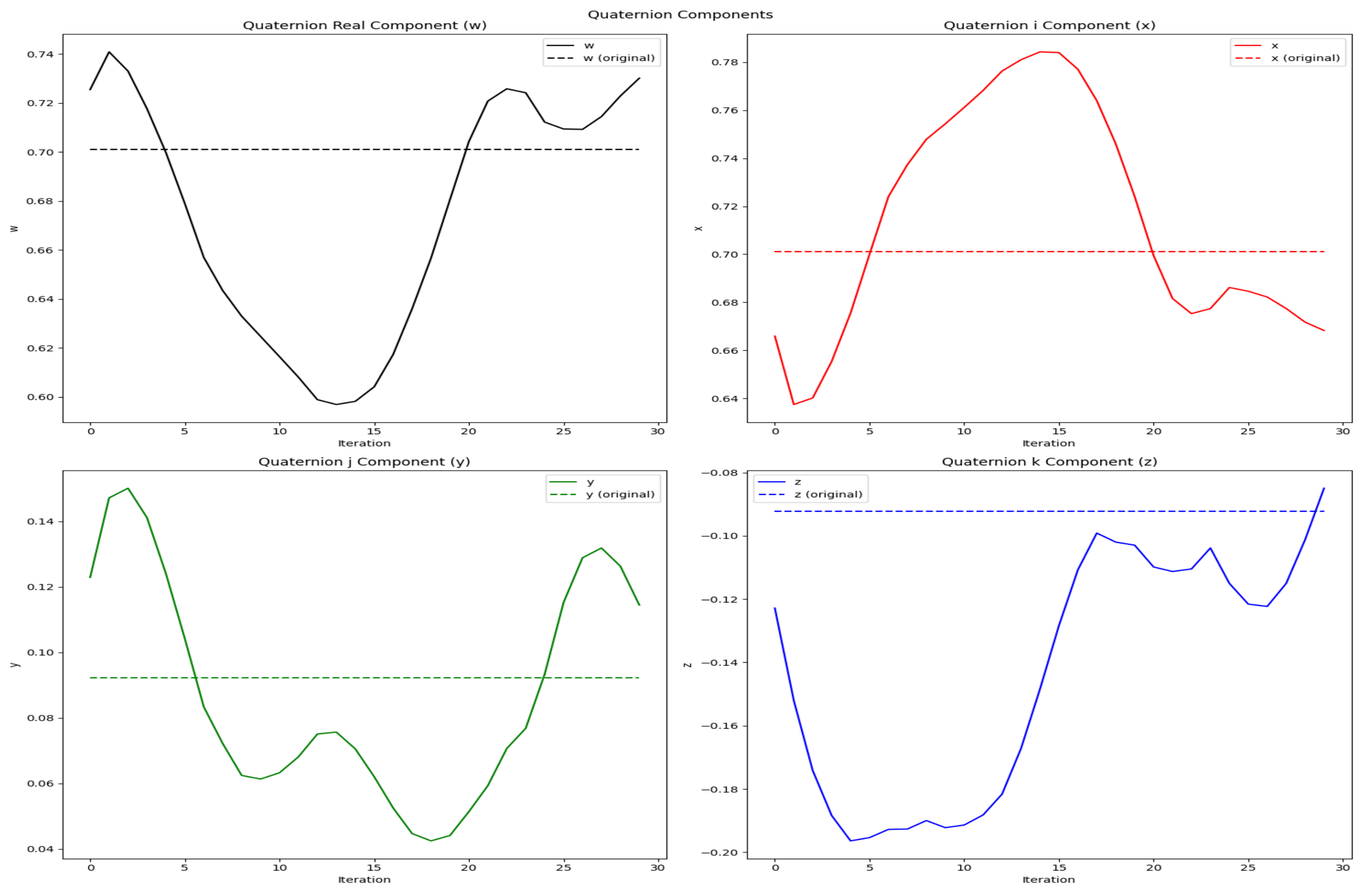

Fig. 16. Update in Rotation components (quaternions) over epochs

IX. Experiments

Initialization Approach using Fundamental Matrix

The approach to obtain a good initial estimate of the 6 Degrees of Freedom (6DOF) pose of a camera with respect to an object in an image. The method involves rendering another image of the mesh, finding the fundamental matrix, and using it to determine the best 6DOF pose of the camera.

To estimate the camera pose from a 3D mesh, the mesh is rendered from various viewpoints by applying rotations. Camera extrinsics, including rotation and translation, are defined, and the axis-angle representation is converted to rotation matrices. Keypoints are detected in the rendered images using SIFT, and features are matched between views with filtering to ensure quality matches. The fundamental matrix is computed from the matched points using the 8-point algorithm, and the essential matrix is derived to recover rotation and translation. Multiple camera poses are generated, and the one consistent with the scene geometry is selected.

By following the steps outlined above, we can obtain a good initial estimate of the 6DOF pose of a camera with respect to an object in an image. This approach leverages feature matching, fundamental matrix computation, and pose recovery to achieve accurate pose estimation. But in our case during testing it was failing for multiple cases and the failing conditions were unpredictable.

X. Discussion

A. Advantages of Each Representation

- **Axis-Angle**: Simple to implement and interpret but can suffer from singularities. It has 6 parameters in total.
- **Quaternion**: Avoids singularities and provides smooth interpolation but requires careful normalization during updation steps. This also requires 7 parameters in total.
- **Rotation Matrix**: Direct manipulation of R offers intuitive control but has 12 parameters in total.

B. Numerical Stability

Ensuring numerical stability is essential:

- Adding small ϵ values during normalization prevents division by zero.
- Using Angle Axis for perturbation prevents going out from SO(3) space.

Definition of the Mesh M'

Let M' be the object's mesh defined as:

$$M' := \left\{ v \;\middle|\; \exists\, r \in \text{FOV s.t } v_r^T \cdot r = 0 \text{ and } \|v - o\| < \|v_j - o\|, \; \forall v_j \text{ such that } v_j^T \cdot r = 0,\; v_j \neq v \right\} \quad (5)$$

Here, M' represents the foreground image mesh.

M' is the set of all points where, for all rays r from the camera at position O_c within the field of view (FOV), v is the closest vertex intersecting the ray r.

Loss Gradient with Respect to Parameters q_i

The gradient of the loss with respect to the parameters q_i is given by:

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

The loss L is defined as:

$$L = (I_{\text{GT}} - I_p)^2 \quad \Rightarrow \quad \frac{\partial L}{\partial I_p} = -2\,(I_{\text{GT}} - I_p)$$

where:

- L: Loss function
- I_p: Projected image
- I_{GT}: Ground truth image

Projected Image Differential with Rotation Matrix $\frac{\partial}{\partial R}(I_p)$

$$I_p = P \cdot M'$$

where P is the Projection Matrix

$$P = K\,[R \mid t]$$

and M' is the Foreground Mesh defined before which is also dependent on camera Extrinsic and thus R.

Deriving $\frac{\partial}{\partial R}(PM')$

Given:

$$F = PM',$$

where $P = P(R)$ and $M' = M'(R)$, the derivative with respect to R is given by the product rule:

$$\frac{\partial (PM')}{\partial R} = \frac{\partial P}{\partial R} M' + P \frac{\partial M'}{\partial R}.$$

Components of the Derivative

1) **Derivative of P with respect to R:**

   $\frac{\partial P}{\partial R}$ is a tensor of shape $(n_p \times m_p) \times (n_r \times m_r)$,

   where n_p, m_p are the dimensions of P, and n_r, m_r are the dimensions of R.

2) **Derivative of M' with respect to R:**

   $\frac{\partial M'}{\partial R}$ is a tensor of shape $(n_{m'} \times m_{m'}) \times (n_r \times m_r)$,

   where $n_{m'}, m_{m'}$ are the dimensions of M'.

3) Substituting back into the product rule:

$$\frac{\partial (PM')}{\partial R} = \left(\frac{\partial P}{\partial R}\right) M' + P \left(\frac{\partial M'}{\partial R}\right).$$

Numerical Approximation Using Finite Differences

If an analytical solution is not feasible, finite differences can be used. For a specific element R_{ij}, the finite difference is given by:

$$\left[\frac{\partial P}{\partial R}\right]_{ij} \approx \frac{P(R + \epsilon E_{ij}) - P(R)}{\epsilon},$$

where E_{ij} is a matrix of the same shape as R, with 1 at position (i, j) and 0 elsewhere.

Third Gradient Term

- **For Quaternions:**
$$\frac{\partial R}{\partial q_i}.$$
- **For Rotational Matrix:**
$$\frac{\partial R}{\partial R}.$$
- **For Angle-Axis:**
$$\frac{\partial R}{\partial \mathrm{AA}_i}.$$

XI. Conclusion

We explored pose optimization using different rotation representations and demonstrated their implementation in a PyTorch framework. Each representation has unique considerations during optimization. Understanding these nuances is critical for effective pose estimation in 3D rendering applications.

XII. Contribution of Team Members

- Meet Gera: Implement Differential Rendering Algorithm, Understanding Relevant Literature, Project Report and Documentation, Deriving Math Expression for Backpropagation of Loss.
- Abhinav Raundhal: Implement Differential Rendering Algorithm, Understanding Relevant Literature, Project Report and Documentation, Running experiments on different objects for obtaining results
- Amey Karan: Implement Differential Rendering Algorithm, Understanding Relevant Literature, Project Report and Documentation, Installing Dependencies and Environment creation

References

[1] 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: Springer International Publishing, 2019, pp. 702–715.

[2] I. Shugurov, I. Pavlov, S. Zakharov, and S. Ilic, "Multi-view object pose refinement with differentiable renderer," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2579–2586, Apr. 2021. [Online]. Available: http://dx.doi.org/10.1109/LRA.2021.3062350

[3] 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.