3D Reconstruction and Camera-Guided Object Transformation

Ghulam Nabi

January 31, 2026

1 Introduction

This report presents a complete pipeline for reconstructing a three-dimensional object from a video sequence and performing camera-guided geometric transformations on the reconstructed model. The workflow follows a Structure-from-Motion (SfM) approach using COLMAP, followed by rigid transformations and metric scaling of the resulting point cloud.

The objectives of this task are:

- Extract frames from a video
- Reconstruct a 3D point cloud and camera poses
- Select an arbitrary camera viewing direction
- Translate and rotate the object using this direction
- Resolve scale ambiguity and validate metric scaling

2 Frame Extraction

A video of the object was recorded while moving the camera around it. To ensure diverse viewpoints, frames were sampled uniformly between the first and last frame.

```
indices = np.linspace(0, total - 1, num_frames).astype(int)
cap.set(cv2.CAP_PROP_POS_FRAMES, idx)
ret, frame = cap.read()
```

This approach avoids redundant neighboring frames and improves reconstruction stability.

3 Structure-from-Motion Reconstruction

COLMAP was used to perform Structure-from-Motion, which estimates camera poses and reconstructs a sparse 3D point cloud from the extracted frames.

These commands were used in the terminal to get the 3D point cloud reconstruction.

```
colmap feature_extractor --database_path database.db --image_path
   frames
colmap exhaustive_matcher --database_path database.db
colmap mapper --database_path database.db --image_path frames --
   output_path sparse
colmap model_converter --input_path sparse/0 --output_path sparse_txt
   --output_type TXT
colmap model_converter --input_path sparse/0 --output_path points3D.ply
    --output_type PLY
```

The main outputs used in this work are:

- `points3D.ply`: reconstructed 3D point cloud
- `images.txt`: camera pose information

4 Camera Pose Representation

Each camera pose is represented by a rotation and translation. COLMAP stores rotation as a quaternion:

$$q = (q_w, q_x, q_y, q_z)$$

The quaternion is converted to a rotation matrix using:

$$R = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-zw) & 2(xz+yw) \\ 2(xy+zw) & 1-2(x^2+z^2) & 2(yz-xw) \\ 2(xz-yw) & 2(yz+xw) & 1-2(x^2+y^2) \end{bmatrix}$$

Implemented in code as:

```
R = q2r(q)
```

The camera projection model is:

$$x = RX + t$$

The camera center in world coordinates is computed as:

$$\boxed{C = -R^\top t}$$

```
C = -R.T @ t
```

In camera coordinates, the forward viewing direction is:

$$v_{cam} = (0, 0, 1)$$

This is transformed into world coordinates as:

$$\boxed{v = R^\top v_{cam}}$$

```
v = R.T @ np.array([0.0, 0.0, 1.0])
v = v / np.linalg.norm(v)
```

The vector v is a unit vector indicating the direction in which the camera is facing.

5 Object Translation and Rotation

Let the reconstructed point cloud be:

$$\mathcal{P} = \{p_i \in \mathbb{R}^3\}$$

The object is translated 5 units along the camera viewing direction:

$$\boxed{p_i' = p_i + 5v}$$

```
pcd_tr.translate(5.0 * v, relative=True)
```

The object is rotated clockwise by 60° around the same axis v. Using axis–angle representation:

$$\theta = -60^\circ$$

The rotation is applied about the centroid c of the point cloud:

$$\boxed{p_i' = R(p_i - c) + c}$$

```
R = o3d.geometry.get_rotation_matrix_from_axis_angle(
    v * (-np.deg2rad(60))
)
pcd_tr.rotate(R, center=pcd_tr.get_center())
```

Structure-from-Motion reconstructions are scale ambiguous. The assignment specifies:

$$1 \text{ unit} = 20 \text{ cm} = 0.2 \text{ m}$$

To convert to metric units:

$$\boxed{s = \frac{1}{0.2} = 5}$$

Each point is scaled as:

$$\boxed{p_i' = c + 5(p_i - c)}$$

```
pcd_sc.scale(5.0, center=pcd_sc.get_center())
```

6 Validation

Two self-selected points p_i and p_j are used to validate scaling.

Euclidean distance is computed as:

$$d(p_i, p_j) = \|p_i - p_j\|_2$$

```
d0 = np.linalg.norm(P0[i] - P0[j])
d1 = np.linalg.norm(P1[i] - P1[j])
```

After scaling:

$$\boxed{d_{after} = 5 \cdot d_{before}}$$

The measured distances matched the expected values, confirming correct metric scaling.

7 Overall Transformation Summary

In mathematical terms, the entire pipeline applies a single composite transformation to every point $p \in \mathbb{R}^3$ in the reconstructed point cloud.

The final transformed point p_{final} is given by:

$$\boxed{p_{\text{final}} = 5\left(R\left(p + 5v - c\right) + c\right)}$$

where:

- v is the unit camera viewing direction in world coordinates,
- R is the rotation matrix corresponding to a clockwise rotation of 60° around the axis v,

- c is the centroid of the point cloud,
- the outer factor of 5 corresponds to metric scaling, converting the reconstruction from arbitrary units to meters.

This expression compactly represents the sequence of operations applied in the pipeline: translation of the object along the camera viewing direction, rotation about the same axis with respect to the object centroid, and final uniform scaling to resolve the scale ambiguity inherent in Structure-from-Motion reconstruction.