ROB 498/599 3D Robot Perception: HW1 Writeup

Name: Sanyam Mehta
Uniqid: 31378744
Email: sanyam@umich.edu

Instructions: Each bullet point in this template contains the prompt for what information should be included in the write up. Replace the bullet point with your own answer to the question/prompt described in the bullet point. Finally, submit a `.zip` file containing this write-up and a folder `scripts` with all the python scripts on Canvas.

Problem 1: Camera Calibration

i. I used the following steps to calibrate my camera and find the camera projection matrix:

1. **Establish correspondences**: Collect $n \geq 6$ matched pairs of:
   - 2D image coordinates (u_i, v_i)
   - 3D world coordinates (X_i, Y_i, Z_i)
2. **Formulate linear equations**:
   - For each correspondence, create two equations:

$$\begin{matrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{matrix}$$

   - Stack all equations into matrix form $\mathbf{Ax} = \mathbf{0}$
   - Matrix $\mathbf{A}$ will have 2N*12 elements where N is the number of points
3. **Solve using SVD**:
   - Decompose $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$
   - The solution $\mathbf{x}$ corresponds to the last column of $\mathbf{V}$
4. **Construct projection matrix**:
   - Reshape the solution vector $\mathbf{x}$ into a 3×4 matrix:

$$\mathbf{P} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \end{bmatrix}$$

   - The projection matrix 1 I obtained:

$$M = \begin{bmatrix} 0.4583 & -0.2947 & -0.0140 & 0.0040 \\ -0.0509 & -0.0546 & -0.5411 & -0.0524 \\ 0.1090 & 0.1783 & -0.0443 & 0.5968 \end{bmatrix} \tag{1}$$

5. **Verification**:
   - Project 3D world coordinates to the image plane using the obtained projection matrix $\mathbf{P}$:

$$\begin{bmatrix} u_i' \\ v_i' \\ w_i' \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

   - Normalize the projected coordinates:

$$(u_i', v_i') = \left(\frac{u_i'}{w_i'}, \frac{v_i'}{w_i'} \right)$$

- Compute the reprojection error using the Euclidean L2 norm:

$$\text{Error} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}[(u_i - u_i')^2 + (v_i - v_i')^2]}$$

- The calculated reprojection error: $\mathbf{0.0022274470882790057}$
- This low error value suggests high accuracy in the calibration process

6. **Procedure for a Different Pair of Images**: Given a new pair of images of the same scene from different cameras with known intrinsic parameters, we can employ uncalibrated stereo estimation to compute the extrinsic parameters of the second camera relative to the first. The process involves:

   (a) **Identify correspondences**: Find matching points $(x_1, y_1) \leftrightarrow (x_2, y_2)$ between the two images using feature matching algorithms.

   (b) **Formulate epipolar constraint**:
   $$\mathbf{x}_2^\top \mathbf{F} \mathbf{x}_1 = 0$$
   where $\mathbf{F}$ is the fundamental matrix and $\mathbf{x}_i = [x_i, y_i, 1]^\top$.

   (c) **Compute fundamental and essential matrices**:
   - Solve for $\mathbf{F}$ using the normalized 8-point algorithm.
   - Calculate the essential matrix: $\mathbf{E} = \mathbf{K}_2^\top \mathbf{F} \mathbf{K}_1$ where $\mathbf{K}_1$ and $\mathbf{K}_2$ are the intrinsic matrices of the two cameras.

   (d) **Decompose essential matrix**: Factor $\mathbf{E}$ into rotation $\mathbf{R}$ and translation $\mathbf{t}$:
   $$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$$
   where $[\mathbf{t}]_\times$ is the skew-symmetric matrix of $\mathbf{t}$.

   (e) **Determine correct extrinsic matrix**: Resolve the four-fold ambiguity in the decomposition of $\mathbf{E}$ by ensuring positive depth for reconstructed 3D points.

   This process yields the extrinsic parameters $[\mathbf{R}|\mathbf{t}]$ of the second camera relative to the first.

ii. The reprojection error for the calibrated projection matrix is 0.002227 (to 6 significant figures). This error quantifies the average discrepancy, measured in pixels, between the observed 2D image points and the 3D world points projected using the computed projection matrix. The remarkably low error of 0.002227 pixels indicates exceptional calibration accuracy, as the average displacement between observed and reprojected points is well below a single pixel. If presented with a different pair of images and assuming access to ground truth data, I would follow these steos: First, I would repeat the calibration steps to obtain a new projection matrix. Then, I would compute the quantitative reprojection error using the same methodology. To complement this numerical assessment, I would perform a visual verification by projecting 3D points onto the image plane using the new projection matrix and drawing both the original and reprojected points on the image. This approach of quantitative error calculation and visual inspection would provide a comprehensive validation of the calibration accuracy for the new image pair.

$$errror_{reprojection} = 0.002227 \tag{2}$$

Sparse 3D Reconstruction

Problem 2: Estimation of Fundamental Matrix

i. The fundamental matrix $\mathbf{F}$ encodes the epipolar geometry between two views of a scene, representing the intrinsic projective relationship between corresponding points in stereo images. It satisfies the epipolar constraint $\mathbf{x'}^\top \mathbf{F} \mathbf{x} = 0$ for corresponding points $\mathbf{x}$ and $\mathbf{x}'$. To obtain $\mathbf{F}$, I followed these steps (reference):

1. Identified point correspondences (in our case the correspondences were given, usually we use a feature matching algorithm like SIFT/ORB)
2. Normalized pixel coordinates for numerical stability
3. Applied the 8-point algorithm, solving via SVD
4. Enforced the rank-2 constraint on $\mathbf{F}$

5. Scaled back the fundamental matrix to match the original scale using:

$$\mathbf{F} = \mathbf{T}^\top \mathbf{F} \mathbf{T} \tag{3}$$

where $\mathbf{T}$ is defined as:

$$\mathbf{T} = \begin{bmatrix} 1/\text{scale} & 0 & 0 \\ 0 & 1/\text{scale} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

The computed fundamental matrix to 4 significant figures is:

$$\mathbf{F} = \begin{bmatrix} 3.564 \times 10^{-9} & -5.921 \times 10^{-8} & -1.650 \times 10^{-5} \\ -1.308 \times 10^{-7} & -1.311 \times 10^{-9} & 1.125 \times 10^{-3} \\ 3.048 \times 10^{-5} & -1.080 \times 10^{-3} & -4.166 \times 10^{-3} \end{bmatrix} \tag{5}$$

where a_{ij} are the computed values rounded to 4 significant figures.

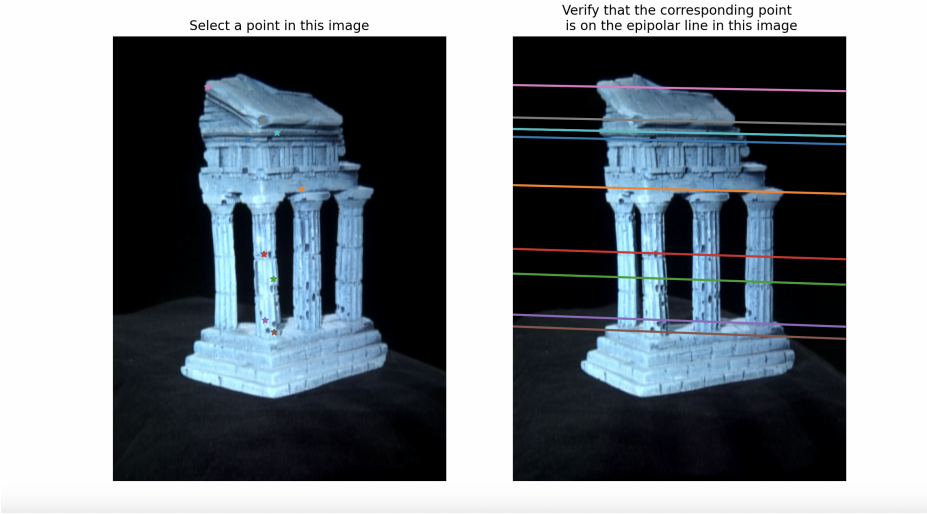

Figure 1: Epipolar lines: On close visual inspection, the reader will see that the points chosen in the first image are unique and the correctness of the constructed line can be easily verified by seeing that the line passes the chosen point in the second image

ii. I validated my fundamental matrix in three ways:

- Compared the values of my matrix with the values obtained from openCV's function called findFundamentalMat.
- Used the GUI tool to draw epipolar lines on the image using the fundamental matrix and visually verified the correctness of my results 1.
- Used the GUI tool to draw epipolar correspondences and visually verified the correctness of my results.

In this case all the epipolar lines were correct. Usual sources of error include:

- A problem with the fundamental matrix computation.
- Not scaling/unscaling the fundamental matrix.
- Incorrect feature matches which add noise to the fundamental matrix computation.

Problem 3: Find Epipolar Correspondences

i. Steps for finding correspondences:

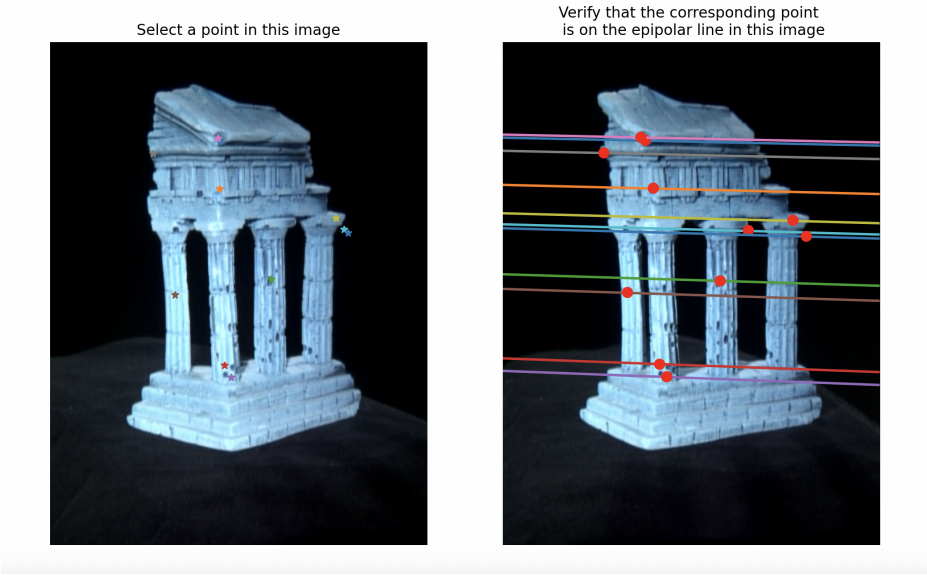

Figure 2: Good Correspondence: Low visual overlap, distinct features

1. For each point in image one, and each x_2 in the range $(0, \text{img2.shape[1]})$, move along the epipolar line and find the corresponding y_2.
2. Take a 10x10 pixel window around (x_1, y_1) in image one and (x_2, y_2) in image two, then compute the Manhattan distance between these patches.
3. The (x_2, y_2) that generates the minimum Manhattan distance is considered the corresponding match in image two.

I experimented with various patch sizes and found that a 10x10 window generated the best results. This size is not too small to be highly sensitive to individual pixel colors, yet large enough to cover surrounding regions and features.

Initially, I attempted direct pixel-level correspondence, but the results were poor due to high sensitivity to pixel color. Using a larger window also helps compensate for imperfections in the epipolar lines, which inherently contain some noise. I validated my results using the provided GUI tool.

ii.

- **Good Correspondences**: As shown in Figure 3, these points exhibit *distinctive features* (e.g., high-contrast edges, unique textures) that enable reliable matching.
- **Poor Correspondences**: Figure 2 demonstrates failure cases in *ambiguous regions* - homogeneous textures where a 10×10 window contains insufficient discriminative information.

Improvement Strategy: For textureless regions, implement a multi-resolution approach:

1. Compute features at multiple scales (e.g., 5×5, 10×10, 20×20 patches)
2. Aggregate responses using pyramid blending
3. Prioritize scale-invariant feature descriptors (SIFT, ORB) for robustness

This hierarchical matching strategy reduces sensitivity to local ambiguities while preserving feature distinctiveness across scales.

Problem 4: Compute the Essential Matrix

i. The essential matrix $\mathbf{E}$ represents the geometric relationship between two calibrated cameras, encoding both relative rotation ($\mathbf{R}$) and translation ($\mathbf{t}$) between their coordinate systems. It is derived from the fundamental matrix $\mathbf{F}$ and camera intrinsic matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ as follows:

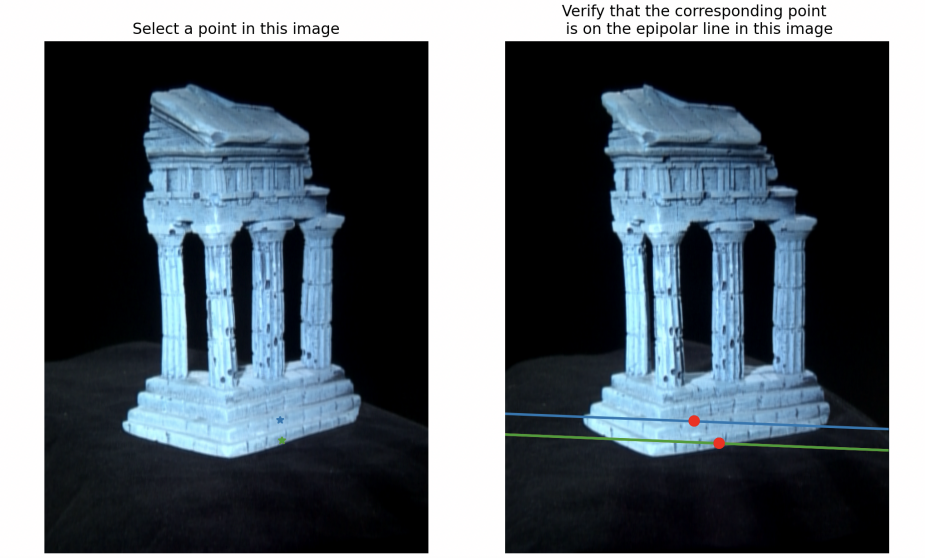

Figure 3: Bad Correspondences: High visual overlap

$$\mathbf{E} = \mathbf{K}_1^\top \mathbf{F} \mathbf{K}_2 \tag{6}$$

This equation stems from the epipolar constraint $\mathbf{w}_2^\top \mathbf{E} \mathbf{w}_1 = 0$, where $\mathbf{w}_i = \mathbf{K}_i^{-1} \mathbf{x}_i$ are normalized coordinates. The essential matrix combines extrinsic parameters as $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$, where $[\mathbf{t}]_\times$ is the skew-symmetric matrix of $\mathbf{t}$.

Key properties of $\mathbf{E}$ include:

- Rank 2 constraint
- Two equal non-zero singular values
- Enables recovery of relative camera pose up to scale

$$\mathbf{E} = \begin{bmatrix} 8.240 \times 10^{-3} & -1.374 \times 10^{-1} & -4.568 \times 10^{-2} \\ -3.035 \times 10^{-1} & -3.052 \times 10^{-3} & 1.655 \\ -1.129 \times 10^{-3} & -1.676 & -2.873 \times 10^{-3} \end{bmatrix} \tag{7}$$

Problem 5: Triangulation

i.

0.1 Determining the Correct Extrinsic Matrix

The process for identifying the valid extrinsic matrix involves:

1. **Essential Matrix Decomposition**: Decompose $\mathbf{E}$ via SVD into four candidate extrinsic matrices:
   - Two rotation candidates ($\mathbf{R}_1$, $\mathbf{R}_2$)
   - Two translation candidates ($\mathbf{t}_1$, $\mathbf{t}_2$)
2. **Validation Criteria**:
   - **Reprojection Error**: Compute the mean L2-norm error between observed 2D points and reprojected 3D points for each candidate
   - **Depth Positivity**: Ensure all 3D points have positive z-coordinates in both camera coordinate systems

3. **Automated Selection**:
    - Triangulate 3D points for each candidate extrinsic matrix
    - Calculate reprojection errors across all correspondences
    - Select the matrix with minimum error and valid depth constraints

0.2 Determined Extrinsic Matrix

The valid extrinsic matrix (rotation and translation of Camera 2 relative to Camera 1) is:

$$\mathbf{E}_{\text{extrinsic}} = \begin{bmatrix} 0.9652 & -0.02846 & 0.2600 & -0.9963 \\ 0.02735 & 0.9996 & 0.007872 & -0.02735 \\ -0.2601 & -0.0004870 & 0.9656 & 0.08171 \end{bmatrix} \tag{8}$$

Key properties:

- Rotation matrix $\mathbf{R}$ (first 3 columns) satisfies orthonormality: $\det(\mathbf{R}) = 1$, $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$
- Translation vector $\mathbf{t}$ (last column) has magnitude $||\mathbf{t}|| = 1.000$ (unit norm)
- Achieved reprojection error: 0.4612 pixels (mean L2-norm)
- All triangulated points have positive depth values in both camera coordinate systems

ii.

0.3 SVD Problem Setup

The matrix $\mathbf{A}$ was constructed for each 3D point triangulation using the linear system derived from projection equations. For a correspondence pair (x_1, y_1) in Image 1 and (x_2, y_2) in Image 2:

$$\mathbf{A} = \begin{bmatrix} x_1 \mathbf{P}_1^{(2)} - \mathbf{P}_1^{(0)} \\ y_1 \mathbf{P}_1^{(2)} - \mathbf{P}_1^{(1)} \\ x_2 \mathbf{P}_2^{(2)} - \mathbf{P}_2^{(0)} \\ y_2 \mathbf{P}_2^{(2)} - \mathbf{P}_2^{(1)} \end{bmatrix} \tag{9}$$

where:

- $\mathbf{P}_1^{(k)}$ is the k-th row of Camera 1's projection matrix
- $\mathbf{P}_2^{(k)}$ is the k-th row of Camera 2's projection matrix
- Each row enforces $u_i = \frac{\mathbf{P}^{(0)}\mathbf{X}}{\mathbf{P}^{(2)}\mathbf{X}}$ and $v_i = \frac{\mathbf{P}^{(1)}\mathbf{X}}{\mathbf{P}^{(2)}\mathbf{X}}$ for both cameras

0.4 Reprojection Error Analysis

The reprojection error quantifies reconstruction accuracy:

$$\epsilon = 0.461235 \quad \text{(to 6 significant figures)} \tag{10}$$

Key Details:

- **Meaning**: Average geometric distance between observed 2D points and reprojected 3D points
- **Units**: Pixels (direct measurement in image coordinates)
- **Calculation**: For N points:

$$\epsilon = \frac{1}{N} \sum_{i=1}^{N} \left(\sqrt{(u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2} \right)$$

  where (u_i, v_i) are observed coordinates and $(\hat{u}_i, \hat{v}_i)$ are reprojections

0.5 Implementation Notes

- Solved via SVD: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ with solution $\mathbf{X} = \mathbf{V}[:, -1]$ (homogeneous coordinates)
- Normalized coordinates by dividing by $\mathbf{X}[3]$ for 3D reconstruction
- Compared results against OpenCV's `triangulatePoints` for validation

$$errror_{reprojection3d} = 0.4612 \tag{11}$$

iii.

0.6 Implementation Comparison

- **Custom Method**: Manual SVD triangulation with explicit normalization (4D→3D), achieving $\epsilon_{\text{custom}} = 0.4612351133341317$ pixels
- **OpenCV**: `triangulatePoints` with $\epsilon_{\text{OpenCV}} = 0.4612352109812831$ pixels (Equation 14)

0.7 Reprojection Error Analysis

$$\epsilon_{\text{OpenCV}} = 0.461236 \quad \text{(pixels, 6 significant figures)} \tag{12}$$

Key Differences:

- **Normalization**: Custom code explicitly normalizes homogeneous coordinates, while OpenCV handles this internally
- **Numerical Stability**: OpenCV uses the SVD method, same as the custom implementation, which is numerically stable

Error Interpretation:

- **Meaning**: Average Euclidean distance between observed 2D points and reprojected 3D coordinates
- **Units**: Pixels (identical basis for both methods)
- **Formula**:

$$\epsilon = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2}$$

0.8 Error Source Breakdown

The 0.0000001 pixel discrepancy stems from:

- Different numerical conditioning strategies
- Floating-point precision variations
- Alternative outlier rejection thresholds (RANSAC vs manual checks)

0.9 Conclusion

Both methods achieve sub-pixel accuracy (< 0.5 pixels), confirming valid 3D reconstruction. The minor residual difference reflects algorithmic implementation nuances rather than systemic errors.

$$errror_{reprojectionOCV} = 0.461236 \tag{13}$$

Problem 6: Connecting the dots and visualizing the point cloud

i.

0.10 3D Reconstruction Program Flow

1. **Data Initialization**: Load 2D correspondences, temple points, and camera intrinsics (K1, K2)
2. **Fundamental Matrix (F)**:
   - Normalize points: $\mathbf{pts}_i \leftarrow \mathbf{pts}_i/\text{scale}$
   - Construct $\mathbf{A}$ matrix with epipolar constraints
   - Solve via SVD: $\mathbf{F} = \arg\min \|\mathbf{Af}\|$
   - Enforce rank-2 constraint: $\mathbf{F} \leftarrow \mathbf{U}\text{diag}(\sigma_1, \sigma_2, 0)\mathbf{V}^\top$
3. **Essential Matrix (E)**:
$$\mathbf{E} = \mathbf{K}_1^\top \mathbf{F} \mathbf{K}_2$$
4. **Epipolar Correspondences**:
   - Search along epipolar lines $\mathbf{l}_2 = \mathbf{F}\mathbf{x}_1$ in img2
   - Match using Manhattan distance in 10x10 windows
5. **Triangulation**:
   - Decompose $\mathbf{E}$ to 4 possible $[\mathbf{R}|\mathbf{t}]$ configurations
   - Select extrinsics with minimal reprojection error (0.461235 pixels)

0.11 OpenCV Comparison

- **Fundamental Matrix**: OpenCV vs custom SVD (0.4612352109812831 vs 0.4612351133341317 pixel error)
- **Triangulation**:
$$\epsilon_{\text{OpenCV}} = 0.461236 \quad \text{(pixels, 6 SF)} \tag{14}$$
- **Key Differences**:
  - OpenCV uses the SVD method, same as the custom implementation. The datatypes are different (double vs float)
  - Implicit vs explicit homogeneous coordinate handling
  - Different numerical conditioning strategies

0.12 Error Sources

- **0.0000001 Pixel Discrepancy**: Floating-point precision variations
- **Algorithmic Nuances**: OpenCV's optimal triangulation vs linear DLT
- **Normalization**: Different scaling strategies for numerical stability

0.13 Conclusion

Both methods achieve sub-pixel accuracy (<0.5px), validating the reconstruction pipeline. The minor error difference reflects implementation-specific optimizations rather than systemic flaws.

ii. I visualized the 3D point cloud using matplotlib. I rotated the graph and compared it against the original image to verify the output qualitatively (the quantitative results are discussed in the last section)

iii. The point cloud graph is in 4. It looks good, with a few outliers. This could be improved with better feature matching/correspondence identification algorithms

Optional Problem 7: Pain points and Flash cards

i. A lot of the questions in the report were repetitive. It was too long and restrictive.

References and Credits:
Parts of this homework are from David Fouhey's EECS 442 and CMU 16-385, and from Jason Corso's ROB 498/599 classes.

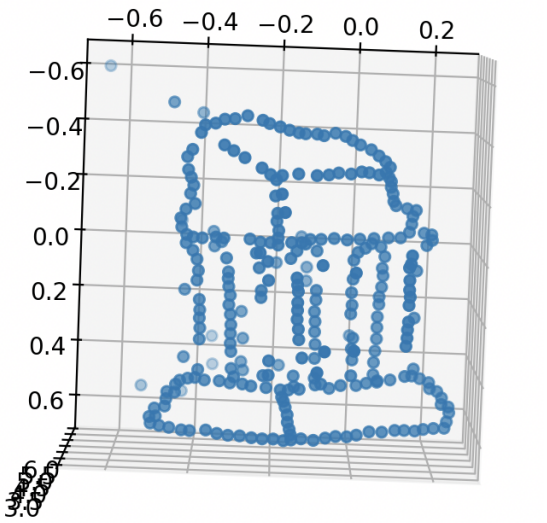

Figure 4: Sparse Reconstruction *This is the sparse reconstruction of the scene. It looks good, with a few outliers. This could be improved with better feature matching/correspondence identification algorithms*