3DCV Homework 2

1 Problem 1

1.1 Structure from Motion

A 30-second video of the CSIE Department was recorded at 30 fps. Because the video was short, all frames were kept for processing to maximize image overlap and feature coverage.
Pipeline:

1. Feature extraction: Obtain the SIFT features of each frame. The simple radial camera model was used.
2. Feature matching: Apply the sequential feature matching, which efficiently handles temporally adjacent frames in the video while maintaining reliable overlap.
3. Incremental reconstruction: Executed COLMAP's mapper to estimate camera poses and produce a sparse point cloud.

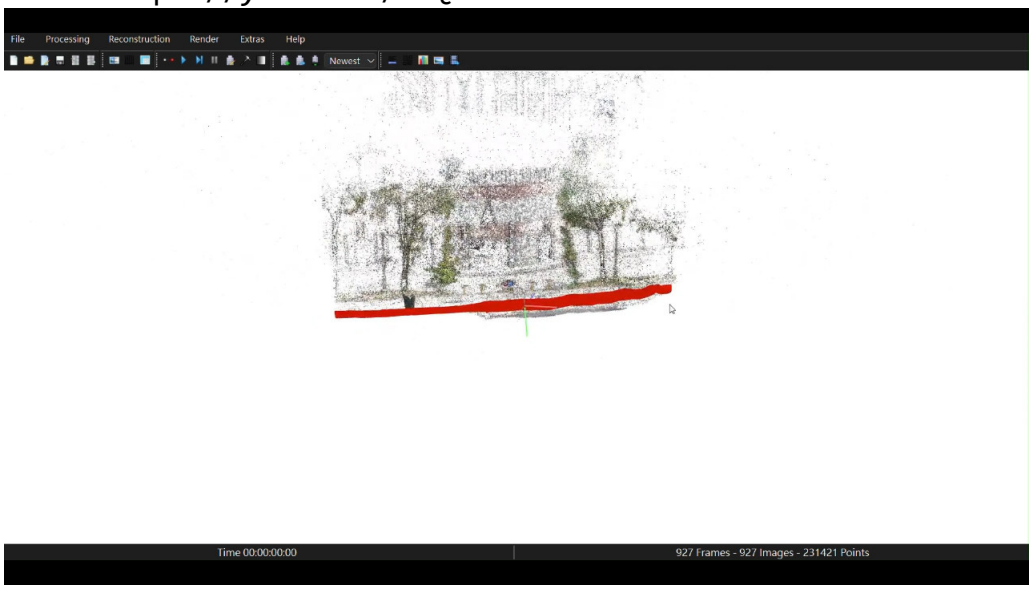
1.2 Mesh Reconstruction

The open-source software CloudCompare was employed for point cloud visualization, editing, and surface reconstruction. It provides interactive tools for segmentation, filtering, normal estimation, and mesh reconstruction, making it well suited for quick processing of SfM outputs.
Pipeline:

1. Load data: Import the point cloud from the .ply file generated by COLMAP.
2. Cropping: Trimmed the cloud to retain only the CSIE building.
3. Noise reduction: Applied statistical outlier removal and spatial down-sampling to reduce noise.
4. Normal estimation: Used a Hough-based orientation method to compute and align surface normals.
5. Poisson reconstruction: Executed Poisson surface reconstruction to obtain the mesh model.

1.3 Results

Video: `https://youtu.be/TEQvZiRv1Ck`



2 Problem 2

Environment and Execution

- python 3.11.13
- open3d 0.19.0
- opencv-python 4.12.0.88
- numpy 2.2.6
- scipy 1.16.2
- pandas 2.3.3
- tqdm 4.67.1

1. Put the dataset under `./data/`.
2. Execute `python transform_cube.py` to adjust the cube location in Q2-2 and generate the vertices array.
3. Execute `python 2d3dmathcing.py` to run the entire problem. It will print the median rotation error and translation error, show the point cloud with camera trajectory and poses in an Open3D window, and play the AR video with a virtual cube.

2.1 Camera Pose Estimation

1. Descriptor Matching: Use FLANN-based matcher with Lowe's ratio test to find the corresponding features between each validation image and the given model.
2. Pose computation: Apply OpenCV's PnP Ransac function to obtain the rotation and translation vectors for each image.
3. Error evaluation: Compute the translational and rotational errors between the estimated poses and the ground truth as required, and report the median values.
4. CCS-to-WCS matrix calculation: Convert the rotation and translation vectors into matrix form and invert it to obtain the camera to world matrix.
5. Camera visualization: Define a base pyramid in camera coordinate system to represent the field of view. Transform the pyramid to the world coordinate with each camera's transformation matrix. Connect the apexes of adjacent cameras to draw the trajectory.

2.2 Augmented Reality video

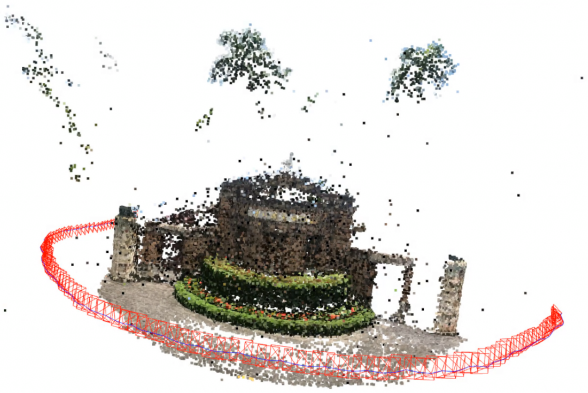
1. Cube point generation: Load the vertices array and use interpolation to generate 10*10 points on each face.
2. Depth calculation: For each image, transform the cube points into camera coordinates using the calculated extrinsic parameters.
3. Point projection: Use the OpenCV's projection function with given intrinsic matrix, distortion coefficients, and calculated extrinsic parameters of each image to project the cube points onto 2D image coordinates.
4. Painter's algorithm: Sort the points by depth and draw from the furthest to the closest. Points located outside the image bounds or behind the camera (negative depth) were skipped.

2.3 Results

Video: `https://youtu.be/UF9zxRv8MYo`

```
(.venv) homework2-fmb123456 ❯ python 2d3dmathcing.py                    main
100%|                                                | 130/130 [02:04<00:00,  1.05it/s]
Median Rotation Error (deg): 0.0023047259431461828
Median Translation Error: 0.0001291439384413469
```

The median rotation error (0.0023 degrees) and median translation error (0.00013) are extremely small, showing that the estimated poses align well with the groundtruth. As a reference, the base size of each red pyramid in the figure below is 0.1 units in length. Such precision suggests that the dataset descriptors were well-matched and that the RANSAC-based PnP solver effectively rejected outliers. The clear structure of the scene (corners and textures) also contributed to this stability.



The red camera pyramids form a smooth and continuous trajectory that naturally curves around the reconstructed front gate model. This result indicates that the estimated poses are spatially consistent and geometrically plausible. By comparing the rendered poses with the corresponding validation images, it can be confirmed that the estimated camera orientations align well with the true viewpoints of the scene.