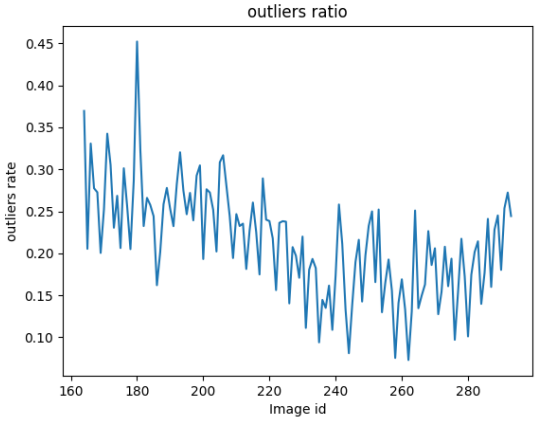
網媒所碩一　薛博文　R11944005

Problem 1: 2D-3D Matching

1. P3P, RANSAC

1. P3P
    - Affine transformation has 6 degree of freedom, so we randomly choose 3 points to match.
    - Compute G_0, G_1, G_2, G_3, G_4 from intrinsic matrix K and corresponding (u, x) pair.
    - Find the roots. (With up to 4 roots)
    - Find a, b, c based on 3D cosine angle and 2D distance.
    - By trilateration theorem, we can compute the translation T. (Three sphere gives 2 solutions)
    - Compute λ, v can be calulated by $v = K^{-1}u$.
    - With $\lambda v = R(x - T)$, rotation matrix R can be further calculated. (Up to 2 possibility)
2. Random Sample Consensus



(a) Outliers ratio

- With many iteration(128) and a large threshold(5) chosen, the outliers appear to be about 20% in average, and 45% at most in our validation dataset. We have 45% outliers in the worse case.
- According to the slide from teacher, we have function

$$1 - (1 - (1 - e)^s)^N = p$$

- By this equation, even the outlier is up to 50%, we can still have 99% chance to choose a inlier set by having 35 iteration.
- By choosing number of iteration=35, we have at least 99% chance to choose a inlier set, and we can save the projected matrix which gives minimal projected error.

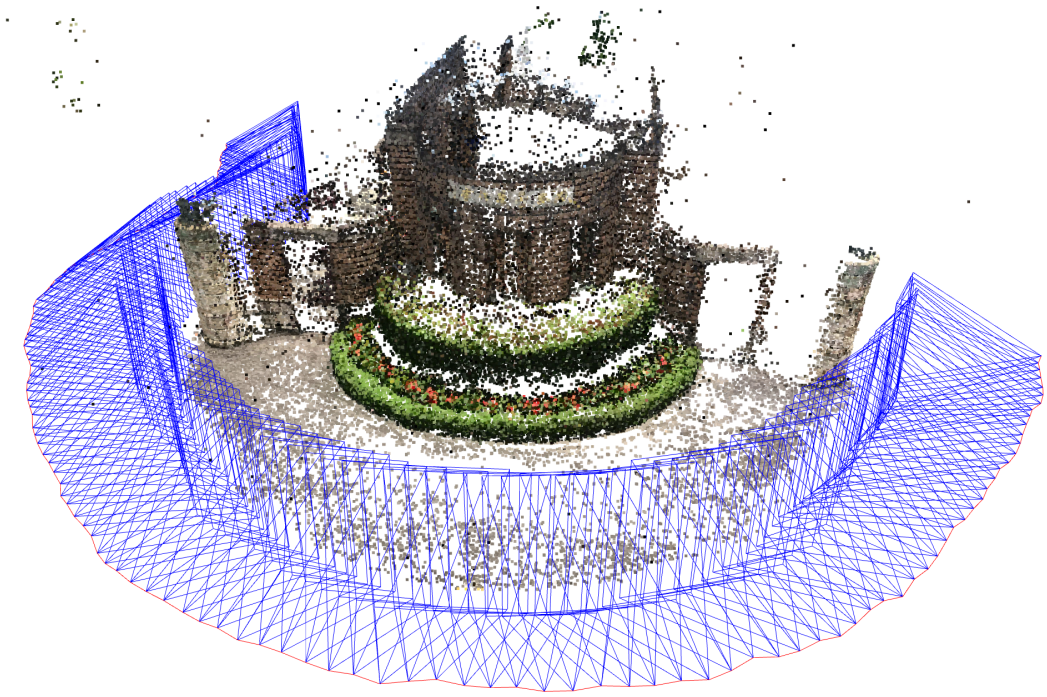
3. Compute the error
    - Compute the 2-norm between ground truth and predicted camera pose. (translation & rotation)
4. Plot the trajectory
    - Given the open3d camera object, intrinsic, extrinsic matrix, following open3d method can plot the camera pose.

$$o3d.geometry.LineSet.create\_camera\_visualization()$$

    - With camera pose is calculate, we have $T_{world} = R^{-1}T_{cam}$, and we can draw the trajectory with

$$o3d.geometry.LineSet()$$

Result



(a) Camera pose and trajectory

Further Discussion

- Median Error of Rotation: 0.0019
- Median Error of Translation: 0.0002
- In RANSAC algorithm, we choose three points as a set randomly. However, it is not very ideal because the error will be magnified if two of the points are very close. Therefore, I skip the case where any of two points in a set is closer than 0.1m in 3D world coordinate. And by doing this, error drops significantly. (Error of Rotation: 0.0044 -> 0.0019, Error of translation 0.0004 -> 0.0002)

Problem 2: Augmented Reality

1. Place the virtual cube

1. Use 'transform_cube.py' to adjust the position of the cube.
2. Read the estimated pose produced by code of problem1.
3. Project 3D points to 2D by the pose of each frame.
4. Implement painter algorithm: points is sort by the depth.
5. Use 'cv2.circle()' to draw points.
6. Plot the cube frame by frame.

How to Run the code

1. For problem 1, use 'python problem1.py' to execute the code, the program will run P3P, RANSAC algorithm, and print out the rotation, translation error. Also, it save the poses of each frame for the problem2 to make video.

2. For problem 2, use 'python problem2.py' to execute the code, the program will make two videos. One is 'gt_video.mp4', and the other is 'pred_video.mp4'.

3. Environment

```
python==3.9.4
numpy==1.19.5
opencv_python==4.6.0.66
matplotlib==3.6.1
scipy==1.9.2
tqdm==4.36.1
```