CSIE5429 3D Computer Vision with Deep Learning Applications

Homework 3 Report

Name: 高榮浩 **ID:** R12922127

- ## Camera Calibration
  - ### Methodology

    I simply calibrated the camera using the provided program to obtain the camera's intrinsic matrix and distortion coefficients.

  - ### Result

```
=> Overall RMS re-projection error: 0.3470685679570199
Camera Intrinsic
[[521.96449056   0.         316.49010244]
 [  0.         523.00978101 182.07710778]
 [  0.           0.           1.        ]]
Distortion Coefficients
[[ 1.28174481e-01 -9.27586722e-01 -1.65340609e-03  4.18645145e-04
   1.71900697e+00]]
```

- ## Feature Matching
  - ### Methodology

    I employ ORB (Oriented FAST and Rotated BRIEF) as a feature extractor, known for its speed advantage of over 10 times compared to SIFT (Scale Invariant Feature Transform). I've adapted the code from the sample provided in the documentation at LINK, as recommended in the homework instructions. Additionally, I implement the detection of outliers to enhance performance and mitigate potential issues.

- ## Pose Estimation using Epipolar Geometry
  - ### Methodology

    I follow the algorithm outlined in the class slides. In the section addressing scale consistency resolution, I refrain from simply picking two matches to calculate the scale. Instead, I explore all probable combinations and choose the median of these scales to mitigate the impact of outliers and ensure robust performance.

■ Pseudo Code

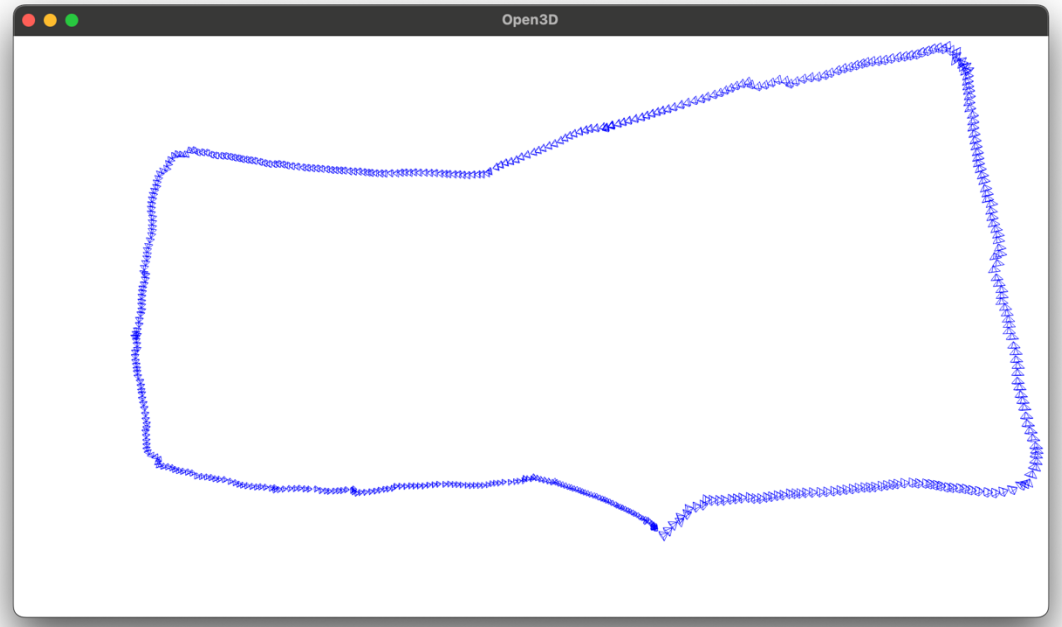
```
let points_past represent the inliers corresponding to keypoints in the previous image
let points_now represent the inliers corresponding to keypoints in the current image
estimate the essential matrix, E, using the input points_past and points_now
decompose the essential matrix, E, into rotation, R, and translation, t, to obtain the relative pose using the input E, points_past and points_now
calculate triangulated points, triangulatedPoints, using the inputs E, points_past, and points_now
match the points from points_past to the points_now of the last iteration, and record these matched points along with their corresponding triangulated points from both this and the last iteration in triPoint_matches
for any two matches in triPoint_matches, calculate the scale and record it in probable_scales
rescale the translation vector, t, by multiplying it with the median of the probable_scales
determine the camera's pose relative to the first camera
```

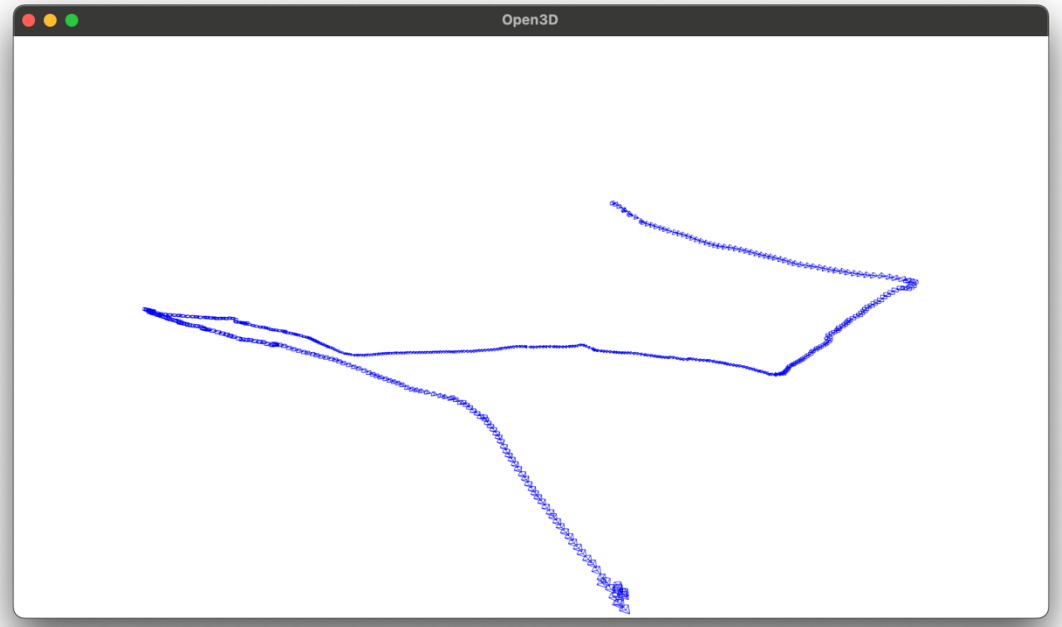
● Results Visualization

■ Methodology

In each camera pose, there are 8 lines that need to be drawn. Initially, I identify and record 5 points. Subsequently, I determine which points will constitute the lines. Finally, I incorporate a LineSet in Open3D to visualize the lines.

■ Result





■ Discussion

From the obtained results, it is apparent that the rough path can be approximately estimated, but some errors persist. In future work, we might explore alternative feature extractors to redo the process and compare performance, taking into account the tradeoff between runtime and accuracy. Additionally, implementing a more robust method for outlier detection could be considered to improve overall performance.

- **YouTube Link**
  https://youtu.be/67kbak3TXyQ

- **Execution**
  - **Command**
    - **Camera Calibration**
      `python ./camera_calibration.py ./calibration_video.avi`

    - **Visual Odometry**
      `python ./visual_odometry.py ./frames`

  - **Environment / Package**
    - `python==3.8.18`
    - `numpy==1.24.4`
    - `opencv-python==4.5.4.58`
    - `open3d==0.14.1`