1. **Camera calibration**

Executed by sample code

```
=> corners found in 14 images
=> Overall RMS re-projection error: 0.22134188646980266
Camera Intrinsic
[[522.42818518   0.         319.274439  ]
 [  0.         522.20859065 175.92882309]
 [  0.           0.           1.        ]]
Distortion Coefficients
[[ 1.71226156e-01 -1.29340241e+00 -2.03148467e-03  2.60487726e-04
   2.61798933e+00]]
```

2. **Feature Matching**

使用 ORB 偵測特徵點，Hamming distance 決定匹配

3. **Pose from Epipolar Geometry (pseudo codes and comments)**

Visual odometry 處理 consistency scale 需要三個 frame，因此這邊會從 ORB 偵測完 keypoint 後分成兩部分解釋

- **Calculate Essential Matrix, Recover Pose**

```
cal_matches_E_pose(keypoint1, descriptor1, keypoint2, descriptor2)
    decide match_points from frame1&2 by Hamming distance
    for match in match_points
        add point to match1 from keypoint1
        add point to match2 from keypoint2
        add descriptor to des2 from descriptor2
    undistort match1, match2
    calculate Essential Matrix by cv2.findEssentialMat(match1, match2, Camera intrinsic matrix)
    recover relative R, relative T, corresponding points by cv2.recoverPose(Essential Matrix, match1, match2, Camera intrinsic matrix)
    get inliers in match2 by corresponding points
    get inliers' descriptor in des2 by corresponding points
    return relative R, relative T, inliers in match2, inliers' descriptor in match2
```

- **Calculate scale**

cal_scale(preFrame, curFrame, posFrame)

init preFrame, curFrame, posFrame projection matrix

decide match_points1 from preFrame & curFrame by Hamming distance

decide match_points2 from curFrame & posFrame by Hamming distance

decide match_points3 for three frames by intersect(match_points1, match_points2)

for match in match_points:

add point to pre_points from preFrame's point

add point to cur_points from curFrame's point

add point to pos_points from posFrame's point

for range(50):

random select two corresponding points in three frames

caculate 3D points use triangulatePoints(preFrame projection Matrix, curFrame projection Matrix, random 2 points in preFrame , random 2 points in curFrame)

transform 3D points to homogenous format

caculate preFrame, curFrame points' distance

caculate 3D points use triangulatePoints(curFrame projection Matrix, posFrame projection Matrix, random 2 points in curFrame , random 2 points in posFrame)

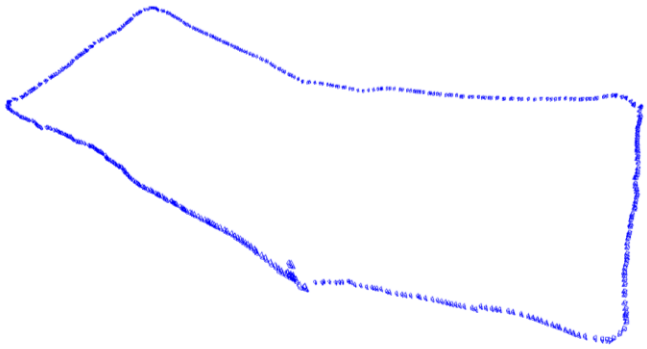
transform 3D points to homogenous format

caculate curFrame, posFrame points' distance

caculate scale by two distance ratio

and return the median of all 50 points scale

4. **Results Visualization**



Video link

https://youtu.be/fvY0gNqICnA

Misc.

- **Python Environment: 3.8.13**
- **Commands**
  - python camera_calibration.py --input calib_video.avi
  - python vo.py --input frames/