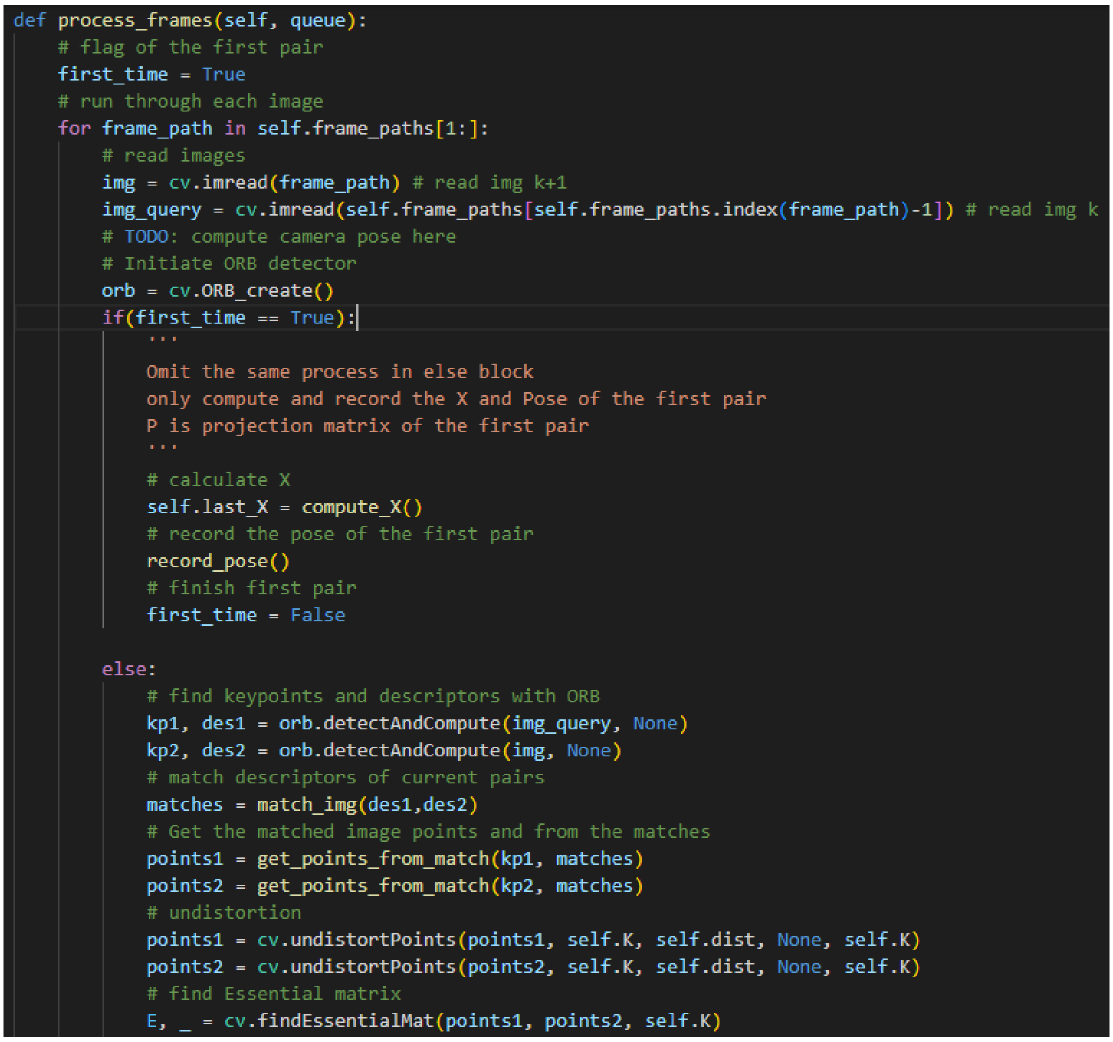
M11152014　蘇珉淙

Report

- Briefly explain your method in each step
    - Camera calibration
      使用助教的 sample code 執行 calibration
    - Feature Matching
      使用 ORB 為 feature extractor
      使用 opencv 的 Matcher.match()來得到最好的 matches
    - Pose from Epipolar Geometry (pseudo codes and comments)
      利用前後影像當作一個 pair，將每個 pair 做特徵點的配對並計算 Essential matrix，分解後得到 relative pose，scale 的問題是將前一個 pair 的 descriptor 記錄下來，並與目前 pair 的 match 最配對，尋找同時在三張影像都出現的點，來計算 relative scale，實驗結果是 scale 有時會不穩定，透過設定 scale 控制在 0 到 3 之間效果較好。

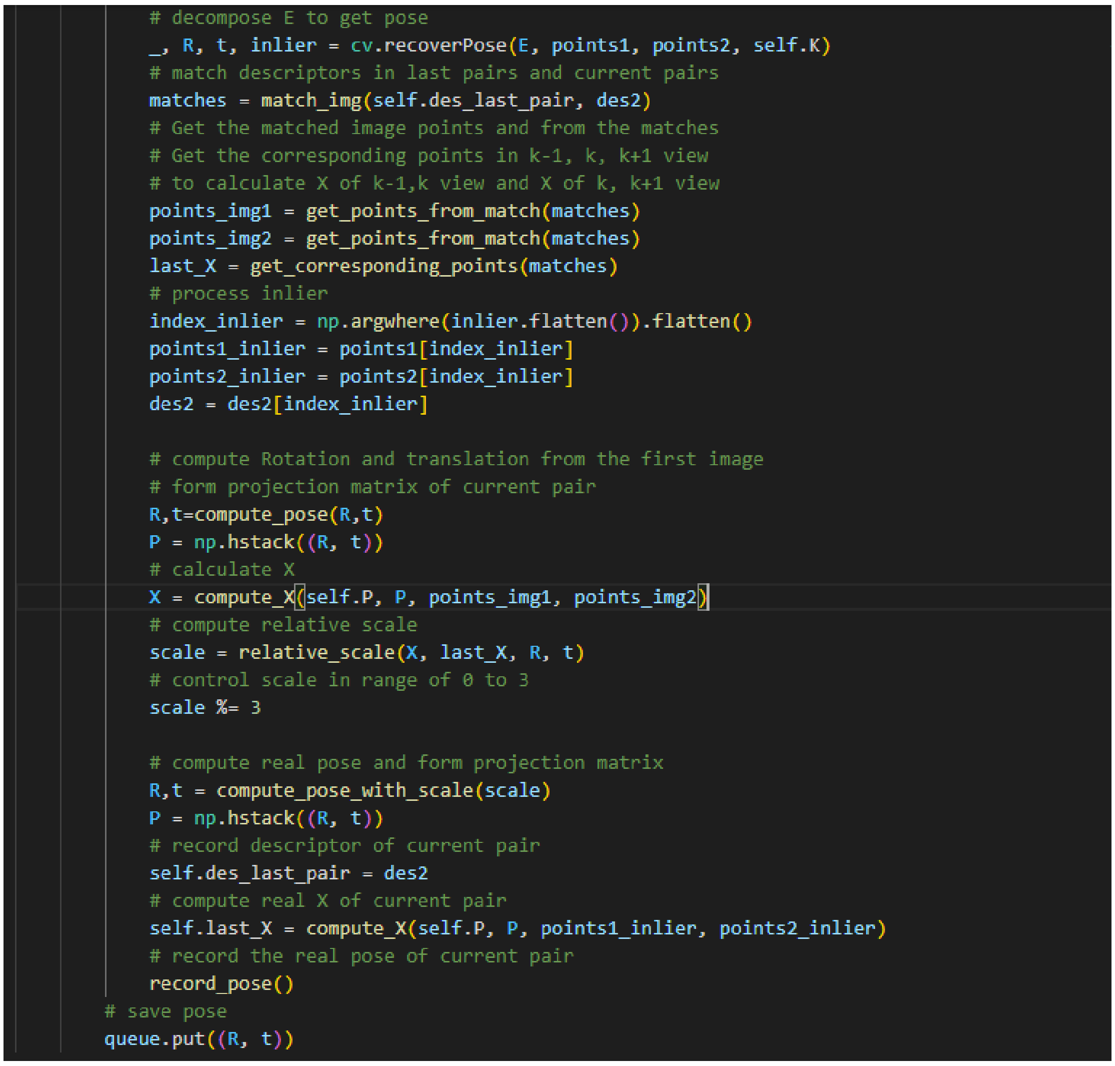
Pseudo code:



```
def process_frames(self, queue):
    # flag of the first pair
    first_time = True
    # run through each image
    for frame_path in self.frame_paths[1:]:
        # read images
        img = cv.imread(frame_path) # read img k+1
        img_query = cv.imread(self.frame_paths[self.frame_paths.index(frame_path)-1]) # read img k
        # TODO: compute camera pose here
        # Initiate ORB detector
        orb = cv.ORB_create()
        if(first_time == True):
            '''
            Omit the same process in else block
            only compute and record the X and Pose of the first pair
            P is projection matrix of the first pair
            '''
            # calculate X
            self.last_X = compute_X()
            # record the pose of the first pair
            record_pose()
            # finish first pair
            first_time = False

        else:
            # find keypoints and descriptors with ORB
            kp1, des1 = orb.detectAndCompute(img_query, None)
            kp2, des2 = orb.detectAndCompute(img, None)
            # match descriptors of current pairs
            matches = match_img(des1,des2)
            # Get the matched image points and from the matches
            points1 = get_points_from_match(kp1, matches)
            points2 = get_points_from_match(kp2, matches)
            # undistortion
            points1 = cv.undistortPoints(points1, self.K, self.dist, None, self.K)
            points2 = cv.undistortPoints(points2, self.K, self.dist, None, self.K)
            # find Essential matrix
            E, _ = cv.findEssentialMat(points1, points2, self.K)
```

M11152014 蘇珉淙



```
            # decompose E to get pose
            _, R, t, inlier = cv.recoverPose(E, points1, points2, self.K)
            # match descriptors in last pairs and current pairs
            matches = match_img(self.des_last_pair, des2)
            # Get the matched image points and from the matches
            # Get the corresponding points in k-1, k, k+1 view
            # to calculate X of k-1,k view and X of k, k+1 view
            points_img1 = get_points_from_match(matches)
            points_img2 = get_points_from_match(matches)
            last_X = get_corresponding_points(matches)
            # process inlier
            index_inlier = np.argwhere(inlier.flatten()).flatten()
            points1_inlier = points1[index_inlier]
            points2_inlier = points2[index_inlier]
            des2 = des2[index_inlier]

            # compute Rotation and translation from the first image
            # form projection matrix of current pair
            R,t=compute_pose(R,t)
            P = np.hstack((R, t))
            # calculate X
            X = compute_X(self.P, P, points_img1, points_img2)
            # compute relative scale
            scale = relative_scale(X, last_X, R, t)
            # control scale in range of 0 to 3
            scale %= 3

            # compute real pose and form projection matrix
            R,t = compute_pose_with_scale(scale)
            P = np.hstack((R, t))
            # record descriptor of current pair
            self.des_last_pair = des2
            # compute real X of current pair
            self.last_X = compute_X(self.P, P, points1_inlier, points2_inlier)
            # record the real pose of current pair
            record_pose()
        # save pose
        queue.put((R, t))
```

- Results Visualization

  按照算出來的 pose 依序與原始設定的起始位置之影像平面座標相乘，來還原 pose 的世界座標，並利用 Open3D 將還原的四個點與鏡心連線，再畫在顯示視窗中。

- Youtube link
  - Your video should be like the video shown in page 13

    https://youtu.be/Q11DWFpI7VI

- Please tell us how to execute your codes, including the package used and the environment

  Environment: Python 3.9

  使用下列 package

  open3d, numpy, opencv, sys, os, argparse, glob, multiprocessing

  How to execute:

  ```
  $ python .\camera_calibration.py .\calib_video.avi –show
  $ python .\vo.py .\frames
  ```