Report for Final Project

Camera Calibration

1. Calibration via OpenCV

(1) Steps

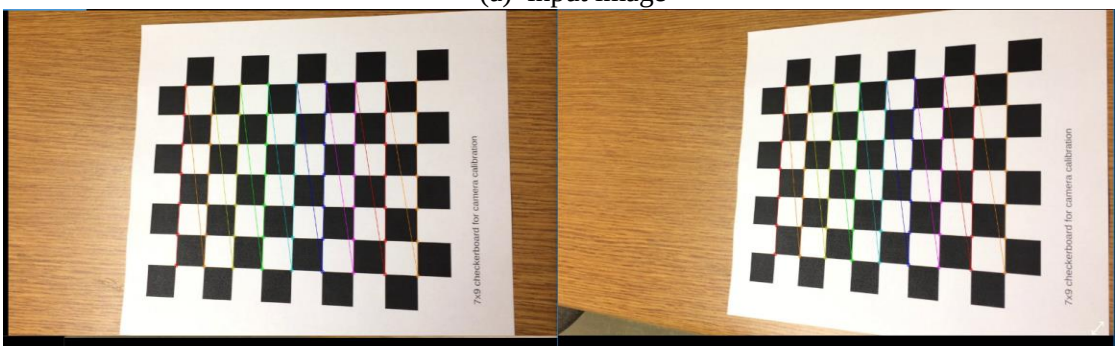
step1: call *findChessboardCorners* to extract chess board corners on the image

step2: call *cornerSubPix* to get sub-pixel corners, then reorder the extracted chess board corners

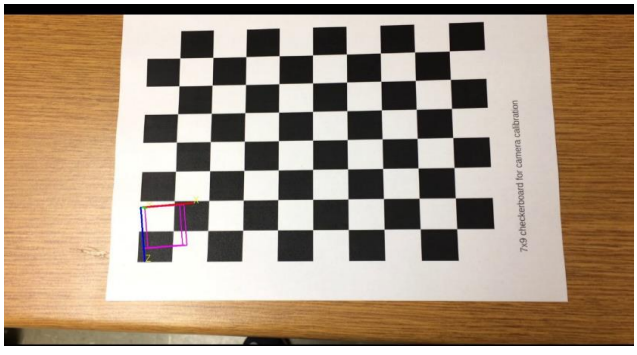
step3: call *calibration* to calibrate for the camera intrinsic parameters

(2) Results

(a) input image



(b) detected chess board corners



(c) project the 3D coordinate system onto the image

(3) Problems

However, after several trials, we found that the calibration result of OpenCV's *calibration* function is not stable, and sometime the result is weird. For example, the calibrated intrinsic matrix of the above chess board images is:

$$K = \begin{bmatrix} 51060.15 & 0.00 & 835.51 \\ 0.00 & 57065.60 & -280.13 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

which is obviously wrong. Thus we turn to the MATLAB Calibration Toolbox for a better solution.

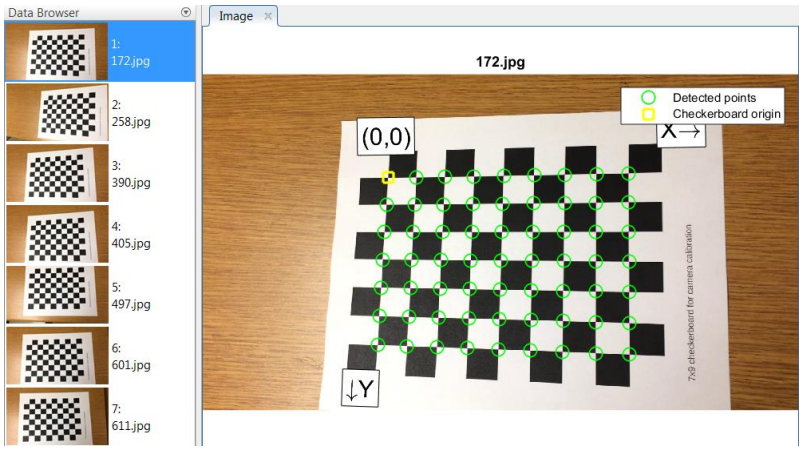
2. Calibration via MATLAB

(1) Steps

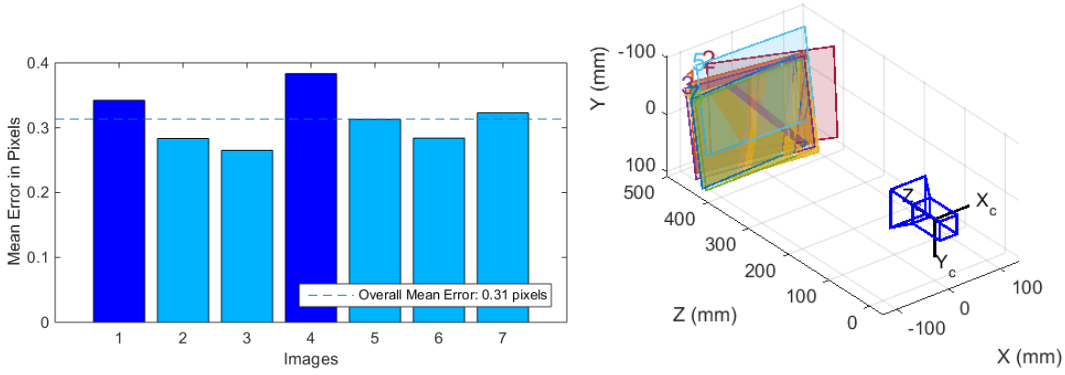
step1: run cameraCalibrator to call the GUI of MATLAB's calibration tool box

step2: follow the instruction step by step

(2) Results



(a) input images and detected chess board corners



(b) mean error and extrinsic results

And the estimated intrinsic matrix is:

$$K = \begin{bmatrix} 1827.60 & 0.00 & 961.95 \\ 0.00 & 1698.00 & 524.53 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

which is much more reasonable now.

3. Conclusion

MATLAB's calibration tool box is much more robust than that of OpenCV.

Visual Odometry

1. Basic idea

   The key issue of SLAM/SFM/VO is the data association between 3D structures in the world and their corresponding 2D observations on image, which is usually achieved by feature matching. For example, in ORB-SLAM, the orb features is used for matching. However, feature tracking methods can also be used to generate association. This project is designed to generate a VO framework based on feature tracking instead of matching.

2. Pipeline

   (1) Initialization

   1) Set the first tracker t_1 for the first frame (a tracker is associated with a key frame)

   2) Track each input frame in t_1, if less than 60% points are tracked, then set the current frame as the second tracker t_2, then generate initial 3D points via the tracked points.

   (2) Tracking

   1) For each new frame, track it in all the existing trackers $t_1, t_2, \dots, t_n$, to find out the matched points with each previous key frame. For those matched points without corresponding 3D points, denote them as $Pts2d$; and for those with 3D points, denote them as $Pts3d$.
   2) If a tracker tracks less than 50 $Pts2d$ and 10 $Pts3d$, we consider it failed, then remove the tracker.

   (3) Localization

   1) Using the $Pts3d$ and corresponding 2d points on the current frame, we can use PnP algorithm to calculate the pose of the current frame.

   (4) New tracker/key frame

   1) If the amount of $Pts3d$ on the newest tracker is less than 80% of that of the tracker, we consider adding a new tracker and a new key frame.
   2) Feature points are extracted on the new key frame via calling *goodFeaturesToTrack* in OpenCV, then the new tracker can be used for tracking the following frame.

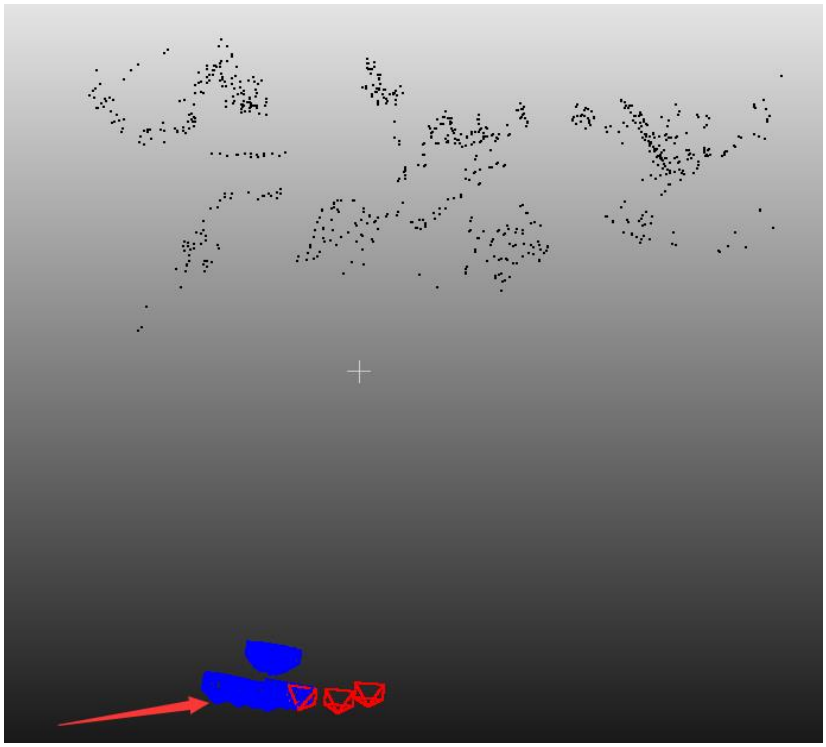
3. Results

   (1) Initial 3D points: frame 1 and frame 35

   We can see that the reconstructed 3D points from the 2D tracked points are generally correct and consistent with the real scene.

(2) Tracking: from frame 35 to frame 83

We can see that the tracked trajectory (the blue ones) of the camera is smooth and consistent with the key-frame locations (the red ones). The gaps between two trajectories are due to the bundle adjustment. Every time a new tracker/key frame is added, the bundle adjustment is applied on all the poses and the 3D points to optimize the solution, which may lead to the offset between posed before and after bundle adjustment.



(3) Result on 700 frames / 56 key frames

We can see that the sparse point cloud of the scene is generally correct, also the trajectory of the key frames (the red ones) is correct. However, due to the bundle

adjustment, tracked trajectory (the blue ones) of the camera is not stable, which is a bug that we will have to solve in the future work.