Exercise Sheet 4

Topic: SfM, Triangulation,
PnP, Bundle Adjustment

Yun-Jin Li

November 23, 2022

Par1: Map Initialization and Landmark Triangulation

See implementation on GitLab.

Part 2: Localizing cameras with PnP

See implementation on GitLab.

Part 3: Bundle Adjustment

According to the documentation on Ceres, the Huber loss is constructed as follows:

$$\rho(s) = \begin{cases} s, & \text{if } s \leq 1 \\ 2\sqrt{s} - 1, & \text{if } s > 1 \end{cases} \tag{1}$$

Since we know that

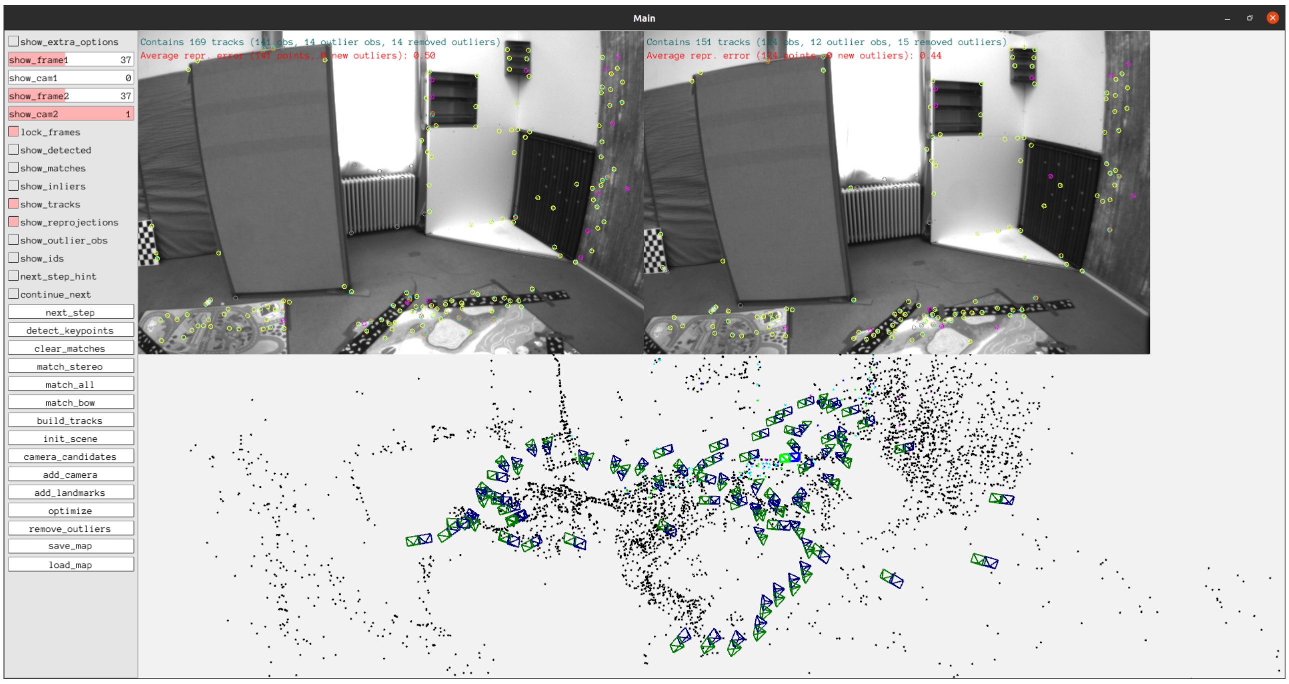
$$s = ||f_i(x_{i_1}, x_{i_2}, x_{i_3}, ..., x_{i_N})||^2, \text{where } f_i \text{ is the cost function.} \tag{2}$$

Based on the equation, we can conclude that if the square of the cost function is larger than 1, then $\rho(s) = 2|\ ||f_i(x_{i_1}, x_{i_2}, x_{i_3}, ..., x_{i_N})||\ | - 1$, whereas when the square of the cost function is smaller or equal than 1, then $\rho(s) = ||f_i(x_{i_1}, x_{i_2}, x_{i_3}, ..., x_{i_N})||^2$. In other words, when the square of the cost function is larger than a threshold, it follows the Mean Absolute Error(MAE); otherwise, it follows the Mean Squared Error (MSE). As a matter of fact, the performance of MSE would be strongly affected by the presence of outliers. However, this problem could be solved by MAE because it only takes the absolute value of the cost function, it wouldn't amplify the error resulting from the outliers. The Huber loss combines the advantages of both MSE and MAE by using MAE when the error is too large, and using MSE when the error is below the threshold so that we could still fit the result pretty nicely among the inliers. The reason why we should use the Huber loss here instead of in the calibration is that there wouldn't be so many outliers presented during calibration (especially RANSAC was also used in the optimization process in calibration); nevertheless, during bundle adjustment, we would encounter so many outliers and we don't want these outliers to affect our approximation too much, so we use Huber loss here.

Part 4: Outlier Filtering

Totally, there are 4 criteria to detect outliers, namely reprojection error much too large, reprojection error too large, distance to a camera too small, the z-coordinate in some camera frame too small. For the first two criteria, these outliers could result from some wrong matches of 2D correspondences. Imagine that we use the wrong matches to do triangulation, which basically means that we do not meet the condition of the 2D-2D correspondence for the triangulation. Undoubtedly, the landmark that we compute would be absolutely wrong; thus, it would give us large reprojection error. Based on the above reason, such kind of landmarks must be removed because they shouldn't be viewed as a "landmark" at all. For the last two criteria, the outliers could result from some sort of local minima during triangulation because we're basically trying to optimize a non-convex function. This kind of landmarks must be removed since in most of our camera models such as Double Sphere Model and Unified Camera Model, there is a parameter called ξ, and we assume that it is so small compared with the 3D point coordinate seen from the camera frame that we can neglect some of its effects in the projection formula. However, when either the distance or the z-coordinate is too small to the camera frame, we wouldn't meet this assumption. Thus, we have to remove them.

Part 5: Building a Map

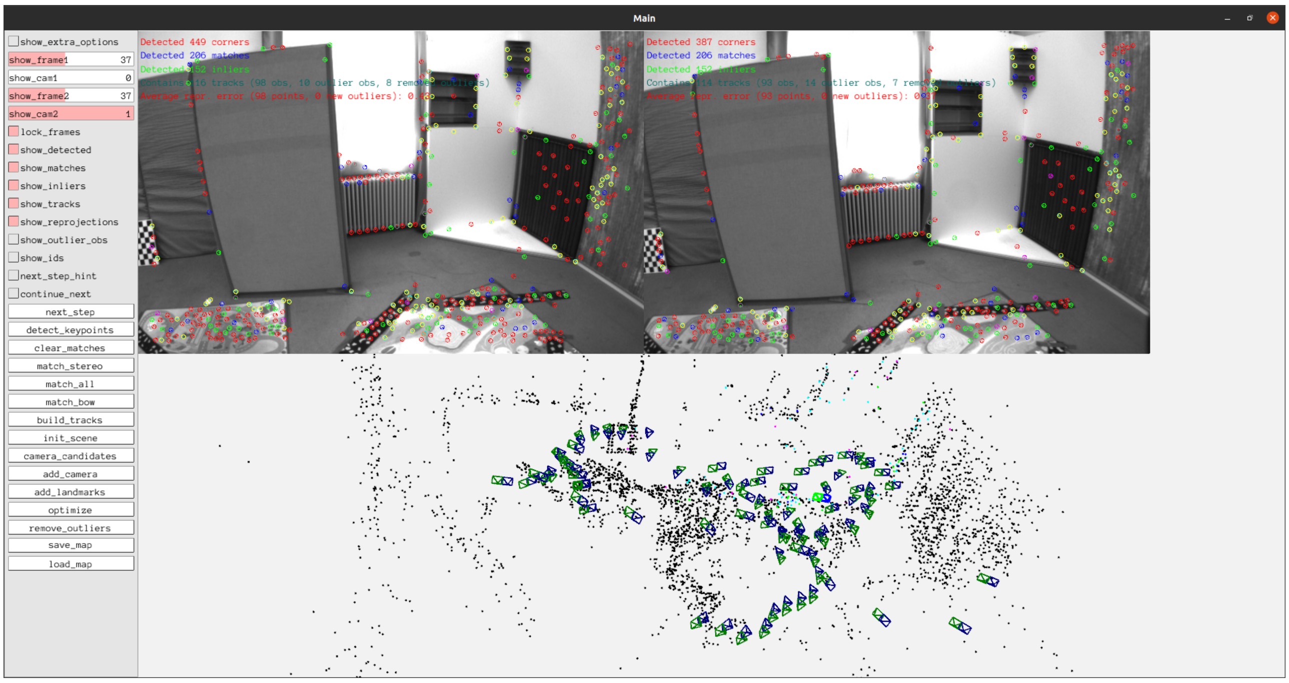


```
Contains 169 tracks (141 obs, 14 outlier obs, 14 removed outliers)
Average repr. error (141 points, 0 new outliers): 0.50
```

```
Contains 151 tracks (124 obs, 12 outlier obs, 15 removed outliers)
Average repr. error (124 points, 0 new outliers): 0.44
```

Figure 1: The screenshot of the completed map (match all)

- There are 164 cameras added to the map.
- Disable –show-gui and run the program for 5 times (see table 1), the average total execution time is 17.4689 seconds. The total execution time is calculated by the time difference between a time flag from the first line of the main function and the other time flag from



```
show_extra_options
show_frame1   37
show_cam1      0
show_frame2   37
show_cam2      1
lock_frames
show_detected
show_matches
show_inliers
show_tracks
show_reprojections
show_outlier_obs
show_ids
next_step_hint
continue_next
```

```
Detected 449 corners
Detected 206 matches
Detected 152 inliers
Contains 116 tracks (98 obs, 10 outlier obs, 8 removed outliers)
Average repr. error (98 points, 0 new outliers): 0.43
```

```
Detected 387 corners
Detected 206 matches
Detected 152 inliers
Contains 114 tracks (93 obs, 14 outlier obs, 7 removed outliers)
Average repr. error (93 points, 0 new outliers): 0.37
```

Figure 2: The screenshot of the completed map (match bow)

the last line of the main function. Each section's execution time is also calculated in a similar manner.

- According to table 1, bundle adjustment takes most of the time.
- Maybe we can reduce the maximum number of iterations from bundle adjustment or maybe try to introduce the stereo camera pair to compute the landmark by triangulation, which might get a more accurate position so that we won't spend too much time on the bundle adjustment.
- According to table 2 and 1, it's obvious to see that using match_bow() would give rise to a more efficient program. Furthermore, the result from match_bow() has 154 cameras, 4019 landmarks, and 19549 observations, while the result from match_all() has 164 cameras, 4718 landmarks, 24031 observations.

	Total(sec.)	BA(sec.)	PnP(sec.)	Triangulation(sec.)
1	17.6377	12.7044	1.01231	1.92807
2	17.6465	12.7276	0.995485	1.8434
3	16.4753	11.6374	1.00112	1.82589
4	17.7761	12.8981	0.996435	1.82603
5	17.8087	12.8475	1.02516	1.87167
Avg.	17.4689	12.5630	1.0061	1.8590

Table 1: Program total and each section in SfM pipeline execution time (match all)

	Total(sec.)	BA(sec.)	PnP(sec.)	Triangulation(sec.)
1	15.1858	11.0145	0.869929	1.24707
2	15.6854	11.3568	0.898483	1.35455
3	16.3915	12.0321	0.904596	1.34151
4	15.2733	10.9368	0.908227	1.33215
5	12.0759	8.33616	0.7488	1.0439
Avg.	14.9224	10.7353	0.8660	1.2638

Table 2: Program total and each section in SfM pipeline execution time (match bow)

References

As suggested in the exercise sheet.