Vision-based Navigation Exercise 4

Kerem Yildirir

May 2021

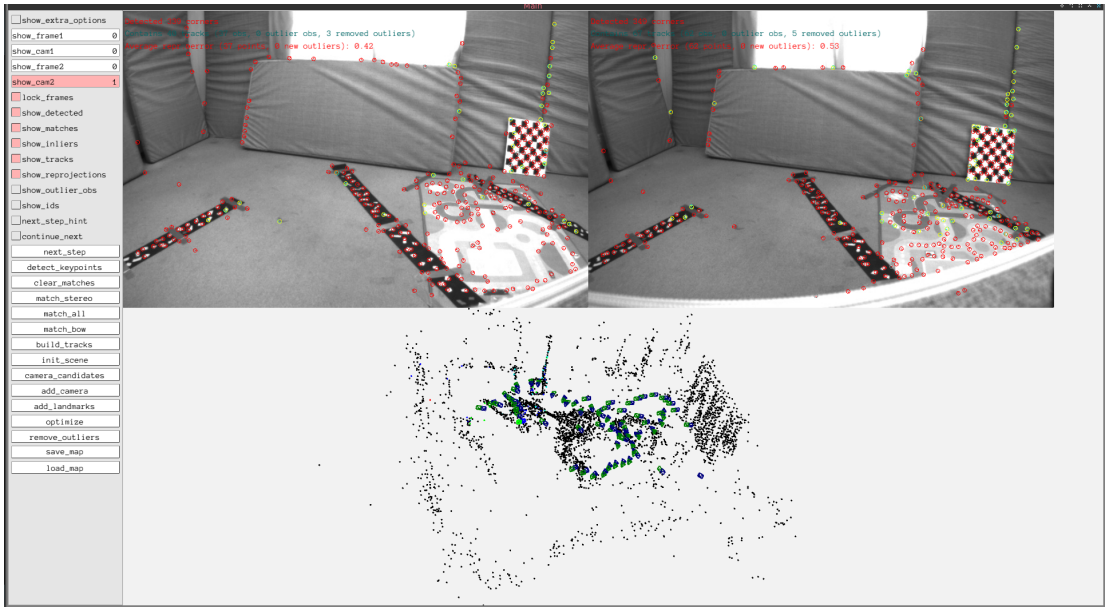
1 Part 3

We need a robust loss function like Huber loss where there we might encounter huge erroneous residuals during our optimization. What it does is, it thresholds the error at some point such that a residual with a huge error is considered an outlier and it's effect on the optimization is weighted down. For smaller values, it acts like a quadratic loss. In calibration, we had the 3D points as ground truth values, hence no outliers. However in bundle adjustment, we are optimizing them as well together with the pose. Since this is a more complicated problem, it produces more outlier residuals. Hence, it makes more sense to use it in bundle adjustment.

2 Part 4

The function checks the reprojection errors of projected landmarks to identify huge and normal outliers. Huge outliers are the ones which the reprojection error is significantly higher than normals, we really need to remove these, because they would affect the quality of the optimization as well. If there are no other specified types, checks for normal outliers with large reprojection errors. Then, checks for points too close to camera or points with too small z coordinate. For points too close to the camera and points with too small z coordinate, small movements will appear too large in the image and which will make it hard to track it, because one small change in the 3D location might mean a big movement pixelwise.

3 Part 5



Matching all frames and bundle adjustment optimization are the steps that takes the most time. Especially since we are adding more and more landmarks and observations, the problem size grows and make things slower, making optimization longer. Computing the whole map took me around 40 minutes. The system added 164 cameras, 4067 landmarks, 21256 observations to the map when used match_all flag. For optimization, I would suggest dropping very old frames, using bag of words instead of brute force matching. When I computed the matches with match_bow, since it uses less frames for matching, it is much faster than brute force matching and optimizing the map was also faster because it used less cameras and less observations to build the map, which makes the optimization problem smaller. I've created a map in around 30 minutes, and it had 145 cameras, 3064 landmarks, 15555 observations.