Coursework 3 of Robot Vision and Navigation

University College London

Coursework, Term 2 2022

Team: W

Yubing Wang 18019783
zcahanr@ucl.ac.uk

Ziwen Li 21039204
zczl625@ucl.ac.uk

18/04/2022

Literature review

1. Introduction

The letters SLAM indicates Simultaneous Localization and Mapping and it is a process where a mobile robot builds a map of the environment and computes its localization at the same time by using the map. [**1**] One of the most important module in the SLAM system is the place recognition that allows the system to do loop closure, which corrects the accumulated error of the model when the place recognition module detects that the sensor has returned to an already mapped area. Visual Slam System has a high popularity in the Computer Vision and Robotics field due to many reasons and one of the reason is that it can be easily experimented. In the SLAM system, we need a monocular camera is used to predict the surrounding environments and the position of vehicles. The convenience of the experiments can be achieved as its main sensor is only a monocular camera which is the cheapest sensor and it is also the smallest which makes the experiments even easier.

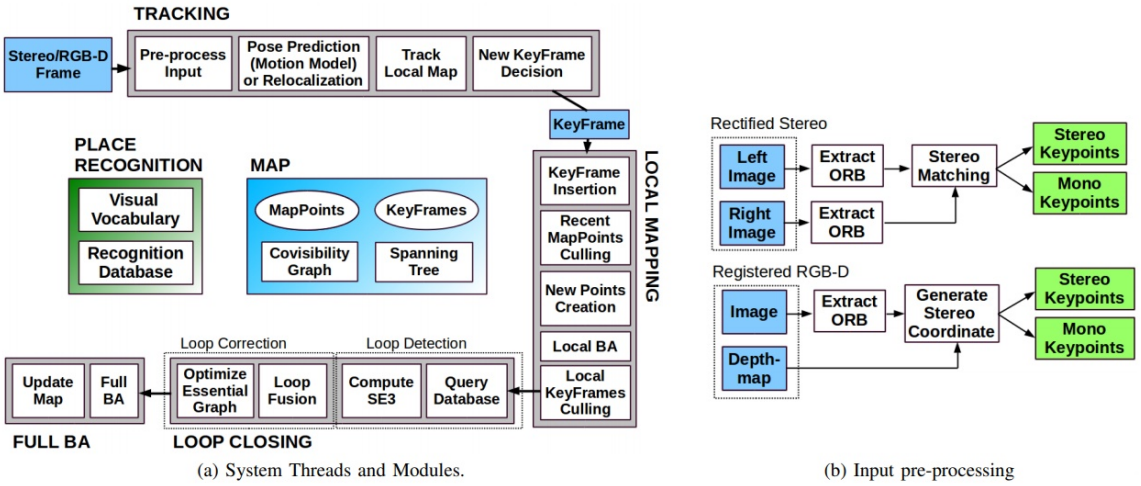
Monocular SLAM system has been researched for a long time, from the initial filtering approaches to the most modern key-frame based approaches. [**2**] PTAM (Parallel Tracking and Mapping) is still viewed as the gold standard in the Monocular Slam system.[**3**] The ORB-SLAM2 used in this report is based on the PTAM algorithm.

In this coursework, we will run the MONOCULAR ORB-SLAM system and do experiments about changing the following three important parts: 1. reducing the amounts of ORB-features; 2. turning off the outliers rejection stage; 3. turning off the loop closure stage.

2. Tracking thread

1. Overview of the ORB-SLAM system

The structure of the ORB-SLAM2 system is shown in Fig1. We can see that the system contains three main parallel threads: Tracking, Local Mapping(with local Bundle Adjustment) and Loop Closing. And an additional thread to perform a full BA is added after the overall optimization.



(a) System Threads and Modules. (b) Input pre-processing

Figure 1: ORB-SLAM system overview

The backbone of the ORB-SLAM system is the Tracking thread. It can communicate with other threads and arrange their work. [**4**] More specifically, the tracking thread is used to calculate the position of cameras with every frame and decide whether to insert another keyframe. [**5**] To achieve this purpose, there are four steps in the tracking module: input pre-processing, pose prediction, local map tracking and new keyframe decision. Next, I will show the details of implementing the tracking.

(1) Data Pre-Processing

Data pre-processing is a stage of ORB-SLAM2, which aims to extract features of keyframes. Then we can discard the original images (this can save lots of space and decrease the computational complexity) and only features extracted are used in the next steps. As shown in Fig1(b), our ORB-SLAM systems can deal with monocular and stereo key points.

Stereo keypoints: For stereo cameras, we can extract orb features for left and right images. For the left image, we set (u_L, v_L) as the coordinates and for the right image, we set u_R as the horizontal coordinate. For every orb feature in the left picture, we find the parts that match them in the picture on the right. If we assume stereo changes images, this step will be very efficient. We can generate the stereo keypoint with the coordinates of the left ORB $((u_L, v_L))$ and the horizontal coordinate of the right match (u_R). Preprocessing data from RGB-D cameras is slightly more cumbersome than stereo. First, we extract ORB features from RGB pictures. After that, we need to find depth values for every coordinate (u_L, v_L). The function is as followed:

$$u_R = u_L - \frac{f_x b}{d} \tag{1}$$

where f_x presents the horizontal focal length, b presents the baseline between the structured light projector and the infrared camera.[**8**] We measure the uncertainty of the depth sensor by calculating the uncertainty of the virtual right coordinate. After the pre-processing stage, features from stereo cameras and RGB-D cameras have the same format, so we can use the same method to handle them on the rest of the ORB-SLAM2 system.

Monocular keypoints only have two coordinates (u_L, v_L) and they are used when the stereo match fails or depth information is valid. These points are only triangulated from multiple

views [**2**]. They contribute to estimating translation and rotation but do not provide any scale information.

(2) Pose prediction or Relocalization

First, we should determine whether the tracking is successful for the last frame. If last time tracking is successful, we should assume the motion of vehicle is uniform linear motion. And we predict the pose of camera based on this assumption. and perform a guided search of the map points observed in the last frame. If the last tracking is unsuccessful, We should query the map points on a larger scale. After that, we optimize the pose through the found correspondences.

Second, if the tracking is lost, we should relocalize the keyframe in our recognition database. The method we use for this is the bag of words model. For finding the pose of the camera, we need to do RANSAC ition.s for every keyframe and calculate the pose by the PnP algorithm[**6**].If the camera poses found by the system can provide us enough information, which means there are adequate inliers in them, we will optimize the poses. After that, the system will also query more matches with the map points of the wider range of keyframes. After finishing all the above processes, we optimize the localization and direction of the camera again and judge whether we need to keep tracking.

(3) Local map tracking

After we predict poses of the cameras and find the feature matched, the system has the ability to project the map into the frame and search for more map point correspondences. As the size of the map grows, so does the complexity. So we need to limit the scope to a local map.

Keyframes set K_1 is contained in the local map and shares map points with the current frame. K_2 is a set of keyframes that are adjacent to keyframes in set K_1 in the graph. Besides, there exists $K_{ref} \in K_1$, sharing most map points. The algorithm to search keyframes set K_1 and K_2 is as followed [**10**]:

1)Compute the map point projection x_p in the current frame. And if x_p is out of the bounds of the image, we discard x_p.

2)Compute the cosine value between the current viewing ray $\hat{v}$ and the mean viewing of map point $\hat{n}$. If the inner product $\frac{(\hat{v},\hat{n})}{\|\hat{v}\|\|\hat{n}\|} < cos(\pi/3)$, then we discard it.

3)Compute the distance d between the centre of the camera and the map point. If $d \notin [d_{min}, d_{max}]$, which means d is out of the region of the map point, then we discard it.

4)Compute the value $\frac{d}{d_{min}}$, which is the scale value of the system.

5)Compare the representative descriptor D of the map point with those unmatched ORB features near x, then find the best match and associate it to D.

(4) New KeyFrame Decision

Now we come to the last step of tracking. We will decide whether convert the current frame to a keyframe. The system inserts a new keyframe if it meets the following 4 conditions:

1) More than 20 frames must have passed from the last global relocalization.

2) Local mapping is idle, or more than 20 frames have passed from the last keyframe insertion.

3) Current frame tracks at least 50 points.

4) Current frame tracks less than 90 percentage points than K_ref.

In this system, we impose a minimum visual change to avoid using a distance criterion to other keyframes such as PTAM(As shown in condition 4)[**11**]. Condition 1 ensure global relocalization contains enough frames. Condition 2 ensures the efficiency of inserting new keyframes. When the system is busy, a signal is sent to stop the local bundle when the system decides to insert a new keyframe. Condition 3 ensures that the current frame is good enough.
ORB-SLAM2 system uses the same strategy as monocular ORB-SLAM [**5**]. It inserts keyframes frequently, then culls excess keyframes. When our cameras are hard to extract features, such as the scene being far from our sensors, the ORB-SLAM2 system can insert new keyframes under the condition depending on the distance information. For example, if the amounts of tracked close points are less than some specific number, ORB-SLAM2 system can create new points and it will insert a new keyframe. The exact parameters will depend on the experiments we design and the environment we meet.

2. Exploit Information

In the rest of the chapter, we will give a detailed description of how to exploit information from cameras and how this information affects the system. This part is based on [**10**].

A. System Bootstrapping

As mentioned in the previous part of the reports, the ORB-SLAM system uses a stereo or RGB-D camera to get information. Compared to the monocular case, we do not need a specific structure to extract features from a single frame. When we get the first frame, we can create the first keyframe and set its pose to (0,0,0) and create a new map.

B. Bundle Adjustment

In the tracking thread, the ORB-SLAM2 system performs bundle adjustment [**7**] to optimize the camera pose (motion-only BA). Besides, the system also performs local BA in mapping thread and a full BA after loop closure, but we will not introduce them in this report.
We use motion-only BA to optimize the pose of the camera [**10**] which contains the camera orientation $O \in SO(3)$ and position $p \in R^3$ and use this information to minimize the reprojection error between keypoint $x^i_{(.)}$ and matched points $X^i \in R^3$ in world coordinates. They match the following equation:

$$\{O, p\} = arg \min_{O,p} \sum_i \rho(\|x^i_{(.)} - \pi_{(.)}(OX^i + p)\|^2_\Sigma) \tag{2}$$

where ρ is the robust Huber cost function[**12**] and Σ is the covariance matrix concerning kepoint's scale. The fuctions, $\pi_{(s)}$ (stereo projection fuction) and $\pi_{(m)}$ (monocular projection fuction) are defined by the following equations [**10**]:

$$\pi_{(s)} \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix} \tag{3}$$

$$\pi_{(m)} \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix} \tag{4}$$

where (f_x, f_y) is the focal length,(c_x, c_y) is the principal point and b is the baseline. We can get these information from calibration[**10**].

C. Bag of words Place Recognition.

A bags of words place recognition module is contained in the ORB-SLAM2 system, which is based on DBoW2 [**8**]. Visual vocabulary is the discretization of descriptor space. Cameras get the images and extract ORB features from the picture and the visual vocabulary is created based on these features. If we get enough images, we can use the same vocabulary in different environments. As shown in [**9**], this strategy can get a satisfactory performance. There is an inverted index contained in the ORB-SLAM system, which can help us query the keyframes efficiently. As mentioned in the section New KeyFrame Decision, we update keyframes frequently. So the inverted index is updated with the insertion of new keyframes and the deletion of old keyframes.

The DBoW2 algorithm [**8**] considers it is difficult to distinguish two adjacent keyframes (because there exists an overlap between them), so it adds up similar frames. However, this algorithm has a significant drawback: it did not distinguish keyframes inserted at the same place but at different times. An efficient way to improve the model is by grouping these keyframes which are connected in the covisibility graph. [**5**]

Another benefit is also shown in [**8**]. We can speed up querying by constraining the brute force matching to features belonging to the same nodes. We can also detect outliers to ensure a coherent rotation by refining the correspondences with an orientation consistency test. [**9**]

Experiments

For the experiments, we will run the MONOCULAR ORB-SLAM system on two different sequences under four different conditions. The sequences we chose are KITTI 07 and TUM3, where KITTI 07 is a dataset about a vehicle moving on the road and TUM3 is a dataset about a camera moving around a desk. We believe sequences from different datasets can reveal more about the MONOCULAR ORB-SLAM system.

We will first experiment with off-the-shelf options(EXP1) and then try with fewer features(EXP2), try without the outlier rejection stage(EXP3), try without loop closing(EXP4).

Since the results produced by the MONOCULAR ORB-SLAM system have some randomness, we will run each experiment at least 3-5 times to find a more general result, and we will present the most general or representative results.

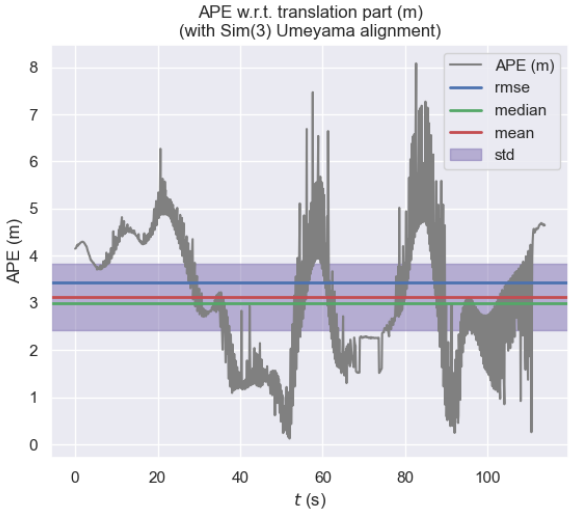
EXP1: Run the system with off-the-shelf options

We will be using the original `KITTI04-12.yaml` for sequence KITTI 07 and the original `TUM3.yaml` for sequence TUM3.

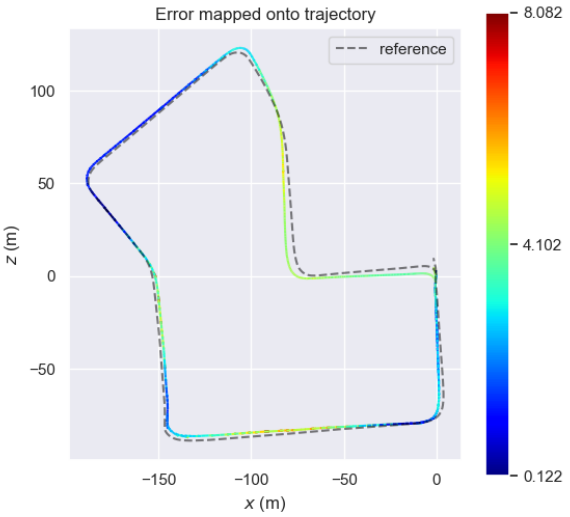
Sequence: KITTI 07

In Figure 2(b), we can see that the trajectory is almost the same as the reference line, which shows that the ORB-SLAM system with this original configuration is good when solving this sequence KITTI 07. On the map, half of the trajectory is blue and the other half is green. We can't observe any near-red parts. All of this shows that the errors are quite small. Also, we may observe that the errors are relatively high during the turns. This could be caused by the fact that when the vehicle took a turn, the keyframes during the turn have fewer same features.

Another evidence of the good performance would be the APE plot. Considering that this dataset is filmed on road, this APE has a relatively low maximum (8 meters), mean(about 3.1 meters) and root mean square error(about 3.5 meters). We can see that there are several big waves which correspond to the turning points in the trajectory map, so the gradient of APE changes sign when the vehicle changes direction. There are also many small fluctuations which can be caused by high feature numbers that could lead to high variance and overfitting. Except for the fluctuations, we may observe a small plateau. From the video, we can see that the vehicle was waiting to turn, so it was stationary for a few seconds.



(a) KITTI07 — APE w.r.t translation part (m)



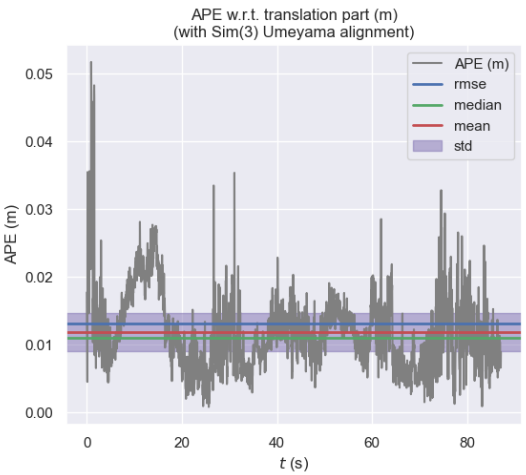
(b) KITTI07 — Error mapped onto trajectory

Figure 2: Result on KITTI07 Sequence

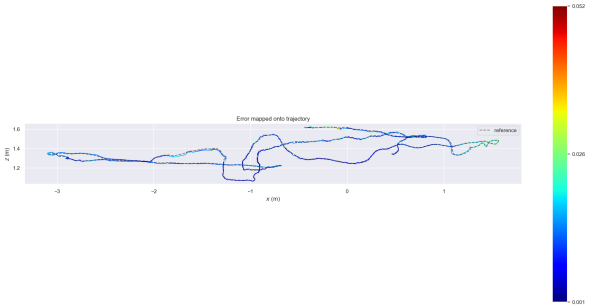
Sequence: TUM3

This TUM sequence is filmed inside an office and is around a desk. So compared to the KITTI 07 sequence, this TUM sequence has data with a way smaller scale. Hence, the error mapped onto the trajectory has a very small scale. Also, due to the small movements, many keyframes may contain many same features. So a higher precision and overall lower APEs are obtained(a maximum around 0.051, a mean around 0.011, a root mean square error around 0.012).

Also, since the video is filmed by a camera held by a human, the camera is unstable, especially compared to the vehicle in KITTI 07 sequence that drives stably. This is observable in Figure 3(b) and a lot more fluctuations in the APE plot (Figure 3(a)) that construct an unstable trend may also prove this. Otherwise, the plots can be explained similarly to the plots for sequence KITTI 07(the goodness of the model and the waves in the APE, etc).



(a) TUM3 — APE w.r.t translation part (m)



(b) TUM3 — Error mapped onto trajectory

Figure 3: Result on TUM3 Sequence

EXP2: Reduce the number of ORB features

For the KITTI07 sequence, we try with 1100, 1300 and 1500 ORB-features, while in the original case we use 2000 ORB-features. For the TUM3 sequence, we try with 600, 700 and 800 ORB features, while in the original case we use 1000 ORB-features.

Sequence: KITTI 07

In order to fully explore the effect of the number of ORB-features, we experimented with many cases. As in the 1000 ORB-features case, the system wasn't able to initialise the map and lost tracking for the whole time in all the experiments we try and it is unable to produce a plot, we present the case with 1100, 1300 and 1500 ORB-features. We did not choose an ORB-features number higher than 1500 as it performs greatly like the original 2000 ORB-features case and a higher number may not show the effect obviously.

Case 1 with 1100 ORB-features

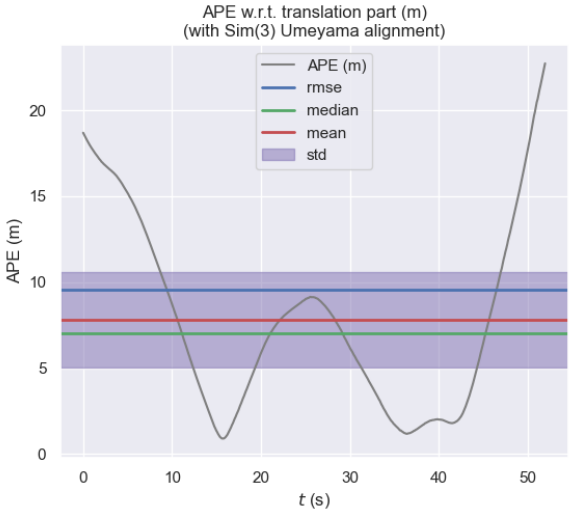
When we only have 1100 ORB-features, the system lost track of a part of the trajectory as the features are not enough for the images to match. Hence, we can only see part of the trajectory and the reference on the map plot, while the APE plot only shows APE for around 55 seconds. As the only part of the map is plotted and they can be plotted with a lower number of ORB-features, the APE is only doubled compared to the original 2000 ORB-features case. However, the shape of the APE is a U-shape and the line is much smoother as no loop closure was performed due to the fact that there is no scenario in which the same features were captured. (more explanation about the loop closure will be presented in Experiment 4)

Case 2 with 1300 ORB-features

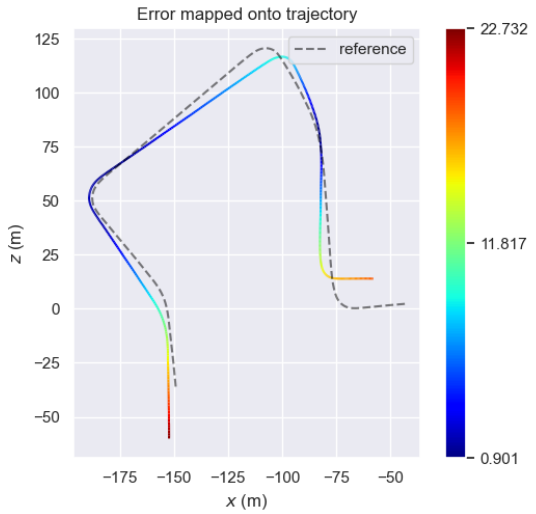
When we have 1300 ORB-feature, there is only a small part where the system lost track and that can be observed from the map plot. Also, the APE plot now shows APE for more than 100 seconds. We may find it surprising that we have a higher maximum APE of around 48 compared to the 18 in Case 1. The reason could be that in case 2 we have a much longer trajectory and its error accumulates more and still without the loop closure, the trajectory won't be corrected. Hence, the beginning point and the endpoint of the APE will have a way larger APE and this is also what we observed from the APE plot (U-shape and smoother line).

Case 3 with 1500 ORB-features

When we have 1500 ORB-features, the plots are almost the same as the original case with 2000 ORB-features, though the APE is slightly higher in the plots presented. In general, the trajectory fits the reference well. However, we observe some circumstances that the system lost track of when we are experimenting with this case many times and it's more frequently than in the original case.

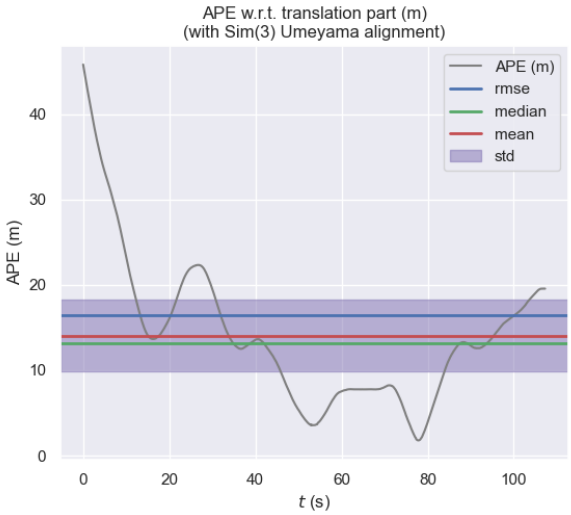


(a) KITTI07 — APE w.r.t translation part (m)

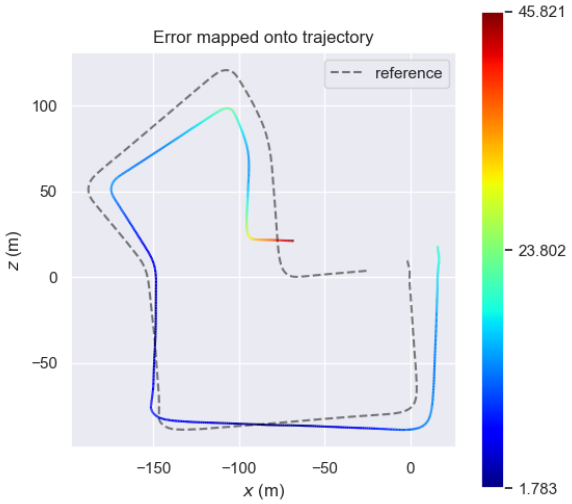


(b) KITTI07 — Error mapped onto trajectory

Figure 4: Result on KITTI07 Sequence

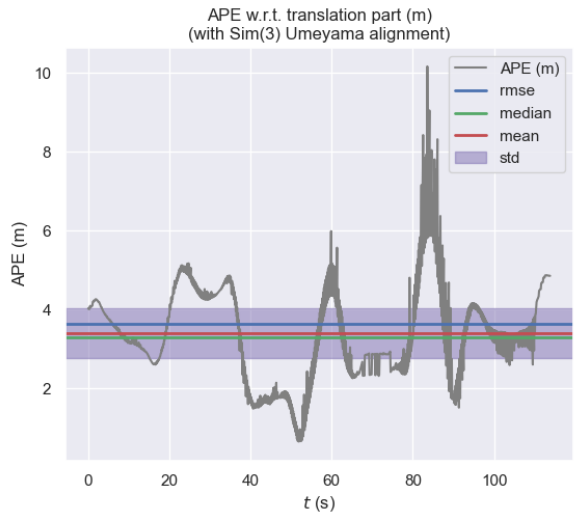


(a) KITTI07 — APE w.r.t translation part (m)

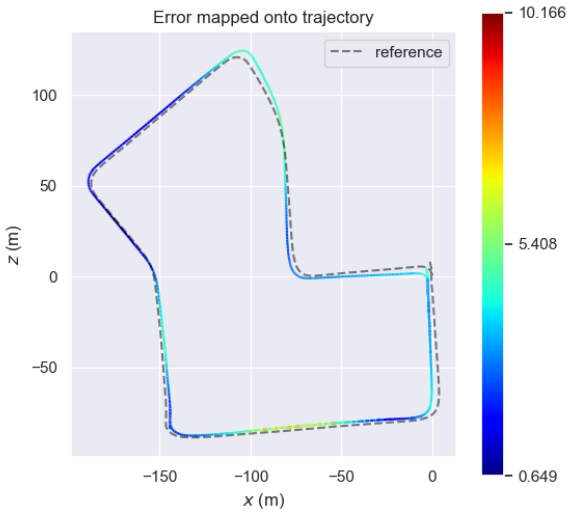


(b) KITTI07 — Error mapped onto trajectory

Figure 5: Result on KITTI07 Sequence



(a) KITTI07 — APE w.r.t translation part (m)



(b) KITTI07 — Error mapped onto trajectory

Figure 6: Result on KITTI07 Sequence

Sequence: TUM3

Similarly, we experimented with many cases. The smallest number we tried is 500 and the plots can't be produced. The figures we present are the case with 600, 700 and 800 ORB-features. We also did not choose an ORB-features number higher than 800 and it performs well as the original case with 1000 ORB-features.

Case 1 with 600 ORB-features

In the 600 ORB-features case, the system lost track of almost half of the trajectory, which can be observed from the map plot. And the APE plot only shows APE for around 36 seconds. Since the data is collected around a desk, the data doesn't carry much information and it is also why we use fewer ORB-features than the KITTI07 sequence. Hence, the APE is also very low and the reason stated in the KITTI07 case 1 is also fitted. However, the shape of the APE is not a U-shape and the line is not smoother like the KITTI07 case. The reason might be that the handheld camera provides unstable data and the trajectory is too short and the error is too small to show a clear trend (so even without the loop closure, the APE doesn't show the U-shape) and the unstable nature also causes the fluctuations in the APE plots.

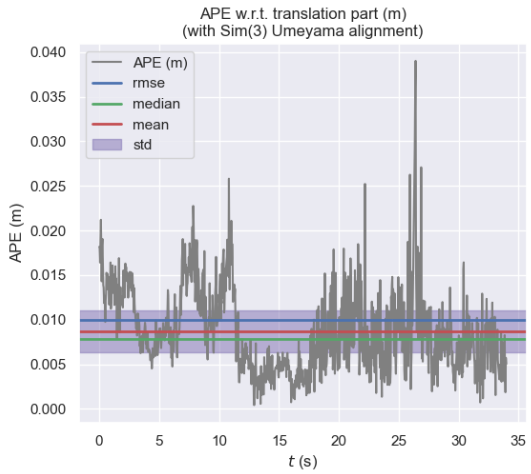
Case 2 with 700 ORB-features

In the 700 ORB-features case, the system lost track of a small part of the trajectory, which can be seen from the map plot (the line from the upper middle part to the upper right corner is missing). And the APE plot now shows APE for more than 80 seconds. Like what we explained in the KITTI07 case 2, the APE increases for a bit. However, the U-shape that wasn't observable is now presented in the APE plot. The reason that the U-shape is now very obvious might be that there are two points on the map that is very close to each other and if the line is connected together, loop closure will be performed. And the highest points in the APE points should correspond to the two points on the map.

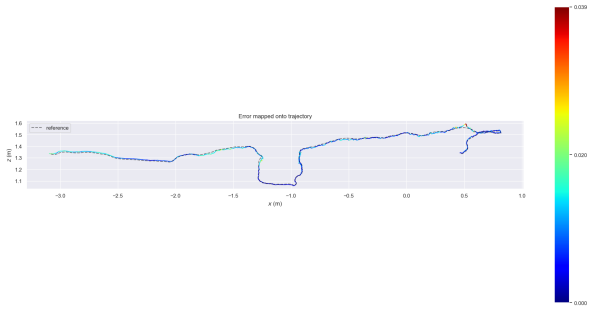
Case 3 with 800 ORB-features

For the 800 ORB-features case, the plots are almost the same as the original case with 1000 ORB-features, and surprisingly the APE is slightly lower in the plots presented. In general, the

trajectory fits the reference well and due to the APE plots, we can even say that it performs better than the original case. However, the plots we present may not speak for the whole case as it has randomness. If this case is indeed better than the original case, the reason could be that with more features, the model has a higher variance and tends to overfitting if the number of ORB-features is really high.

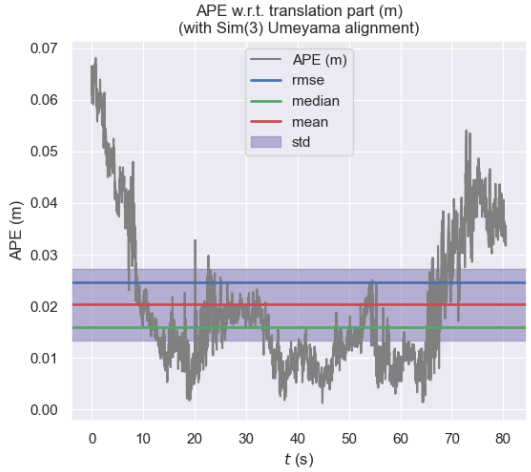


(a) TUM3 — APE w.r.t translation part (m)

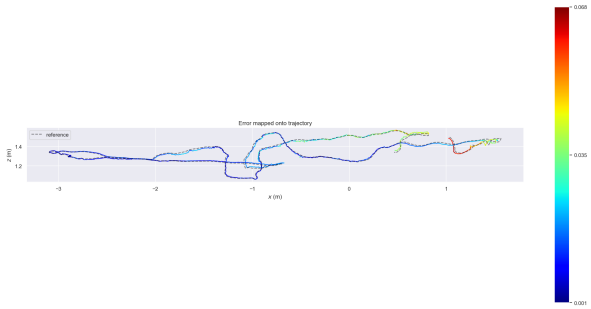


(b) TUM3 — Error mapped onto trajectory

Figure 7: Result on TUM3 Sequence

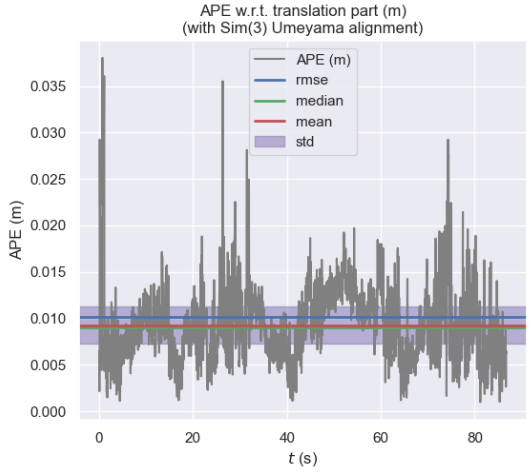


(a) TUM3 — APE w.r.t translation part (m)

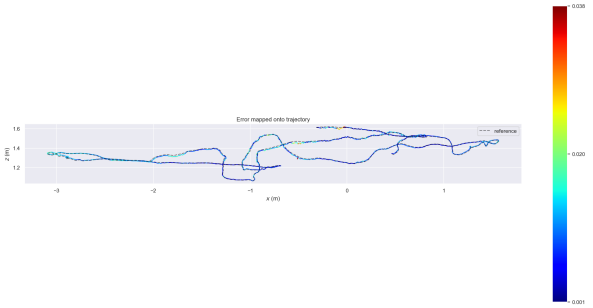


(b) TUM3 — Error mapped onto trajectory

Figure 8: Result on TUM3 Sequence



(a) TUM3 — APE w.r.t translation part (m)



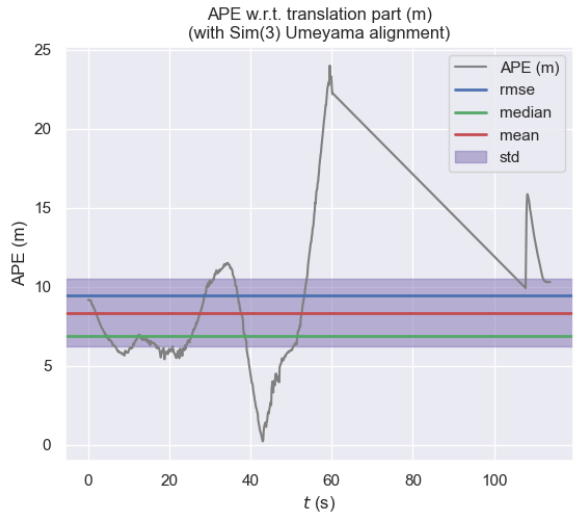
(b) TUM3 — Error mapped onto trajectory

Figure 9: Result on TUM3 Sequence

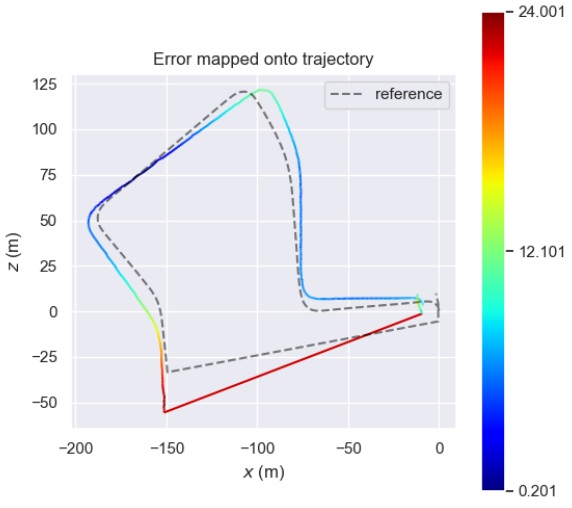
EXP3: Turn off the outlier rejection stage

From the APE plots, we can see when we turn off the outlier rejection stage, there exists some large increases or decreases due to the outlier. And unlike the usual fluctuated line, the part of the line, where large increases or decreases appear, looks linear in both plots. In the map plots, the shape of the reference has changed, especially observable in the KITTI07 sequence plot. But more importantly, the trajectory fitted poorly with the reference, since the outliers cause the estimates to diverge from the ground truth and the loop closing corrects the trajectories in the end. For the TUM3 map plot, it's more obvious that the trajectory is poor as it looks not fit for most of the line.

Sequence: KITTI 07



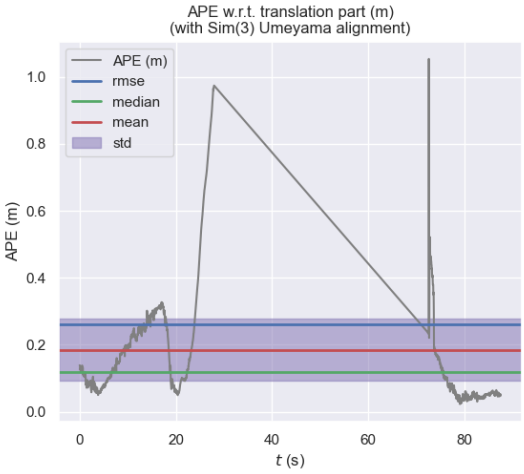
(a) KITTI07 — APE w.r.t translation part (m)



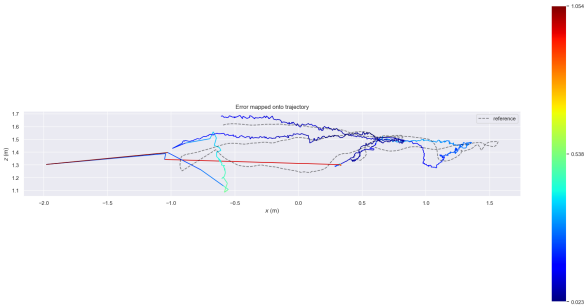
(b) KITTI07 — Error mapped onto trajectory

Figure 10: Result on KITTI07 Sequence

Sequence: TUM3



(a) TUM3 — APE w.r.t translation part (m)



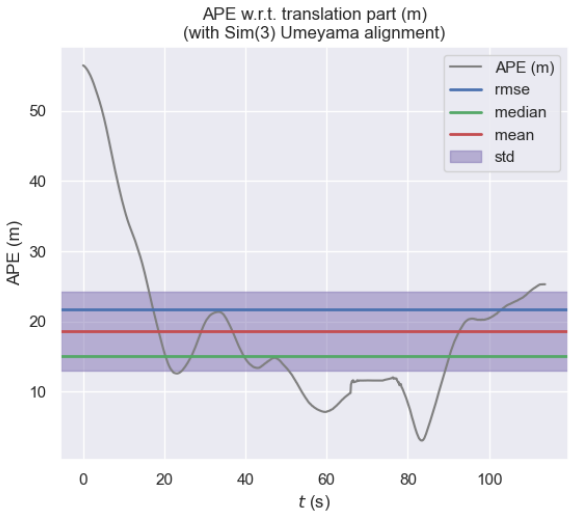
(b) TUM3 — Error mapped onto trajectory

Figure 11: Result on TUM3 Sequence

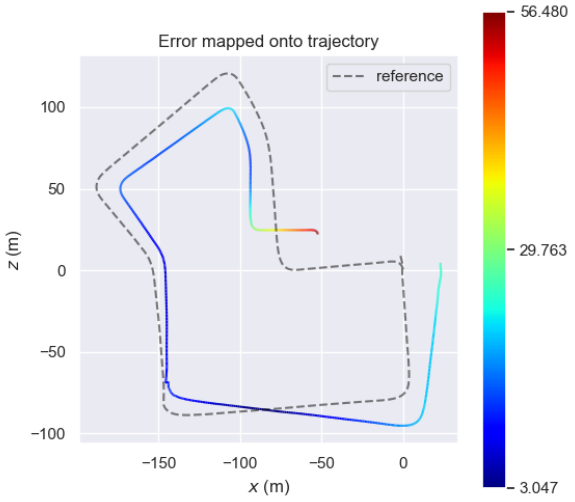
EXP4: Turn off the loop closure

Since the loop closure is turned off, the trajectories will no longer be corrected at the end of the journey. From the map plots, we can see that the trajectory is not closed as the error accumulates and without loop closure, the endpoint of the trajectory and the start point of the trajectory won't be the same as the reference. The map for TUM3 is not that obvious and the reason might be that the camera is handheld and not stable. However, we can observe that the APE increases slightly though it is not comparable with the KITTI07 APE, which has a maximum of around 58 (6 times more than the original case in Experiment 1). Also, the effect of turning off the loop closure is observable in the shape of the APE plots, where they present a U-shape that the beginning and the end part have a higher APE and it decreases in the middle. Except for the general shape, the KITTI07 has a more smooth line and the TUM3 has a more fluctuated line, which might due to the nature that the TUM3 data was collected with a handheld camera and it is not stable.

Sequence: KITTI 07



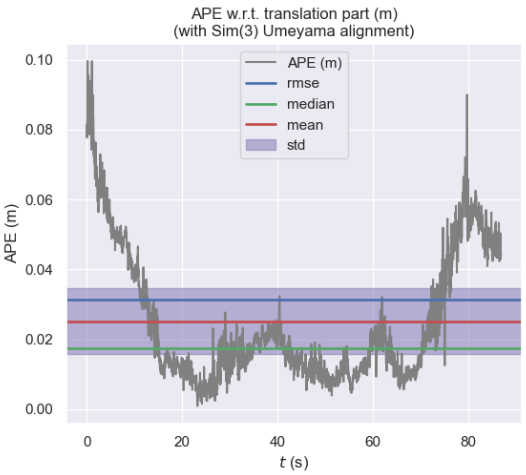
(a) KITTI07 — APE w.r.t translation part (m)



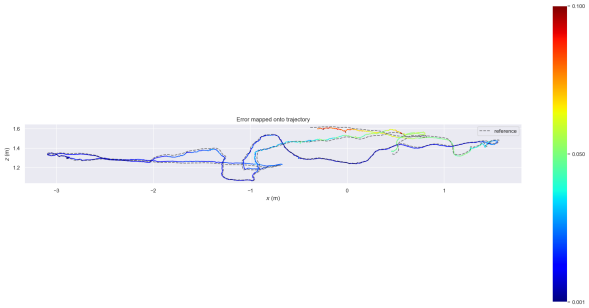
(b) KITTI07 — Error mapped onto trajectory

Figure 12: Result on KITTI07 Sequence

Sequence: TUM3



(a) TUM3 — APE w.r.t translation part (m)



(b) TUM3 — Error mapped onto trajectory

Figure 13: Result on TUM3 Sequence

Conclusion

We describe how the monocular SLAM system work and highlight the tracking thread in chapter 1. And in chapter 2, we implement experiments when off-the-shelf options, reduce the number of ORB features, turn off the outlier rejection stage and turn off the loop closure and evaluate the performance of the ORB-SLAM system .

The ORB-SLAM system has three main steps, which are tracking, local mapping and loop closing. A fourth thread is performing a full bundle thread after the pose-graph optimization, to compute the optimal structure and motion solution. Among the four threads, we introduce the tracking thread in detail. There are four steps in implementing tracking: data pre-processing,

map initialisation, pose estimation, local map tracking and new keyframe decision. We discuss mostly the pose estimation and the map tracking, where its aim is to localize the camera with every frame by finding feature matches and minimizing the re-projection error by motion-only bundle thread.

According to the results of four experiments, we can see the performance of ORB-SLAM is highly dependent on parameters. We work on both sequence KITTI 07 and sequence TUM3 and the results are completely different. Therefore, if we want to get better results from ORB-SLAM, we need to select appropriate parameters according to the environment. Besides, we can learn from experiments 3 and 4 that the outlier rejection stage and loop closure play important roles in the ORB-SLAM system in both environments. It is essential to keep them in the ORB-SLAM system.

Appendices

Listing 1: changes made in each experiments

```
Case 1: No change


Case 2:
For KITTI 07 sequence:
In the file Refactored_ORB_SLAM2\Install\etc\orbslam2\Monocular
KITTI04-12.yaml:
Line 29: ORBextractor.nFeatures: 2000
# Change 2000 to 1100, 1300 and 1500 (3 levels)

For TUM3 sequence:
In the folder Refactored_ORB_SLAM2\Install\etc\orbslam2\Monocular
TUM3.yaml:
Line 29: ORBextractor.nFeatures: 1000
# Change 2000 to 600, 700 and 800 (3 levels)


Case 3:
In the folder Refactored_ORB_SLAM2\Source\Libraries\ORB_SLAM2\src
Optimiser.cc:
1. Comment line 385-392 and add the following as line 394 395:
      pFrame->mvbOutlier[idx] = false;
      e->setLevel(0);

This is the revised code from line 385 to 395:
    //  if (chi2 > chi2Mono[it]) {
    //    pFrame->mvbOutlier[idx] = true;
    //    e->setLevel(1);
    //    nBad++;
    //  } else {
    //    pFrame->mvbOutlier[idx] = false;
    //    e->setLevel(0);
    //  }

      pFrame->mvbOutlier[idx] = false;
      e->setLevel(0);

2. Comment line 412-419 and add the following as line 421 422:
      e->setLevel(0);
      pFrame->mvbOutlier[idx] = false;

This is the revised code from line 412 to 422:
    //  if (chi2 > chi2Stereo[it]) {
    //    pFrame->mvbOutlier[idx] = true;
    //    e->setLevel(1);
    //    nBad++;
    //  } else {
    //    e->setLevel(0);
    //    pFrame->mvbOutlier[idx] = false;
```

```
    //  }

      e->setLevel(0);
      pFrame->mvbOutlier[idx] = false;
```

Case 4:
In the folder Refactored_ORB_SLAM2\Source\Libraries\ORB_SLAM2\src
LocalMapping.cc:
Comment line 80:

```
      // mpLoopCloser->InsertKeyFrame(mpCurrentKeyFrame);
```

Tracking.cc:
Comment line 1389:

```
  // mpLoopClosing->RequestReset();
```

System.cc:
1. Comment line 146 147 148 159 162 164 165:
This is the revised code from line 145 to 165:

```
  // Initialize the Loop Closing thread and launch
  // mpLoopCloser = new LoopClosing(mpMap, mpKeyFrameDatabase, mpVocabulary,
  //                                mSensor!=MONOCULAR);
  // mptLoopClosing = new thread(&ORB_SLAM2::LoopClosing::Run, mpLoopCloser);

  // Initialize the Viewer thread
  if(bUseViewer){
    mpViewer = new Viewer(this, mpFrameDrawer, mpMapDrawer, mpTracker,
                          strSettingsFile);
    mpTracker->SetViewer(mpViewer);
  }

  // Set pointers between threads
  mpTracker->SetLocalMapper(mpLocalMapper);
  // mpTracker->SetLoopClosing(mpLoopCloser);

  mpLocalMapper->SetTracker(mpTracker);
  // mpLocalMapper->SetLoopCloser(mpLoopCloser);

  // mpLoopCloser->SetTracker(mpTracker);
  // mpLoopCloser->SetLocalMapper(mpLocalMapper);
```

2. Comment line 338 346 347 **and** add the following as line 348:

```
  while(!mpLocalMapper->isFinished()){
```

This is the revised code from line 336 to 350:

```
void System::Shutdown()
{
  mpLocalMapper->RequestFinish();
  // mpLoopCloser->RequestFinish();
  if(mpViewer)
  {
    mpViewer->RequestFinish();
    while(!mpViewer->isFinished())
```

```
    this_thread::sleep_for(chrono::milliseconds(5));
}

// Wait until all thread have effectively stopped
// while (!mpLocalMapper->isFinished() || !mpLoopCloser->isFinished() ||
//        mpLoopCloser->isRunningGBA()) {
while (!mpLocalMapper->isFinished()){
  this_thread::sleep_for(chrono::milliseconds(1));
}
```

Bibliography

[1] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," in IEEE Robotics Automation Magazine, vol. 13, no. 3, pp. 108-117, Sept. 2006, doi: 10.1109/MRA.2006.1678144.

[2] Mur-Artal, R., Tardós, J. D. (2014, July). ORB-SLAM: tracking and mapping recognizable features. In Workshop on Multi View Geometry in Robotics (MVIGRO)-RSS (Vol. 2014, p. 2).

[3] Mur-Artal, R., Tardós, J. D. (2014). Orb-slam: Tracking and mapping recognizable. Work. Multi VIew Geom. Robot.-RSS 2014.

[4] Daoud, H. A., Md. Sabri, A. Q., Loo, C. K., Mansoor, A. M. (2018). SLAMM: Visual monocular SLAM with continuous mapping using multiple maps. PloS one, 13(4), e0195878.

[5] Mur-Artal, R., Montiel, J. M. M., Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE transactions on robotics, 31(5), 1147-1163.

[6] Lepetit, V., Moreno-Noguer, F., Fua, P. (2009). Epnp: An accurate o (n) solution to the pnp problem. International journal of computer vision, 81(2), 155-166.

[7] Triggs, B., McLauchlan, P. F., Hartley, R. I., Fitzgibbon, A. W. (1999, September). Bundle adjustment—a modern synthesis. In International workshop on vision algorithms (pp. 298-372). Springer, Berlin, Heidelberg.

[8] Gálvez-López, D., Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics, 28(5), 1188-1197.

[9] Mur-Artal, R., Tardós, J. D. (2014, May). Fast relocalisation and loop closing in keyframe-based SLAM. In 2014 IEEE International Conference on Robotics and Automation (ICRA) (pp. 846-853). IEEE.

[10] Mur-Artal, R., Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE transactions on robotics, 33(5), 1255-1262.

[11] Pire, T., Fischer, T., Castro, G., De Cristóforis, P., Civera, J., Berlles, J. J. (2017). S-PTAM: Stereo parallel tracking and mapping. Robotics and Autonomous Systems, 93, 27-42.

[12] Petrus, P. (1999). Robust Huber adaptive filter. IEEE Transactions on Signal Processing, 47(4), 1129-1133.