# Implementing ORB-SLAM

Practical Course: Vision-based Navigation WS2019/20

Anselm Coogan, Sean Wang

March 14, 2020

## 1 Introduction

Simultaneous Localization and Mapping (SLAM) is used by robots to map their environment in real-time and localize themselves in it simultaneously. Originally, SLAM was developed using LiDARs [2], however, the advent of small, mobile robots (i.e. drones) has required computational systems that rely on less expensive and heavy sensory equipment. This need was met with Visual SLAM - a SLAM system based on a monocular or stereo RGB camera setup. One of the groundbreaking Visual SLAM systems is ORB-SLAM (OS) which was originally proposed in 2015 [6] and further developed in [7]. The main contributions of OS are feature-reusing throughout the entire pipeline, local optimization, real-time capable tracking, mapping, loop closing, robust initialization and localization, and techniques for compact map representation to allow for long-term operation. The illumination, scale, rotation and translation invariant ORB features[8] are used for tracking, mapping, relocalization and loop closing - the main SLAM steps. Local pose tracking and map optimization are done through a novel, so-called Covisibility Graph. A subset of this graph, the Essential Graph, is exploited for real-time capable loop-closing. For monocular systems - with which the original OS paper concerned itself - the authors propose a heuristic algorithm for scene initialization that produces a unique, robust result. Furthermore, for scene recognition they propose a Bag of Words (BoW) [4] approach. In our project we extended the stereo Visual Odometry (VO) system developed throughout the semester to an OS-like setup. Concretely, we implemented the Covisibility Graph used in tracking and mapping and the Essential Graph used in loop closing. Additionally, instead of the BoW approach of OS, we designed a place recognition system based on pose similarity and 3D-point correspondences. Finally, we adhered to the original authors suggestions for removing redundant keyframes whilst adding new keyframes more frequently than in the VO system. This added functionality results in increased computational time, though notwithstanding we were able to keep the system running in real-time at 20 Hz. Overall, our system was compared with its parent - the VO implementation - on the EUROC V1 1 dataset [1] and showed improved map consistency and comparable pose and hence trajectory estimates. The implementation can be found on Gitlab.[1]

## 2 System Overview

As noted, the presented system was built based on and inspired by [6] and [7], the main differences being that the original papers consider loop closures as disjunct from map building (operating in a separate thread), and that they use a more complex method for detecting loop closures - namely BoW versus pose similarity. The map representation is made up of 3D points in space, termed landmarks, keyframes (camera poses and associated landmark observations), and a Covisibility Graph connecting camera poses with each other based on similarity of viewpoints, see Figure 1. A high-level overview of the system flow is depicted in Figure 2. It is divided into two main components: I) tracking and II) mapping with possible loop closure. The tracking task consists of extracting features from the image frames, projecting a local map onto the current camera, matching existing landmarks in the map with detected features, and finally estimating the camera

[1] https://gitlab.vision.in.tum.de/visnav_ws19/w0031/visnav.git, branch: develop

pose based on these matches. This is done on a per-frame basis. If a frame is to be added as a new keyframe because it contains enough new information, the mapping part of the system is activated. It is made up of several parts: first, it adds new observations of existing landmarks and creates new landmarks based on feature matches between the current left and right frames. Then it necessarily adds the current frame as a keyframe and updates the covisibility graph. This is followed by detecting whether a loop closure can be performed. Subsequently the local map is optimized using Bundle Adjustment (BA) [10] and finally, redundant keyframes are removed. The following sections discuss the methods (those that are different from VO) in more detail.

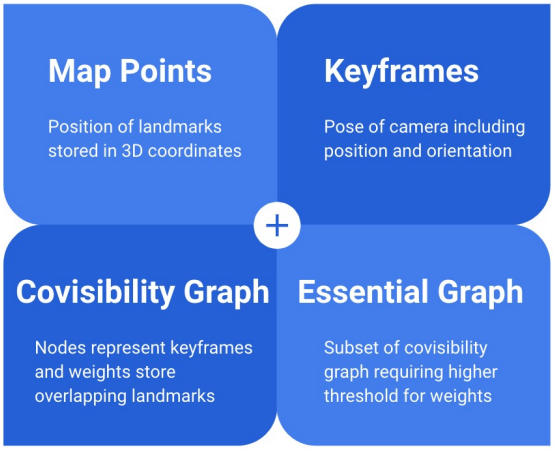


Figure 1: The map is represented by landmarks (or map points), keyframes, and the Covisibility and Essential Graphs


Figure 2: The system consists of two main parts, tracking and mapping. Tracking is done for every frame whereas mapping is done only if a frame is added as a keyframe

# 3 Tracking

Odometry tracking is done for every frame. The pose estimate is initialized through the previous frames pose. Further pose estimation is conducted using ORB-features found in the left frame. Features detected in the right frame are only used in mapping. However, since feature detection is computationally intensive, ORB-features are computed for both left and right frame concurrently, so that if mapping is concluded, the detected features are available without further computation by the mapping part. 2000 features are computed per frame.

## 3.1 Covisibility Graph

The Covisibility Graph is one of the central contributions by the original Orb-SLAM authors. In contrast to visual odometry where tracking (and mapping) is done based on temporal proximity, the Covisibility Graph considers spatial closeness. So instead of doing tracking and mapping

by exploiting information only from the n previous keyframes and their associated landmarks, keyframes with similar scenes and poses are selected. This is achieved by constructing a weighted, undirected graph where keyframes are nodes and an edge $e_{ij}$ between two keyframes $k_i$ and $k_j$ exists if they have a similar view, i.e. share a minimum amount of landmark observations $\theta_{min} = 30$. If an edge $e_{ij}$ exists, its weight is equal to the number of shared landmarks. See Equations 1 and 2 where $obs_i$ are the unique landmark IDs observed by keyframe $i$.

$$\widehat{e}_{ij} = \sum_{obs_i} \sum_{obs_j} \mathbb{1}_{obs_i = obs_j} \tag{1}$$

$$e_{ij} = \begin{cases} \widehat{e}_{ij}, & \text{if } \widehat{e}_{ij} \geq \theta_{min} \\ 0 & \text{else} \end{cases} \tag{2}$$

A visualization of the graph can be seen in Figure 3. The Covisibility Graph is used to construct a local map in which the current frame can be localized. A subset of this graph, the Essential Graph, is used later on in the system when propagating pose updates during loop closure. An edge in the Essential Graph exists if the number of shared landmarks is equal to or greater than $\theta^{EG}_{min} \geq \theta_{min}$, in our case $\theta^{EG}_{min} = 60$. A graphical representation is given in Figure 4.

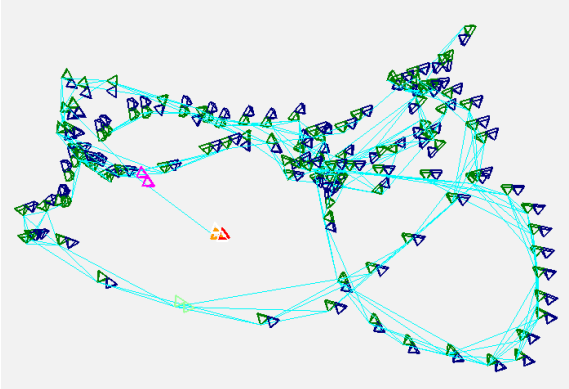

Figure 3: The Covisibility Graph. Nodes are keyframes and edges between keyframes with number of shared observations exceeding $\theta_{\text{min}}$.

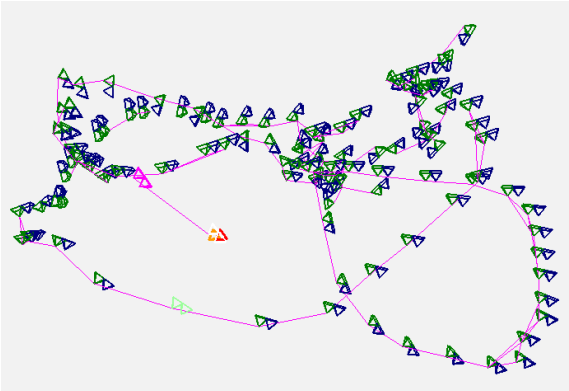

Figure 4: The Essential Graph. A subset of the Covisibility Graph. For an edge to exist, keyframes must share more than $\theta^{\text{EG}}_{\text{min}}$ landmark observations.

## 3.2 Local Map

To generate a local map, two steps are necessary. First, the landmarks seen by the most recent keyframe are projected into the current frame and matched with the current frames ORB-features.

Based on this matching, the current pose is calculated in an initial estimation. Second, the matches are used to find the keyframe(s) $k_1$ that share at least $k_1^{min} = 10$ landmark observations with the current frame. Then, the local map is created by combining all landmarks seen by the keyframes in $k_1$ and by the set of neighbors of $k_1$ in the Covisibility Graph. This local map is then projected into the current frame, and the current pose is estimated based on matched ORB-features. A graphical abstraction of the local map is given in Figure 5.

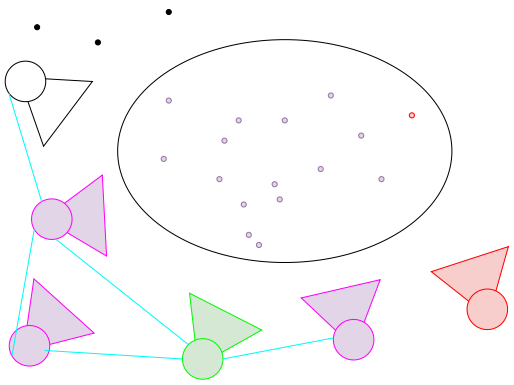

Figure 5: The local map used to localize the current frame. The current frame is highlighted in red, the (single) $k_1$ frame is in green, $k_1$'s neighbors in the Covisibility Graph are in purple, and a not-included keyframe is given in white. The shared point between $k_1$ and the current frame is red and the local map consists of the red point and the points seen by the neighbors, highlighted in purple.

### 3.3 Keyframe Decision

Tracking is completed by deciding whether to add the current frame as a keyframe. Overall, the high-level goal is to add only keyframes with enough new information. If the mapping thread is currently busy, the frame is discarded regardless of its potential value. If it is not, the frame is selected as a keyframe if either of the following conditions hold:

- there were more than $m_{min} = 130$ matches with the local map and more than $f_{min} = 50\%$ features of the frame are unmatched
- the current number of keyframes is less than or equal to $kf_{min} = 5$ ,
- more than $kf_{max} = 40$ frames have passed since the last keyframe insertion

The first part of the first condition ensures good tracking, whereas the second part asserts that only keyframes with new information are being added. However, this can result to losing track if too rapid changes occur. To combat this, the second condition is introduced: if too much time has passed since the last insertion (and hence tracking was lost because condition one didn't hold) the keyframe is inserted regardless. Finally, the third conditions ensures that aggressive keyframe removal, especially in the beginning, doesn't become a problem.

## 4 Mapping

If a frame was selected to be added as a keyframe, the mapping part of the system comes into play. First, feature descriptors between the left and right frame are matched. As previously mentioned, ORB-feature detection for the right image is done concurrently to tracking so that both feature sets are available at this point. New landmarks are added based on stereo matches that have not yet been associated with an existing landmark in the preceding tracking step and observations of existing landmarks are added as well. In order to disregard outliers as quickly as possible, the distance to the left camera needs to be between $d_{min} = 0.1m$ and $d_{max} = 10m$. Once this is

```
FOR keyframe in keyframes
    SET overlapCounter to 0
    FOR landmark in landmarks of keyframe:
        IF observationCount of landmark > 3 THEN
           INCREMENT overlapCounter
        END IF
    END FOR
    SET overlapPercentage to (DIVIDE overlapCounter by (SIZE of landmarks))
    IF overlapPercentage > 0.9 THEN
       REMOVE keyframe from keyframes
    END IF
END FOR
```

Algorithm 1: The algorithm used for keyframe removal.

completed, further mapping steps, namely redundant keyframe removal, loop closure and local bundle adjustment, are computed in a separate thread.

## 4.1 Keyframe Removal

Keyframes with redundant information produce two undesirable effects. First, they unnecessarily inflate the systems memory footprint. Second, the redundant keyframes' pose estimates are likely to have an adverse outcome on pose and map estimation. Thus, keyframes are removed if more than $k_{max}^{overlap} = 90\%$ of the landmarks they observe, are also seen by at least three other keyframes. For better understanding, pseudo-code for this procedure is given in Algorithm 1.

## 4.2 Loop Closure

Every estimation step will introduce an error - however slight it may be. As time progresses, the error will accumulate and cause a problem known as drift, as exemplified in Figure 6. This error is present in both the map and the pose estimates. A technique known as loop closure is used to combat this. The main idea is to recognize a previously explored place and then correct the error

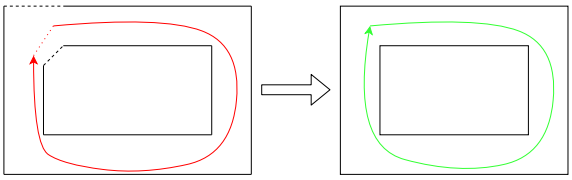

Figure 6: Accumulated error in map and trajectory estimation known as drift seen on the left. To combat this, previously seen places (e.g. landmarks) need to be recognized so that they aren't added as new landmarks. Once this happens, drift can be corrected via loop closure (right).

that has accumulated in between viewings. In OS this is achieved by recognizing places based on the BoW feature representations between frames. Our system uses a simpler, but possibly more intuitive, method based on pose similarity. In the first step, possible loop closure candidates are identified. This is done by first computing the absolute pose difference $p$ between the current keyframe and all its neighbors in the covisibility graph as given by Equation 3, where $A$ and $B$ are the poses, $I$ is the identity matrix and $\| * \|_{\mathrm{F}}$ is the Frobenius norm.

$$p = \|(A^{-1}B) - I\|_{\mathrm{F}} \tag{3}$$

The largest pose difference is then compared to the pose differences between the current keyframe and all other non-neighboring keyframes; keyframes with smaller pose differences are considered as loop closure candidates. For each candidate, its ORB descriptors are matched with those of the current keyframe - only features that are associated with an existing landmark are considered. This

```
CREATE queue
ADD currentFrame to queue
ADD neighbors of currentFrame to queue

CREATE visited
ADD currentFrame to visited
ADD neighbors of currentFrame to visited

WHILE queue not empty DO
      SET currentNode to queue.popFront()
      FOR neighbor of neighbors in essentialGraph of currentNode
          IF neighbor in visited
             CONTINUE WHILE
          END IF
          SET relativePose to RELATIVE_POSE between neighbor and currentNode
          SET absolutePose of neighbor to currentNode.pose TIMES (INVERSE of relativePose)
          ADD neighbor to visited
          ADD neighbor to queue
      END FOR
END WHILE
```

Algorithm 2: Breadth-First Search applied to pose update propagation via the Essential Graph.

results in 3D-3D correspondences and the resulting point clouds are aligned using RANSAC [3]. If this alignment is supported by enough inliers $n_{min}^{inliers} = 12$ and it has more inliers than its competing loop closure candidates, it is accepted as a loop closure keyframe. Closing a loop is composed of two steps. First, landmarks are merged: from the matched 3D points between the loop closure frame and the current keyframe the both observations are counted towards the associated landmark of the loop closure frame. This landmark gets new observations from the current keyframe and its neighbors in the Covisbility Graph that previously observed the erroneous landmark. The second part is correcting the drift. This is done by updating the pose of the current keyframe based on the new landmark matches and then propagating this result using Breadth-First Search (BFS) [5] through the Essential Graph. The algorithm is given in Algorithm 2. This feature is accomplished by keeping the relative poses between landmarks constant and only updating their poses w.r.t. the world coordinate system. However, the pose update is done separately from loop closure detection after the map has been optimized with local BA and in the main thread (not the separate mapping thread).

### 4.3 Local Bundle Adjustment

After redundant keyframes have been removed and a possible loop closure has been found, the map and keyframe poses are optimized using Bundle Adjustment (BA). Again, the Covisibility Graph is used to ensure spatially local map optimization. The optimization is thus done on the current keyframe and its neighbors in the Covisibility Graph plus all landmarks observed by these keyframes. Non-neighboring Keyframes that observe any of the optimized landmarks are also included but kept fixed during optimization.

## 5 Evaluation

In the scope of this project, the developed system was evaluated on the first route of the EUROC V1 dataset [1] and compared against the precedingly developed Visual Odometry system. This was done on an eight-core 2.3 GHz Intel Core i9 processor running with 32 GB RAM.

## 5.1 Real-time Capability

The system should be able to run in real-time. For the considered dataset, this evaluates to a framerate of 20Hz or 50ms of computation time per frame. On average, our system is able to achieve this, running at 30.48 per frame. However, around 1% of frames exceed the real-time threshold, with single spikes at up to 61.0 ms. The most computationally intensive tasks are ORB feature extraction and local map localization with maximum runtime values of 26.1ms and 30.0ms, respectively. Hence, significant speed-up is achieved by running ORB-feature extraction in multiple threads. Optimizing the ORB-feature extraction algorithm and the data structures and routines used throughout the system should be able to bring down the worst-case runtimes so that the real-time constraint is achieved for all frames. A detailed timing analysis can be found in Table 1.

Table 1: Timing analysis for the different functionalities of our system ($\sigma$ is the standard deviation, $\sigma^2$ the variance). A distinction is made between the main thread that processes ever frame, a separate thread for ORB-feature extraction in the right frame that is also run every frame, and finally the separate mapping thread. Note that only the main subtasks are considered.

| Step | Avg. t [ms] | min. t [ms] | max. t [ms] | $\sigma$ [ms] | $\sigma^2$ [ms$^2$] |
|---|---|---|---|---|---|
| **MAIN THREAD** | **3.05E1** | **1.93E-2** | **6.10E1** | **7.99** | **6.39E1** |
| BFS pose update | 1.83E-2 | 4.76E-2 | 4.70E-4 | 1.07E-2 | 1.13E-4 |
| *Tracking* | *2.90E1* | *1.14E1* | *5.70E1* | *7.42* | *5.50E1* |
| ORB (left) | 1.35E1 | 6.82 | 2.6E10 | 2.85 | 8.14 |
| Tracking (prev. landmarks) | 4.72 | 1.44E1 | 1.90E-2 | 1.77 | 3.13 |
| Tracking (local map) | 1.07E1 | 3.57E-3 | 3.00E1 | 4.49 | 2.02E1 |
| Keyframe decision | 2.33E-4 | 1.04E-3 | 1.18E-4 | 9.60E-5 | 9.21E-9 |
| ORB (right) (separate thread) | 1.34E1 | 2.66E1 | 6.70 | 2.90 | 8.35 |
| *Mapping* | *8.19* | *9.39E-1* | *2.01E1* | *4.27* | *1.82E1* |
| Stereo ORB matching | 1.57 | 1.39E-1 | 1.30E1 | 1.77 | 3.12 |
| Adding landmarks & observations | 4.21E-1 | 7.03E-2 | 1.22 | 2.82E-1 | 8.00E-2 |
| Adding keyframe | 1.36 | 4.88E-2 | 3.04 | 7.44E-1 | 5.54E-1 |
| **MAPPING THREAD** | **8.74E1** | **2.14E-1** | **2.67E2** | **4.88E1** | **2.38E2** |
| Detecting loop closure | 6.27 | 3.90E-3 | 1.57E2 | 2.20E1 | 4.87E2 |
| Local bundle adjustment | 6.77E1 | 1.46E-1 | 1.85E2 | 3.74E1 | 1.40E3 |
| Removing redundant keyframes | 8.63 | 5.16E-4 | 2.04E1 | 5.14 | 2.64E1 |

## 5.2 Map Consistency

Map consistency is somewhat more difficult to evaluate quantitatively as no ground truth map of the environment exists. Hence, other heuristics are considered. Since the scene under consideration is indoors and does not exhibit long hallways, one such heuristic are the distances of the map points to the origin (the first frame pose) and the underlying distribution. Both systems create a similar amount of points (OS: 21563, VO: 20630). Additionally, the most extreme outliers have similar distances of $2.07 \cdot 10^7$ and $2.49 \cdot 10^7$ for OS and VO, respectively. Furthermore, due to these similarities the distributions are very similar on the surface as well with standard deviations being $2.28 \cdot 10^5$ and $2.50 \cdot 10^5$, variances of $5.21 \cdot 10^{10}$ and $6.26 \cdot 10^{10}$, and medians lying at 4.32 and 4.24, for OS and VO, respectively. However, as indicated by the slightly higher variance and standard deviation of the VO system, our system exhibits fewer outliers. Again, defining outliers is a bit tricky with no ground truth available. However, qualitatively inspecting the (non-symmetric) map makes it possible to consider points with distances greater than $6.9m$ to be outliers and points with distances greater than $50m$ to be extreme outliers. For these metrics our system shows significant improvement with 7 vs. 17 extreme outliers and 95 vs. 125 regular outliers (note that these contain the extreme outliers). A layover of both maps with highlighted outliers depicts this difference in Figures 7 and 8.

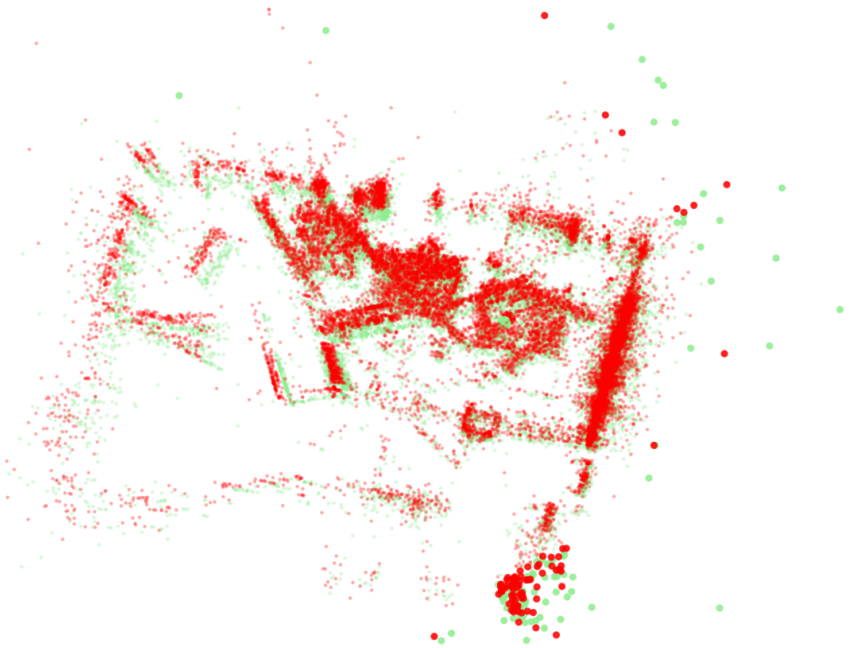

Figure 7: The resulting maps from the EUROC V1 1 dataset for our system (red) and VO (lightgreen) seen in a birds-eye view. Outliers are marked in bold in the respective colors.

Figure 8: The resulting maps from the EUROC V1 1 dataset for our system (red) and VO (lightgreen) as viewed from the side. Outliers are marked in bold in the respective colors.

## 5.3 Trajectory Accuracy

For the trajectory a ground truth exists, and hence, evaluation is more straightforward. Two common metrics to evaluate trajectory estimates are the Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) [9]. ATE is a metric to calculate global consistency of the estimated trajectory. It is computed by first aligning the estimated trajectory $E$ with the ground truth trajectory $G$ through least-squares minimization. The resulting rigid body translation $T$ is then used to compute the remaining error between the trajectories. The formal definition is given in 4[9] with $n$ being the number of frames in the trajectories and $G_i$ and $E_i$ the poses for frame $i$ for the ground truth and estimated poses, respectively. The $trans()$ operator returns the translational component of a transformation.

$$ATE = (\sum_{i=1}^{n} \|\text{trans}(G_i^{-1} T E_i)\|^2)^{\frac{1}{2}} \tag{4}$$

Note that ATE does not take rotational errors into consideration. The Relative Pose Error is a measure for drift. A single error is computed over an interval $\Delta$ and given in Equation 5. For a single pose $i$ the error is then computed for a given interval $\Delta$ and all pose error are averaged as given by 6 with $m = n - \Delta$. Again, only the translational component is considered.

$$RPE_i = (G_i^{-1} G_{i+\Delta})^{-1} (E_i^{-1} E_{i+\Delta}) \tag{5}$$

$$RPE_\Delta = (\frac{1}{m} \sum_{i=1}^{m} \|\text{trans}(RPE_i)\|^2)^{\frac{1}{2}} \tag{6}$$

$$RPE = \frac{1}{n} \sum_{\Delta=1}^{n} RPE_\Delta \tag{7}$$

Finally all possible time intervals $\Delta$ are taken into account and an overall average is computed, see Equation 7. Instead of using the trajectory of the non-optimized first pose estimates for every frame, only the final keyframe poses are used for the path. Necessarily, this results in less smooth trajectories. Although local BA, loop closure and local map localization are in use, based on the Absolute Trajectory and Relative Pose Error, no significant improvement of our system compared to VO can be seen. APE for our system and VO is 0.089761 and 0.106138 respectively, a mere 15% decrease. For RPE, the values come out to 0.256769 and 0.253313 for our system and VO, respectively. Here the error even increased by 1.3%. This is likely due to the map being a closed room where even a VO system accumulates no significant drift (as it would in looping hallways). The poses are depicted in Figures 10 and 9

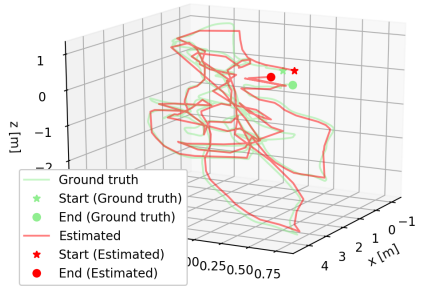


Figure 9: The trajectory of our system on the EUROC V1 1 dataset compared to the ground truth trajectory.

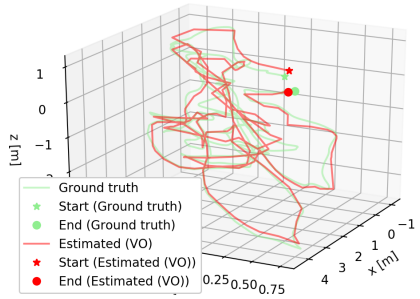


Figure 10: The trajectory of the VO system compared to the ground truth trajectory is qualitatively as well as quantitatevly very similar to our system's result.

# 6 Conclusion

During the final stretch of the semester we were able to extend the hitherto developed Visual Odometry system to include a large part of the contributions of ORB-SLAM: the Covisibility Graph for local tracking and mapping, Loop Closure for drift compensation via the Essential Graph and BFS pose update propagation, and finally Keyframe Removal for compact map representation. Through performance optimization (most notably by running feature detection for left and right frames concurrently) we were able to withstand the added functionality and keep the system running at 20 Hz for 99% of the time. Evaluating the previous VO system to ours showed improved map consistency and similar numerical results for trajectory accuracy, likely due to the map at hand. Comparing the systems on different datasets from the EUROC datasets was not possible as both systems weren't robust enough to manage the more difficult tracks (although notably VO failed far sooner than our OS system). There are two performance improvements that the timeframe did not permit to implement but would likely result in a comparable system to the original ORB-SLAM: first, using a BoW representation of the ORB-feature space for place recognition in Loop Closure, and secondly, to run Loop Closure truly in a third thread (instead of having it be part of mapping).

# References

[1] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35, 01 2016.

[2] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006.

[3] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381395, June 1981.

[4] Dorian Gálvez-López and Juan D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28:1188–1197, 2012.

[5] C. Y. Lee. An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers*, EC-10(3):346–365, Sep. 1961.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

[7] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[8] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. pages 2564–2571, 11 2011.

[9] Juergen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. pages 573–580, 10 2012.

[10] B. Triggs, Philip Mclauchlan, R. Hartley, and Andrew Fitzgibbon. Bundle adjustment - a modern synthesis. *ICCV '99 Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pages 198–372, 01 2000.