Stereo ORB SLAM

Qiao Qiao

Robotics, Cognition, Intelligence - Technische Universität München

Abstract

This stereo ORB SLAM project is one of the projects of TUM practical course: Vision-based Navigation. It is an extension of the ORB odometry algorithm which was already implemented as part of the practical course exercises. Some useful concepts inspired by the ORB SLAM papers [14] [15] are added to this implementation to realize the loop closure, which reduces drifts and improves the trajectory accuracy. The system is able to recognize the places where it has visited before and combine the new and old visual information together. It uses stereo images as input and efficiently uses the same ORB features for all SLAM tasks: tracking, mapping and loop closing. Thanks to the additional redundant keyframes and old landmarks discarding strategy, the map only grows if the scene content changes, which allows lifelong operation. By using covisibility graph, the local set of keyframes in local mapping are chose based on their visual information while in odometry they are chose only based on time. Being tested on several stereo sequences of EuRoC dataset[3], SLAM trajectory estimation is more accurate than the odometry's.

1 Introduction

Simultaneous Localization and Mapping (SLAM) means updating a map of an unknown environment while simultaneously keeping track of an agent's location within it. [4] SLAM can be used in a wide range of fields, for instance, robot navigation, virtual reality and autonomous driving. Although at first glance, it seems to be unsolvable because it is one of the famous chicken-and-egg problems. However, as time goes by, it appears several methods that can at least approximately solve this problem, for example, using Particle Filter [2] and Extended Kalman Filter[13]. Nowadays, most SLAM approaches use bundle adjustment to jointly estimate poses and landmark positions. ORB SLAM also belongs to this kind of approaches. Among the different sensor modalities, for example, laser scanning, sonar sensors, cameras or even tactile sensors, cameras are cheap and provide rich information of the environment that allows for robust and accurate place recognition. Visual SLAM can be performed by using only a monocular camera, which is the cheapest and smallest sensor setup. However, we don't have access to depth information from one single camera, the scale of the map and estimated trajectory is unknown. Monocular SLAM suffers from scale drift, at the same time, it may fail if the camera only rotates and has no translation. By using a stereo or an RGB-D camera, the scale problem mentioned above can be solved. Since the previous implemented exercises use stereo cameras setup for odometry, for convenience, my SLAM extension will also use stereo cameras.

What are the differences between odometry and SLAM? In odometry, there are no global mapping, the tracking is added incrementally. The system is not able to perform loop closure, which means, if the camera moves to somewhere it has been before, it won't be able to recognize the previous visited place. So, the system is accumulated to drift. However, in SLAM algorithm, beside the normal tracking part, the system will check the visual similarity between current frame and all the previous keyframes in order to recognize a previous visited location to close the loop. In this way, the system is able to

correct the drifts or errors which accumulated over time and improve the accuracy of the estimated trajectory. The keyword here is "loop closure": if we sacrifice loop closures, SLAM reduces to odometry. [4]

Why we use ORB features? Key point detection and matching is a very important part of SLAM. ORB feature [5] (Oriented FAST and Rotated BRIEF) is a combination of FAST detector [20] [21] and BRIEF [11] descriptor, which is rotation invariant and robust to noise. As a 256 bits descriptor, it has much lower computational cost compared to SIFT [10] and SURF [1]. FAST stands for Features from Accelerated Segment Test. It is a corner detector, which is several times faster than other existing corner detectors. However, it is not robust to high levels of noise. FAST detects the corner by comparing the brightness of the candidate point and a circle around it depending on its intensity and a threshold. FAST feature does not have an orientation component. In order to have the orientation information, ORB feature computes the intensity weighted centroid of the patch with located corner at center. The direction of the vector from this corner point to centroid gives the orientation. BRIEF stands for Binary Robust Independent Elementary Features. It uses simple binary tests between pixels in a smoothed image patch. It provides high recognition rate unless there is large in-plane rotation. ORB revamps the BRIFE to be rotation-aware.

2 Related Work

2.1 Stereo SLAM

Depth information of stereo system can be accessed through triangulation. However, an early stereo system finished by Paz et al. [18] showed that landmarks can be reliably triangulated if their depth is less than 40 times the stereo baseline. For this reason, the close and far landmarks should be treated differently. In ORB SLAM2 paper [15], close landmarks are used to provide scale, translation and rotation information, at the same time, far landmarks provide accurate rotation information but weak scale and translation information. However, in this implementation, I first aim to focus on small-scale dataset, so, there is no distinction between close and far points.

In order to enable lifelong operation and large environments mapping ability, for mapping, bundle adjustment is performed only on a local set of keyframes to make the calculation complexity independent to the map size. Many other stereo SLAM works also use local bundle adjustment for scalability, like RSLAM of Mei et al. [12] and S-PTAM by Pire et al [19]. However, they both lack global consistency, which is one of the goal of ORB SLAM2 [15]. In order to get a globally consistent map, when performing the loop closure, the system first aligns both sides, so that the tracking can continue localizing using the old map and then performs a global BA. In the ORB SLAM2 paper, they perform a pose-graph optimization before global BA. However, in this implementation, I skip the pose-graph optimization step because I want to firstly focus on small-scale dataset, in which the accumulated drift that need to be corrected is not that large.

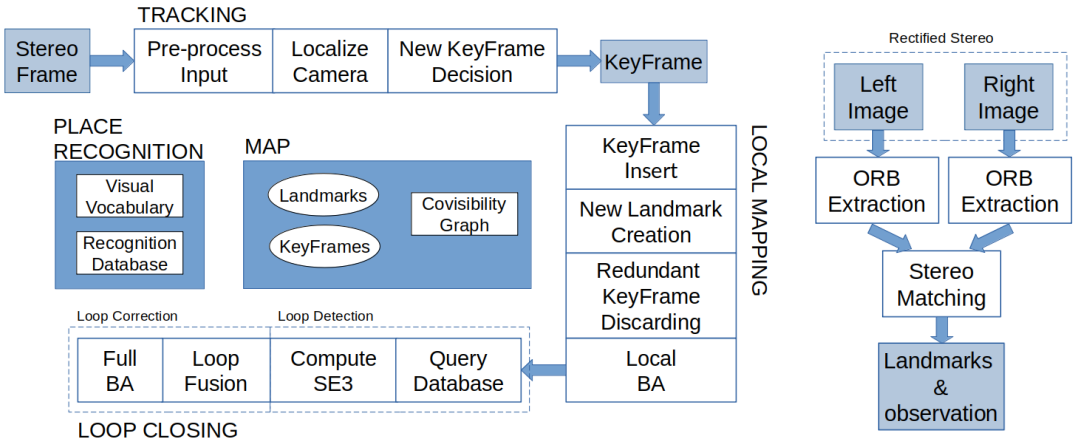
2.2 Place Recognition

Place Recognition aims to find similar images from a large amount of images, is one of the critical part of loop closure. For appearance based methods, bags of words technique [16] is one of the most popular place recognition techniques because of its high efficiency. The idea of bags of words is to discretize the feature-descriptor space by hierarchical clustering in a "vocabulary tree". In our case, each feature can be seen as visual "words",

which correspond to leaf node. For each image, we count the occurrence of each "word" and store it as a bag-of-words vector. In this way, when comparing images, we only need to compute the distance of normalized bag-of-words vectors, which make it very efficient. Those images whose bag-of-words vectors are close means they are potentially similar. However, when the image amount is very large, it is still not efficient to compare every image. To avoid comparing every image, we can query the image with "inverse index". It means, in addition to cache the word count for each image, we also store list of images for each word.

DBoW2 [7] used for the first time bags of binary words obtained from BRIEF descriptors [11] and FAST feature detector [21], making it very efficient and robust. However, BRIEF descriptors lacks rotation and scale invariant, which limited the system to in-plane trajectories and loop detection from similar viewpoints. ORB SLAM solve this problem by replacing the descriptor and detector by ORB feature, which is invariant to rotation and scale in a certain range.

3 Method description



(a) System Overview (b) Input Pre-processing

Figure 1: Stereo ORB SLAM is composed of two main parallel threads: tracking + loop closing + Full BA, local mapping.

A general overview of the system is shown in Figure 1. The original ORB SLAM system [14] has three main parallel threads: tracking, local mapping and loop closing. Loop closing thread launches a fourth thread to perform full BA after the pose-graph optimization. However, due to development time constraint, my implementation only have two threads. One thread is the combination of tracking and loop closing. Tracking localizes every frame by finding feature matches to the local map and minimizing the reprojection error by applying motion-only BA Loop closing tries to detect large loops and correct the accumulated drift by directly performing a full bundle adjustment. Another thread is local mapping, which optimizes the local map by performing local BA.

3.1 Basic Concept

3.1.1 Landmarks

Landmarks are obtained by reprojecting the detected corner to 3D space. Since we don't reconstruct the object surface in the map, landmarks are those remarkable features that actually represent the map. There is a trackId for every landmark.

Each landmark stores:

- Its 3D position in the world coordinate system.
- Inlier observations in the current map: stores as a collection of all images that observed this feature and the corresponding feature index in that image.
- Outlier observations in the current map: stores as a collection of all images that are outliers for this feature and the corresponding feature index in that image.

3.1.2 Keyframes

Due to the high frame-rate, it is unfeasible to use all the frames to perform optimization. So, we choose keyframes, the frames which store rich information of the map. And the map should be well presented by only using keyframes.

Each Keyframe stores:

- Camera pose: which is a rigid body transformation that transforms points from the camera to the world coordinate system.
- Keypoint information: include corner position, corner angle and corner descriptor, which are indexed by FeatureId. FeatureId is ids for 2D features detected in images.

3.1.3 Covisibility Graph

Covisibility graph, shown in Figure 2, is a very useful concept in the implementation, which is presented as an undirected weighted graph. Each node is a keyframe and an edge between two keyframes exists if they share some observations of the same landmarks (the threshold can be adjusted depending on dataset). The weight θ of the edge is the number of common landmarks that they both observe.

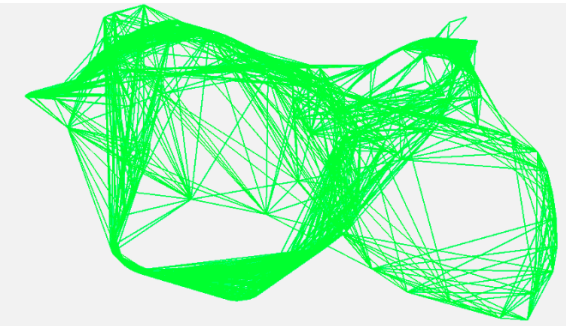


Figure 2: Covisibility Graph

3.1.4 Bags of Words Place Recognition

Loop closure is one of the critical part of SLAM algorithm. The system has used bags of words techniques, based on DBoW2 [7], to perform loop detection. Visual words are just a discretization of the descriptor space, which is known as the visual vocabulary. The vocabulary is created offline with the ORB descriptors extracted from a large set of images. In this case, we just use the same ORB vocabulary, which is created and shared by the author of ORB SLAM paper.

3.1.5 Bundle Adjustment (BA)

Bundle Adjustment is used to provide accurate estimates of camera poses and a sparse geometrical reconstruction [23], given enough matches and good initial guesses.

Landmark 3D locations $X_{w,j} \in R^3$ and keyframe poses $T_{iw} \in SE(3)$, where w stands for the world reference, are optimized by minimizing the reprojection error with respect to the matched keypoints $x_{i,j} \in R^2$.The error term for the observation of a landmark j in a keyframe i is:

$$e_{i,j} = x_{i,j} - \pi_i(T_{iw}, X_{w,j})$$

where π_i is projection function, projects the 3D points into 2D image plane. The cost function to be minimized is:

$$C = \sum_{i,j} \rho_h(e_{i,j}^T e_{i,j})$$

where ρ_h is the Huber robust cost function. Practically, we use Ceres Solver to solve the non-linear optimization problem. To deal with outliers, a standard technique is to use a LossFunction. Loss functions like Huber reduce the influence of residual blocks with high residuals, usually the ones corresponding to outliers.

$$\rho_h(s) = \begin{cases} s & s \leq 1 \\ 2\sqrt{s} - 1 & s > 1 \end{cases}$$

In case of global BA (the implement in section 3.5.4) we optimize all landmarks and keyframes, but keeping the first keyframe fixed as the origin. In local BA (see section 3.4.4) the landmarks and keyframes included in the local area are optimized, while a subset of keyframes outside the region is fixed. In pose optimization (motion-only BA), (see section 3.3.3) all landmarks are fixed and only the current camera pose is optimized.

3.2 Map Initialization

Compared to monocular SLAM, map initialization by stereo camera is relative simple. Since we already get the depth information of landmarks through triangulation. To set up the system, we take the very first frame as a keyframe, set the left camera pose as the origin, and create an initial map from all stereo landmarks.

3.3 Tracking

In this section, I will describe about the different steps of the tracking thread that are performed with every image frame, not only the keyframe. The work in this section are mostly done in the previous exercises. The camera pose optimization, mentioned in several steps, also named as motion-only BA. By performing motion-only BA, all points are fixed and only the camera pose is optimized.

3.3.1 ORB Extraction

Af first, a list of the keypoints (1500, this number can be adjusted) is detected using the Shi-Tomasi algorithm [9], which can be implemented by the goodFeaturesToTrack() function in OpenCV. Afterwards, the corner point angle computation and descriptor computation are done to get the ORB features, which is a binary descriptor with 256 bits containing 0 and 1. The detailed method for computation can be found in ORB descriptor paper [5].

3.3.2 Finding Stereo Matches

Once having the ORB features of each image, we want to match the features of left and right stereo image. Since we only have two images here, the number of features is not large. We could simply use brute force matching. For every descriptor in left camera image, we want to find a correspondent descriptor in right camera image, that has the minimum Hamming distance. There are three conditions that need to be fulfilled for a valid match.

- The distance should be smaller than the predefined threshold.
- The distance to the second best match should be larger than the smallest distance multiplied by dist_to_best.
- If descriptor P in left image match to descriptor Q in right image. Then descriptor Q in right image should also match to descriptor P in left image.

3.3.3 Pose Estimation From Previous Frame

Firstly, the landmarks was projected to the image plane using the camera pose of last frame. And then the matches are searched between projected landmarks and detected keypoints in the current frame by performing a guided search. Once we get the 3D-2D correspondences, we can use it to estimate the current camera pose by using Perspective-n-Points (PnP) in a RANSAC scheme in OpenGV'CentralAbsoluteAdapter to be robust against the outliers matches. RANSAC stands for RANdom SAmple Consensus. RANSAC algorithm randomly selects a subset of data points and fits the camera pose based on these selected data points for N iterations. [6] This pose estimation also names as motion-only bundle adjustment, which keeps all the landmarks fixed and only optimizes the camera pose, details are explained in section 3.1.5 . It counts inliers and keep the camera pose with largest number of inliers. In the end, a refinement of the camera pose is done by using all inlier matches from RANSAC by minimizing re-projection error.

3.3.4 New Keyframe Decision

Compared to the ORB SLAM paper [14], our new keyframe decision step is relative simple. I keep this part same as the odometry algorithm. New frames are added based on visual change. If the matches between image and landmarks have too few inliers, the camera is probably in a new location. So, if the number of inlier image-to-landmark matches is under than the predefined threshold (which can also be directly adjusted in user interface) and the bundle adjustment optimization is not running, the next frame will be a new keyframe and be added to the map.

3.4 Local Mapping

In this section, I will describe how I perform local mapping with every new keyframe. In this section, except for adding new landmarks and observations, other works are implemented on my own.

3.4.1 New Keyframe Insertion

In order to insert new keyframe, we first update the covisibility graph, adding a new node for current frame and updating the edges if the new frame observe same landmarks (more than a adjustable threshold) with other frames. And then we update active cameras, active landmarks, fixed cameras to prepare for the local bundle adjustment. Active cameras include the currently processed keyframe and all the keyframes connected to it in the covisibility graph. Active landmarks are all the landmarks seen by those active cameras. All other keyframes that see those active landmarks but are not connected to the current keyframe in the covisibility graph are included in the optimization but remain fixed, that is fixed cameras.

3.4.2 Adding New Landmarks And Observations

We already have stereo matches for the current frame and feature-to-landmark matches for the left camera. For all inlier feature-to-landmark matches, we add the observations (current left frame Id) to the existing landmarks. If the left camera's feature appears also in stereo matches inliers, then add both observations (current left and right frame Id) to this landmark. For all inlier stereo observations that were not added to the existing landmarks, we triangulate them and add new landmarks.

3.4.3 Redundant Keyframes Deleting

Bundle adjustment complexity grows with the number of keyframes. Redundant keyframes slows down the optimization and can not really provide much useful information. So, we try to detect redundant keyframes and delete them in order to enable lifelong operation. In this way, the number of keyframes will not grow unbounded, unless the scene changes. We discard all the keyframes in whose 90 % of the map points have been seen by at least other three keyframes, since most of its information is shared by other keyframes. The first frame at the very beginning will not be deleted. Only one keyframe will be discarded at one time.

3.4.4 Local Bundle Adjustment

It is unfeasible to achieve real-time on a single CPU if full bundle adjustment was used to optimize the map whenever a new keyframe is added. So, local BA, choosing a local set of keyframes, is a better choice. The local BA optimizes active cameras and active landmarks using the equation in section 3.1.5. It optimizes the camera poses and landmarks position in a local map. In odometry, they use a relative simple strategy by always choosing the 10 latest keyframes as active cameras. However, in this work, active cameras are the currently processed keyframe and all the keyframes connected to it in the covisibility graph. Active landmarks are all the landmarks seen by those active keyframes. All other keyframes that see those active landmarks but are not directly connected to the currently processed keyframe in covisibility graph are fixed cameras. Fixed cameras are included in the optimization but remain fixed.

3.5 Loop Closure

The loop closing thread takes the current keyframe after processing by the local mapping and tries to detect and close loops. The work in this section are completely implemented on my own.

3.5.1 Loop Candidates Detection

We use the bag of words techniques to compare the keyframe appearances. In order to make the system more robust, we first compute the similarity between the bag of words vector of current frame and all its neighbors in the covisibility graph (θ min = 30, which can be adjusted) and get the lowest score s_min . Then we discard all those keyframes whose score is lower than s_min because their appearances are potentially not similar, they are unlikely to be at the same place. All those keyframes directly connected to current frame are also discarded from the results because we want to detect large loop. To accept a loop candidate, we must detect consecutively three loop candidates that are consistent, which means keyframes connected with each other in the covisibility graph. There can be several loop candidates if there are several places with similar appearance to current frame.

3.5.2 Compute the Transformation

In stereo SLAM there are six degrees of freedom in which the map can drift, three translations and three rotations. We don't need to consider scale factor in contrast to monocular SLAM. To close a loop, instead of similarity transformation, we only need to compute a rigid body transformation from the current keyframe to the loop keyframe that informs us about the error accumulated in the loop. We first compute correspondences between ORB associated to map points in the current keyframe and the loop candidate keyframes to get the 3D to 3D correspondences for each loop candidate. We perform RANSAC iterations with each loop candidate, trying to find a rigid body transformation using the method of Horn [8], a closed-form solution. In practice, we use point-cloud alignment in OpenGV to get the transformation between two frames. If we find a transformation with enough inliers, we optimize it by minimizing re-projection error using Ceres Solver.

$$e_1 = x_{1,i} - \pi_1(T_{12}, X_{2,j})$$

$$e_2 = x_{2,i} - \pi_1(T_{12}^{-1}, X_{1,i})$$

and the cost function to minimize is:

$$C = \sum_n (\rho_h(e_1^T e_1) + \rho_h(e_2^T e_2))$$

ρ_h is the Huber robust cost function. And then a guided search is performed to search for more correspondences, which means projecting the landmarks that observed by current frame into loop candidates. For projecting landmarks observed by current frame into loop candidates, we first convert the 3d point into current frame (using current's world pose), then we convert them to loop candidate frame using their transformation, and then project the 3D points into 2D image plane. After getting more correspondences, we optimize it again using Ceres Solver as before. If the transformation is supported by enough inliers, the loop with this loop candidate is accepted.

3.5.3 Loop Fusion

In order to correct the loop, we need to fuse duplicated landmarks and update the covisibility graph. In particular, to insert new edges in the covisibility graph that will attach the loop closure. At first, the current keyframe pose is corrected with the SE3 transformation from last step and this correction is propagated to all its neighbors in the covisibility graph, so that both sides of the loop get aligned. All landmarks seen by the loop keyframe and its neighbors are projected into current keyframe and its neighbors. Matches are searched in a narrow area around the projection, as done in the tracking part 3.3.3. All those landmarks matched and those that were inliers in the computation of transformation are fused. For fusion, I added imageIds which observes the newer landmark to the older landmark observation and then delete the newer landmark. Finally, we need to update the covisibility graph.

3.5.4 Global Bundle Adjustment

To close the loop, I directly perform a global bundle adjustment after loop fusion using all the keyframes and landmarks with the equation in section 3.1.5. The first keyframe will remain fixed as the origin. The original ORB SLAM paper [14] for monocular camera only uses a pose-graph optimization to effectively close the loop. In ORB SLAM2 paper [15], they perform a global bundle adjustment in a separate thread after pose-graph optimization. However, I first focus on solving the problem in small-scale dataset. Considering that the drift isn't very large, I simply perform a global BA without pose-graph optimization.

3.6 Visualization

A good visualization in GUI (graphical user interface) helps us debug, observe the progress, understand how the system works, analyze what can be improved. There are buttons in GUI for users to decide whether the corresponding object will be shown or not.

Original odometry visualizes the following objects:

- Keyframes: are shown as two pyramids, which represents stereo cameras.
- Current Keyframes: same shape as Keyframes, in a different color.
- Landmarks: 3D point.
- Old Landmarks: landmarks that will not be used any more, 3D points in a different color.

In addition to the objects which are visualized in odometry, my SLAM project visualizes the following objects:

- Ground Truth Trajectory: Since the ground truth data and the images have different time interval. I choose the ground truth frame which has closest timestamp compared to the keyframe to do the alignment with keyframes. But for visualization, every ground truth frame is shown.
- Camera Trajectory: polyline connected by every keyframes.
- Active Cameras: cameras that are used in the local bundle adjustment for local mapping.

- Active Landmarks: landmarks that are used in the local bundle adjustment for local mapping.
- Old Landmarks: landmarks that will not be used any more.
- Covisibility Graph: poly-line connected by covisibility graph neighbor keyframes.
- Loop Closure Edges: the loop closure edges are highlighted with a different color.

4 Evaluation

In order to evaluate the algorithm, I load the ground truth and compare it with the estimated poses. The given ground truth data might not have the exactly same timestamp as image frame, so I pick the ground truth which has closest time to the image frame. After loading the ground truth, we need to do the alignment between ground truth and trajectory using all poses.

There are two different ways to align the estimated trajectory and the ground truth trajectory:

- SE(3) alignment: The trajectories are aligned with a rigid-body transformation.
- Sim(3) alignment: The trajectories are aligned with a similarity transformation.

Since we only use stereo cameras, which provide us the depth information to mitigate the scale problem which will occur by only using mono-camera, we can choose SE(3) alignment instead of Sim(3) alignment.

There are different error metrics for evaluation. Two major methods are the absolute trajectory error (ATE) and the relative pose error (RPE). The ATE is well-suited for measuring the performance of visual SLAM systems. In contrast, the RPE is well-suited for measuring the drift of a visual odometry system. [22] [17] So, we are going to use ATE in this case.

I use ATE RMSE error metric to evaluate the algorithm, ATE RMSE stands for Absolute trajectory root-mean-square-error. The absolute trajectory error directly measures the difference between points of the ground truth and the estimated trajectory. This is the root-mean-square-error of the estimated pose translations and the ground truth pose translations.

The algorithm is able to run on the V1_01_easy and MH_01_easy stereo sequences in the EuRoC dataset [3]. It gets a better performance on the V1_01_easy sequence. Compared to the ground truth data, SLAM gets much lower Root Mean Square Error than odometry (0.03 - 0.04 compare to 0.06 - 0.1). I also tested some other sequences in the EuRoC dataset, but the tracking was lost both for SLAM and odometry. The tracking part for SLAM and odometry are almost the same. The problem might be that the tracking algorithm is not powerful enough to due with fast motion, blur, low-texture environments or some other issues. Or some parameters need to be fine tuned for different sequences to get better results.

Because the trajectory of SLAM is simply visualized by a polyline of all keyframes. At some places, when the keyframes are relative sparse, there will be a gap between the trajectory and ground truth in visualization.

Figure 3: SLAM

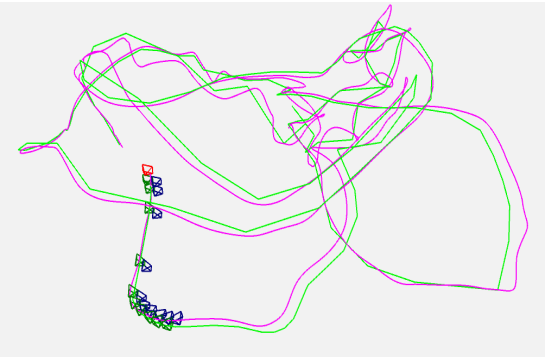


Figure 4: Odometry

5 Conclusion and Discussion

Having six weeks to extend the Odometry algorithm in the exercises to SLAM, I was able to achieve several goals:

- Instead of always using the 10 latest keyframes in odometry, I use covisibility graph information to select keyframes which share most information with the current frame for local mapping.
- The stereo ORB SLAM is able to detect loop and correct loop. Reduce the accumulated drift and achieve higher trajectory accuracy in comparison of odometry, which has no loop correction ability.
- The algorithm enables lifelong operation. Map only grows when the camera sees new scene. This is verified by MH_01 sequence in the EuRoC dataset [3]. At the beginning of MH_01 sequence, the camera just moves around at the same place for a long time. By implementing redundant keyframes discarding, our map doesn't grow unbounded when there are no new scenes.

However, due to the limited time for development, there are several unfinished parts compared to the original paper [14] [15]. These can be considered as future works to improve the system.

- In the ORB SLAM2 paper [15], they treat close and far points differently. If the depth of one point is less than about 40 times the stereo baseline, the point will be defined as 'close' point and can be triangulated to provide scale, translation and rotation information. A 'far' point will only provide accurate rotation information. However, since I first focus on running the algorithm on the dataset sequence whose environment is a relative small room. I didn't distinguish close and far points at the beginning to simplify the implementation. At the end, I didn't have time left to finish this part. That could be one of the reasons that why the algorithm doesn't work well on some sequences.
- In the original ORB SLAM paper for monocular camera, they perform a pose graph optimization over the Essential Graph to effectively close the loop, which is a rough approximation of a full BA. For stereo camera, they first perform the pose graph optimization, and then a full BA. Although I already implement the essential graph, I didn't have time to implement the pose graph optimization. Considering that the drift is not that large, I skip the pose graph optimization and directly use full BA.

- I wasn't able to implement the loop detection and global bundle adjustment in separate threads. The original ORB-SLAM2 paper [15] have four separate threads for the realization of real-time. I only have two separate threads, tracking + loop detection + global BA and local mapping, so this stereo ORB SLAM is not really real time. It will stop for a while when doing global bundle adjustment.
- Also, there is no automatically re-localization when the tracking is lost. In some more difficult sequences, because of fast motion or other reasons, the camera tracking might be lost. In this case, the SLAM can't continue because its lack of automatic re-localization. Based on the ORB SLAM paper [14] Re-localization is basically the same method used in loop detection. Given more time, Re-localization should be possible to realize.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[2] J.L. Blanco, J.A. Fernández-Madrigal, and J. Gonzalez. A novel measure of uncertainty for mobile robot slam with rao—blackwellized particle filters. *The International Journal of Robotics Research*, 27(1):73–89, 2008.

[3] 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.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[5] K. Konolige E. Rublee, V. Rabaud and G. Bradski. Orb: an efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, Barcelona, Spain, 2011.

[6] 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):381–395, June 1981.

[7] D. Galvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[8] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[9] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[10] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.

[11] C. Strecha M. Calonder, V. Lepetit and P. Fua. Brief: Binary robust independent elementary features. In *European Conference on Computer Vision (ECCV)*, page 778–792, Hersonissos, Greece, 2010.

[12] Christopher Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. Rslam: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, 94:198–214, 2010.

[13] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007.

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

[15] Raul Mur-Artal and Juan D. Tardos. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[16] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168, 2006.

[17] Chair of Computer Vision and Artificial Intelligence of TUM. Useful tools for the rgb-d benchmark.

[18] L. M. Paz, P. PiniÉs, J. D. TardÓs, and J. Neira. Large-scale 6-dof slam with stereo-in-hand. *IEEE Transactions on Robotics*, 24(5):946–957, 2008.

[19] T. Pire, T. Fischer, J. Civera, P. De Cristóforis, and J. J. Berlles. Stereo parallel tracking and mapping for robot localization. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378, 2015.

[20] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, Jan 2010.

[21] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012.

[23] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment — a modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 298–372, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.