ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

V4RL VISION FOR ROBOTICS LAB
Vision for Robotics Lab
Prof. Margarita Chli

Semester Project

Map Fusion for Collaborative UAV SLAM

Autumn Term 2016

Supervised by:
Patrik Schmuck
Prof. Dr. Margarita Chli
Prof. Dr. Roland Siegwart

Author:
Andreas Ziegler

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Map Fusion for Collaborative UAV SLAM

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Andreas	Ziegler

Student supervisors

Patrik	Schmuck
Marco	Karrer

Supervising lecturers

Margarita	Chli
Roland	Siegwart

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date	Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Abstract

A correct and accurate common map is crucial for multiple robots to collaboratively performing tasks.

In this semester project, an existing multi agent Simultaneous Localisation and Mapping (SLAM) system should be extended to fuse maps of single robots in a way that no false map alignment is guaranteed and that an optimal alignment is achieved by using multiple place matches. Also redundant information, a consequence of the fusion of two maps, should be removed in order to get a good performance of the optimization routines e.g. Bundle Adjustment (BA).

To achieve the first goal, a new approach is proposed which uses multiple KeyFrame Matchs (KFMs) to fuse two maps. This proposed approach also use a novel idea to spread the KFMs over a bigger area by skipping KeyFrames (KFs) between the detection of KFMs.
To remove redundant information, KF culling is performed after all KFMs were detected and before the main map fusion. This way information which were present in both maps appear only once in the fused map and the Pose Graph Optimization (PGO) and the BA does not have unnecessary data to process.
To reduce the runtime of the optimization part (PGO and BA), the usage of Powell's dog leg (DL) non-linear least squares technique instead of the Levenberg-Marquardt (LM) optimization was evaluated and the system was adapted in a way, that the performance is increased.

The proposed map fusion approach reduces drift and achieves better accuracy compared to the previous approach. With the implemented KF culling, the number of KFs which the PGO and the BA have to process is decreased significantly and therefore better timing is achieved. The usage of DL non-linear least squares technique reduced the runtime of the optimization furthermore.

Acronyms

SLAM Simultaneous Localisation and Mapping

UAV Unmanned Aerial Vehicle

KF KeyFrame

KFM KeyFrame Match

MP Map Point

PGO Pose Graph Optimization

BA Bundle Adjustment

LM Levenberg-Marquardt

DL Powell's dog leg

ROS The Robot Operating System

RMSE Root Mean Squared Error

Chapter 1

Introduction

Simultaneous Localisation and Mapping (SLAM) is the task of moving in a previously unknown environment while mapping the robot's workspace and simultaneously estimating its position in this map. This task is one of the most important challenges on the way to full autonomy for a robot.

Most SLAM systems use onboard sensors as this allows them not to depend on external systems, with which they might loose connection and in this case also their sensing. Vision has become a popular sensor choice for mobile robots as it provides rich information about the environment, while having a low power consumption and providing portability.

The SLAM task is in general platform-agnostic but it is of special interest for Unmanned Aerial Vehicles (UAVs), as they are capable of reaching remote places and are highly agile. This makes them a good choice for tasks like inspection, search and rescue or maintenance in inaccessible areas.
Another reason why SLAM systems are often designed for UAVs, especially in research, is because they are the most challenging platform for SLAM systems as they are more unstable and have faster dynamics to track compared to other types of robots and can suffer from enormous drift, specially if the camera is mounted lateral and the UAV performs rotations. These challenges make UAVs an interesting target for research.

With SLAM systems for a single UAV reaching considerable maturity, multi UAVs systems are gaining increasingly interest. Collaborative SLAM systems for example can boost the efficiency of a mission by sharing the workload of tasks amongst all UAVs. Furthermore, multiple UAVs also allow to increase the robustness of a SLAM system as the UAVs can benefit from the measurements recorded by the other UAVs. Such a collaborative SLAM systems enables a team of UAVs collaboratively performing tasks, e.g. inspection of a structure, maintenance or search and rescue.

In a collaborative SLAM system, a goal is to build a common map which can be used amongst the team of UAVs. To create this map, a place recognition system detects overlaps between maps constructed by different UAVs. If enough correspondences are detected between two maps, these maps can be fused into one which is further on used by multiple UAVs simultaneously.

This work aims to implement a pipeline that, given two SLAM maps with multiple place matches, produces an optimized fused map. This includes finding an optimal

alignment of the two maps by a Sim(3) transformation, applying optimization techniques for improved alignment and identifying and fusing redundant information in the fused map. A Sim(3) transformation is needed, as in monocular SLAM there are seven degrees of freedom (three translations, three rotations and a scale factor). To allow for the map fusion to run online in real-world UAV experiments, furthermore the efficiency of the fusion algorithm will be taken into account.

Chapter 2

Related work

This chapter briefly discusses the work which is related with this semester project, that the reader gets more familiar with already existing approaches and methods.

2.1 ORB-SLAM(2)

ORB-SLAM, proposed in [1], and the extension ORB-SLAM2, proposed in [2], are feature-based (monocular) SLAM systems, which operate in real time and which are designed for indoor and outdoor environments. ORB-SLAM(2) uses the same features for all SLAM tasks (tracking, mapping, relocalization and loop closing) which makes ORB-SLAM(2) more efficient, simple and reliable. ORB-SLAM(2) uses ORB features which are extremely fast to compute and match and have a good invariance to viewpoint. ORB-SLAM(2) generates a compact and trackable map that only grows if the scene content changes and therefore allows a lifelong operation. For this purpose, ORB-SLAM(2) uses a survival of the fittest strategy to select the points and KeyFrames (KFs) which represent the reconstruction best.

2.2 Bags of Words

ORB-SLAM(2) uses a bags of words place recognition module, which is based on [3]. ORB-SLAM(2) uses this module to perform loop detection and relocalization. A visual vocabulary is created offline with the ORB descriptors extracted from a large set of images. Visual words are then a discretization of the descriptor space, which is the visual vocabulary. If the images from which the vocabulary is built are enough general, the same vocabulary can be used for different environments getting a good performance. With the bags of words approach, querying the database can be done very efficiently as the system builds incrementally a database that contains an invert index, which stores in the vocabulary for each visual word, in which KFs it has been seen.

2.3 Co-visibility Graph and Essential Graph

In a co-visibility graph, each node/vertex represents a KF and there is an edge between two KFs if they share observations of the same Map Points (MPs). With the usage of such a co-visibility graph for the tracking and the mapping in ORB-SLAM(2), this two tasks focus in a local co-visible area, independent of the global map size.

An essential graph, described in [1], is built from a spanning tree, loop closure links and strong edges from the co-visibility graph. Loop closing based on the optimization of an essential graph can be performed in real time as the essential graph retains all the KFs but only a reduced number of edges compared to the co-visibility graph and therefore the optimization is less computationally demanding.

2.4 g^2o

The optimization framework g^2o [4] is used by ORB-SLAM(2) for the different optimization tasks. g^2o is a general framework for optimizing graph-based nonlinear error functions as they often appear in many popular problems in robotics, such as SLAM or Bundle Adjustment (BA).

2.5 Levenberg-Marquardt vs. Powell's dog leg

As in many other SLAM systems, ORB-SLAM(2) uses the Levenberg-Marquardt (LM) optimization algorithm for the BA and the Pose Graph Optimization (PGO) by default. As argued in [5] the Powell's dog leg (DL) optimization algorithm has some advantages over the LM algorithm and substituting the LM algorithm in the implementation of the BA with the DL algorithm can lead to computational benefits.

2.6 Multi-UAV Collaborative Monocular SLAM

The multi-UAV Collaborative SLAM system proposed in [6] consists of multiple UAVs and a central ground station. In the proposed system, each agent is able to independently explore the environment running a limited-memory SLAM system onboard, while sending all collected information to the ground station which has increased computational resources. The ground station manages the maps of all the UAVs, detecting loop closures between them, and if necessary, triggering map fusion, optimization and distribution of information back to the UAVs. A single UAV therefore can insert observations from the other UAVs in its SLAM system on the fly.

Chapter 3

Approach

3.1 Map fusion

In this section, the proposed map fusion approach is explained. Firstly, the reader will be introduced to the terms needed to explain map fusion in general and the proposed map fusion approach especially. Then, the previous map fusion approach is introduced followed by the new proposed approach.

3.1.1 Terms

Map Point (MP) / Landmarks

The terms Map Point (MP) and Landmarks are used interchangeably in this report and are defined as uniquely identifiable objects in the world, whose location can be estimated by a sensor.

KeyFrame (KF)

KeyFrames (KFs) are the most representative poses of the trajectory of a robot. As an example in the sketch in figure Figure 3.1, x_0 and x_3 are KFs whereas x_1 and x_2 are not KFs as they are not enough representative because they do not contribute additional (unseen) MPs.

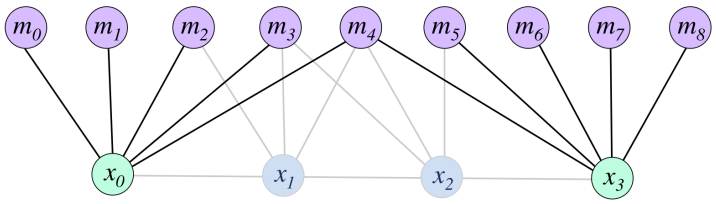


Figure 3.1: Sketch of the KF principle

In ORB-SLAM(2) [1], [2], on which this semester project is based, the following conditions must be fulfilled to insert a new KF:

- More than 20 frames must have passed from the last global relocalization
- Local mapping is idle, or more than 20 frames have passed from last KF insertion
- Current frame tracks at least 50 points

- Current frame tracks less than 90% points than the reference KF

KeyFrame Match (KFM)

A KeyFrame Match (KFM) is if two KFs, one per client, observed the same location (cf. Figure 3.2). If a KFM was detected, a transformation $T \in \text{Sim}(3)$ can be obtained from the poses of the KFs, which relates the maps of the two clients to each other (cf. Figure 3.3). With the transformation T the maps than can be aligned (cf. Figure 3.4).

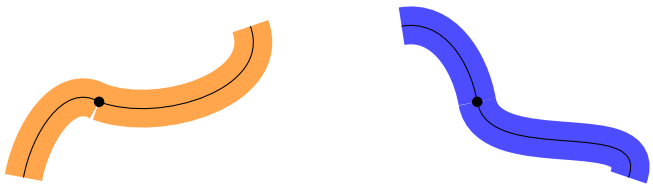


Figure 3.2: Two clients, each with own landmarks and KFs with one KF observing the same location

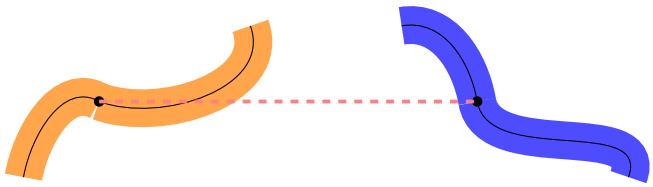


Figure 3.3: Transformation T obtained from the poses of the two KFs

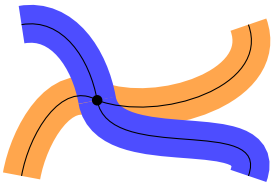


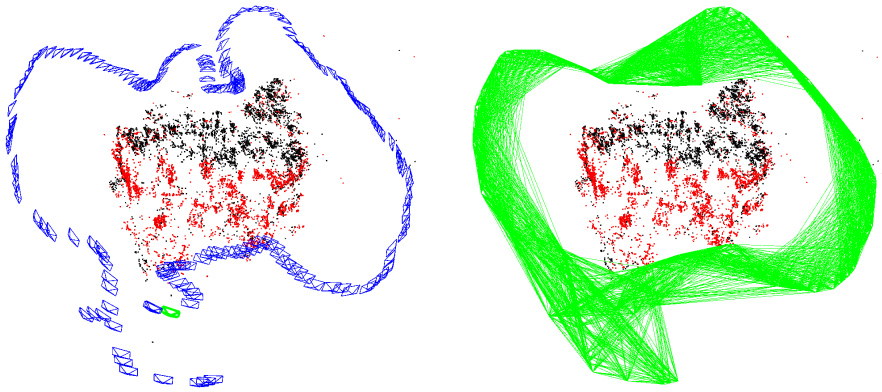
Figure 3.4: The two maps aligned

In conclusion, a KFM contains:

- Two KFs (One per map/client)
- The transformation ($T \in \text{Sim}(3)$) between the two KFs

Co-visibility graph

Co-visibility is represented as an undirected, weighted graph, a co-visibility graph. Each node/vertex is a KF and an edge between two KFs exists if they share observations of the same MPs. The weight of an edge is the number of common MPs [1]. An example of KFs and the resulting co-visibility graph is shown in Figure 3.5.



(a) KFs (blue), Current Camera (gree), MPs (black, red), Current Local MPs (red)

(b) Co-visibility Graph

Figure 3.5: KFs and resulting co-visibility graph (Figures taken from [1])

3.1.2 Previous approach

In the previous approach, proposed in [6], as soon as a KFM was detected, the maps were fused. In a real world example this procedure looks the following:

1. Two clients observed the same location (shown in figure Figure 3.6)

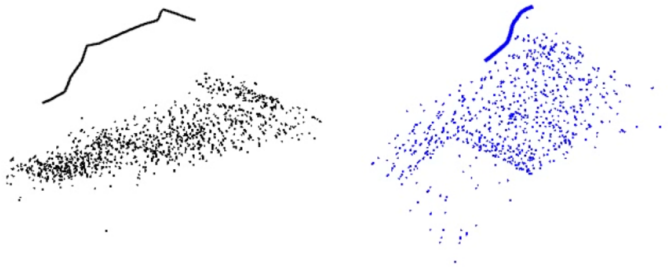


Figure 3.6: Two clients, each with own landmarks and KFs observing the same location

2. The transformation between the two maps can be obtained (shown in figure Figure 3.7)

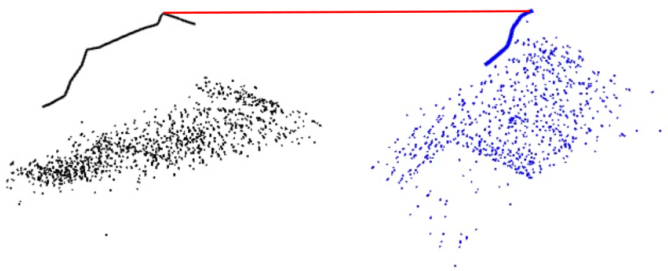


Figure 3.7: Transformation T obtained from the poses of the two KFs

3. The maps can be aligned (shown in figure Figure 3.8)

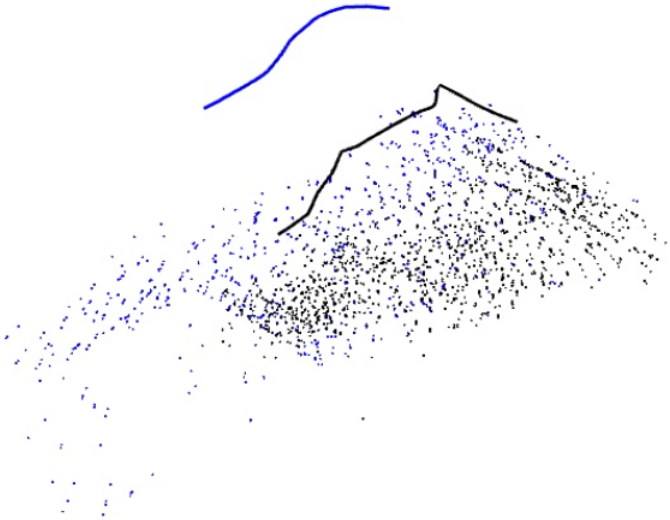


Figure 3.8: Aligned maps

4. Perform Pose Graph Optimization (PGO)
5. Perform Bundle Adjustment (BA)

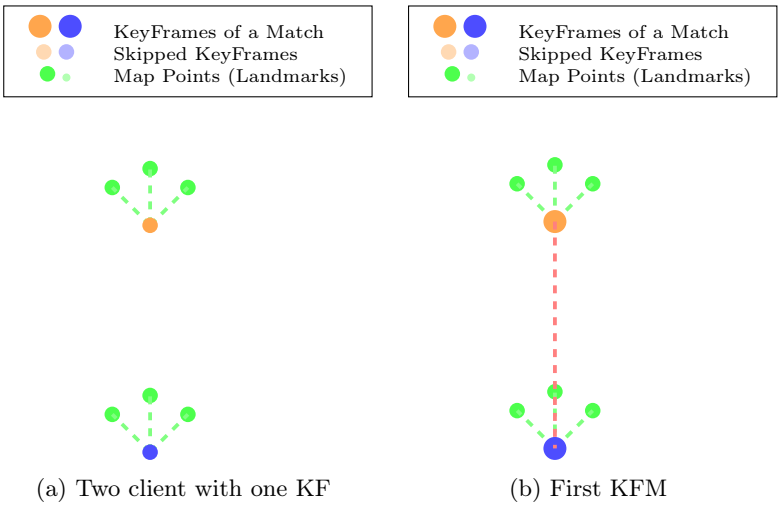
3.1.3 Proposed approach

The general idea of the proposed approach is to use multiple KFMs to guarantee no false map alignment, to reduce drift and to achieve higher accuracy. If maps are only fused when multiple KFMs were detected, one false KFM won't result in a wrong aligned fused map, which could happened with the previous approach. The usage of multiple KFMs provides the PGO and the BA with more information which should lead to a reduced drift and a higher accuracy.

The proposed map fusion approach works as follow:

1. After a KFM is found, skip m KFs before processing the next KF
2. Wait until n KFMs were detected
3. Fuse the two maps with the transformation of one of the KFMs
4. Fuse the map points in the $(n-1)$ KFMs
5. Perform PGO
6. Perform BA

To give the reader a better intuition about how the proposed map fusion approach works, it will be visualized in a synthetic example step by step.
The example starts with two KFs, one per client, which are checked if they observe the same location (cf. Figure 3.9a). If a KFM was detected, as in Figure 3.9b, m, in this example five, KFs are skipped before processing the next KFs.



(a) Two client with one KF (b) First KFM

Figure 3.9: First two steps of the map fusion example

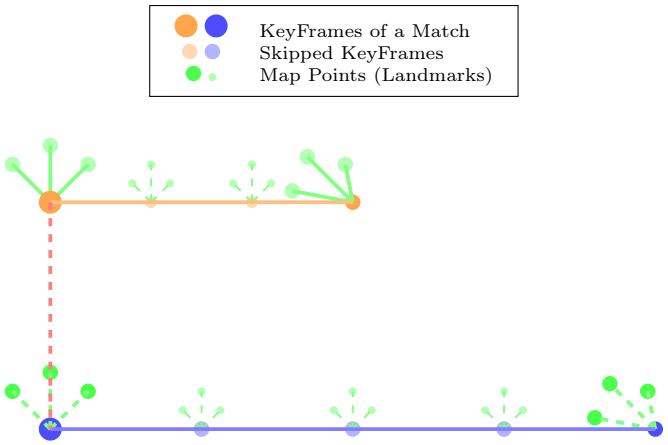


Figure 3.10: Skipped five KFs

After $m(= 5)$ KFs are skipped, the system checks again, if a KFM can be detected, shown in Figure 3.10. Figure 3.11 then shows the case, where the next KFM was detected.

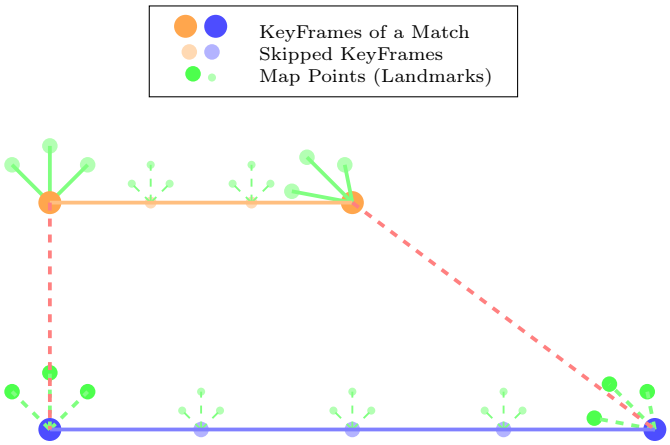


Figure 3.11: Next KFs

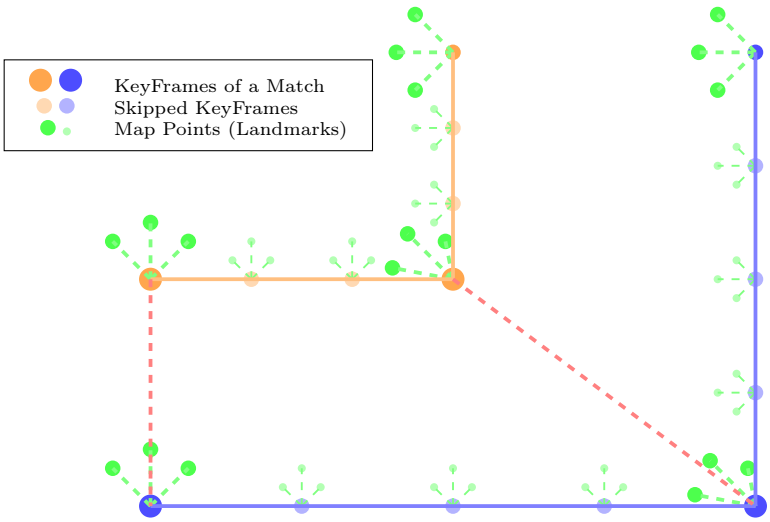


Figure 3.12: Skipped five KFs

Then again m KFs are skipped, as shown in Figure 3.12, before the last KFM was detected, shown in Figure 3.13.

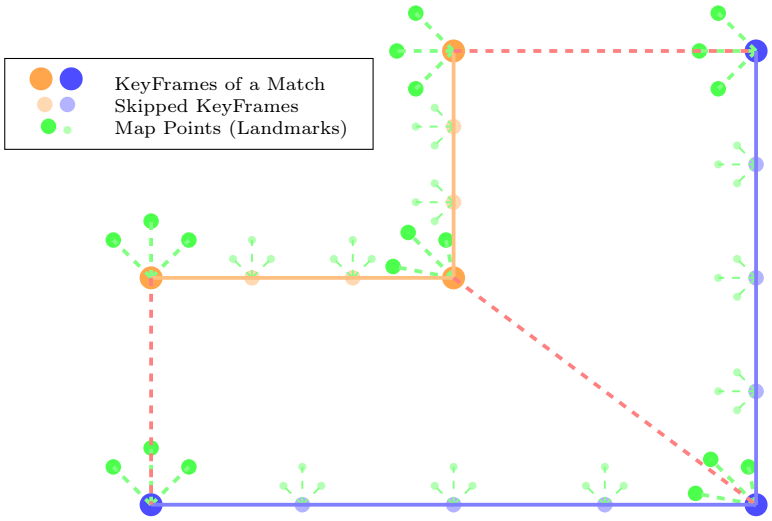


Figure 3.13: Last KFM

After the last KFM was detected, the maps will be aligned with the transformation T from the first KFM and the MPs of the KFs belonging to a KFM are fused, as shown in Figure 3.14.

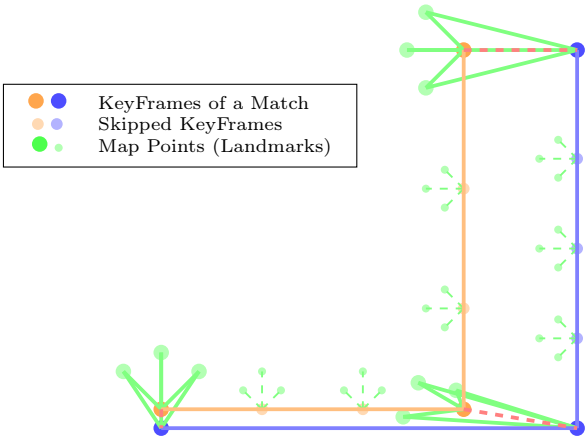


Figure 3.14: Maps aligned

After the two maps are aligned (possibly still with some miss alignment as in Figure 3.14), the PGO will be performed which optimizes the poses of the KFs and afterwards the BA will be performed which optimizes the poses of both, the KFs and the MPs.

The idea behind the usage of multiple KFMs was already discussed in contrary to the idea of skipping KFs after a KFM was detected. This idea will be discussed next.
If KFs are skipped after a KFM was detected, the next KFM will potentially have a bigger distance to the previous KFM. This causes that the KFMs will be more apart from each other and therefor they will cover a bigger area. The KFMs from a bigger area provide the PGO and the BA with more information which hopefully leads to a reduction in drift and an increased accuracy.
To visualize the influence of this idea, Figure 3.15 shows the co-visibility graph of an

experiment where only one KF was skipped after a KFM was detected. Figure 3.16 on the other hand shows an experiment where 10 KFs were skipped after a KFM was detected. One can easily see, that the KFMs in Figure 3.16 are more apart from each other and cover a bigger area compared to the KFMs in Figure 3.15.

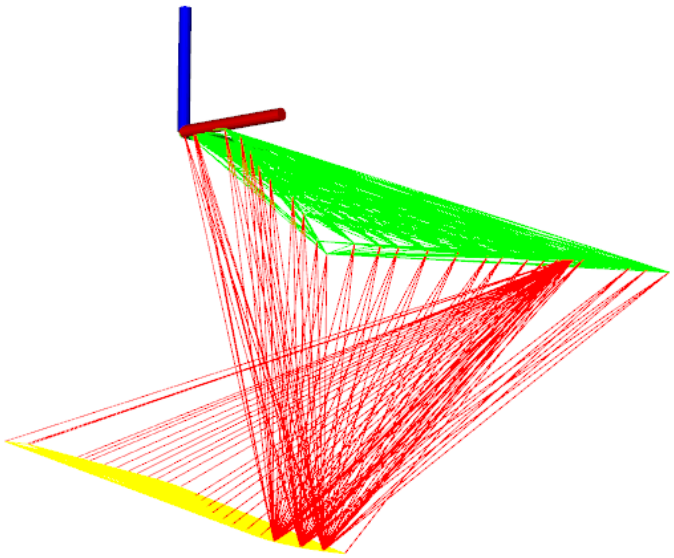


Figure 3.15: One KF skipped after a KFM was found
green: Co-visibility graph of first map
yellow: Co-visibility graph of second map
red: Co-visibility between the KFMs

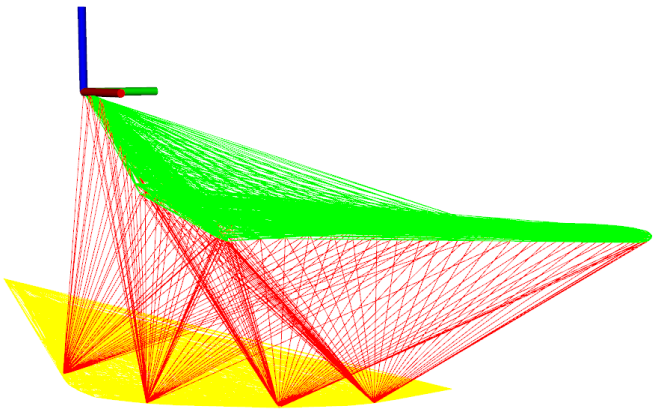


Figure 3.16: 10 KFs skipped after a KFM was found
green: Co-visibility graph of first map
yellow: Co-visibility graph of second map
red: Co-visibility between the KFMs

3.2 Culling

Culling is motivated by the fact, that the BA complexity grows with the number of KFs and that it enables lifelong operation in the same environment as the number of KFs will not grow unbounded [1].

ORB-SLAM(2) [1], [2] already implemented a KF culling mechanism which works locally. As the proposed new approach uses multiple KFMs to fuse two maps, KF culling within the fusion procedure could remove redundant KFs from a bigger area compared to the standard KF culling approach of ORB-SLAM(2).

3.2.1 Approach

The culling of KFs is an additional step in the map fusion procedure which is executed after n KFMs were found and before the maps are aligned and the MPs are fused.

The culling during the map fusion follows the same approach as the one used for the local KF culling in ORB-SLAM(2) [1], [2]: All the KFs which observe a certain percentage of MPs, the redundancy threshold, which have also been seen in at least three other KFs in the same or finer scale, are discarded.

Culling of KFs is performed for every KFM separately. Therefore the KFs which are considered for culling per KFM are the two KFs of the KFM and their direct neighbors in the co-visibility graph. Figure 3.17 visualizes this, where the blue ellipses indicate the direct neighbors in the co-visibility graph and the red ellipse surrounds all the KFs which are considered for the culling for one KFM.

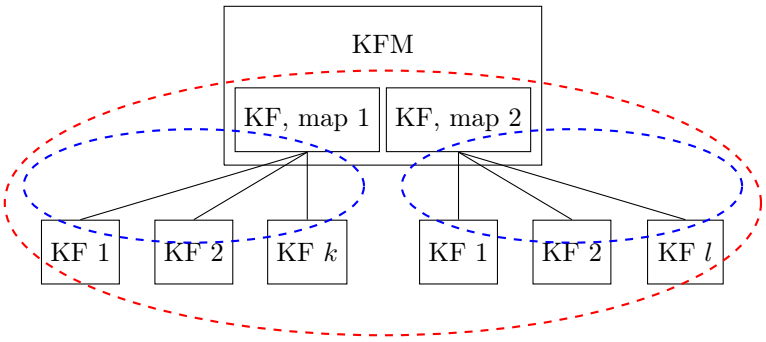


Figure 3.17: KFM with its KFs and their direct neighbors in the co-visibility graph

To illustrate the culling procedure, Figure 3.18 shows the maps of two robots in a toy example.

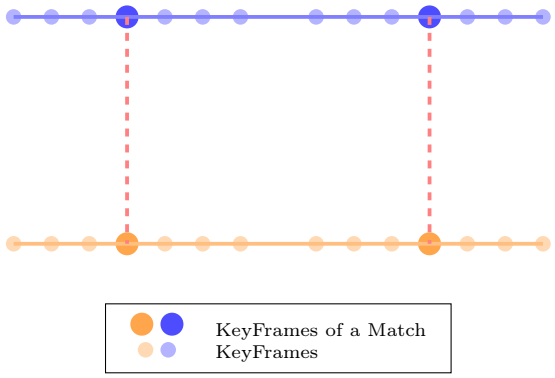


Figure 3.18: Toy example of the maps of two clients

Figure 3.19 then shows the KFs which are directly connected to the KFs of a KFM indicated by dashed rectangles.

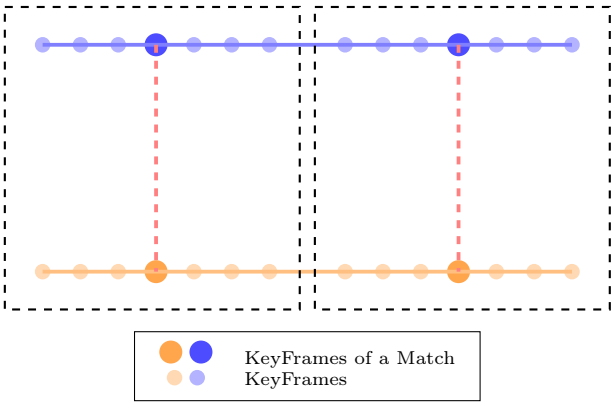


Figure 3.19: Two KFMs with their neighborhood

The toy example with culled KFs is shown in Figure 3.20, where the x represent culled KFs.

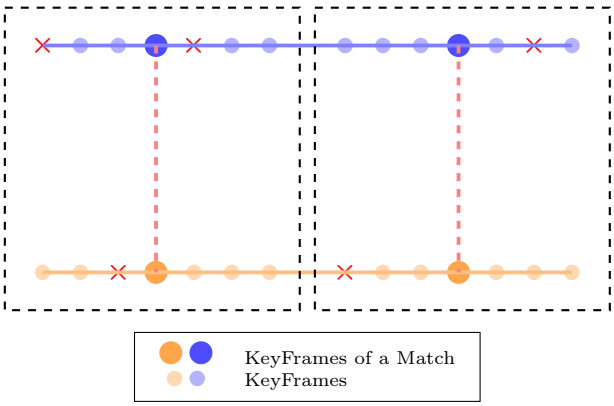


Figure 3.20: KFs culled

3.3 Optimization

Optimizations such as Pose Graph Optimization (PGO) and Bundle Adjustment (BA) are typically the most time consuming computations arising in a SLAM system like ORB-SLAM(2) [1], [2] on which this semester project is based on.
BA leads to a large scale optimization problem that is solved by simultaneously refining the 3D structure and viewing parameters, to achieve a reconstruction which is optimal. The optimization is obtained by using non-linear least squares algorithms, of which Levenberg-Marquardt (LM) has become a very popular choice [5].

In [5] it is argued, that considerable computational benefits can be gained by substituting the LM algorithm in the implementation of BA with a variant of Powell's dog leg (DL) non-linear least squares technique.

ORB-SLAM(2) also uses the LM algorithm for the PGO and the BA and therefore as stated in [5], this semester project could also benefit from better timing by using the DL algorithm instead of the LM algorithm.

As the reader might not be familiar to the LM and the DL algorithm in detail, a short description of them, based on [5], will follow in the next two sections.

3.3.1 Levenberg-Marquardt (LM) algorithm

The principle of the LM algorithm is to solve a non-linear optimization problem by iteratively solving a linear approximation of the original problem [7]. Therefore instead of solving $\min ||f(\mathbf{p})||_2$, f is replaced by a linear Taylor series expansion

$$f(\mathbf{p} + \delta_\mathbf{p}) \approx f(\mathbf{p}) + \mathbf{J}\delta_\mathbf{p} \tag{3.1}$$

where $\mathbf{J}$ denotes the Jacobian matrix $\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}$. In each iteration, the algorithm has to find the step $\delta_\mathbf{p}$ that minimizes

$$||\mathbf{x} - f(\mathbf{p} + \delta_\mathbf{p})|| \approx ||\mathbf{x} - f(\mathbf{p}) - \mathbf{J}\delta_\mathbf{p}|| \tag{3.2}$$

The solution to this minimum least square problem is obtained when $\mathbf{J}\delta_\mathbf{p} - \mathbf{x} + f(\mathbf{p})$ is orthogonal to the column space of $\mathbf{J}$, therefore $\mathbf{J}^T(\mathbf{J}\delta_\mathbf{p} - \mathbf{x} + f(\mathbf{p})) = \mathbf{0}$. Reordering this equation leads to:

$$\mathbf{J}^T\mathbf{J}\delta_\mathbf{p} = \mathbf{J}^T(\mathbf{x} - f(\mathbf{p})) \tag{3.3}$$

which is also known as the normal equations. The solution of the normal equations is the Gauss-Newton step $\delta_\mathbf{p}$. The LM method solves a variation of Equation 3.3, the so-called augmented normal equations:

$$(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\delta_\mathbf{p} = \mathbf{J}^T(\mathbf{x} - f(\mathbf{p})), \text{ with } \mu > 0 \tag{3.4}$$

Where $\mathbf{I}$ is the identity matrix and μ is called the damping term and makes the LM algorithm to a combination of the Gradient-descent and the Gauss-Newton method. When the current solution is far from a local minimum, the damping term μ is chosen large and the LM algorithm becomes a Gradient-descent method. If the current solution is close to a local minimum, the damping term μ is chosen small and the LM algorithm behaves like a Gauss-Newton method.

The LM algorithm controls the damping term μ the following way. If the updated parameter $\mathbf{p} + \delta_\mathbf{p}$ with $\delta_\mathbf{p}$ calculated from Equation 3.4 leads to a reduction in the error, the update is accepted and the process repeats with a decreased damping term μ. Otherwise, if the damping term μ is increased, the augmented normal

equations are solved again and the process iterates until a value of $\delta_{\mathbf{p}}$ is found, which decreases the error.

The disadvantage of the LM algorithm is, that Equation 3.4 has to be solved repeatedly until the error has decreased, in every iteration. As the result of Equation 3.4 can't be used if the error was not decreased, the LM performs a lot of unproductive effort.

3.3.2 Powell's dog leg (DL) algorithm

Similar to the LM algorithm, the DL algorithm also tries combinations of the Gauss-Newton and Gradient-descent directions. In contrast to the LM algorithm, the DL algorithm controls the combination of Gauss-Newton and Gradient-descent via the use of a trust region.

In a trust region framework, information regarding the function f is gathered and used to construct a quadratic model function L whose behavior in the neighborhood of the current point is similar to that of f. Within a hypersphere of radius Δ around the current point, the model function L is trusted to accurately represent f.

A new candidate step minimizing f is found by minimizing L over the trust region. The model function is

$$L(\delta) = 2(\frac{1}{2}(\mathbf{x} - f(\mathbf{p}))^T(\mathbf{x} - f(\mathbf{p})) - (\mathbf{J}(\mathbf{x} - f(\mathbf{p})))^T\delta + \frac{1}{2}\delta^T\mathbf{J}^T\mathbf{J}\delta) \quad \text{subjected to } ||\delta|| \leq \Delta \tag{3.5}$$

and the candidate step is the solution of the following subproblem:

$$\min_{\delta} L(\delta), \text{ subjected to } ||\delta|| \leq \Delta \tag{3.6}$$

The radius of the trust region is crucial to the success of a step and chosen based on the success of the model in approximating the objective function during the previous iterations.
If the model is accurately predicting the behavior of the objective function, the radius is increased to allow longer steps. On the other hand, if the model fails to predict the objective function over the current trust region, the radius of the latter is reduced and Equation 3.6 is solved again.
The solution of Equation 3.6 as a function of the trust region radius is a curve, shown in Figure 3.21. Powell [8] proposed to approximate this curve with a piecewise linear trajectory consisting of two line segments. The first line segment goes from the current point to the Cauchy point, given by

$$\delta_{sd} = \frac{\mathbf{g}^T\mathbf{g}}{\mathbf{g}^T\mathbf{J}^T\mathbf{J}\mathbf{g}}\mathbf{g} \tag{3.7}$$

The second runs from δ_{sd} to the Gauss-Newton step δ_{gn}, given by the solution of

$$\mathbf{J}^T\mathbf{J}\delta_{gn} = \mathbf{g} \tag{3.8}$$

For $\kappa \in [0, 2]$ the dog leg trajectory is then defined as

$$\delta(\kappa) = \begin{cases} \kappa\delta_{sd} & 0 \leq \kappa \leq 1 \\ \delta_{sd} + (\kappa - 1)(\delta_{gn} - \delta_{sd}) & 1 \leq \kappa \leq 2 \end{cases} \tag{3.9}$$

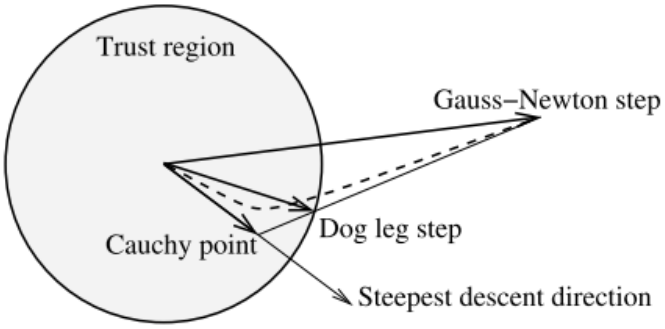


Figure 3.21: Dog leg approximation of the curved optimal trajectory (shown dashed) (Figure taken from [5])

The dog leg step is defined as follows: If the Cauchy point lies outside the trust region, the DL step is chosen as the truncated Cauchy step, i.e. the intersection of the Cauchy step with the boundary of the trust region. Otherwise, and if the Gauss-Newton step is within the trust region, the DL step is taken to be equal to it. Finally, when the Cauchy point is in the trust region and the Gauss-Newton step is outside, the next trial point is computed as the intersection of the boundary of the trust region and the straight line joining the Cauchy point and the Gauss-Newton step, as shown in Figure 3.21. In every case, the dog leg path intersects the boundary of the trust region at most once and the point of intersection can be determined analytically, without search.

The advantage of the DL algorithm is, that once the Gauss-Newton step has been determined, the DL algorithm can solve the subproblem for various values of Δ, without the need to resolve Equation 3.8.

3.3.3 Powell's dog leg (DL) vs. Levenberg-Marquardt (LM)

In the previous sections, the disadvantage of the LM algorithm and the advantage of the DL algorithm were already mentioned. In this section they will be recapitulated shortly and a conclusion will be drawn.

When a LM step fails, the LM algorithm has to resolve the augmented normal equations with an increased damping term. In other words, every update to the damping term requires that the augmented normal equations are solved again. Therefore failed steps entail unproductive effort.
Opposite to this, once the Gauss-Newton step has been calculated, the DL algorithm can solve the subproblem for various values of Δ without solving Equation 3.8 again.
Another fact is, that when the truncated Cauchy step is taken, the DL algorithm can avoid solving Equation 3.8, while the LM algorithm always needs to solve Equation 3.4, even if it chooses a step smaller than the Cauchy step.

Since the PGO and the BA problem involves many parameters, linear algebra costs dominate the computational overhead associated with every inner iteration of both algorithms (LM and DL) and therefore reducing the number of times that Equation 3.8 needs to be solved is crucial for the overall performance of the minimization process.
For the mentioned reasons, the DL algorithm is a more promising implementation of the non-linear minimization arising in PGO and BA in terms of the required

computational effort.

Chapter 4

Experimental Results

4.1 Experimental Setup

In this section it will be described how the proposed approaches were compared against each other and against the previous approach and how they were evaluated. First the data sets used for the comparison and the evaluation are described, then the The Robot Operating System (ROS) setup will be explained and in the last section, the evaluation procedure itself will be described.

4.1.1 Data sets

Both of the used data sets were captured in a room equipped with a motion tracking system, VICON, which provided the ground truth for the comparison and the evaluation. As the whole setup is based on ROS, the data sets were recorded with rosbags. The rosbags contain at least the following topics:

- The grayscale images as "sensor_msgs.msg.Image" in the topic "/cam0/image_raw"
- The ground truth as "geometry_msgs.msg.PointStamped" in the topic "/camera_imu/vrpn_client/estimated_transform"

The recorded rosbags were afterwards splitted into parts which do not contain any loop closure as loop closures would prohibit a fair comparison between the previous and the proposed new map fusion approach.

For the evaluation each client took the data from a seperate splitted rosbag simulating two clients running the same time.

Hand held

The hand held data sets (vi_loops_close and vi_loosp_far) were recorded with a camera rig walking loops while facing the wall. In the data set “vi_loops_close”, a frame of it shown in Figure 4.1, the camera was close to the wall. In the data set “vi_loops_far”, a frame of it shown in Figure 4.2, the camera was far from the wall.

Figure 4.1: Frame from the hand held data set close to the wall

Figure 4.2: Frame from the hand held data set far from the wall

Unmanned Aerial Vehicle (UAV)

The UAV used to record this data set (vi_loops_uav) is shown in Figure 4.3. In this data set loops were flown while the camera of the UAV was facing the wall. A frame of the recorded data set is shown in Figure 4.4.

Figure 4.3: The used UAV

Figure 4.4: Frame from the UAV data set

4.1.2 The Robot Operating System (ROS) setup

As already mentioned, the data sets for the two clients were created by splitting a bigger data set recorded with only one client. Because the split data sets don't have the same timing, a re-timing ROS node, explained in the next subsection, was written to work around this issue.
To get the ground truth in a usable format, a record ROS node, explained shortly, was written, which receives the ground truth position from the re-timing ROS node and writes it into a text file.

The whole ROS setup is illustrated in Figure 4.5.

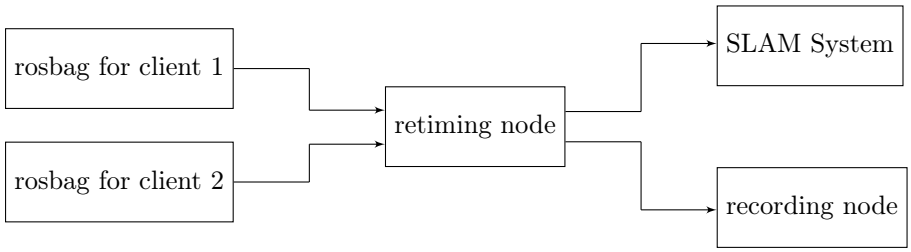


Figure 4.5: ROS setup

Re-timing node

The re-timing ROS node subscribes to the topics providing the ground truth position and the greyscale image and republishes the messages of these topics with the time stamp set to the current system time.

Recording node

The recording ROS node subscribes to the topic, which contains the re-timed ground truth position. It saves the ground truth positions into a list and at the end writes all the positions together with their time stamps into a text file.

4.1.3 Evaluation

To evaluate the proposed new map fusion approach and to measure the influence of the culling as well as the change of the optimization algorithm to the accuracy, an evaluation procedure was developed. The evaluation procedure performs the following steps:

1. Load ground truth and SLAM positions
2. Find corresponding (in time) ground truth and SLAM positions
3. Perform a 7DoF alignment between the ground truth and the SLAM positions
4. Calculate the Root Mean Squared Error (RMSE) between the ground truth and the SLAM positions

In the following subsections each step will be described in more detail.

Loading of the ground truth and SLAM positions

This step involves the loading of the positions from text files, applying the time offset t_{off} and applying the camera-to-marker transformation $T_{\text{camera\_marker}}$.

The offset t_{off} has to be applied as there is a time offset, emerging from the fact that the ROS messages with the greyscale image and the ROS messages with the ground truth position do not have the same time of travel, which has to be compensated. For the vi_loops_uav data set the time offset varies a lot as both topics are received over a unreliable wireless LAN connection. To counteract this situation different time offsets are tried by brute-forcing and the time offset which results in the smallest RMSE is taken.

As the camera and the marker of the motion tracking system do not have the same position, the transformation $T_{\text{camera\_marker}}$ has to be applied that the ground truth positions and the SLAM positions are in the same coordinate system for the evaluation.

Find corresponding ground truth and SLAM positions

In this step, for every SLAM position the first ground truth position with a time stamp higher than the one of the SLAM position is searched (line 14). After that, the absolute time difference between the time stamp of the SLAM position and the time stamp of the ground truth position, right before the one which was found in the previous step, is calculated (line 16). If this absolute time difference is smaller than the specified accuracy, the ground truth and the SLAM positions are saved in a new array (lines 16-18). The full procedure is listed in Listing 4.1.

Listing 4.1: Find corresponding positions in time

```
1  def time_matching(list_gt_x,list_slam_x,gt_coordinates,
2                    slam_coordinates,accuracy):
3
4    # Matching of the datasets according to their time stamps
5    gt = np.zeros((len(list_gt_x), 4))
6    slam = np.zeros((len(list_slam_x), 4))
7
8    pos = 0
9    offset = 1
10   break_flag = False
11   for i in range(len(list_gt_x)):
12     for j in range(offset, len(list_slam_x)):
13       # value where slam time > gt time
14       if gt_coordinates[i,0] < slam_coordinates[j,0]:
15         # check the one before
16         if abs(gt_coordinates[i,0]-slam_coordinates[j-1,0])
                                                  < accuracy:
17           gt[pos] = gt_coordinates[i]
18           slam[pos] = slam_coordinates[j - 1]
19           pos = pos + 1
20           if pos >= len(slam_coordinates):
21             break_flag = True
22             break
23           offset = j
24           continue
25     if break_flag==True:
26       break
27
28   # Remove empty rows
29   slam = slam[~(slam == 0).all(1)]
30   gt = gt[~(gt == 0).all(1)]
31
32   return slam, gt
```

Perform a 7DoF alignment

A 7DoF alignment is necessary as monocular SLAM systems, as the one this semester project is based on, doesn't provide a scale and position and rotation of the SLAM coordinate system and the one of the ground truth doesn't have to be the same one.

To perform the 7DoF alignment an implementation of the algorithm proposed in [9] was used. This algorithm determines the transformation and the scaling which re-

sults in the least RMSE. With this transformation and scaling, the SLAM positions are transformed and scaled accordingly.

Calculate the Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) between the ground truth and the SLAM positions is calculated according to

$$\mathrm{RMSE} = \sqrt{\frac{1}{N-1}\sum_{i}^{N}\left((x_{\mathrm{SLAM},i} - x_{\mathrm{gt},i})^2 + (y_{\mathrm{SLAM},i} - y_{\mathrm{gt},i})^2 + (z_{\mathrm{SLAM},i} - z_{\mathrm{gt},i})^2\right)} \tag{4.1}$$

where N is the total number of positions, $x_{\mathrm{SLAM},i}$, $y_{\mathrm{SLAM},i}$ and $z_{\mathrm{SLAM},i}$ are the positions of the SLAM system and $x_{\mathrm{gt},i}$, $y_{\mathrm{gt},i}$ and $z_{\mathrm{gt},i}$ are the ground truth positions provided by a motion tracking system.

4.2 Experiments

In this section, the performed experiments which show the performance of the proposed new map fusion approach, of the culling and of the different optimization algorithm settings are presented.

For every experiment, three runs were performed from which an average for the two maps was calculated. These averages as well as the average of the averaged map results are listed for every experiment.

4.2.1 Evaluation of suitable settings

Description

With the first data set, the vi_loops_close data set, a variety of different settings (number of KFMs and number of KFs to skip after a KFM) was evaluated to get an idea of the influence of the two parameters and to choose a selection of suitable settings for the other experiments.

Results

The results of this experiment are listed in Table 4.1.

Discussion

In Table 4.1 one can see, that with only three KFMs the RMSE gets higher. This may result from the fact, that with only three KFMs not much more information is gained while the computational effort is increased, as the transformation for every KFM is calculated and optimized. If this increased computational effort occurs more or less at once, this could lead to short time overload of the system which then results in a higher RMSE. The fact that the RMSE is decreased if KFs are skipped, supports this reasoning as with skipped KFs the increased computational effort is more spread over time because the time between the transformations of the KFMs are calculated and optimized is increased due to the skipping of KFs.
The setting with five KFMs and no KFs skipped also results in a worse RMSE which could be explained with the same reason mentioned before.
The settings with five and more KFMs and skipped KFs results in a lower RMSE and therefore a higher accuracy compared to the old approach.

# KFMs	# KFs skip	RMSE Map 1 [m]	RMSE Map 2 [m]	RMSE Average [m]
1	0	0.499 8	0.507 3	0.503 6
3	0	0.470 3	0.617 5	0.543 9
3	5	0.445 3	0.609 1	0.527 2
3	10	0.400 3	0.639 2	0.519 7
3	20	0.465 7	0.628 0	0.546 9
5	0	0.542 4	0.650 0	0.596 2
5	5	0.465 3	0.523 1	0.494 2
5	10	0.408 1	0.512 8	0.460 5
5	20	0.366 8	0.490 7	0.428 8
5	30	0.409 4	0.445 5	0.427 5
5	40	0.312 9	0.414 3	0.363 6
7	10	0.439 4	0.514 9	0.477 2
7	20	0.396 6	0.531 5	0.464 0
10	5	0.460 5	0.536 5	0.498 5
10	10	0.428 3	0.438 5	0.433 4
15	5	0.432 9	0.486 2	0.459 6
15	10	0.440 0	0.510 6	0.475 3

Table 4.1: RMSEs with the vi_loop_close data set

The selected settings which will also be evaluated in the other experiments are listed in Table 4.2.

# of KFMs	# of KFs skipped
5	20
10	5
10	10

Table 4.2: The selected settings

The selected settings may not be the ones performing best on the first data set (vi_loop_close) but they are also well suited for other data sets as they e.g. do not skip too many KFs or do not require too many KFMs which could lead to no map fusion in data sets with less overlap.

4.2.2 Proposed map fusion approach

Description

To evaluate the performance of the proposed map fusion approach, the approach was applied to the vi_loop_far data set and to the more challenging vi_loops_uav data set, with the chosen settings described in subsection 4.2.1.

Results

The results of the experiment with the vi_loop_far data set are listed in Table 4.3. In Table 4.4 the results of the experiment with the vi_loops_uav data set are listed.

# KFMs	# KFs skip	RMSE Map 1 [m]	RMSE Map 2 [m]	RMSE Average [m]
1	0	0.553 2	0.461 7	0.507 4
5	20	0.337 3	0.353 1	0.345 2
10	5	0.377 2	0.312 7	0.344 9
10	10	0.356 9	0.342 9	0.349 9

Table 4.3: RMSEs with the vi_loop_far data set

# KFMs	# KFs skip	RMSE Map 1 [m]	RMSE Map 2 [m]	RMSE Average [m]
1	0	0.114 1	0.148 1	0.131 1
5	20	0.073 6	0.108 7	0.091 2
10	5	0.112 9	0.134 2	0.123 6
10	10	0.084 6	0.107 5	0.096 1

Table 4.4: RMSEs with the vi_loop_uav data set

Discussion

In Table 4.3 and Table 4.4 one can see, that the RMSEs are also decreased for the selected settings with the data set vi_loop_far, respective vi_loop_uav.

The presented results show therefore, that with the usage of multiple KFMs and the skipping of KFs, the RMSE can be reduced, respective the accuracy can be increased. The reduced drift and the increased accuracy is visible, when comparing Figure 4.6 and Figure 4.7

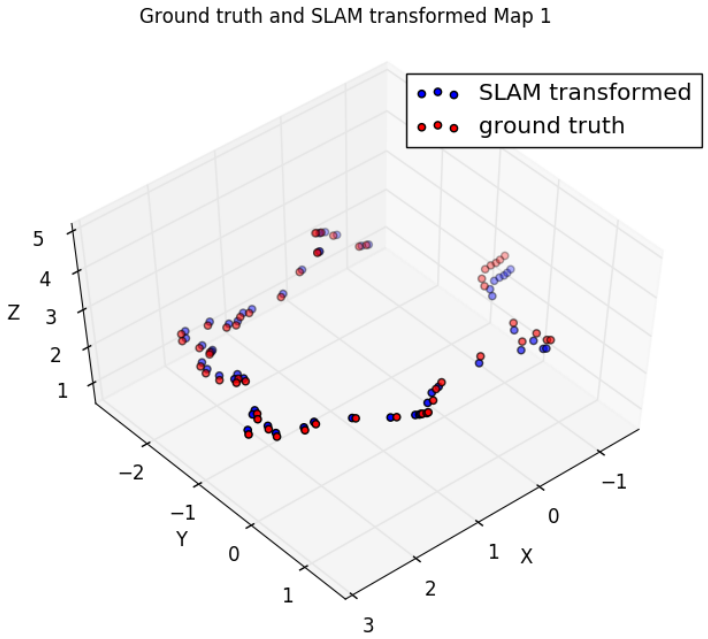


Figure 4.6: Drift of a trajectory with the previous approach

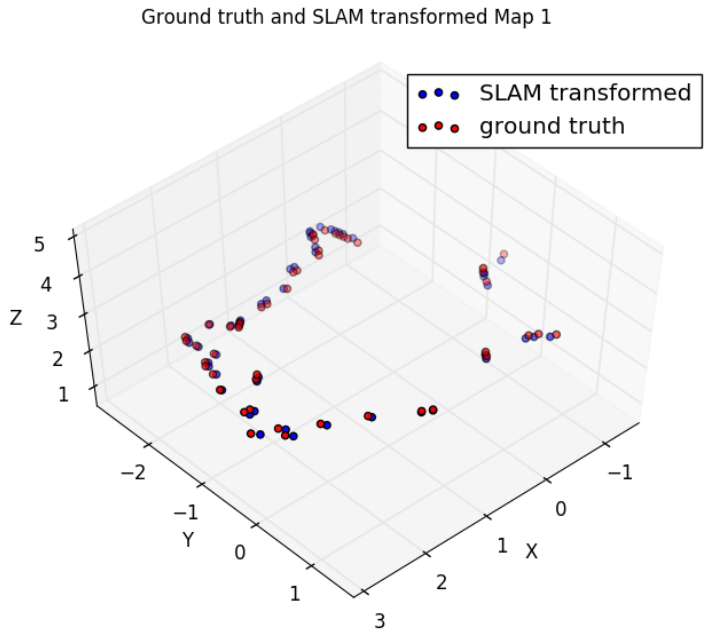


Figure 4.7: Drift of a trajectory with the proposed new approach

4.2.3 Culling

Description

To determine the influence of the culling, the proposed map fusion approach without and with culling was applied to the vi_loop_uav data set. The influence was measured by counting the number of edges and vertices the PGO and the BA had to process, by measuring the timing of the PGO and the BA and by calculating the RMSEs.

Results

Table 4.5 lists the number of edges and vertices the PGO and the BA had to process without culling for two representative settings and Table 4.6 lists the results with culling.

# KFMs	# KFs skip	PGO # edges	PGO # verteces	BA # edges	BA # verteces
1	0	754	59	22 193	2 308
10	10	1 949	150	49 765	4 806

Table 4.5: Number of edges and vertices in the PGO and in the BA without culling with the vi_loop_uav data set

# KFMs	# KFs skip	PGO # edges	PGO # verteces	BA # edges	BA # verteces
1	0	283	36	13 222	1 893
10	10	657	93	30 453	4 165

Table 4.6: Number of edges and vertices in the PGO and in the BA with culling with the vi_loop_uav data set

The timings of the fusion of Map Points (MPs), the PGO and the BA without and with culling are listed in Table 4.7 and Table 4.8.

# KFMs	# KFs skip	MPF [ms]	PGO [ms]	BA [ms]
1	0	0.00	142.25	163.89
5	20	44.31	649.52	2 112.18
10	5	57.41	372.42	1 334.04
10	10	98.12	532.28	3 659.48

Table 4.7: Timings of the MPF, the PGO and the BA with the vi_loop_uav data set without culling

# KFMs	# KFs skip	MPF [ms]	PGO [ms]	BA [ms]
1	0	0.00	43.01	72.33
5	20	10.08	119.78	410.91
10	5	16.24	86.09	209.41
10	10	33.78	178.83	1 098.37

Table 4.8: Timings of the MPF, the PGO and the BA with the vi_loop_uav data set with culling

Table 4.9 lists the RMSEs of the proposed map fusion approach with culling with the vi_loop_uav data set.

# KFMs	# KFs skip	RMSE Map 1 [m]	RMSE Map 2 [m]	RMSE Average [m]
1	0	0.199 1	0.238 2	0.218 7
5	20	0.115 0	0.131 5	0.123 2
10	5	0.075 0	0.088 3	0.081 7
10	10	0.083 1	0.109 9	0.096 5

Table 4.9: RMSEs with culling with the vi_loop_uav data set

Discussion

With culling, the number of edges and vertices the PGO and the BA have to process is reduced significantly compared to without culling, as shown in Table 4.5 and Table 4.6. The reduction of the number of edges in the PGO and in the BA is more than 60%, respectively 38% and the reduction of the number of vertices in the PGO and in the BA is up to 38%, respectively 13%, as shown in Table 4.10 and Table 4.11.

# KFMs	# KFs skip	PGO edges red. [%]	PGO vertices red. [%]
1	0	62.47	38.98
10	10	66.29	38.00

Table 4.10: Reduction of edges and vertices in the PGO with culling in percent

The reduction in edges and vertices then results in a decrease of the runtime as shown in Table 4.7 and Table 4.8

# KFMs	# KFs skip	BA edges red. [%]	BA vertices red. [%]
1	0	40.42	17.98
10	10	38.81	13.34

Table 4.11: Reduction of edges and vertices in the BA with culling in percent

As the culling removes information, one has to be careful not to remove to many information. With a redundancy threshold of 90%, the accuracy of the setting with 10 KFMs and 10 KFs skipped got worse. Because of this reason, another experiment with a redundancy threshold of 94% was performed instead, which showed a satisfactory result.

With the right choice of the redundancy threshold, the number of edges and vertices in the optimization can be reduced significantly which results in a reduced runtime while achieving the same accuracy.

4.2.4 Different optimization algorithms

Description

To compare the performance of the LM algorithm with the performance of the DL algorithm for the PGO and the BA in the proposed map fusion approach with culling, the timings and the RMSEs for all the cases were measured.

Results

The timings when the PGO and the BA both use the LM algorithm with the vi_loop_uav data set are listed in Table 4.8. In Table 4.12, the timings when the PGO and the BA both use the DL algorithm are listed.

# KFMs	# KFs skip	MPF [ms]	PGO [ms]	BA [ms]
1	0	0.00	48.75	68.24
5	20	7.89	139.30	226.00
10	5	14.27	92.79	164.97
10	10	35.29	243.35	446.75

Table 4.12: Timings of the MPF, the PGO and the BA both using the DL algorithm with the vi_loop_uav data set with culling

In Table 4.9, the RMSEs when the PGO and the BA both use the LM algorithm with the vi_loop_uav data set are listed. The RMSEs when the PGO and the BA both use the DL algorithm are listed in Table 4.13.

# KFMs	# KFs skip	RMSE Map 1 [m]	RMSE Map 2 [m]	RMSE Average [m]
1	0	0.190 7	0.254 5	0.222 6
5	20	0.075 9	0.109 9	0.092 9
10	5	0.065 6	0.089 1	0.077 3
10	10	0.061 6	0.095 2	0.078 4

Table 4.13: RMSEs with PGO using the LM algorithm and BA using the DL algorithm with the vi_loop_uav data set with culling

Discussion

The timings of both optimization procedures (PGO and BA) with the LM and the DL algorithm were recorded and presented in Table 4.8 and Table 4.12.
The best results are achieved if for the PGO the LM algorithm and for the BA the DL algorithm is used. The timings of this setup are summarized in Table 4.14.

# KFMs	# KFs skip	MPF [ms]	PGO [ms]	BA [ms]
1	0	0.00	43.83	68.77
5	20	6.74	99.80	204.36
10	5	16.00	93.45	164.58
10	10	33.95	178.70	383.54

Table 4.14: Timings of the MPF, the PGO and the BA, PGO using the LM algorithm and BA using the DL algorithm with the vi_loop_uav data set with culling

That only the BA performs better with the DL algorithm may come from the fact that the BA is a large scale, classically non-linear, optimization problem [5, 10], whereas the PGO in comparison is a smaller optimization problem, as the MPs stay fixed and are not optimized, and the LM algorithm performs better for smaller optimization problems.

With the change of the BA using the DL algorithm instead of the LM algorithm, the runtime of the optimization was decreased and therefore also the runtime of the whole system.

Chapter 5

Conclusion and Outlook

5.1 Conclusion

In this semester project, a novel map fusion approach was developed which uses multiple KFMs and skips KFs after each detected KFM. With the skipping of KFs, the KFMs are spread over a bigger area and together with the usage of multiple KFMs, more information can be gathered and provided to the optimization procedures which results in a reduction of drift and an improved accuracy. The novel approach therefore outperforms the previous approach in terms of accuracy.

The second part of this semester project, the introduced culling approach, removes redundant KFs and therefore the optimization procedures (PGO and BA) have to process less edges and vertices. This culling approach reduced the runtime significantly while maintaining the accuracy.

The evaluation of the LM and the DL algorithm for the optimization procedures has shown, that the best timing can be achieved if the PGO uses the LM algorithm and the BA uses the DL algorithm. With this setting the timing of the system was further improved.

With the new map fusion approach, the culling of KFs and the usage of optimal optimization algorithms with respect to the runtime, an existing multi agent SLAM system was improved which was the goal of this semester project.

5.2 Limitations

In the proposed map fusion approach, the number of KFMs and the number of KFs which are skipped after a KFM depends on the data set. If a data set contains a big overlapping area between the map of the clients, the number of KFMs and the number of KF skips can be chosen high and the map fusion approach will be able to fuse the maps with high accuracy.

If the same settings are now used with a data set that contains only a small overlapping area, the map fusion approach will probably fail as it won't be able to find the required number of KFMs.

Contrary, if a setting with a small number of KFMs and KF skips is used, the map fusion approach will be able to fuse the maps with both data sets but it might

achieve a lower accuracy on the data set with a large overlap compared to a setting with a higher number of KFMs and KF skips.

5.3 Outlook

The proposed map fusion approach takes the transformation of the first detected KFM to align the two maps. A more sophisticated heuristic approach could be used to determine the transformation of the KFMs which will result in the best alignment.

In the implementation of the KF culling only the first order neighbors (in the co-visibility graph) of the KFs of the KFMs are considered for the redundancy check. This could be adapted, that higher order neighbors in the co-visibility graph are also taken into consideration. In this case, one has to be careful to set the redundancy threshold to a value that the optimization procedures can still be provided with enough information not to reduce the accuracy.

Bibliography

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras," *arXiv*, no. October, 2016.

[3] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[4] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[5] M. I. A. Lourakis and A. A. Argyros, "Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?" in *Proceedings of the IEEE International Conference on Computer Vision*, vol. II, 2005, pp. 1526–1531.

[6] P. Schmuck and M. Chli, "Multi-UAV Collaborative Monocular SLAM," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017.

[7] W. Dahmen and A. Reusken, *Numerik für Ingenieure und Naturwissenschaftler*, ser. Springer-lehrbuch. Berlin: Springer, 2006.

[8] M. J. Powell, "A hybrid method for nonlinear equations," *Numerical methods for nonlinear algebraic equations*, vol. 7, pp. 87–114, 1970.

[9] S. Umeyama, "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[10] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment — A Modern Synthesis Vision Algorithms: Theory and Practice," *Vision Algorithms: Theory and Practice*, vol. 1883, pp. 153–177, 2000.