Institute for
Communications
and Navigation

Visual SLAM with Ranging Aid

Chiraz Nafouki

Supervised by Dr. Gabriele Giorgi
M.Sc. Chen Zhu

November 18, 2016

Institute for Communications and Navigation
Univ.-Prof. Dr. sc. nat. Christoph Günther

Technische Universität München
Theresienstrasse 90
80333 Munich

Submitted for the degree *Master of Science*
at the faculty of *Elektrotechnik und Informationstechnik*
at *Technische Universität München (TUM)*
Email: chiraz.nafouki@tum.de
Layout by LaTeX 2_ε

Abstract

In this master thesis, a bundle adjustment-based SLAM-with-ranging-aid algorithm is presented. The underlying theory and implementation details are given. The algorithm combines both visual cues and ranging measurements to a fixed reference in order to reduce the drift in visual SLAM. A description of the whole workflow from image acquisition and rectification to trajectory and map update is given. The algorithm is tested on real scenes using a VI-sensor as well as on scenes extracted from KITTI data set. We show that integrating range measurements into bundle adjustment allows us to reduce the drift in the trajectory.

Contents

Chapter 1

Introduction

The estimation of the movement of a camera and the construction of a map of the environment are two main tasks in robotics and autonomous visual navigation. These tasks are usually performed using Simultaneous Localization and Mapping (SLAM), the process by which a mobile robot builds a map of its environment and simultaneously localizes itself within the map. In this thesis, we focus on a particular category of SLAM which is visual SLAM (V-SLAM). Visual SLAM systems use cameras as sensors for trajectory and map estimation and can be classified into two main categories: monocular SLAM and stereo SLAM. Monocular SLAM systems use a monocular camera and can therefore estimate the trajectory and map only up to a scale. Stereo SLAM systems rely on stereo cameras which, unlike monocular cameras, do not introduce a scale ambiguity, but still result in a drift in the trajectory estimation due to the dead reckoning process inherent to SLAM. Dead Reckoning is the estimation of the robot's current position based on a previously determined position and on the relative estimated motion between the previous and current positions. This process is subject to cumulative error [SS10] which increases the drift. Therefore, drift correction is still a core topic in SLAM and many approaches have been proposed to deal with it. Some of the navigational aid methods for drift reduction combine V-SLAM with other sensors such as GPS or IMU or use loop closure detection, which relies on recognizing revisited places and previously observed landmarks for trajectory and map update. In this work, another approach for drift reduction is proposed, which allows us to update the trajectory and map even in absence of loop closure. The approach combines visual cues with ranging measurements that give the distance over discrete intervals of time to a fixed reference which has a known position. Ranging measurements are usually taken using a ranging equipment: an antenna placed at a known reference point in the environment to be explored emits signals that can be converted to ranging measurements by a robot placed at an arbitrary distance from the reference antenna and equipped with a wireless radio receiver. These ranging measurements are combined with the V-SLAM algorithm to update both the trajectory and map. This approach can work even in GPS denied environments and does not require expensive high-

accuracy IMUs or laser scanners.

In visual SLAM, there are two main approaches that are used to update the trajectory: filtering methods and non-filtering methods [SF11]. Filtering methods such as the Extended Kalman Filter (EKF) use all images and update the trajectory at each frame using a probabilistic framework. On the other side, non filtering-methods are optimization methods that check either the local or global consistency of the map and trajectory. In this thesis, we use a non-filtering method which is bundle adjustment. Our bundle adjustment algorithm uses visual cues as well as ranging measurements and performs a non-linear optimization that aims at minimizing the error between the predicted and measured parameters. This work gives the theory and implementation details of a bundle adjustment based V-SLAM algorithm that combines visual information with ranging measurements for drift mitigation. The report is organized as follows: Some related work in the field of drift reduction in SLAM is given in Chapter 2. The underlying theory of this thesis is given in Chapter 3. Chapter 4 describes the method of bundle adjustment with ranging aid used to reduce the drift. In Chapter 5, we present the workflow and give some implementation details. Experimental results using VI-sensor as well as scenes from KITTI data set are presented in chapter 6. Finally, the work is concluded in Chapter 7.

Chapter 2

Related Work

Different approaches have been proposed in ~~the~~ literature to deal with the drift issue in SLAM. One of the most known approaches is loop closure detection. This approach is based on recognizing that the autonomous agent has returned to a previously visited location. This is be usually achieved using a combination of visual and spatial appearance of local scenes [HN06]. Once a loop closure is correctly detected, the map and trajectory can be both updated based on the correlation of errors that charachterizes the SLAM problem. However, the method of loop closing constraints the movement of the robot by assuming that previously seen places are revisited, which should not always be the case. Furthermore, loop closing detection requires a good data association process to ensure a robust detection. Data association is however a computationally heavy operation that consumes a lot of memory and time. In [KSC13], additionally to loop closing detection, RGB-D cameras are used for a dense feature matching over all pixels by exploiting both photometric and depth information. Furthermore, a graph optimization method using g2o framework is suggested for drift reduction in the trajectory. However, the problem with RGB-D sensors such as Kinect is that they can not operate outdoors due to their infrared (IR) based ranging. Another commonly used approach is to integrate additional navigational aid into the SLAM algorithm. For example, integrating GPS [BAC13] or IMUs [Pet14] information into the visual SLAM system can help reduce the drift in the trajectory estimation. GPS provides an additional absolute localization which can reduce the impact of the drift for large scale motions. However, GPS can not be used in some environments such as indoor environments or on Mars. It also requires an additional sensor fusion process in order to be included with visual SLAM. On the other hand, due to noise, the errors in the IMU accumulate with time and inertial measurements are thus also subject to drift. Moreover, high-accuracy IMUs which have small errors in the trajectory estimation are ~~very expensive~~.

Recently, in [SZM+13] a swarm system of autonomous platforms was proposed for navigation on Mars. Apart from using inertial sensors, cameras and laser scanners, the swarm navigation system uses relative radio positioning systems. The radio

positioning system is exploited to get measurements of the distance between the swarm elements, which allows them to avoid collisions. In [ESZ14], a swarm system that exploits wireless signals as well as a hybrid time division (TDMA) and frequency division (FDMA) access scheme with ranging measurements was proposed. The ranging measurements are processed online using a particle filter and calculated using round-trip delay (RTD) approach. Round-trip delay measures the distance between two cooperative autonomous agents : a master and a slave. The master emits an OFDM modulated data packet to the slave which returns this packet. Based on time stamps of the transmission and reception of the packet, the master estimates the round-trip delay and deduces the range value between the two agents. In such a swarm system, the master clock must be stable in order to accurately estimate the ranging. In this thesis, we only rely on one fixed reference for range estimation. Therefore, this approach can work with only one agent but also with multiple agents as long as they can all receive the range information from the fixed reference. Furthermore, our approach does not rely on loop closure or RGB-D images and does not require the integration of IMU or GPS. Therefore, it can be used in both outdoor and indoor environments.

Chapter 3

Relevant Theory

In this Chapter the theory that underlies our work is presented. First, the required frames and notations are defined and the problem of this thesis is formulated. Moreover, some theoretical background which is useful for understanding the coming chapters is given. In the following, vectors and matrices appear in bold and $\mathbf{v}^T$ denotes the transpose of $\mathbf{v}$.

3.1 Problem Definition and Notations

3.1.1 Navigation Frame and World Frame

In this section, we examine a two dimensional simplification of the problem of a robot navigating autonomously in an unknown environment. This two-dimensional hypothesis is also accurate for cases in which the robot moves without a significant change in attitude, i.e. when the robot motion can be considered as planar. The problem considered in this work may be formulated as follows: a static base station emits ranging measurements that are captured by a moving robot. These measurements are used in addition to the V-SLAM algorithm which is based on the images provided at discrete time instants i.e. discrete frames by a calibrated camera (monocular or stereo camera) attached to the robot. For simplicity, we also assume that the position of the robot and the position of the camera are identical i.e. the camera coordinate frame is the same as the robot's coordinate frame. In case of using a stereo camera, we assume that the camera coordinate system is that of the left camera.
The V-SLAM algorithm uses visual odometry to give an initial estimation of the robot trajectory. This trajectory is calculated in the navigation frame denoted by (N). The Navigation frame is a fixed 2D orthonormal direct frame which has as origin the initial position of the robot and as horizontal axis the initial heading of the robot. We denote by x' and y' respectively the horizontal and vertical axis of (N). The trajectory of the robot obtained by visual odometry is therefore estimated with respect to the initial position of the robot. In case of a planar motion, the position

of the robot at the i^{th} frame is described by 3 variables in (N): two Cartesian coordinates x'_i and y'_i and an angle θ'_i which describes the relative heading of the robot with respect to (N). In these notations, the index i varies between 0 and $m-1$, where m represents the total number of image frames.Thus, the initial position of the robot is expressed in (N) by the triplet (x'_0, y'_0, θ'_0)
The trajectory estimated by the V-SLAM algorithm and expressed in (N) is however subject to drift which is due to the error propagation and to the accumulated errors in the estimation of the relative rotation and translation between each two successive camera frames. To reduce the drift and update the trajectory and map, we introduce ranging measurements that are estimations over discrete time intervals of the distance that separates the robot to a fixed reference that has a known position. The reference can for example be a base station antenna that emits the range measurements in form of radio signals. In order to be able to integrate the ranging information, the definition of another global reference frame called World Frame (W) is needed. The World Frame is also an orthonormal direct fixed frame and has as origin the base station antenna. The orientation of its axes is arbitrary. The distance between the origin of (W) and the robot at frame i is given by the ranging measurement ρ_i. The aim of our work is, given an initial trajectory estimation of the robot (x'_i, y'_i, θ'_i) obtained by a V-SLAM algorithm and a set of ranging measurements ρ_i, to be able to reduce the drift in the initial trajectory estimation and update both the trajectory and map. The trajectory and map update is done using the bundle adjustment method which will be detailed in section 3 of this chapter. Fig.3.1 shows both Navigation Frame and World Frame. The trajectory of the robot is depicted over 3 successive positions and the ranging measurements ρ_0, ρ_1 and ρ_2 corresponding to the three initial positions are also depicted in this figure.

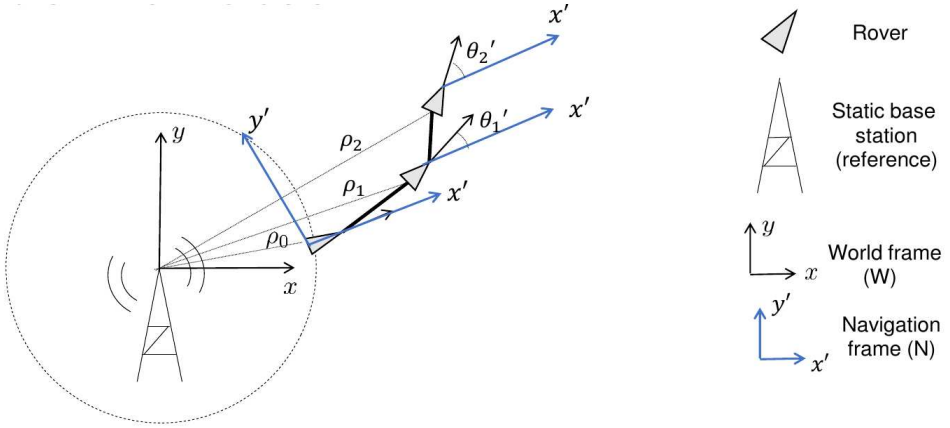


Figure 3.1: Navigation and World Frames

3.1.2 Initial Position Ambiguity

Let (x_i, y_i, θ_i), be the triplet that describes the i^{th} position of the robot in (W). If the initial position of the robot (x_0, y_0, θ_0) is known, then the trajectory of the robot in (W) can be deduced from the estimated trajectory in (N) (x'_i, y'_i, θ'_i) by applying a rigid-body transformation. However, we are usually only given the estimated trajectory in (N), which is provided by the visual odometry algorithm. The initial position of the robot in (W) (x_0, y_0, θ_0) is thus usually unknown. This uncertainty in the initial position of the robot in (W) can be reduced if we are given the initial ranging measurement ρ_0, which restricts the initial position of the robot to the circle φ of origin the origin of (W) and of radius ρ_0. However, given only the ranging measurements ρ_i and the estimated trajectory in (N) (x'_i, y'_i, θ'_i), it is impossible to know the exact initial position of the robot within the circle φ. Fig.3.2 visualizes the ambiguity in the initial position of the robot: given an estimated trajectory in (N) and a set of ranging measurements ρ_i, the whole trajectory of the robot in (W) can be rotated along the circle φ without any change in the ranging measurements ρ_i nor in the expressed trajectory in (N). In order to avoid this ambiguity in the initial position of the robot in (W), we assume that the robot is initially situated within the right half of the x-axis of (W). Therefore, the initial position of the robot in (W) is given by $(1, 0, \theta_0)$, where θ_0 expresses the initial heading of the robot with respect to (W) i.e. the angle between the x-axis and the x'-axis. This angle is not ambiguous and can be determined using the ranging measurements ρ_i and the estimated trajectory (x'_i, y'_i, θ'_i) in (N).

3.2 Projection of World Coordinates into Image Coordinates

3.2.1 Camera Model

In this section, we describe the model of the camera attached to the navigating robot. The model used is the perspective camera model which assumes a pinhole projection system. In a pinhole projection model, the image is formed by the intersection of the focal plane with the light rays coming from the objects and traversing the center of the lens [HZ03]. Furthermore, we assume that we have a calibrated camera, i.e. that the intrinsic parameters of the camera are known. Camera intrinsic parameters can be encapsulated in the camera matrix $\mathbf{K}$. For image planes with square pixels, the camera matrix $\mathbf{K}$ is given by:

$$K = \begin{bmatrix} f & 0 & P_x \\ 0 & f & P_y \\ 0 & 0 & 1 \end{bmatrix}$$

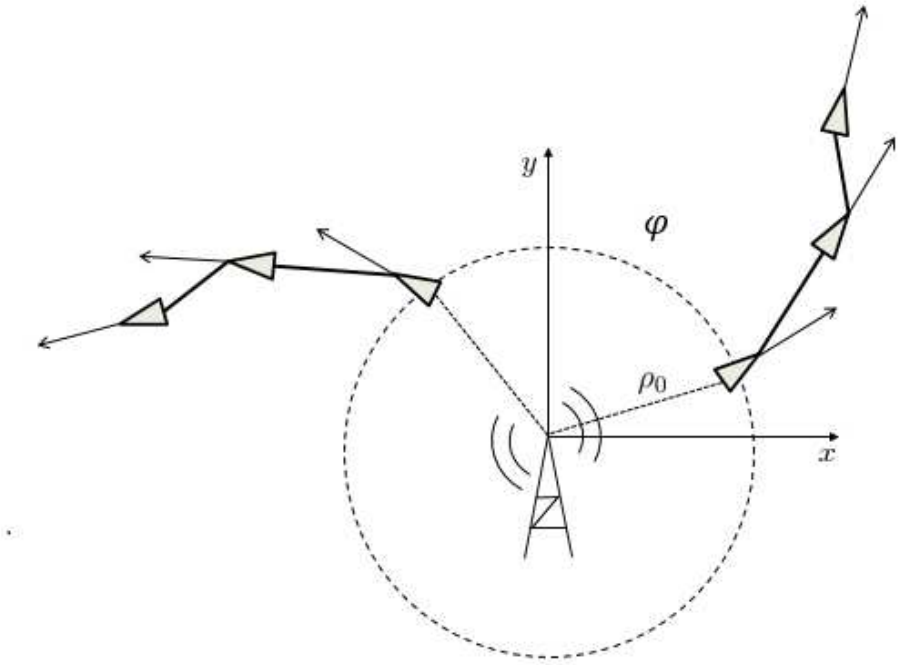


Figure 3.2: Position Ambiguity

where f is the focal length of the camera, P_x and P_y are the image coordinates of the principal point. The principal point is the point at the intersection of the optical axis of the camera with the image plane and is usually situated at the image center.

3.2.2 Projection of World Coordinates into Image Coordinates

As the rover moves in the Navigation Frame (N), the camera is subject to a rotation $\mathbf{R}$ and to a translation $\mathbf{t}$. $\mathbf{R}$ is a 3×3 rotation matrix describing the orientation of the camera frame wrt. (N) and $\mathbf{t}$ is a vector in $\mathbb{R}^3$ representing the location of the camera with respect to the origin of (N). These rotation and translation transformations form the camera extrinsic parameters and are used to define the camera projection matrix $\mathbf{P}$. The projection matrix $\mathbf{P}$ is obtained by multiplying the camera matrix $\mathbf{K}$ with the extrinsic parameters as follows:

$$\mathbf{P} = \mathbf{K}\ [\mathbf{R}|\mathbf{t}]$$

where $\mathbf{P}$ and $[\mathbf{R}|\mathbf{t}]$ are 3×4 matrices and $\mathbf{K}$ is a 3×3 matrix. The real-world 3D coordinates of the landmarks observed by the camera are projected onto the 2D image plane of the camera following the principle of projective geometry. The projection is given in homogeneous coordinates by:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

where $(u\ v\ 1)^T$ are the homogeneous image coordinates, $\mathbf{P}$ is the projection matrix and $(X\ Y\ Z\ 1)^T$ are the homogeneous coordinates of the 3D point.
Apart from matrix representation, a three-dimensional rotation can also be represented using quaternions. A quaternion $\mathbf{q}$ is generally represented in the form:

$$\mathbf{q} = \mathbf{q_w} + \mathbf{q_x}\mathbf{i} + \mathbf{q_y}\mathbf{j} + \mathbf{q_z}\mathbf{k}$$

where q_w, q_x, q_y and q_z are real numbers and $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are the fundamental quaternion units. Quaternion representation of rotations is more compact, more numerically stable and faster to interpolate than matrix representation. It also avoids the problem of gimbal lock related to the representation of rotations by Euler angles. Relations between matrix representation and quaternion representation can be found in appendix B.

3.3 Rigid Body Transformation

3.3.1 From Camera Frame to Navigation Frame

As mentioned in the previous sections, the trajectory of the robot and the map of the environment are both estimated in the Navigation Frame (N). If we denote by (k) the k^{th} camera frame, and by $\mathbf{C_k^{(N)}}$ the three dimensional vector representing the position of the camera at the k^{th} frame in (N), the extrinsic parameters $(\mathbf{R_k}, \mathbf{t_k})$ of the camera at frame k are given by:

$$\mathbf{R_k} = \mathbf{R}_{\mathbf{(N)\rightarrow(k)}} \tag{3.1}$$

$$\mathbf{t_k} = -\mathbf{R_k}\mathbf{C_k^{(N)}} \tag{3.2}$$

where $\mathbf{R}_{\mathbf{(N)\rightarrow(k)}}$ is the rotation matrix representing the rotation from (N) to (k). Furthermore, if we denote by $\mathbf{X^{(k)}}$ the coordinates of a 3D point in the k^{th} camera frame and by $\mathbf{X^{(N)}}$ the coordinates of this point in (N), then we have the following rigid body transformation between $\mathbf{X^{(k)}}$ and $\mathbf{X^{(N)}}$:

$$\mathbf{X^{(N)}} = \mathbf{R_k^T}.\mathbf{X^{(k)}} + \mathbf{C_k^{(N)}} \tag{3.3}$$

In homogeneous coordinates, this relation is given by:

$$\begin{pmatrix} \mathbf{X^{(N)}} \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R_k^T} & \mathbf{C_k^{(N)}} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \mathbf{X^{(k)}} \\ 1 \end{pmatrix} = \mathbf{\Pi_k} \begin{pmatrix} \mathbf{X^{(k)}} \\ 1 \end{pmatrix} \tag{3.4}$$

where $\mathbf{\Pi_k}$ is a 4×4 matrix that represents the pose of the camera at frame k i.e. the position and orientation of the camera relative to (N).

3.3.2 From Navigation Frame to World Frame

Let's denote by $\mathbf{C_k^{(W)}}$ the position of the robot at the k^{th} frame in (W). In section 3.1.2, we mentioned that in case the initial position of the robot in (W) i.e. $\mathbf{C_1^{(W)}}$ is known, the trajectory of the robot in (W) can be deduced from the estimated trajectory in (N) by applying a rigid-body transformation. In case of a planar motion and assuming that $\mathbf{C_1^{(W)}}$=[1 0]^T, this rigid body transformation is given by:

$$\mathbf{C_k^{(W)}} = \mathbf{C_k^{(N)}}\mathbf{R}(\alpha) + \mathbf{C_1^{(W)}} \tag{3.5}$$

where $\mathbf{R}(\alpha)$ is the rotation matrix representing the initial relative heading of the robot w.r.t. (W). This relative orientation can be described using a single angle α in case of a two-dimensional motion.

3.4 Visual Odometry

Visual Odometry (VO) allows a robot to estimate its ego-motion using only image frames as input. The image frames can be obtained using either a monocular or a stereo camera. In VO, the camera pose is incrementally estimated at each new position of the camera by computing the relative motion between the previous and current camera frame [SF11]. The relative motion between each two successive frames is estimated by detecting and matching features between images and then by examining the changes in the matched features. Therefore, the frequency of image frames should be high enough to ensure sufficient features overlaps between each two successive frames.
In monocular VO, the use of a single camera results in estimating the map and trajectory only up to a scale. Despite this drawback, a main advantage of using monocular cameras is that they have a significant less weight and cost. Moreover, stereo-cameras are "short-sighted": stereo VO degenerates to the monocular case when the distance to the scene is much larger than the camera baseline [HZ03]. On the other side, the interest in stereo VO is not only due to the absence of scale ambiguity, but also to the fact that, in stereo VO, coordinates of the 3D features can be easily obtained at each frame and for every stereo pair by triangulation. However, in monocular VO, the 3D point triangulation can be only obtained based on 2D point correspondences between two successive frames [HZ03]. Triangulation in the stereo case is further detailed in Chapter 5. For the monocular case, more details about point triangulation are given in Appendix C.
There are several methods in VO to compute the relative motion between the previous and current image. Based on the kind of feature correspondences, we distinguish three main methods [HN94]:

- **2D to 2D correspondences:** where features in both the previous and current images are taken in 2D coordinates.
- **3D to 3D correspondences:** where features in both the previous and

current images are specified in 3D coordinates.

• **3D to 2D correspondences:** where features of the previous image are specified in 3D coordinates, while features of the current image are given in 2D and correspond the reprojection of the 3D features of the previous image.

In this work, we use the method of 3D to 2D correspondences for relative motion estimation. This method is usually preferred over 3D to 3D correspondences because it was proven that 3D to 2D correspondences is more accurate especially for the stereo case [DNB04]. Moreover, this method is much faster than using 2D to 2D correspondences because it uses a lower number of point correspondences for motion estimation [SF12].

The method of 3D to 2D correspondences aims at finding the relative translation and rotation that minimize the sum of reprojection errors between the current and the previous image. This is equivalent to solving the following minimization problem:

$$\underset{\mathbf{t_{k-1,k}},\mathbf{R_{k-1,k}}}{\operatorname{argmin}} \sum_{i=1}^{n} ||\mathbf{u_i^k} - \mathbf{P_k}.\mathbf{X_i^{k-1}}||^2 \tag{3.6}$$

where $||.||$ denotes the euclidean norm, n is the total number of triangulated 3D points at frame $k-1$, $\mathbf{X_i^{k-1}}$ is a triangulated 3D point whose coordinates are expressed at the camera frame $k-1$. These coordinates can be obtained directly using equation 5.3 for the stereo case. In case of using a monocular camera, the point $\mathbf{X_i^{k-1}}$ is triangulated based on feature matching between frames $k-2$ and $k-1$ (see Appendix C). $\mathbf{P_k}$ is the projection matrix at frame k given by:

$$\mathbf{P_k} = \mathbf{K}[\mathbf{R_{k-1,k}}|\mathbf{t_{k-1,k}}]$$

where $\mathbf{K}$ is the camera matrix, $\mathbf{t_{k-1,k}}$ and $\mathbf{R_{k-1,k}}$ are respectively the relative translation and rotation from camera frame $k-1$ to camera frame k. $\mathbf{u_i^k}$ is the 2D feature at frame k corresponding to the point $\mathbf{X_i^{k-1}}$, obtained by feature matching between frames $k-1$ and k. In case of stereo VO, equation 3.6 considers only the left image projections and the relative motion parameters are estimated between the left camera frames. However, it is also possible to take into account the image projections in the right image frames by solving the following problem:

$$\underset{\mathbf{t_{k-1,k}},\mathbf{R_{k-1,k}}}{\operatorname{argmin}} \sum_{i=1}^{n} ||\mathbf{u_i^{k,l}} - \mathbf{P_{k,l}}.\mathbf{X_i^{k-1}}||^2 + ||\mathbf{u_i^{k,r}} - \mathbf{P_{k,r}}.\mathbf{X_i^{k-1}}||^2 \tag{3.7}$$

where $\mathbf{u_i^{k,r}}$ and $\mathbf{u_i^{k,l}}$ denote respectively the image projections in the right and left image frames, and $\mathbf{P_{k,r}}$ and $\mathbf{P_{k,l}}$ are respectively the projection matrices of the right and left cameras. Therefore, 3.7 aims at minimizing the sum of reprojection errors for both the right and left camera frames in the stereo case.

Solving the least square minimization problem of equation 3.6 or equation 3.7 refers to minimizing the error between the predicted image projections of the 3D points

$\mathbf{X_i^{k-1}}$ into the current frame and the measured image projections. As we will see in the next chapter, this least square problem, known as perspective from n points (PnP), is similar to the one solved in bundle adjustment. There are different possible ways to solve it in the literature. The standard method is called perspective from three points (P3P). It uses 3 3D to 2D point correspondences and is described in [LKS11].

The result of the relative motion estimation i.e. the estimated translation and rotation between each two successive frames are incrementally concatenated to get the pose at the current frame w.r.t. (N). Let $\mathbf{\Pi_{k-1}}$ and $\mathbf{\Pi_k}$ be respectively the pose of the camera at frames $k-1$ and k, and let $\mathbf{t_{k-1,k}}$ and $\mathbf{R_{k-1,k}}$ be respectively the relative translation and rotation between frames $k-1$ and k, obtained by VO. We have then the following relation between $\mathbf{\Pi_k}$ and $\mathbf{\Pi_{k-1}}$:

$$\mathbf{\Pi_k} = \mathbf{\Pi_{k-1}} \begin{bmatrix} \mathbf{R_{k-1,k}} & \mathbf{t_{k-1,k}} \\ 0 & 1 \end{bmatrix}^{-1} \tag{3.8}$$

Through this recursive relation, it is possible to get all camera poses $\mathbf{\Pi_{1:m}}$, where m is the total number of frames and where $\mathbf{\Pi_1} = \mathbf{I_4}$, $\mathbf{I_4}$ being the 4×4 identity matrix.

3.5 Outlier Rejection

The estimation of relative motion through visual odometry requires matching features between successive frames. This matching is however subject to outliers mainly due to noise, occlusions and change in illumination or viewpoint. An important number of outliers can have a significant impact by making the motion estimation erronous. Therefore, we need to integrate an outlier rejection step into the motion estimation process in order to make it more accurate and robust against wrong correspondences. Random Sample Consensus (RANSAC) [AB81] is considered as the standard method for outlier removal. Using this method, it is possible to estimate the motion parameters accurately even in presence of outliers. RANSAC is based on the idea of taking a subset of random samples at each iteration and on computing the parameters of the model to be estimated based on these samples. Then, the accuracy of the model is verified using the other data points. The model that shows the highest consensus with the other data is taken as solution and data points that are outside a certain "acceptance region" of the subgroup with largest consensus are eliminated as outliers. In our case, in order to compute the relative motion between two camera positions, RANSAC selects at each iteration some subsets of 3 random 3D to 2D correspondences [BKL10]. Based on these correspondences, the motion parameters (relative rotation and translation) are computed. The number N of used subsets is given by:

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)} \tag{3.9}$$

where s is the minimum number of data points needed for motion estimation (here $s = 3$), p is the probability that at least one sample has no outliers and ϵ is the assumed percentage of outliers in the data set. For each estimated motion parameters, the number of inliers using the Euclidean reprojection error. A feature is considered as an inlier, if the Euclidean reprojection error (introduced in the previous section) is lower than a certain threshold δ. The motion parameters that give the highest number of inliers is retained and a final estimation step with all inliers of the best sample is performed to give the final egomotion estimation.

Chapter 4

Bundle Adjustment with Ranging Measurements

In this chapter, the original version of bundle adjustment as known in the state-of-the-art is presented as well as our extension of this method to include ranging measurements. We focus particularly on solving the BA problem using Levenberg Marquardt algorithm and give some details of this algorithm.

4.1 Bundle Adjustment

4.1.1 Problem and Notations

Bundle adjustment (BA) is a non-linear optimization method that aims at refining both the map and camera poses by minimizing the sum of reprojection errors for a subset of frames [LA04]:

$$\underset{\{\mathbf{X_i^{(N)}}\}_{\mathbf{i}\in[\mathbf{1}:\mathbf{n}]},\{\mathbf{P_j}\}_{\mathbf{j}\in[\mathbf{1}:\mathbf{m}]}}{\operatorname{argmin}} \sum_{j=1}^{m}\sum_{i=1}^{n} ||\mathbf{u_i^j} - \mathbf{P_j}.\mathbf{X_i^{(N)}}||^2 \tag{4.1}$$

add that we assume that all points are visible at all frames but its also applicable if not replace the whole difference by 0
where $||.||$ denotes the euclidean norm, m is the total number of frames considered in the optimization, n is the total number of 3D points, $\mathbf{X_i^{(N)}}$ is the i^{th} triangulated 3D point whose coordinates are expressed in (N). $\mathbf{P_j}$ is the projection matrix at frame j which contains the camera extrinsic parameters expressed w.r.t. (N):

$$\mathbf{P_j} = \mathbf{K}[\mathbf{R_j}|\mathbf{t_j}]$$

where $\mathbf{K}$ is the camera matrix, $\mathbf{t_j}$ and $\mathbf{R_j}$ are respectively the translation and rotation of camera frame j w.r.t.(N). Their definitions were given in 3.1 and 3.2. In this thesis, we assume that we have a calibrated camera i.e. that the intrinsic camera

matrix $\mathbf{K}$ is known. Thus, only the extrinsic motion parameters $\mathbf{t_j}$ and $\mathbf{R_j}$ need to be estimated. However, it is possible to apply BA to optimize over the camera intrinsic parameters in addition to the structure and extrinsic parameters. $\mathbf{u_i^j}$ is the image point of the 3D point $\mathbf{X_i^{(N)}}$ measured at the j^{th} frame. The error function (or cost function) to be minimized in 4.1 is the reprojection error and is similar to the cost function considered in 3.6. However, there are three main differences between the problem of BA and the minimization problem solved at each step of the VO algorithm and expressed in 3.6. The first difference is that BA does not only optimize over the motion parameters but also over the structure i.e. over the 3D points of the map. The second difference is that the cost function is minimized over the structure and poses w.r.t. the Navigation Frame (N), while the motion parameters in 3.6 were estimated at each frame w.r.t. the previous camera frame. The last difference is that the optimization in BA is not done for only two successive images but for more than two images. This results in a better drift reduction because the consistency with the measurements is checked over more than two image frames [SF12]. Based on the number of images considered in the BA problem, we distinguish two main categories of BA [EL09]:

• **Global BA:** where the optimization is done considering all camera images. Global BA is usually performed offline as a post processing step after the V-SLAM algorithm in order to refine the whole trajectory and map at the same time.

• **Local (or windowed) BA:** where the optimization is performed only over a window of the last m frames. Therefore, only a part of the trajectory and 3D map is refined.

In the rest of our work, we will consider global BA i.e. all images, but the underlying theory that will be presented in this chapter is also applicable for local BA. To simplify the notations and to avoid matrix representations, we will denote in the coming sections by $\mathbf{a_j}$ the vector representing the camera motion parameters at frame j, estimated using BA. If we use quaternions instead of matrices to represent rotations, we can see that only 4 parameters are needed to represent a three-dimensional rotation. Moreover, a translation in space is represented using three parameters. In sum, $\mathbf{a_j}$ has a size of 7: 4 rotation parameters and 3 translation parameters. we will denote by $\mathbf{b_i}$ the vector representing the i^{th} 3D point. $\mathbf{b_i}$ has a size of 3 and contains the three coordinates X_i, Y_i and Z_i of the i^{th} point w.r.t.(N). Let $\mathbf{p}$ be the vector containing all the motion and structure parameters to be estimated by BA:

$$\mathbf{p} = (\mathbf{a_1^T}, \mathbf{a_2^T}, ..., \mathbf{a_m^T}, \mathbf{b_1^T}, \mathbf{b_2^T}, ..., \mathbf{b_n^T})^\mathbf{T}$$

where m is the total number of frames and n is the total number of 3D points. Therefore, the total number of unknowns i.e. the size of $\mathbf{p}$ is $7m + 3n$. Let $\mathbf{Q}$ be the projection function which projects the i^{th} 3D point into the j^{th} frame as follows:

$$\mathbf{Q}(\mathbf{a_j}, \mathbf{b_i}) = \mathbf{P_j}.\mathbf{X_i^{(N)}}$$

. We will also substitute the notation $\mathbf{u_i^j}$ by $\mathbf{u_{i,j}}$ and we will denote by $\mathbf{U}$ the vector containing all measurements $\{\mathbf{u_{i,j}}\}_{\mathbf{i}\in[\mathbf{1},\mathbf{n}],\mathbf{j}\in[\mathbf{1},\mathbf{m}]}$, defined as follows:

$$\mathbf{U} = (\mathbf{u_{1,1}^T}, ..., \mathbf{u_{1,m}^T}, \mathbf{u_{2,1}^T}, ..., \mathbf{u_{2,m}^T}, ..., \mathbf{u_{n,1}^T}, ..., \mathbf{u_{n,m}^T})^T$$

where $\mathbf{u_{i,j}}$ is the image projection of the i^{th} 3D point into the j^{th} frame. If a point i is not seen at frame j , the element $\mathbf{u_{i,j}}$ is simply replaced by the null vector. Since $\mathbf{u_{i,j}}$ is expressed in pixel coordinates, it has a size of 2. Therefore, the maximum number of measurements i.e the size of $\mathbf{U}$ is $2 \times m \times n$. The problem to be solved in 4.1 can then be reformulated as follows using the new notations:

$$\underset{\mathbf{a_j},\mathbf{b_i}}{\operatorname{argmin}} \sum_{j=1}^{m}\sum_{i=1}^{n} ||\mathbf{u_{i,j}} - \mathbf{Q}(\mathbf{a_j}, \mathbf{b_i})||^2 \tag{4.2}$$

Given an initial motion and structure parameters estimation, the aim of BA is to refine the values of the parameters $\mathbf{a}_j$ and $\mathbf{b}_i$. In other terms, BA aims at finding the vector $\mathbf{p}$ that minimizes the reprojection error i.e. the difference between the predicted image projections $\mathbf{Q}(\mathbf{a_j}, \mathbf{b_i})$ and the measurements $\mathbf{u_{i,j}}$. This is a non-linear optimization problem because the projection function $\mathbf{Q}$ is non-linear in $\mathbf{a}_j$ and $\mathbf{b}_i$. In order to solve this problem, BA uses Levenberg-Marquardt (LM) algorithm, which is the standard technique for non-linear least-squares problems [LA04]. In the following section, some elements of the solution using LM algorithms are given.

4.1.2 Levenberg-Marquardt Algorithm

Levenberg-Marquardt Algorithm is an iterative minimization technique that aims at finding a parameter vector $\mathbf{p}$ that minimizes the squared error between an observed measurement vector $\mathbf{U}$ and the output of an objective function $\mathbf{f}$, where the objective function is a non-linear function in $\mathbf{p}$ that returns the predicted measurement vector [W.M63]. In our case, the objective function $\mathbf{f}$ returns the predicted projections of the n 3D points into the m image frames and is defined by:

$$\mathbf{f(p)} = (\mathbf{Q(a_1, b_1)^T}, ..., \mathbf{Q(a_m, b_1)^T}, \mathbf{Q(a_1, b_2)^T}, ..., \mathbf{Q(a_m, b_2)^T}, ..., \mathbf{Q(a_1, b_n)^T}, ..., \mathbf{Q(a_m, b_n)^T})^\mathbf{T} \tag{4.3}$$

where $\mathbf{p}$, $\mathbf{a}_i$ and $\mathbf{b}_j$ are the parameter vectors defined in the previous section. Therefore, LM algorithm aims at solving the following problem:

$$\underset{\mathbf{p}}{\operatorname{argmin}} ||\mathbf{U} - \mathbf{f(p)}||^2 \tag{4.4}$$

which is equivalent to solving the problem 4.2 of BA.

Given an initial estimate $\mathbf{p}_0$ of $\mathbf{p}$ and a measurement vector $\mathbf{U}$, LM algorithm uses a combination of steepest descent and of Gauss Newton methods: at each iteration,

if the solution is far from the correct one (i.e if the reprojection error is relatively high), LM behaves as the steepest descent algorithm. The steepest descent is slower than Gauss Newton method but its convergence is guaranteed. On the other side, if the current solution is close to the correct one, LM becomes equivalent to Gauss Newton method which is characterized by a fast convergence. In order to find the optimal $\mathbf{p}$, LM looks at each iteration in the neighborhood of $\mathbf{p}$ and finds a vector $\delta_{\mathbf{p}}$ that minimizes the reprojection error $||\mathbf{U} - \mathbf{f}(\mathbf{p} + \delta_{\mathbf{p}})||$. This expression can be simplified usings the Taylor series expansion:

$$\mathbf{f}(\mathbf{p} + \delta_{\mathbf{p}}) = \mathbf{f}(\mathbf{p}) + \mathbf{J}\delta_{\mathbf{p}} \tag{4.5}$$

where $\mathbf{J}$ is the Jacobian matrix $\frac{\partial \mathbf{f}(\mathbf{p})}{\partial \mathbf{p}}$ and where $\delta_{\mathbf{p}}$ is a vector with an infinitesimal norm. Therefore, this equation gives us a linear approximation of $\mathbf{f}$ in the neighborhood of $\mathbf{p}$. Using the equality in 4.5, we have:

$$||\mathbf{U} - \mathbf{f}(\mathbf{p} + \delta_{\mathbf{p}})|| = ||\mathbf{U} - \mathbf{f}(\mathbf{p}) - \mathbf{J}\delta_{\mathbf{p}}|| \tag{4.6}$$

Let's denote by ϵ the error $\mathbf{U} - \mathbf{f}(\mathbf{p})$. Therefore, at each iteration step of the LM algorithm, the optimal $\delta_{\mathbf{p}}$ that minimizes the quantity $||\epsilon - \mathbf{J}\delta_{\mathbf{p}}||$ is found and the vector $\mathbf{p}$ is substituted by $\mathbf{p} + \delta_{\mathbf{p}}$. It was proven that the solution $\delta_{\mathbf{p}}$ is the same solution to the so-called augmented normal equation [LA04]:

$$\mathbf{N}\delta_{\mathbf{p}} = \mathbf{J}^{\mathbf{T}}\epsilon \tag{4.7}$$

with the matrix $\mathbf{N}$ defined as follows:

$$\mathbf{N_{ij}} = \begin{cases} [\mathbf{J}^{\mathbf{T}}\mathbf{J}]_{\mathbf{ii}} + \alpha & if\ i = j \\ [\mathbf{J}^{\mathbf{T}}\mathbf{J}]_{\mathbf{ij}} & else \end{cases} .$$

where α is a damping parameter adjusted at each iteration by the LM algorithm in a way that reduces the error ϵ [Lam97]: if ϵ is high, the value of α is increased and LM behaves as the steepest descent algorithm. Otherwise, the damping value is kept small and in this case LM algorithm is equivalent to Gauss-Newton method. LM algorithm terminates in case one of these three conditions is fulfilled [LA04]:

- The norm of $\mathbf{J}^{\mathbf{T}}\epsilon$ drops below a certain threshold.
- The relative change in $||\delta_p||$ drops below a threshold.
- A fixed maximum number of iterations is reached.

We will not go into more details of solving the augmented normal equation using LM algorithm. The interested reader can find more details in Appendix 6 of [HZ03]. However, it is worth mentioning that, in order for the LM algorithm to be able to converge fast to the global minimum, good initial motion and structure estimates should be provided as input. If the initial trajectory or map estimation is very poor i.e. if they are very far from the global minimum, the algorithm could converge to a local minimum and the trajectory and map estimation will not be

necessarily improved. In order to avoid this case, a good visual odometry algorithm with an outlier rejection scheme need to be used and only triangulated points with low uncertainty should be included into the map.

Since the number of unknowns in the case of BA increases linearly with the number of frames and with the number of 3D points, the computational cost of using LM algorithm can rapidly increase in case of a large map or a relatively important number of frames. Indeed, solving the augmented normal equation 4.7 in one iteration of LM algorithm has an $O(M^3)$ complexity, with M being the total number of parameters, i.e. $7m + 3n$ in the case of bundle adjustment for motion and structure. However, thanks to the lack interaction between the different motion and structure parameters, a sparse version of the LM algorithm can be used, which significantly reduces the computational time [HZ03]. In the following section, we give a simplified example with a reduced number of frames m and 3D points n in order to explain the sparseness characteristic in the BA problem which makes it possible to use a sparse LM algorithm.

4.1.3 A Sparse Levenberg-Marquardt Algorithm

Let's consider a smplified case of BA with only 3 frames (the minimum number of frames in a BA problem) and 4 3D points. Furthermore, we assume that all points are observed in all frames. Normally, in order to be able to solve a BA problem, the number of measurements should be bigger than the number of unkowns. In case of 3 frames and 4 3D points, the number of unkowns is $7 \times 3 + 3 \times 4 = 33$ and the number of measurements is $2 \times 4 \times 3 = 24$. It is clear then that 4 points is not enough to solve this BA problem because the number of measurements is less than that of unknowns. Indeed, if we assume that all points are seen in all frames, the minimum required number of 3D points is 7. In this case the number of unkonwns and measurements are both equal to 42 and thus, it is possible to find the unknown motion and structure parameters using LM algorithm. However, in order to avoid big matrices and for simplification, we will consider only 3 frames and 4 points but the coming reasoning can be applied for a bigger number of frames and 3D points. Thus, the parameter vector $\mathbf{p}$ is given by:

$$\mathbf{p} = (\mathbf{a_1^T}, \mathbf{a_2^T}, \mathbf{a_3^T}, \mathbf{b_1^T}, \mathbf{b_2^T}, \mathbf{b_3^T}, \mathbf{b_4^T})^\mathbf{T}$$

and the measuerement vector $\mathbf{U}$ is given by:

$$\mathbf{U} = (\mathbf{u_{1,1}^T}, \mathbf{u_{1,2}^T}, \mathbf{u_{1,3}^T}, \mathbf{u_{2,1}^T}, \mathbf{u_{2,2}^T}, \mathbf{u_{2,3}^T}, \mathbf{u_{3,1}^T}, \mathbf{u_{3,2}^T}, \mathbf{u_{3,3}^T}, \mathbf{u_{4,1}^T}, \mathbf{u_{4,2}^T}, \mathbf{u_{4,3}^T})^T$$

Solving the augmented normal equation 4.7 by LM algorithm requires the computation of the Jacobian matrix $\mathbf{J}$ of the objective function $\mathbf{f}$. The Jacobian can be either computed analytically or approximated using the finite-differences method. The analytical estimation of the Jacobian requires to know the partial derivatives of

the objective funtion $\mathbf{f}$ which is possible since we know the analytical expression of $\mathbf{f}$. On the other hand, finite-differences is a numerical method used to approximate the elements of the Jacobian matrix without needing to compute the analytical derivatives of $\mathbf{f}$. This method consumes less computation time but is not as acccurate as the analytical method. It uses the Forward Difference Approximation [BD71]. Let's denote by $\mathbf{A_{ij}}$ the 2×7 matrix $\frac{\partial \mathbf{Q}(\mathbf{a_j}, \mathbf{b_i})}{\partial \mathbf{a_j}}$ and by $\mathbf{B_{ij}}$ the 2×3 matrix $\frac{\partial \mathbf{Q}(\mathbf{a_j}, \mathbf{b_i})}{\partial \mathbf{b_i}}$. Using these notations the Jacobian matrix $\mathbf{J}$ in block representation is given by:

$$\mathbf{J} = \frac{\partial \mathbf{f}(\mathbf{p})}{\partial \mathbf{p}} = \begin{pmatrix} \mathbf{A_{11}} & \mathbf{0} & \mathbf{0} & \mathbf{B_{11}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A_{12}} & \mathbf{0} & \mathbf{B_{12}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A_{13}} & \mathbf{B_{13}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A_{21}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B_{21}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A_{22}} & \mathbf{0} & \mathbf{0} & \mathbf{B_{22}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A_{22}} & \mathbf{0} & \mathbf{B_{23}} & \mathbf{0} & \mathbf{0} \\ \mathbf{A_{31}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B_{31}} & \mathbf{0} \\ \mathbf{0} & \mathbf{A_{32}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B_{32}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A_{33}} & \mathbf{0} & \mathbf{0} & \mathbf{B_{33}} & \mathbf{0} \\ \mathbf{A_{41}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B_{41}} \\ \mathbf{0} & \mathbf{A_{42}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B_{42}} \\ \mathbf{0} & \mathbf{0} & \mathbf{A_{43}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B_{43}} \end{pmatrix} \tag{4.8}$$

It is clear that the Jacobian matrix $\mathbf{J}$ has a sparse structure since there are a lot of zero blocks. This is due to the fact that the projection of a 3D point i onto a frame j depends only on the position of the camera at the j^{th} frame and not on its positions at other frames. Moreover, the projection of the of a 3D point i onto a frame j is independent of the positions of other 3D points. Mathematically, this means that $\frac{\partial \mathbf{Q}(\mathbf{a_j}, \mathbf{b_i})}{\partial \mathbf{a_k}} = \mathbf{0}$ for $k \neq j$ and that $\frac{\partial \mathbf{Q}(\mathbf{a_j}, \mathbf{b_i})}{\partial \mathbf{b_k}} = \mathbf{0}$ for $k \neq i$. We remind that $\mathbf{J}$ in 4.8 is expressed in case all points are seen in all images. If, additionally, some 3D points are not visible in some images, some of the $\mathbf{A_{ij}}$ and $\mathbf{B_{ij}}$ may be equal to null matrix and the the Jacobian matrix $\mathbf{J}$ will have a sparser structure. For example, if the first 3D point $\mathbf{b_1}$ is not visible at the third frame then $\mathbf{Q}(\mathbf{a_3}, \mathbf{b_1}) = \mathbf{0}$. Therefore, $\mathbf{A_{13}} = \frac{\partial \mathbf{Q}(\mathbf{a_3}, \mathbf{b_1})}{\partial \mathbf{a_3}} = \mathbf{0}$ and $\mathbf{B_{13}} = \frac{\partial \mathbf{Q}(\mathbf{a_3}, \mathbf{b_1})}{\partial \mathbf{b_1}} = \mathbf{0}$. This sparseness in the Jacobian matrix enables the BA problem to use a sparse version of LM algorithm which saves a lot of computational power. Details about the sparse LM algorithm can be found in [LA04].

4.2 Bundle Adjustment with Ranging Measurements

As seen in the previous section, the original version of BA is meant to find the optimal motion and structure parameters that minimize the reprojection error. In this section, we extend the original version of the BA problem in order to consider not only the reprojection error but also ranging information.

4.2.1 Problem and Notations

Compared to the original BA problem expressed in equation 4.1, our extended version of BA integrates ranging measurements into the BA cost function and aims at simultaneously minimizing the reprojection error and the difference between the predicted ranging and the the measured ones:

$$\underset{\{\mathbf{X_i^{(N)}}\}_{\mathbf{i}\in[\mathbf{1:n}]},\{\mathbf{P_j}\}_{\mathbf{j}\in[\mathbf{1:m}]}}{\operatorname{argmin}} \sum_{j=1}^{m}\sum_{i=1}^{n} ||\mathbf{u_{i,j}} - \mathbf{P_j}.\mathbf{X_i^{(N)}}||^2 + ||\rho_j - \hat{\rho_j}||^2 \tag{4.9}$$

where ρ_j denotes the ranging measurement and where $\hat{\rho_j} = ||\mathbf{C_j^{(W)}}||$ is the predicted ranging at frame j with $\mathbf{C_j^{(W)}}$ being the camera position in World Frame (W) at frame j. This optimiation problem is also a non-linear least squares problem that adds to the sum of squared reprojection errors the sum of the squared errors between the predicted and measured ranging data. The number of unknowns to be found in this problem is still the same i.e $7m+3n$ but measurements now also include ranging measurements and the maximum number of measurements is therefore $2\times m\times n+m$. This additional ranging information has the advantage of ensuring less trajectory drift even in absence of loop closure because it only requires the distance information to a fixed reference and does not rely on revisiting previously seen landmarks for the trajectory and map update. To solve the non-linear optimization problem in 4.9, LM algorithm is used. In the following, the projection function $\mathbf{Q}$ is substituted by the function $\mathbf{Q'}$ defined as follows:

$$\mathbf{Q'}(\mathbf{a_j}, \mathbf{b_i}) = (\mathbf{Q}(\mathbf{a_j}, \mathbf{b_i})^\mathbf{T}, \hat{\rho_j})^\mathbf{T}$$

Notice that $\hat{\rho_j}$ depends only on the extrinsic camera parameters i.e. on $\mathbf{a_j}$ and is independent of the 3D point $\mathbf{b_i}$ of the map. The measurement vector $\mathbf{U}$ is substituted by the vector $\mathbf{U'}$ defined as follows:

$$\mathbf{U'} = (\mathbf{u\prime_{1,1}^T}, ..., \mathbf{u\prime_{1,m}^T}, \mathbf{u\prime_{2,1}^T}, ..., \mathbf{u\prime_{2,m}^T}, ..., \mathbf{u\prime_{n,1}^T}, ..., \mathbf{u\prime_{n,m}^T})^T$$

where $\mathbf{u'_{i,j}}$ is the vector containing the measurements for the the j^{th} frame and i^{th} point defined as:

$$\mathbf{u'_{i,j}} = (\mathbf{u_{i,j}^T}, \rho_j)^\mathbf{T}$$

Using these new notations, the "BA with ranging measurements" problem expressed in 4.9 can be reformulated as follows:

$$\underset{\mathbf{a_j},\mathbf{b_i}}{\operatorname{argmin}} \sum_{j=1}^{m}\sum_{i=1}^{n} ||\mathbf{u'_{i,j}} - \mathbf{Q'}(\mathbf{a_j}, \mathbf{b_i})||^2 \tag{4.10}$$

In the following section, some elements of the solution using LM algorithms for our extended version of BA are given.

4.2.2 Solution using LM Algorithm

The parameter vector $\mathbf{p}$ searched by the LM algorithm in case we add ranging information remains the same as defined before:

$$\mathbf{p} = (\mathbf{a_1^T}, \mathbf{a_2^T}, ..., \mathbf{a_m^T}, \mathbf{b_1^T}, \mathbf{b_2^T}, ..., \mathbf{b_n^T})^\mathbf{T}$$

The objective function $\mathbf{f}$ however is now defined as follows:

$$\mathbf{f(p)} = (\mathbf{Q'(a_1, b_1)^T}, ..., \mathbf{Q'(a_m, b_1)^T}, \mathbf{Q'(a_1, b_2)^T}, ..., \mathbf{Q'(a_m, b_2)^T}, ..., \mathbf{Q'(a_1, b_n)^T}, ..., \mathbf{Q'(a_m, b_n)^T})^\mathbf{T} \tag{4.11}$$

In this section, we also consider the simplified example of 3 frames and 4 points. Furthermore, we denote by $\mathbf{A'_{ij}}$ the 3×7 matrix $\frac{\partial \mathbf{Q'(a_j, b_i)}}{\partial \mathbf{a_j}}$ and by $\mathbf{B'_{ij}}$ the 3×3 matrix $\frac{\partial \mathbf{Q'(a_j, b_i)}}{\partial \mathbf{b_i}}$. Using $\mathbf{A_{ij}}$ and $\mathbf{B_{ij}}$ notations of section **4.1.3**, we have:

$$\mathbf{A'_{ij}} = \begin{bmatrix} \mathbf{A_{ij}} \\ \frac{\partial \hat{\rho_j}}{\partial \mathbf{a_j}} \end{bmatrix}$$

and

$$\mathbf{B'_{ij}} = \begin{bmatrix} \mathbf{B_{ij}} \\ \mathbf{0} \end{bmatrix}$$

where $\frac{\partial \hat{\rho_j}}{\partial \mathbf{a_j}}$ is a 1×7 matrix. Notice that the last row of $\mathbf{B'_{ij}}$ is all zeros due to the fact that $\frac{\partial \hat{\rho_j}}{\partial \mathbf{b_i}} = 0$, $\forall i$. The Jacobian matrix $\mathbf{J}$ in block representation is then given by:

$$\mathbf{J} = \frac{\partial \mathbf{f(p)}}{\partial \mathbf{p}} = \begin{pmatrix} \mathbf{A'_{11}} & \mathbf{0} & \mathbf{0} & \mathbf{B'_{11}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A'_{12}} & \mathbf{0} & \mathbf{B'_{12}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A'_{13}} & \mathbf{B'_{13}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A'_{21}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B'_{21}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A'_{22}} & \mathbf{0} & \mathbf{0} & \mathbf{B'_{22}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A'_{22}} & \mathbf{0} & \mathbf{B'_{23}} & \mathbf{0} & \mathbf{0} \\ \mathbf{A'_{31}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B'_{31}} & \mathbf{0} \\ \mathbf{0} & \mathbf{A'_{32}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B'_{32}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A'_{33}} & \mathbf{0} & \mathbf{0} & \mathbf{B'_{33}} & \mathbf{0} \\ \mathbf{A'_{41}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B'_{41}} \\ \mathbf{0} & \mathbf{A'_{42}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B'_{42}} \\ \mathbf{0} & \mathbf{0} & \mathbf{A'_{43}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B'_{43}} \end{pmatrix} \tag{4.12}$$

Therefore, the Jacobian matrix of the BA with ranging measurements problem has also a very sparse structure which can be exploited using the sparse version of LM algorithm.

4.3 Keyframe Selection

In the previous sections, the bundle adjustment method was assumed to be performed considering all image frames. However, in case of very small relative camera motion between adjacent frames, the uncertainty in the estimation of the triangulated 3D points increases [SF12]. Moreover, we have seen that the computation cost of bundle adjustment grows fast with the number of images. Thus, it is usually more robust and more computationally efficient in BA to skip some frames for which there is no important relative change in the camera position. This process is called keyframe selection. The first keyframe is generally chosen as the first frame [TBW04]. The next selected keyframe is the most recent frame such that the number of matches between this frame and the previous keyframe does not drop below a certain threshold. This means that the relative motion between the newly selected keyframe and the previous one is neither very small nor very big and thus difficult to be accurateley estimated. The procedure of keyframe selection allows the BA problem to have a fewer number of unknowns and thus can save a significant amount of time and memory for the LM algorithm while keeping the trajectory and map estimation accurate and robust.

4.4 Error Modeling

So far in this chapter, we have assumed that the measurements recorded in the vector $\mathbf{U}$ or $\mathbf{U}'$ are not prone to error. However, in real case scenarios, this assumption is not correct. In fact, pixel measurements as well as ranging measurments have usually some uncertainty which is due to noise or to a lack of precision in the measurement method. To model this uncertainty, we need to know the covariance matrices associated to the measurements. The covariance matrices are then taken into consideration in the BA problem by replacing the euclidean norm with a weighted norm. For example the BA with ranging aid problem stated in 4.9 becomes after modeling the uncertainties:

$$\underset{\{\mathbf{X_i^{(N)}}\}_{\mathbf{i}\in[\mathbf{1}:\mathbf{n}]},\{\mathbf{P_j}\}_{\mathbf{j}\in[\mathbf{1}:\mathbf{m}]}}{\operatorname{argmin}} \sum_{j=1}^{m}\sum_{i=1}^{n} ||\mathbf{u_{i,j}} - \mathbf{P_j}.\mathbf{X_i^{(N)}}||^2_{\mathbf{\Sigma_u}} + ||\rho_j - \hat{\rho_j}||^2_{\mathbf{\Sigma}_\rho} \tag{4.13}$$

where $\mathbf{\Sigma_u}$ is the covariance matrix corresponding to the image projections measurements in pixele coordinates, $\mathbf{\Sigma}_\rho$ is the covariance matrix of the ranging measurements and $||.||_{\mathbf{\Sigma}}$ is the Mahalanobis distance defined as follows:

$$||\mathbf{x}||_{\mathbf{\Sigma}} = \sqrt{\mathbf{x^T \Sigma^{-1} x}}$$

where $\mathbf{x}$ is a vector and $\mathbf{\Sigma}$ is a covariance matrix (a symmetric positive semi-definite matrix). The LM algorithm solves the non-linear least squares problem 4.13 by

substituting the augmented normal equation 4.7 with a weighted augmented normal equation [LA04]:

$$\mathbf{N}'\delta_{\mathbf{p}} = \mathbf{J}^{\mathbf{T}}\boldsymbol{\Sigma}_{\mathbf{U}'}^{-\mathbf{1}}\epsilon \tag{4.14}$$

where $\boldsymbol{\Sigma}_{\mathbf{U}'}$ is the covariance matrix of the measurements vector $\mathbf{U}'$ (including both image projections and range measurements noise) and where the matrix $\mathbf{N}'$ is defined as follows:

$$\mathbf{N}'_{\mathbf{ij}} = \begin{cases} [\mathbf{J}^{\mathbf{T}}\boldsymbol{\Sigma}_{\mathbf{U}'}^{-\mathbf{1}}\mathbf{J}]_{\mathbf{ii}} + \alpha & if\ i = j \\ [\mathbf{J}^{\mathbf{T}}\boldsymbol{\Sigma}_{\mathbf{U}'}^{-\mathbf{1}}\mathbf{J}]_{\mathbf{ij}} & else \end{cases} .$$

In absence of any further knowledge, we assume that $\boldsymbol{\Sigma}_{\mathbf{u}}$ is the identity matrix (no noise in the image projections measurements). For the range measurements, if the covariance matrix is unknown, the Cramér-Rao lower bound [Cra99] can be used as an approximation. Furthermore, if we assume that the ranging noises are uncorrelated, $\boldsymbol{\Sigma}_{\rho}$ becomes a diagonal matrix which significantly simplifies the computations for the LM algorithm.

Chapter 5

Workflow and Implementation Details

In this chapter, our general work pipeline for the trajectory and map estimation is described. Here, we consider a calibrated stereo set-up but the same pipeline is also valid for the monocular case.

The pipeline of V-SLAM with ranging aid for trajectory and map estimation is presented in Fig. 5.1. The workflow is composed of two main parts: The first part is a V-SLAM part where the initial trajectory and map estimations are obtained. The second part is a post-processing part which consists in applying BA with ranging measurements on all image frames to reduce the drift in the trajectory and update the map estimation. The pipeline begins with the left and right image acquisition at each frame for the stereo case. In case of using a monocular camera, only one image is provided at each acquisition step. For our experiments with VI-sensor, we used Robot Operating System (ROS) to interface the sensor and divide the recorded videos into pairs of stereo images. Then, the image pairs are subject to undistortion and rectification operations also using ROS. After that, feature detection, extraction and matching as well as 3D points triangulation are performed using the rectified and undistorted image pairs. The triangulated points which form the map are used along with their corresponding image projections by the visual odometry algorithm to provide an initial egomotion estimation. The initial trajectory and map estimations are then provided with the recorded range measurements as input to the bundle adjustment algorithm. Finally, the BA adjustment with ranging aid algorithm gives us the updated trajectory and map. The coming sections will explain and give implementation details of the main steps in our workflow.

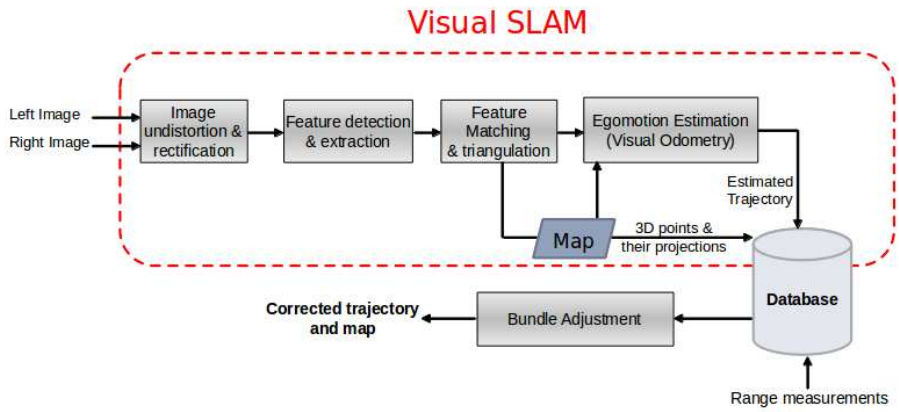


Figure 5.1: Pipeline of V-SLAM with ranging aid for trajectory and map estimation

5.1 Image Undistortion and Rectification

5.1.1 Image Undistortion

In the pinhole camera model that we have adopted in this thesis, it is assumed that lenses are perfectly linear. However, this assumption does not hold with real lenses. In fact, real lenses introduce some distortions to the images, mainly a radial distortion. Under this distortion, parallel lines in 3D world are no longer parallel in image plane. Let's denote by u_d and v_d the image coordiantes distorted by radial distortion and by $\hat{u}$ and $\hat{v}$ the ideal image coordiantes without distortion. We have then the following relation [HZ03]

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = L(r) \begin{pmatrix} \hat{u} \\ \hat{v} \end{pmatrix} \tag{5.1}$$

where r is the radial distance from the center of radial distortion and $(u_c, v_c)^T$ and L is a non-linear function of r which has the following truncated Taylor expansion:

$$L(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + k_4 r^4$$

where k_1, k_2, k_3 and k_4 are the radial distortion coefficients. These coefficients are usually given as part of the camera internal calibration parameters. The correction of radial distortion is given in pixel coordinates by:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} u_c \\ v_c \end{pmatrix} + L(r) \begin{pmatrix} u - u_c \\ v - v_c \end{pmatrix} \tag{5.2}$$

where $(u, v)^T$ are the measured coordinates and $(u', v')^T$ are the corrected coordinates.

5.1.2 Rectification

Rectification is an important pre-processing step for stereo-images. In fact, in the stereo case, image planes of the right and left cameras are usually not coplanar. Therefore, the search of matches between left and right images must be performed on non parallel epipolar lines which results in a higher computational load. Performing rectification consists in applying an homography to the two images in order to restore parallelism of epipolar lines [LZ99].In this way, matching points in right and left images have the same x-coordinate, which facilitates the feature matching and the points triangulation for stereo pairs.

5.2 Feature Detection and Extraction

We used Harris corner detector for feature detection. This method has the disadvantage of giving us the location of keypoints without subpixel accuracy (i.e. the pixel coordinates of the keypoints are integers). However, it is computationally efficient and suitable for real-time processing. The feature descriptor was made using responses to an 11×11 Sobel filter [AGS11]. Here, we assume a smooth movement of the camera (no sudden rotation or scale change) which makes it unnecessary to use computationally expensive feature descriptors such as SIFT [G.L04] or SURF [HBG08]. To reduce the number of features per image and thus reduce the computational cost of the feature extraction and matching steps, bucketing method [BKL10] is used. Bucketing consists in dividing an image into rectangular subregions called buckets. Then, only a certain number of features is retained per bucket. This have the advantage of reducing the total number of features per image while keeping the distribution of features uniform over the whole image. Therefore, the computation load of the feature matching process is significantly reduced while keeping the egomotion estimation accurate.

5.3 Feature Matching and Tracking

5.3.1 Feature Matching

Feature matching is performed on each stereo pair for points triangulation and on each pair of successive images for egomotion estimation. Given the Sobel filter responses of two feature points, the feature matching is performed using the sum of absolute differences (SAD) metric. When matching between the left and right images, the epipolar constraint (matching points have nearly the same x-coordinates) is additionally used with an error tolerance of 1 pixel.

5.3.2 Feature Tracking

In order to be able perform BA, features need to be tracked along the images. Unlike in VO, features are not tracked along only two images but over the maximum possible number of images. To achieve this aim, we used a table structure for feature tracking, where the first column represents the indexed 3D points which form the map. Each row of the first column contains the 3D coordinates of the corresponding point. The other columns contain the corresponding tracked 2D features (image projections) for each 3D point. At each new coming image, we first look for every detected 2D feature for an eventual feature match in the previous image. Once the corresponding 2D feature coordinates in the previous image are found, we search for these 2D feature coordinates in the table at the column corresponding to the previous image. If they are found, it means that the relative 3D point has been seen before and is still trackable at the current frame. Thus, we only need to add the 2D feature coordinates to the table at the column representing the current frame and at the row corresponding to the relative 3D point. If we do not find the 2D coordiantes in the table, it means that we are dealing with a new 3D point that has not been saved into the map yet. Therefore, we need to add a new row to the table corresponding to this new 3D point and fill the columns of the current and the previous image with the relative measured image projections. The other columns corresponding to previous images are filled with zeros since the 3D point has not been seen in old images.

5.4 Triangulation

For a stereo setup, the 3D points are triangulated as follows:

$$\begin{aligned} X &= (u_l - P_x).\frac{b}{d} \\ Y &= (v_l - P_y).\frac{b}{d} \\ Z &= f.\frac{b}{d} \end{aligned} \tag{5.3}$$

where X,Y and Z are the 3D coordinates of the triangulated point, (u_l, v_l) are the 2D feature coordinates of the left image, (P_x, P_y) the coordinates of the principal point, b is the camera baseline and d is the disparity defined by:

$$d = u_l - u_r \tag{5.4}$$

where (u_r, v_r) are the corresponding feature coordinates at the right image. For a rectified image, we have $v_r = v_l$. Using the formula in 5.3 and the stereo image pairs, we first express the triangulated points in the current camera local frame. Then, using the rigid body transformation formula in 3.3, the 3D points are expressed in the Navigation Frame (N) and saved into the map. Triangulated points with very small disparities are not considered in the map since have high uncertainty.

5.5 Egomotion Estimation

This step was explained in Chapter 3 (section 3.4.). Here, we give more implementation details. We implemented the egomotion estimation using the C++ library LIBVISO2 (Library for Visual Odometry 2) [AGS11]. At each new frame, RANSAC is used to find the best 3 3D to 2D correspondences between the previous and current frame. The relative rotation and translation is then estimated by minimizing the reprojection error 3.7 for all inliers using Gauss-Newton iterative optimization method. For the monocular case, the problem 3.6 is solved instead. The relative motions estimated between successive images are then concatenated iteratively to get the estimation of the current camera pose w.r.t. the Navigation Frame (N) as in 3.8. The coordinate systems definitions adopted by LIBVISO2 are depicted in Fig. 5.2. A left-handed camera coordinate system is assumed. The z-axis is aligned with the camera optical axis. For the image plane, the axes u and v of pixel coordinates are also depicted in Fig. 5.2.

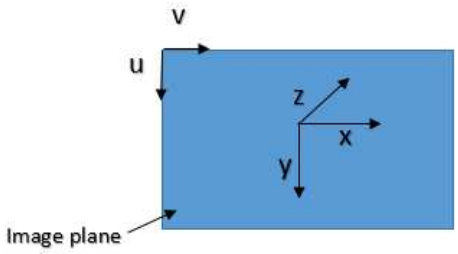


Figure 5.2: Camera coordinate systems definitions for the visual odometry algorithm

5.6 Range Measurements

As explained in Chapter 3, range measurements are estimations of the distance that separates the camera to a fixed reference at a certain time. In KITTI data set, the ground truth of the camera pose at each time frame k is provided. To get the range estimations, we only need to consider a fixed point and calculate the distance of the ground truth positions to this point. For simplification, the fixed reference was chosen as the initial postion of the camera provided by the ground truth. In this case, the World Frame (W) and the Navigation Frame (N) defined in Chapter 3 are identical. If we denote by $\mathbf{C_k^{(W)}}$ the ground truth position of the camera at the k^{th} frame expressed in (W), the ground truth range value ρ_k at frame k is given by:

$$\rho_k = ||\mathbf{C_k^{(W)}}|| \tag{5.5}$$

where $||.||$ denotes the euclidean norm. In order to simulate real case scenarios some noise can be added to the ground truth range values. The range measurements with

noise can then be modeled using the covariance matrix $\mathbf{\Sigma}_\rho$. As explained in Chapter 4, this noisy range measurements are then given as input to the BA problem along with the other measurements and the weighted minimization problem 4.13 is solved.

When it comes to our experiments using the VI-Sensor, we do not have the ground truth position of the sensor at each time frame k, but only a ground truth trajectory which s independent of time. Therefore, the method used to get the range measurements for the KITTI data set can not be used in this case. Instead, we try to simulate the range measurements using the sensor itself since we do not have a ranging device in the laboratory. In order to simulate the range measurements, we use a checkerboard as the fixed reference. In our experiments, we fix the checkerboard at a certain position in a way that it remains visible by the VI-Sensor during its motion. At each time frame, the VI-Sensor measures takes an image of the scene which includes the checkerborad. We then use OpenCV routines to detect the corners in the checkerboard (see Fig.5.3). Once the corners are detected, we estimate the distance to the checkerboard using the following triangle similarity formula:

$$d = \frac{f.L}{l} \tag{5.6}$$

where d is the estimated distance from the VI-Sensor to the checkerboard in m, f is the focal length of the sensor in pixels, L is the real length of a grid edge of the checkerboard in m and l is the measured length of a grid edge in the image plane of the sensor in pixels. This formula assumes that the image plane and the checkerboard plane are parallel which is difficult to perfectly ensure in real scenarios. In order to evaluate the accuracy of this method, we measured the real distance between the camera and the checkerboard at different cameras positions and compared the real range values to the estimated values obtained using 5.6. Fig.5.4 shows the variation of the range measurements error in m in function of the distance between the VI-Sensor and the checkerboard. It can be seen that error increases with the distance to the checkerboard. This is due to the fact that the accuracy of corner detection in the image decreases with the size if the checkerboard. Therefore, when the camera gets further to the checkerboard, its size in the image plane decreases and thus the error in the range estimation increases.

5.7 Bundle Adjustment

The BA method and its extended version using ranging aid were both detailed in Chapter 4. In this section, we give rather implementation details of the BA with ranging aid algorithm. For implementation, we used the Sparse Bundle Adjustment (sba) C++ package [LA09]. sba is composed of a set of drivers that allow the user a certain flexibility in defining the parameters of BA and LM algorithm. We used sba to update both the motion and structure and we provided as input a certain number of parameters. As parameters, we provide the total number of frames m,

Figure 5.3: Corner Detection of a checkerboard for range estimation using a VI-Sensor

the total number of points n, the initial estimations of the 3D points in (N) and the size of a measurement $\mathbf{u_{i,j}}$, which is 2 for BA without ranging aid and 3 if we include the range information. Additionally, a visibility matrix $\mathbf{V}$ that specifies for each 3D point i whether it is visible at the j^{th} frame is provided as follows:

$$\mathbf{V_{ij}} = \begin{cases} 1 & \textit{if point } i \textit{ is visible at frame } j \\ 0 & \textit{else} \end{cases} .$$

If $\mathbf{V_{ij}} = 1$, we also provide as input the measured image projection of the i^{th} point at the j^{th} frame in pixel coordiantes. Furthermore, we provide at each frame j the initial estimation of the pose of the camera (orientation and position). We also provide the objective function $\mathbf{f}$ 4.11, which includes both the projection function and the predicted range value. For the Jacobian matrix estimation, we used the approximate method of finite-differences because we found that it was more computationally efficient than the analytical approach while giving similar accuracy in the trajectory and map estimation. Finally, we provide as input the range measurements which are estimated as explained in the previous section.

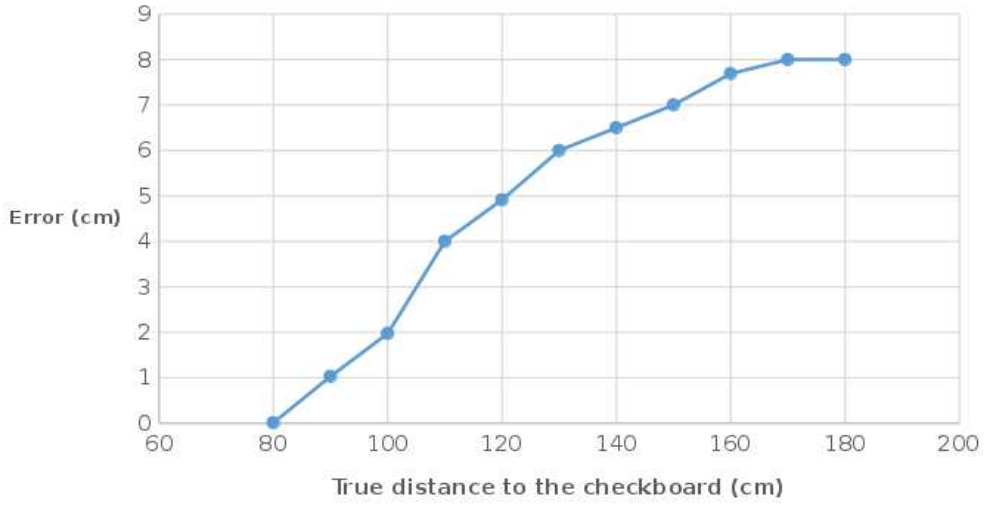


Figure 5.4: Error in range measurements (cm) in function of the distance (cm) between the VI-Sensor and the checkerboard

Chapter 6

Experimental results

6.1 Experimental Results on KITTI Data set

6.1.1 KITTI Data set

KITTI data set [GLSU13] is a set of stereo sequences recorded from a moving vehicle in Karlsruhe and captured using Pointgrey Flea2 firewire cameras. Fig.6.1 shows some examples of scenes taken from KITTI data set. The stereo sequences are saved as images and the calibration parameters as well as the baseline of the cameras are given . Additionally, the ground truth of the camera position at each frame is provided using an OXTS RT 3000 GPS/IMU system. The ground truth trajectory is given in a coordinate system which is rotated and translated w.r.t. (N). Therefore, in order to compare our estimated trajectory to the ground truth trajectory, we perform a rotation and translation transformation on the ground truth coordinate system to align it with the coordinate system (N).

6.1.2 Results

For the egomotion estimation in KITTI Data set, we used 200 RANSAC iterations with an inlier threshold δ of 2 pixels. For the feature matching, we used buckets of size 50×50 pixels with a maximum number of 2 features per bucket. To simulate the range measurements, we also added white Gaussian noise to the range values calculated from the ground truth trajectory. Fig.6.2 shows experimental results obtained on a stereo sequence from KITTI data set. The blue trajectory is the ground truth trajectory after being rotated and translated to align with (N). The red trajectory is the trajectory obtained without BA (using only visual odometry) and the yellow trajectory was obtained using BA but without ranging aid. The green trajectory is the trajectory obtained using BA with ranging aid. The range measurements in this sequence were obtained by adding noise to the true range values such that the signal-to-noise-ratio is 20 dB. For this sequence, bundle adjustment was performed on 250 frames, 35827 3D points and 71809 image projections. LM

Figure 6.1: Examples of scenes from KITTI data set

algorithm converged after 128 iterations and the total computation time of the bundle adjustment with ranging aid algorithm was 92.46 seconds. To evaluate the accuracy of our algorithm, we calculated the error in the trajectory estimation using the root mean square error (RMSE) as measure. Table 6.1 shows the average errors in centimeters corresponding to the trajectories represented in Fig.6.2. The results show that BA even without ranging aid allows us to significantly reduce the drift. By adding ranging measurements to the BA algorithm, the average drift was further reduced by 2.5 centimeters. Therefore, we notice that BA gives us satisfactory results in drift reduction and that by adding the range values, the trajectory is further refined without a significant change in the computation time w.r.t. the BA algorithm without ranging aid.

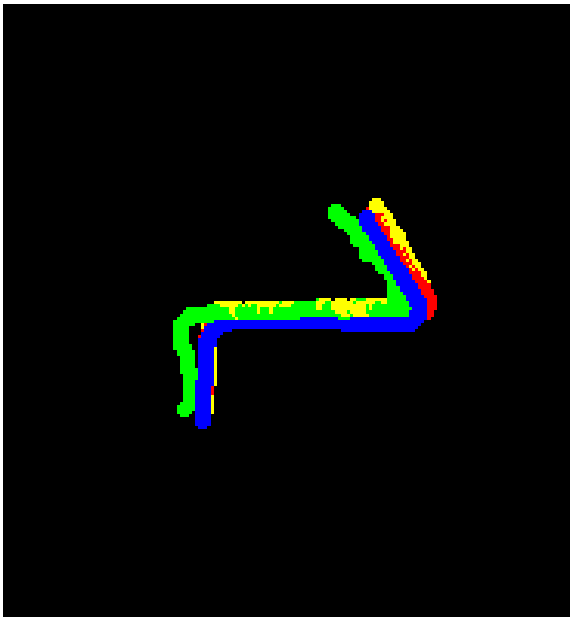


Figure 6.2: Ground truth trajectory of a sequence from KITTI dataset (blue) and estimated trajectories without BA (red), with BA but without ranging aid (yellow) and with BA and ranging aid (green)

Method	Average error (drift) in cm
VO without BA	41.33
BA without ranging aid	18.82
BA with ranging aid	16.32

Table 6.1: Average errors in centimeters corresponding to the trajectories represented in Fig.6.2

6.2 Experimental Results using VI-Sensor

6.2.1 VI-Sensor

The VI-Sensor (Visual-Inertial Sensor) shown in Fig.6.3 is a sensor developed by Skybotix AG and which incorporates a calibrated stereo camera, an FPGA system as well as a time synchronized and calibrated inertial measurement system [NRB+14]. In our experiments, we only use the stereo camera which provides us with stereo images at a rate of 20 fps. The images have a resolution of 752×480 and are taken using an Aptina MT9V034 sensor. We interface the sensor using ROS and perform rectification on the images.

Figure 6.3: VI-Sensor (front view)

6.2.2 Experimental Setup

Fig.6.4 shows a typical experimental setup using the VI-Sensor. We move the camera along a known path which is predefined using markers on a table as shown in Fig.6.4. The markers are used for the ground truth estimation of the trajectory. Furthermore, we hang a checkerboard at a board in front of the VI-Sensor in a way that it can be seen by the sensor during its trajectory. This checkernoard is used for the range estimation as explained in section 5.6.. Some features (pen, cisor...) are also placed near the sensor to facilitate the relative motion estimation. At each time frame, a stereo image pair as well as a range value using the left image are obtained. Fig.6.5 shows an example of an image captured using the left camera of the VI-Sensor while following a predifined path.

6.2.3 Results

For the egomotion estimation using the VI-Sensor, we used 200 RANSAC iterations with an inlier threshold δ of 5 pixels. For the feature matching, we used buckets of size 50×50 pixels with a maximum number of 20 features per bucket. Since the ranging measurments estimated using the checkerboard are already noisy as shown in section 5.6, we did not add noise to the measurements. Fig.6.6 shows an example of a predifined trajectory using markers (Fig.6.6a) as well as the estimated trajectory using the VI-Sensor after applying our algorithm (Fig.6.6b). The ground truth trajectory in Fig.6.6a was generated by drawing a straight line between each two consecutive ground truth positions of two markers. However, while following the desired trajectory with the VI-Sensor, it is difficult to follow a perfectly straight line between each two consecutive markers. In addition to that, the noise in the images captured by the VI-Sensor, which is mainly due to illumination changes, as well as the errors in the range estimation using the checkerboard has made our estimated trajectory shown in Fig.6.6b relatively noisy compared to the predifined trajectory (Fig.6.6a). For this predefined trajectory, bundle adjustment was performed on 172 frames, 21034 3D points and 68551 image projections. LM algorithm converged

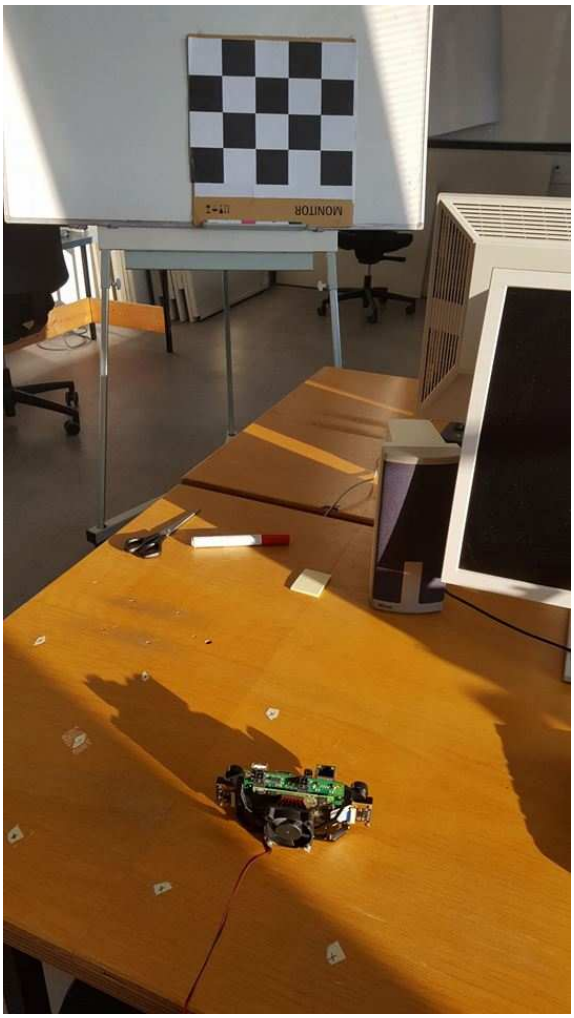
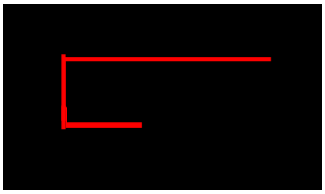


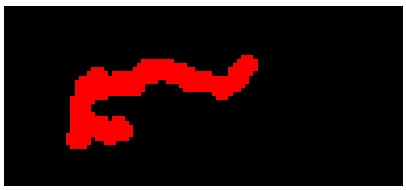
Figure 6.4: Experimental Setup using the VI-Sensor

after 67 iterations and the total computation time of the bundle adjustment with ranging aid algorithm was 38.07 seconds. In order to evaluate the accuracy of our estimated trajectory w.r.t. the predefined trajectory, we also calculated the error in the trajectory estimation using the root mean square error (RMSE) as measure. Table 6.2 shows the average errors in centimeters. These results also confirm the fact that the BA algorithm by itself allows us to significantly reduce the drift and that by adding ranging measurements to the BA algorithm, the average drift is further reduced without increasing the computation time.

Figure 6.5: Example of a scene captured using the left camera of the VI-Sensor



(a) Predifined trajectory



(b) Estimated trajectory

Figure 6.6: (a) A predifined trajectory for the VI-Sensor using markers (b) Estimated trajectory using the VI-Sensor after applying our BA with ranging aid algorithm

Method	Average error (drift) in cm
VO without BA	7.05
BA without ranging aid	3.35
BA with ranging aid	2.83

Table 6.2: Average errors in centimeters for the estimated trajectories using the VI-Sensor

Chapter 7

Conclusion and Future Work

In this thesis, a bundle adjustment-based V-SLAM algorithm with ranging aid for drift reduction was presented, implemented and tested on real scenarios. Our approach is based on integrating the range w.r.t. to a fixed reference with known position into the cost function of bundle adjustment. We showed that the bundle adjustment with ranging aid problem has a sparse structure which allowed us to use a sparse version of the Levenberg-Marquardt algorithm. Furthermore, the objective function, the projection function, the Jacobian matrix and the measurements vector of the original bundle adjustment problem were all adapted to include the ranging information. To test our bundle adjustment with ranging aid algorithm, we implemented a pipeline composed of a V-SLAM part and of a post-processing part, which is the bundle adjustment with ranging aid algorithm for trajectory and map update. The tests were performed on sequences from KITTI data set as well as on real data obtained from a VI-Sensor in the laboratory. It was also compared to the pure V-SLAM algorithm and to the bundle adjustment-based approach without ranging aid. For the KITTI data set, the range values were simulated by adding white Gaussian noise to the range values calculated using the provided ground truth trajectory. For the experiments with the VI-Sensor, range measurements were obtained using the images captured by the VI-Sensor by detecting a checkerboard as a fixed reference in the scene and by estimating the range based on the proportion between the measured length of a grid edge in the image plane and its real length. Experimental results show that bundle adjustment is able to significantly reduce the drift in the trajectory even without range measurements. Moreover, the drift is further reduced in case we integrate the ranging information. The decrease in the drift by adding ranging aid is though not as significant as the decrease in error using bundle adjustment w.r.t to the error obtained without bundle adjustment. However, we noticed that the computational time practically does not increase if we add the ranging measurements to the original global BA algorithm. Thus, combining ranging aid with bundle adjustment allows us to reduce the drift without increasing the computation load, which makes it a useful step for drift reduction in V-SLAM algorithms. Since we did not have a ground truth estimation of the map,

we were not able to evaluate the accuracy of the updated map. However, since bundle adjustment aims at minimizing the reprojection error w.r.t. both the trajectory and the 3D points, reducing the drift in the trajectory estimation should normally correspond to reducing the error in the map as well.

Since we used a checkerboard and the images provided by the VI-Sensor and KIITI data set to obtain the range measurements, there was no need for time synchronization between the ranging values and the image frames. However, in our experiments using the VI-Sensor, the noise in our ranging measurements increased significantly with the distance to the checkerboard. This noise in addition to the noise in the images made our estimated trajectory relatively noisy w.r.t. the ground truth trajectory. Using a ranging device will require more effort for synchronisation but should allow us to have more accuracy in the range estimation and therefore a better performance of the bundle adjustment with ranging aid algorithm.

Appendices

Appendix A

Quaternions

Here give additional material on quaternions.

Appendix B

Triangulation in monocular SLAM

Here give additional material on triangulation in monocular case.

List of Figures

List of Tables

Bibliography

[AB81] Martin A.Fischler and Robert C. Bolles. RANSAC sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[AGS11] Julius Ziegler Andreas Geiger and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, June 2011.

[BAC13] Guillaume Bresson, Romuald Aufrère, and Roland Chapuis. Making visual slam consistent with geo-referenced landmarks. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, June 2013.

[BD71] Kenneth M. Brown and J. E. Dennis. Derivative free analogues of the levenberg-marquardt and gauss algorithms for nonlinear least squares approximation. *Numerische Mathematik*, 18(4):289–297, 1971.

[BKL10] Andreas Geiger Bernd Kitt and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *2010 IEEE Intelligent Vehicles Symposium (IV)*, June 2010.

[Cra99] Harald Cramér. *Mathematical methods of statistics*. Princeton University Press, 1999.

[DNB04] Oleg Naroditsky David Nistér and James Bergen. Visual odometry. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2004.

[EL09] Alexander Eudes and Maxime Lhuillier. Error propagation for local bundle adjustment. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, pp. 2411-2418*, 2009.

[ESZ14] Armin Dammann Emanuel Staudinger, Siwei Zhang and Chen Zhu. Towards a radio-based swarm navigation system on mars – key technologies and performance assessment. In *2014 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, October 2014.

[G.L04] David G.Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[HBG08] Tinne Tuytelaars Herbert Bay, Andreas Ess and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding 110*, 110(3):346–359, 2008.

[HN94] Thomas Huang and Arun Netravali. Motion and structure from feature correspondances: A review. In *Proceedings of IEEE, vol.82, no2*, pages 252–268, 1994.

[HN06] Kin Leong Ho and Paul Newman. Loop closure detection in SLAM by combining visual and spatial appearance. *Robotics and Autonomous Systems*, 54:740–749, July 2006.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, New York, United States of America, 2003.

[KSC13] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013.

[LA04] Manolis I.A. Lourakis and Antonis A.Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical report, August 2004.

[LA09] MANOLIS I. A. LOURAKIS and ANTONIS A. ARGYROS. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Transactions on Mathematical Software*, 36(1), 2009.

[Lam97] Michael Lampton. Damping-undamping strategies for the levenberg-marquardt nonlinear least-squares method. *Computers in Physics*, 11:110–115, January 1997.

[LKS11] David Scaramuzza Laurent Kneip and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2011.

[LZ99] Charles Loop and Zhengyou Zhang. Computing rectifying homographies for stereo vision. Technical report, April 1999.

[NRB+14] Janosch Nikolic, Joern Rehder, Michael Burri, Pascal Gohl, Stefan Leutenegger, Paul T. Furgale, and Roland Siegwart. A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, June 2014.

[Pet14] Arne Petersen. *A Specialized Kalman Filter Framework for IMU Aided Stereo SLAM*. Number 2014/3 in Kiel Computer Science Series. Department of Computer Science, CAU Kiel, 2014. Dissertation, Faculty of Engineering, Kiel University.

[SF11] David Scaramuzza and Friedrich Fraundorfer. Visual Odometry Part I: The First 30 Years and Fundamentals. *IEEE Robotics and Automation Magazine*, pages 81–90, December 2011.

[SF12] David Scaramuzza and Friedrich Fraundorfer. Visual Odometry Part II: Matching, Robustness, Optimization, and Applications. *IEEE Robotics and Automation Magazine*, pages 83–86, June 2012.

[SS10] Ulrich Steinhoff and Bernt Schiele. Dead Reckoning from the Pocket - An Experimental Study. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, May 2010.

[SZM+13] S. Sand, S. Zhang, M. Mühlegg, G. Falconi, C. Zhu, T. Krüger, and S. Nowak. Swarm exploration and navigation on mars. In *2013 International Conference on Localization and GNSS (ICL-GNSS)*, June 2013.

[TBW04] Thorsten Thormählen, Hellward Broszio, and Axel Weissenfeld. *Keyframe Selection for Camera Motion and Structure Estimation from Multiple Views*, pages 523–535. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[W.M63] Donald W.Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal of Applied Mathematics*, 11(2):431–441, June 1963.