École Centrale de Nantes

Università degli Studi di Genova

MASTER ERASMUS MUNDUS

EMARO "European Master in Advanced Robotics"
2014/2015

Master Thesis Report

Presented by

Nived Chebrolu

On 28/08/2015

Title

Collaborative Visual SLAM

JURY

President:	Gaëtan Garcia	Maître de conférences, ECN
Evaluators:	Philippe Martinet	Professeur, IRCCyN, ECN
	David Marquez-Gamez	Ingénieur de Recherche, IRT Jules Verne
	Gaëtan Garcia	Maître de conférences, ECN
	Arnaud Hamon	Ingénieur de Recherche, IRCCyN
	Salvador Dominguez	Ingénieur de Recherche, IRCCyN

Supervisors:
Philippe Martinet, Professeur, École Centrale de Nantes
Laboratory: Institut de Recherche en Communication et Cybernétique de Nantes
David M. Gamez, Ingénieur de Recherche, IRT Jules Verne

Co-Supervisor:
Fabio Solari, Assistant Professor, Università degli Studi di Genova
Dipartimento di Informatica, Bioingegneria, Robotica, Ingegneria Dei Sistemi (DIBRIS)

UNIVERSITÀ DEGLI STUDI
DI GENOVA

EMARO "European Master on Advanced Robotics"
2014/2015

Master Thesis
FINAL REPORT

Collaborative Visual SLAM

Author:
Nived Chebrolu

Supervisors:
Prof. Philippe Martinet
Dr. David Marquez-Gamez
Co-Supervisor:
Prof. Fabio Solari

August 2015

Abstract

Multi-robot systems are envisioned to be used for challenging applications in large unstructured environments. For example, a team of mobile robots deployed to explore a damaged structure in a disaster situation with the aim of providing risk assessment to the first responders. For such applications, the ability to localize and map the environment (SLAM) collaboratively as a team becomes essential. A collaborative system extends the sensing capability of an individual robot by allowing them to *look* beyond the direct reach of their on-board sensors.

In this thesis, we present a framework for collaborative visual SLAM using monocular cameras for a team of mobile robots. Each robot performs SLAM individually using its on-board processors. During this process, it estimates the seven degrees of freedom (three for rotation, three for translation and the scale) for the motion of the camera and creates a map of the environment as a pose-graph of keyframes. The collaboration between robots is facilitated through a centralized mechanism. Each robot communicates to a central server by sending local keyframe information. The central server merges them when a visual overlap is detected in the scene and creates a global map. In the background, the global map is continuously optimized using a bundle adjustment step and the updated pose information is communicated back as feedback to the individual robots.

The main components of the overall system are: (1) A monocular SLAM process for each robot, (2) Detection of scene overlap amongst several robots, (3) Computation of the global map using measurements from all team members. The key contribution of the thesis is in the integration of several existing solutions to develop a flexible framework for collaborative Visual SLAM.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Philippe Martinet for giving me this opportunity of working on such an interesting topic. His constant motivation and guidance throughout the project has been invaluable.

I would like to express my heartfelt gratitude to Dr. David Marquez-Gamez for his guidance, constant help and encouragement in completing this work. He has always encouraged me to try out different ideas and critically think about the problem. He has shown a lot of patience and care throughout the entire process. His mentorship has been a throughly fulfilling experience for me over the last year.

I would like to take this opportunity to thank my co-supervisor Prof. Fabio Solari who has been constant source of support and guidance.

Thanks to all the members of the IRCCyN lab for providing a fun-filled working environment. My special thanks to Dr. Salvador Dominguez for sharing great many ideas and useful programming tips.

Thanks to the EMARO program for providing a great opportunity to learn from the most dedicated professors and students coming from diverse backgrounds.

I am thankful to my friends Francesco, Harsh, Praveen and Simona who have not only tolerated me, but have made my life far richer than it could otherwise be.

Most of all, I would like to thank my family for their constant support and encouragement.

Contents

List of Figures

List of Algorithms

Abbreviations

SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
IMU	**I**nertial **M**easurement **U**nit
EKF	**E**xtenden **K**alman **F**ilter
UAV	**U**nmanned **A**erial **V**ehicle
MAV	**M**icro **A**erial **V**ehicle
SFM	**S**tructure **F**rom **M**otion
VO	**V**isual **O**dometry
SVD	**S**ingular **V**alue **D**ecomposition
BA	**B**undle **A**djustment
RANSAC	**RAN**dom **SA**mple **C**onsensus
SIFT	**S**cale -**I**nvaraint **F**eature **T**ransform
SURF	**S**peeded **U**p **R**obust **F**eatures
DSM	**D**ecoupled **S**tochastic **M**apping
NCFM	**N**etwork **C**oupled **F**eature **M**aps
PTAM	**P**arallel **T**racking **A**nd **M**apping
GPU	**G**raphics **P**rocessing **U**nit
RGB-D	**R**ed **G**reen **B**lue -**D**epth
DoF	**D**egrees **o**f **F**reedom
ROS	**R**obot **O**perating **S**ystem
VR	**V**irtual **R**eality
LSD-SLAM	**L**arge **S**cale **D**irect - **SLAM**
FAB-MAP	**F**ast **A**ppearance **B**ased - **MAP**ping
PCA	**P**rincipal **C**omponent **A**nalysis
BoW	**B**ag **o**f **W**ords
ROI	**R**egion **O**f **I**nterest

ICP	**I**terative **C**losest **P**oint
GICP	**G**eneralized **ICP**
NDT	**I**terative **C**losest **P**oint
RTK-GPS	**R**eal **T**ime **K**inematic-**GPS**

Chapter 1

Introduction

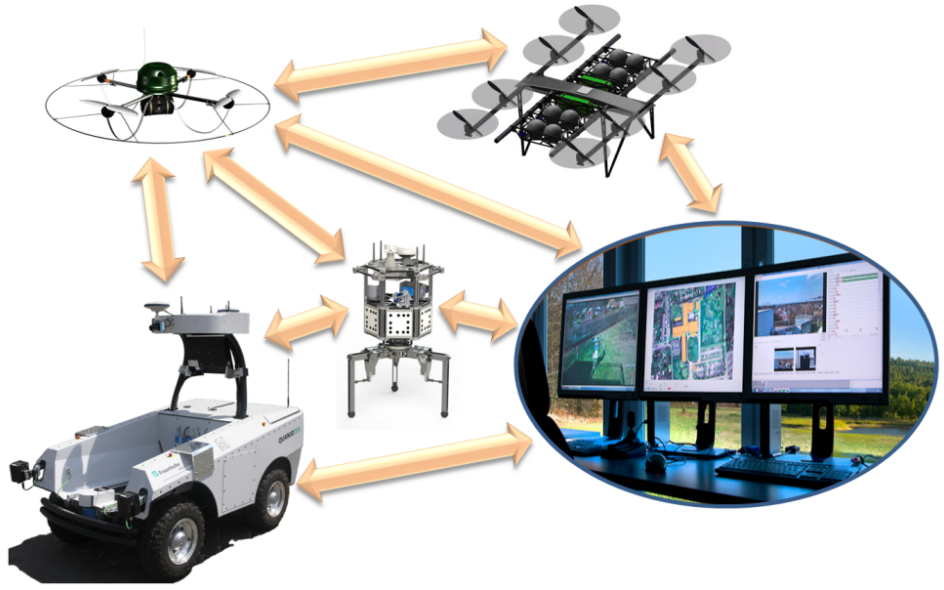
1.1 Motivation

Increasingly, autonomous robots are being used to solve more and more complex problems. In order to deal with these complex situations, a multi-robot system consisting of a team of robots (such as ground mobile robots, aerial vehicles and underwater robots etc.) which are equipped with perception sensors is necessary. Multi-robot systems extend the capabilities of a single robot by merging measurements from several team members and providing each robot with information beyond the capability of their individual sensors. It facilitates more efficient usage of resources and achieves tasks which are not feasible for a single robot system.

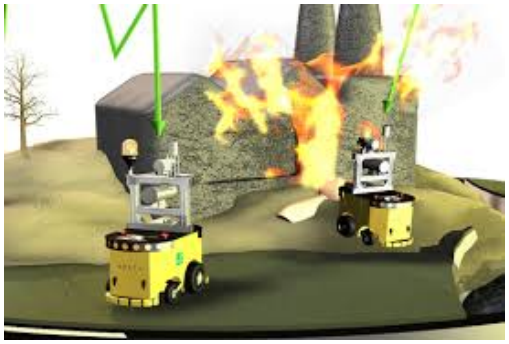
The potential applications for autonomous multi-robot systems are many (Figure 1.1). For example, in a disaster situation such as an earthquake, a team of robots autonomously able to navigate and map a damaged building could provide vital information to the rescuers. The information could be the location of the survivors, a safe path to reach them or a warning about severely damaged portions of the building which may be dangerous to pass through. Similarly, they may be employed in hazardous industrial situations where toxic substances must be handled. Such a system could protect human workers by avoiding direct exposure to the toxic elements.

To perform these tasks successfully, the robots must know where they are at all times and should be able to perceive the surroundings using its sensors. In addition, the

different team members should be able to communicate with each other effectively and share useful information.



(A) Multi-robot system for disaster relief operations [1]



(B) Industrial hazard scenario [1]

(C) Post-Earthquake scenario [2]

FIGURE 1.1: Applications of multi-robot systems

1.2 Problem Statement

The main objective of the thesis is to develop a framework for a multi-robot system which can collaboratively explore an unknown environment. This framework would form the foundation block for applications mentioned in section 1.1. The final framework should support the following functionalities:

1. Ability of an individual team member to localize in the environment: Each member of the team should individually be able to localize in the environment using only its on-board sensors and processor. In addition it should also be able to map the environment in which it navigates. In particular, the aim is to provide this ability using only vision sensors such as cameras.

2. Ability to detect other robots and places visited by them: The robots must be able to detect other members of the team when they encounter each other. Moreover, they should also be able to detect the places which have been visited by other robots. This ability would facilitate aggregation of information from other team members.

3. Ability to communicate with team members: A communication framework should be designed where the robots in the team can efficiently talk to one another and exchange useful information.

4. Flexibility in operation: The individual robots should be able to join and leave the team asynchronously. They should also be able to explore collaboratively without having any prior knowledge of each others location.

5. Ease of extension: The framework must be designed in a manner such that it can be extended easily to accommodate different kinds of robots, sensors and applications.

1.3 System Requirements

In order to develop and test a multi robot system, several components are required. Here we list out the main components used during the project:

- Mobile Robotic Platform: The final system is tested on a team of ground mobile robots. For this purpose, we use the mobile robotic platform "*Turtlebot*"(Figure 1.3a). The technical specifications of the platform are summarized in Table 1.1. During the testing stage several experiments were conducted using a hand-driven cart as the mobile platform. (Figure 1.3b)

- Perception Sensor: Of the several vision sensors available, monocular camera was chosen as the perception device. This choice was made taking into account several

Category	**Specifications**
Size and Weight	• External Dimensions: 354 x 354 x 420 mm • Weight: 6.3 kg • Wheels(Diameter): 76 mm • Ground Clearance: 15 mm
Speed and Performance	• Max Payload: 5 kg • Max Speed: 0.65 m/s • Max Rot Speed: 180 deg/s
Power System	• Battery: 2200 mAh Li-Ion • User power: 12 V (1.5A)
Sensors	• 3D Vision sensor (ASUS Xtion PRO LIVE) • Encoders • Gyro • Others: Forward bump, cliff and wheel drop sensors
Computer	• Memory: 4 GB • Processor: Intel Core i3-4010U • Graphics: Intel HD Graphics • Hard Drive: 500 GB • WiFi: 802.11n

TABLE 1.1: Turtlebot technical specifications

(A) Turtlebot Platform (B) Hand-driven cart

FIGURE 1.2: Mobile robotic platforms used for testing.

advantages of the monocular camera such as (a) Flexibility to be used for both indoor and outdoor applications, (b) Compact and lightweight, (c) low-power, (d) cost effective and ubiquitous availability. For our tests, we use an *uEye* camera which captures images at a frequency of 30Hz and with a resolution of 640 $\times$ 480 pixels. In addition, a wide angle lens (130 deg Field of view) is fitted to the camera.

- Computational Resources: Each robot is equipped with a notebook which performs all local processing. The back-end computations are done on a more powerful Core 2 Duo laptop.

(A) Monocular camera (B) Wide angle lens

FIGURE 1.3: Perception sensor: Monocular Camera

- Communication Hardware: All the robots are equipped with a standard WiFi transmitter/receiver for communication.
- Software Requirement: The framework for the system is developed in Linux environment (Ubuntu 12.04/14.04). The software is written in the ROS framework and uses several open source libraries such as OpenCV, *g2o* etc.

A summary of the overall requirement is given in Table 1.2 .

Component	**Details**
Mobile Robotic Platform	• Two Turtlebots • Hand driven cart for testing
Perception Sensor	• UEye monocular camera with 640 x 480 resolution • Wide eye lens (130 deg Field of view)
Computational Resources	• Notebook on each Turtlebot • Core 2 Duo laptop for back-end processing
Communication Hardware	• Standard WiFi transmitter/receiver
Software Requirements	• Linux support (Ubuntu 12.04/14.04) • ROS(Robot Operating System) as messaging framework • Vision: OpenCV support • Optimization: *g2o* library

TABLE 1.2: System Requirements

1.4 Main Contributions

The main contribution of this thesis is the development of a collaborative visual SLAM framework for a multi-robot system. At its core, a system for fusing multiple visual SLAM processes running on individual robots into a single consistent map is presented. The framework integrates many existing components and demonstrates their combined performance in a complete system. The overall framework can be broken into several modules, each of which is self-contained and can be used independently. These are:

- A monocular visual SLAM algorithm running independently on individual robots.
- A place recognition system which detects scene overlap among different robots.
- A merge procedure where individual SLAM processes are fused together to produce a global consistent map.
- A feedback mechanism wherein the global information is communicated back to the individual robots.

The framework is finally validated by conducting several experiments with a team of two ground mobile robots operating in an indoor environment.

1.5 Organization

The report is organized in the following manner. Chapter 1 introduced the idea of multi-robot systems and its potential applications. It presented the problem statement of the thesis. Chapter 2 will give an overview of the important concepts and the state of the art techniques related to the thesis. Chapter 3 will describe the monocular SLAM running on the individual robots. Chapter 4 discusses the place recognition system which would be used to detect the scene overlap among different robots. Chapter 5 describes the various techniques used to compute the relative transformation between robots which share a common view. Chapter 6 integrates all the tools developed in Chapters 3,4 and 5 and explains the overall framework. Finally, Chapter 7 provides a summary of the project and scope for future work.

Chapter 2

Background

In this chapter, we discuss the important concepts and state of the art techniques related to the thesis. Section 2.1 introduces the topic of SLAM and formulates the problem in a probabilistic framework. Several existing solutions are then described. Section 2.2 gives an overview of the state of the art techniques in visual SLAM. The techniques are presented in an evolutionary manner emphasizing how they were built upon the previous techniques. Finally, section 2.3 describes the work in the field of collaborative visual SLAM in the past few years.

2.1 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) is a process which answers two fundamental questions: "Where am I?" and "What am I seeing?". These questions must be answered continuously as this information is required by the robot to navigate in the environment or take some decision autonomously.

In our daily lives, we as humans perform SLAM continuously. We observe the environment through our senses: vision, hearing, smell, touch and taste. Our brain then combines these observations with prior knowledge while making some assumptions to give us information regarding our location and the surroundings, both qualitatively and quantitatively. Mostly we perform this process routinely without even noticing. Take the example of riding a bicycle on the street. The localization process answers the questions like where are we located on the street and how fast are we moving etc. While the

mapping process gives information regarding the location of other vehicles or obstacles. Both these information put together is useful as it allows us to take a decision to avoid the obstacles and navigate safely.

While designing a synthetic SLAM system for a robot, we have to explicitly specify what information it must provide as output and an algorithm to compute it. In a typical example of autonomous navigation for a mobile robot, the SLAM algorithm should provide an estimate of the pose of the robot at each instant and information regarding the structure of the environment. It should also explicitly give a measure of the uncertainty in these estimates as it can further assist in decision making.

If the structure of the environment is known, the process of finding the robot's pose (localization) is fairly straightforward. Similarly, if the robot's pose is exactly known, the problem of building a representation of the environment (mapping) is equally approachable. However, it is more challenging to find both the pose of the robot and the information regarding the structure of the environment while exploring an unknown territory using noise-ridden sensor measurements. The SLAM process aims to find a solution in this situation.

2.1.1 A Probabilistic Approach

A robot operating autonomously in a real environment must deal with uncertainty. These uncertainties commonly arise from the dynamic and unpredictable nature of the environment, the resolution and technological limits of the sensors, the inaccuracies in robot actuation and approximations in the models and algorithms used for computations (Thrun et al. [3]). The ability to cope these uncertainties plays a key role in the robot's success in an unstructured environment. This leads us to the realm of *Probabilistic Robotics* where the key idea is to explicitly represent uncertainty using the calculus of probability theory.

Initial work in this direction was described in the seminal paper by Smith et al. [4] which established a statistical basis to explain the relations between geometric quantities while providing an explicit representation of the inherent uncertainties. The main idea is to represent relationship between uncertain quantities as a probability distribution over spatial parameters. For most applications, an approximation of the original probability

distribution given by the first and second moments is sufficient. A convenient choice for this representation is through the mean vector and covariance matrix with dimensions defined by the number of uncertain parameters. The system using such a representation can be built incrementally by adding a new element each time to the mean vector and a corresponding column to the covariance matrix. As a whole, the resulting state is called as the stochastic map. Therefore SLAM can essentially be seen as the process of representing and maintaining the stochastic map and fusing sensor observations into it.

Given such a representation, the SLAM process can be treated as a state estimation problem where the robot pose and structure of the environment must be inferred from the sensor measurements which carry only partial information about the states and are normally corrupted by noise. The probability state estimation algorithms compute a belief distribution over all possible world states. The overall aim of this state estimation problem is to jointly estimate the robot position and positions of a set of remarkable objects (called landmarks) in the environment which are supposed static. Figure 2.1 shows the different steps of SLAM:

1. The robot observes the three landmarks for the first time (the corners of the squares).
2. The robot then moves and it knows its position with some uncertainty.
3. The robot re-observes these landmarks.
4. The fusion of this observation with the map previously constructed, reduces both uncertainties on the positions of landmarks and the position of the robot.

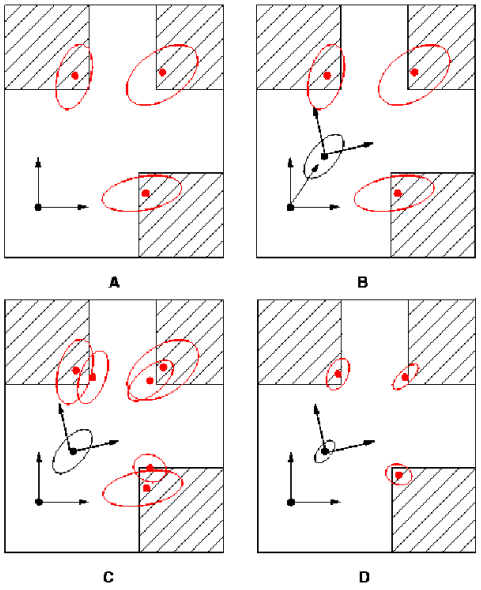


FIGURE 2.1: Main steps of SLAM

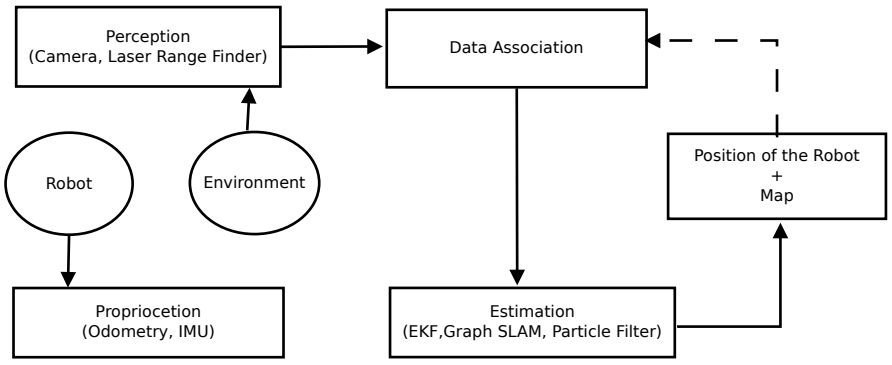


FIGURE 2.2: Various blocks of a SLAM algorithm

Figure 2.2 shows a general view of a SLAM algorithm. The various blocks are:

1. *Perception:* Perception algorithms provide from the sensors (camera) a set of observations of the environment, called landmarks.

2. *Data Association:* Given an observation, we must determine if this observation is a landmark already present in the map, or if it is a new landmark to add to the map.

3. *Proprioception*: Proprioceptive data are internal measures from the system, and provide information on the evolution of the hidden state. For SLAM, we are interested in the position of the robot whose evolution is typically provided by odometer sensors or an inertial measurement unit (IMU).

4. *Estimation:* The estimation algorithm must integrate data from perception and proprioception to provide an estimate of the position of the robot and the positions of landmarks, and the uncertainties associated with these estimates.

2.1.2 Problem Formulation

Now we introduce the terminology and the notations used to state the SLAM problem in a formal mathematical framework. This notation will be used throughout the thesis. We will take the example of a mobile robot exploring an unknown territory to establish the idea. Consider the robot moving through the environment taking relative measurements of a number of unknown objects (*landmarks*) using an on-board sensor. At any time instant t, the following quantities are defined:

- $\mathbf{x_t}$, the state vector which describes the pose of the robot at a time t. This could be the position and orientation of the mobile robot.
- $\mathbf{u_t}$, the control vector which is applied at time $(t-1)$ in order to drive the robot to a state $\mathbf{x_t}$ at time t. The control data for a mobile robot typically consists of velocity information or odometry data which gives a measure of the revolution of the wheels.
- $\mathbf{z_t}$, the measurements taken by the robot at time t. It could consist of camera images, range scans etc. captured from an on-board sensor.
- $\mathbf{m_i}$, the vector describing the location of the i th landmark whose true location is assumed to be time invariant.

In addition, we also define the following sets:

- $\mathbf{x_{0:t}} = \{\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_t}\}$, which holds all the robot poses upto time t.
- $\mathbf{u_{0:t}} = \{\mathbf{u_0}, \mathbf{u_1}, \ldots, \mathbf{u_t}\}$, which holds all the control data upto time t.
- $\mathbf{m} = \{\mathbf{m_1}, \mathbf{m_2} \ldots, \mathbf{m_N}\}$, the set of all landmarks.
- $\mathbf{z_{0:t}} = \{\mathbf{z_0}, \mathbf{z_1}, \ldots, \mathbf{z_t}\}$, the set of all landmark observations made by the on-board sensors.

A key concept in probabilistic robotics is that of a *belief*. It reflects the robot's internal knowledge about the state of the world (i.e. both its pose and information of the surroundings). In our case, the robot does not have any direct knowledge of its pose and it must infer this information from the measurements given by its sensors. This inference defines the robot's belief. Typically the belief is represented as a conditional probability distribution. The belief assigns a probability to each possible hypothesis with respect to the true state.

In this framework, the localization problem can be posed as finding the distribution $p(\mathbf{x_t}|\mathbf{z_{0:t}}, \mathbf{u_{0:t}}, \mathbf{m})$. Here we assume that the landmark locations m are known exactly and the aim is to compute robot's pose $\mathbf{x_t}$ with respect to these landmarks. While the mapping problem can be defined as computing the probability distribution $p(\mathbf{m}|\mathbf{x_{0:t}}, \mathbf{z_{0:t}}, \mathbf{u_{0:t}})$ assuming that the pose of the robot is known at each time instant.

These two probability distributions refer to the localization and mapping problem separately. Considering them together the overall SLAM problem can be formulated as computing $p(\mathbf{x_t}, \mathbf{m}|\mathbf{z_{0:t}}, \mathbf{u_{0:t}})$. This probability distribution describes the joint posterior density over the robot poses and the landmark locations at a time t given the recorded observations and the control input until time t.

In the probabilistic robotics literature, there are two main forms of the SLAM problem. First, the *Online SLAM* problem where we want to compute the posterior over the momentary pose, i.e. $p(\mathbf{x_t}, \mathbf{m}|\mathbf{z_{0:t}}, \mathbf{u_{0:t}})$. Its referred to as online because it computes variables that persist at time t only. Many of the online SLAM algorithms are incremental, i.e they discard the past measurements and controls once they are processed. Second, the *full SLAM* problem where we seek to compute $p(\mathbf{x_{1:t}}, \mathbf{m}|\mathbf{z_{0:t}}, \mathbf{u_{0:t}})$ which is the posterior over the entire path $\mathbf{x_{1:t}}$ along with the map. This subtle difference between the two forms makes a big difference in terms of the algorithms that can be employed. In particular, the online SLAM solution can be obtained by integrating out past poses from the full SLAM problem:

$$p(\mathbf{x_t}, \mathbf{m}|\mathbf{z_{0:t}}, \mathbf{u_{0:t}}) = \int \int \dots \int p(\mathbf{x_{1:t}}, \mathbf{m}|\mathbf{z_{0:t}}, \mathbf{u_{0:t}}) \mathbf{dx_1 dx_2} \dots \mathbf{dx_{t-1}} \tag{2.1}$$

Here, we discuss a solution for the online SLAM problem. The Bayesian filtering framework gives us an algorithm to compute the solution for SLAM is a recursive manner.

The algorithm consists of two main steps. The first one being a prediction step where the control data is used to predict the current state from the past state. This step is also called as the control update. Followed by a correction step where the measurements from the sensors are used to improve the prediction. This step is known as the measurement update. The overall computation requires two models to be defined. First, a motion model describing how the control input $\mathbf{u_t}$ affects the state transition from $\mathbf{x_{t-1}}$ to $\mathbf{x_t}$. Second, an observation model describing how the measurements are made by the sensors.

The motion model gives the distribution $p(\mathbf{x_t}|\mathbf{x_{t-1}}, \mathbf{u_t})$. Here, we assume that state transition to be a Markov process such that the present state $\mathbf{x_t}$ only depends on the previous state $\mathbf{x_{t-1}}$ and the control input $\mathbf{u_t}$. Also, it is independent of the observations $\mathbf{z_t}$ and the map $\mathbf{m}$. Whereas, the observation model describes the probability of making an observation $\mathbf{z_t}$ when the current robot pose and the landmark locations are known. Its described by the distribution $p(\mathbf{z_t}|\mathbf{x_t}, \mathbf{m})$.

The procedure giving a recursive solution for SLAM in the Bayesian framework is summarized in Algorithm(1). The input to the recursive Bayes filter algorithm is the belief at $t-1$, and it computes as output the belief of the robot at time t with the help of a prediction step using motion model (line 3) and a correction step using the observation model (line 4).

Algorithm 1 Recursive solution for SLAM using Bayes Filter.

Input: $p(\mathbf{x_{t-1}}, \mathbf{m}|\mathbf{z_{0:t-1}}, \mathbf{u_{0:t}}), \mathbf{u_t}, \mathbf{z_t}$

Output: $p(\mathbf{x_t}, \mathbf{m}|\mathbf{z_{0:t}}, \mathbf{u_{0:t}})$

1: **for** all $\mathbf{x_t}$ **do**

2: // Prediction Step

3: $p(\mathbf{x_t}, \mathbf{m}|\mathbf{z_{0:t-1}}, \mathbf{u_{0:t}}) = \int p(\mathbf{x_t}|\mathbf{x_{t-1}}, \mathbf{u_t}) p(\mathbf{x_{t-1}}, \mathbf{m}|\mathbf{z_{0:t-1}}, \mathbf{u_{0:t}}) dx_{t-1}$

4: // Correction Step

5: $p(\mathbf{x_t}, \mathbf{m}|\mathbf{z_{0:t}}, \mathbf{u_{0:t}}) = \eta p(\mathbf{z_t}|\mathbf{x_t}, \mathbf{m}) p(\mathbf{x_t}, \mathbf{m}|\mathbf{z_{0:t-1}}, \mathbf{u_{0:t}})$

6: **end for**

7: Return $p(\mathbf{x_t}, \mathbf{m}|\mathbf{z_{0:t}}, \mathbf{u_{0:t}})$

2.1.3 Some Existing Solutions

In this section, we will mention some of the frequently used solutions to solve the SLAM problem. Firstly, we will see a SLAM solution based on Extended Kalman Filter. Then, we give a short introduction to the graph based SLAM approach and finally, we describe an approach based on the Monte Carlo methods known as the particle filter.

2.1.3.1 EKF SLAM

Firstly, we will see how an Extended Kalman Filter (EKF) is applied to solve the SLAM problem. Historically, EKF based SLAM has been one of the first formulations and was introduced by Smith et al. [4] and Moutarlier and Chatila [5].

As mentioned earlier, the state vector consists of the robot pose and the positions of all the features in the map. As the robot moves and takes measurements, the state vector and the associated covariance matrix are updated using the standard equations of the Extended Kalman Filter. Maps created using EKF SLAM are normally based on features such as points, lines and planes etc. As new features are observed they are added to the state vector. The covariance matrix representing the uncertainty grows quadratically with the number of features and as a result the size of the map is limited to less than a thousand features in order to keep it computationally viable. However, many approaches have been proposed to deal with this limitation. We will discuss some of these techniques in the context of visual SLAM in the next chapter.

Initially as the robot takes the first few measurements, the covariance matrix is populated by these features assuming that they are uncorrelated. However when the robot starts moving, the robot pose and the features start becoming correlated. This makes the covariance matrix non-sparse (dense matrix with few non-zero entries). The reason for this correlation is that the uncertainty of the features in the map depend on the uncertainty in the robot pose. It also depends on the uncertainty of the other features which have been used to estimate the current robot pose. This implies that not only the features observed from the current pose are related but also those which are not observed directly become related. As more and more observations are made, the denser the covariance matrix becomes. The importance of these correlations which plays a key

role in the convergence of the SLAM algorithm is discussed in detail by Castellanos et al. [6] and Dissanayake et al. [7].

We will see the working of EKF SLAM by going through one cycle of the algorithm. Assume that the robot starts from an initial position (which may be considered as the origin of the global frame) with complete certainty of the initial pose. From this starting position, the robot observes some features and maps it with an uncertainty related to the sensor error model. As the robot moves its uncertainty increases due to errors in odometry. Later when the robot observes some new features it maps them with an uncertainty obtained due to the combination of the measurement error and the uncertainty in the pose. Supposing the robot moves back towards the starting position and re-observes a feature it knows with high certainty , the robot becomes more sure of its current location and as a result the pose uncertainty shrinks. Using this idea, the EKF based SLAM has been successfully implemented in various scenarios involving ground, aerial and underwater robots.

2.1.3.2 Graph SLAM

Graph based SLAM emerges from the idea that the SLAM problem can be interpreted as a sparse graph of nodes and constraints between the nodes. The nodes are the robot poses and features in the map. The constraints are the relative position between consecutive robot poses and the features observed from these locations. This formulation was first introduced by Lu and Milios [8] which has since influenced many other implementations. The graph based SLAM methods generally attempt to solve the full SLAM problem however some implementations for online SLAM have also been proposed.

The constraints in the graph based SLAM method are not treated as rigid but instead as soft constraints which can be relaxed in order to find a best estimate of the robot pose and the environment. The graph can be thought as an elastic net between the robot poses and features for which a solution can be found by computing the state of minimal energy of this net (Golfarelli et al. [9]). Many optimization techniques based on gradient descent approach have been proposed to solve this problem. An open source implementation with a very efficient optimization procedure was presented by Grisetti et al. [10].

The update time of graph based techniques is constant with a linear memory requirement in the number of features as compared to the quadratic complexity of maintaining the covariance matrix in an EKF approach. However, the full graph optimization problem becomes computationally expensive if the robot path is long. Still many graph based SLAM implementations presented by Bosse et al. [11], Duckett et al. [12], Thrun and Montemerlo [13] have shown successful results with millions of features.

2.1.3.3 Particle Filter SLAM

The SLAM solutions discussed till now have a parametric representation for landmarks and poses. Instead the state can be represented in a parameterless framework using particle filtering. Here the state estimate is represented by a set of samples (particles) drawn randomly from the robot's belief distribution. The advantage of using this representation is that it can model any sort of distribution and can also handle non-linear transformations.

The particle filter maintains a fixed number of particles at each time instant. Each particle consists an estimate of the robot pose as well as the features. When the robot moves, the motion model is applied to each particle to generate its new location and later when an observation is made a factor of importance is computed for each particle. The particles which explain the observation will get a higher importance. Finally a re-sampling step is performed where the current set of particles are replaced by another set which takes into account the importance factor which were determined.

Particle filters originate from the Monte Carlo methods and based on the work of Rao & Blackwell they were adapted to suit the SLAM problem. This is the reason why sometimes these filters are also referred to as Rao-Blackwellized particle filters. This Rao-Blackwellized filter was first applied to the SLAM problem in the bayesian framework by Doucet et al. [14]. Using the particle filter formulation, Montemerlo and Thrun [15] present an efficient algorithm called as *FastSLAM*. This method recursively estimates the full posterior over the robot pose and landmarks. Moreover, the algorithm scales logarithmically with the number of landmarks which allows thousands of landmarks to be mapped in real-time.

2.1.4 Open Challenges And Current Research

Most SLAM algorithms assume that the map is static. However, in the real world environments we have moving entities such as vehicles, humans etc. A good SLAM algorithm must be robust to these dynamic elements. A common approach to deal with dynamic objects is to treat them as outliers. Ideally, we would like to recognize these objects and predict their trajectory. This would result in a more efficient and informative map. We discuss some methods dealing with dynamic objects in the context of collaborative visual SLAM.

Although laser scanners are the most widely used sensors in SLAM, cameras offer an excellent alternative because they are cheaper and provide much richer information. Moreover they are also lighter allowing them to be deployed on small robots such as Micro Aerial Vehicles (MAVs). However, monocular vision has unique challenges of its own for example they provide bearing only measurements without any depth information, so the solution of SLAM will be true only upto a scale. In chapter 2.2, we will see in detail how this problem is tackled and other challenges in implementing visual SLAM .

One of the major issues in SLAM is its sensitivity to to incorrect data associations which is particularly important at loop closure when the robot returns to a previously explored area. We will see techniques employed for vision based SLAM to deal with this problem.

Another interesting direction of research is mapping using with multiple robots where local measurements from many robots can be fused to create a larger and more accurate global map. Chapter 2.3 is dedicated to such techniques.

2.2 Visual SLAM

SLAM is related to the problem of *Structure From Motion* (SFM) studied in the computer vision community. Its aim is to reconstruct, from a series of images, a three-dimensional model of an object. For this we must also reconstruct the camera motion (trajectory). The main differences between the two approaches is from the field of application: SFM does not have a real-time constraint and can use batch processing algorithms whereas the SLAM is intended to be implemented on a real-time platform. In addition, the robot needs to build incrementally its world model. Initially, we will

review techniques from the area of SFM using batch methods. Later we mention some recursive techniques for the SFM problem. And finally we see SLAM techniques using vision as the primary sensor.

2.2.1 Structure From Motion

The main goal of Structure from Motion (SFM) is to obtain the camera pose and scene structure from a sequence of images. Given some correspondences between these images, the idea is to compute the optimal estimate of motion and scene parameters.

Tomasi and Kanade [16] present an efficient factorization for structure and motion for orthographic cameras. Normally the scene geometry is specified as the distance between the camera and the feature points in the scene. However, this representation suffers from noise sensitivity which worsens with increasing depth. The proposed method avoids this problem by using the shape information (coordinates of the feature points in a world reference frame) instead of the depth values relative to the camera. In this approach, interest points are tracked through a sequence of images and then all the feature tracks are processed in parallel. Once the image measurements are available, the structure and motion estimates can be separated via Singular Value Decomposition (SVD) technique. The results obtained are robust to measurement noise but this method only holds for orthographic projection cameras and not for a general perspective projection where the projection is non-linear in depth.

Taylor et al. [17] tackle the problem of SFM for a simplified case of 2D motion and structure. This is relevant for indoor mobile robots which create a map of the environment using vertical line correspondences as features. The camera positions and the feature points are assumed to be in two dimensions. Like most of the subsequent SFM approaches it minimizes the re-projection error over the observations. This is formulated as a least squares minimization problem and an iterative approach is used to solve it. At each iteration, the current estimates of motion and structure provide a linearization point for the optimization algorithm. Since the re-projection error is taken as the objective function, all the measurement data is used for the estimation process. The authors suggest optimizing camera and scene parameters independently at each iteration. This

transforms the problem into a much lower dimensional linear equations at each iteration. Overall this approach showed better results than the EKF as it re-linearizes at each iteration around a better estimate.

A general SFM framework for 3D motion and structure estimation for calibrated perspective cameras is provided by Szeliski and Kang [18]. The perspective projections are expressed in a object-oriented coordinate system, as was previously done by Tomasi and Kanade [16], which ensures that the estimation is well conditioned even if the scene depth is small relative to the viewing distance. Following the idea of Taylor et al. [17], an iterative non-linear (least squares) optimization technique is used on all the observations (consisting of point and line features) to minimize the re-projection error. It employs the Levenberg-Marquardt algorithm which solves the full dimensional linear equation. The method showed rapid convergence even in the absence of any apriori knowledge of the shape or motion. A robust performance is achieved by gating and re-weighting the observations after convergence and then re-iterating.

McLauchlan and Murray [19] show that the Levenberg-Marquardt global optimization framework (a particular case of bundle adjustment) can be transformed such that each iteration takes cubic time in number of landmarks or number of poses (whichever is smaller). Either the motion or the structure parameters can be factored out of the main linear equation using Gaussian elimination to obtain a reduced order equation. The factored out parameters can then be computed using the other parameter. Finally, the per iteration complexity for N landmarks and M poses is $\mathcal{O}(\min\{N^3 + M, N + M^3\})$.

Pollefeys et al. [20] extend the bundle adjustment framework to accommodate time varying and unknown internal camera parameters. In the first step, projective recovery is performed which does not depend on knowing the internal parameters and then the camera parameter can be recovered using the absolute conic method proposed by Faugeras [21]. The absolute conic is defined as projective circle which is invariant to affine transformations and can be used to recover intrinsic parameters. Once the intrinsic parameters are found, projective reconstruction can be done at a metric level (without any scale ambiguity). An absolute reconstruction helps in searching for dense correspondences between pairs of images. These dense points can later also be triangulated to give a textured surface approximation to the scene.

Fitzgibbon and Zisserman [22] provide an efficient method to perform calibration and reconstruction from closed image sequences where camera poses form a loop. This approach is different from the previous SFM methods in the sense that the images are not processed in a sequential manner. Instead a hierarchical approach is employed using image triplets and trifocal tensors which distribute the error optimally over the whole image sequence. The main advantage of this method is that it can deal with closed loop sequences where extra constraints are brought in due to a part of the scene being revisited.

A detailed survey of bundle adjustment methods including advantages, trade-offs and implementation details is provided in Triggs et al. [23].

2.2.2 Recursive SFM Methods

The bundle adjustment (BA) methods we have seen in the last section require the entire sequence to be available as a result of which online recovery is not possible. Moreover, iterative optimization is computationally very expensive making it unsuitable for many applications. Therefore recursive methods which produce (structure and motion estimates) using data only upto the current time are necessary. In this section, we will review such recursive online estimation procedures which eventually lead to the online Visual SLAM methods.

Broida et al. [24] present one of the first recursive algorithms for estimating structure and motion of a rigid object. This approach makes the assumption that the object moves with a constant translational and rotational velocity but all other parameters are estimated from feature matching. The estimate is computed recursively using an iterated EKF which involves multiple iterations for each update to find the optimal point of linearization. The work has been demonstrated to estimate the pose of a moving object in front of a fixed camera however it can be easily adapted to the case of a moving camera observing a fixed object. In general the method showed good convergence however in the case of high sensor (observation) noise it diverges due to linearization errors (including the case where uncertainty in observations is correctly modelled).

Azarbayejani and Pentland [25] extend the framework of structure and motion estimate taking into account an unknown but static focal length. A reduced representation for 3D

point is proposed. A single parameter per point is estimated in the filter which encodes the depth of a point along a fixed ray in the direction of the first frame. This results in a filter which is three times smaller and therefore a 27 times reduction in number of computations (since each update is cubic in the dimension of the state vector). Later, however Chiuso et al. [26] revisit this one parameter representation and find that by not considering the angular uncertainty, the map generated will diverge in the general case.

Bouguet and Perona [27] evaluate the performance of a monocular recursive motion estimation technique for vehicle navigation. The proposed method uses a sliding-window approach where optimization is performed over the frames in the window and the older states are maintained by an EKF. Unlike in the batch approaches, all estimates older the window length are not re-linearized around the updated estimates. Therefore, the resulting estimate is suboptimal but instead provides for faster computations. The motion estimates for the camera are computed from the optical flow from each frame and tracked over time. Experiments were performed on video sequences taken from a vehicle moving on the highway and the results showed that it is possible to obtain a sufficiently accurate estimation of motion using a simple sensor such as a camera.

McLauchlan and Murray [19] propose a generalized statistical framework for recursive SFM methods. The approach maintains a variable dimension state vector which allows new features to be added and removed as the sequence progresses. Certain subsets of the estimated states are marginalized out periodically which corresponds to deletion of rows and columns of the covariance matrix in the EKF framework. However, the resulting filter is suboptimal for non-linear models (relative to an iterative batch approach). This is the case for all recursive type of SFM methods. The results also demonstrate that computing both motion and structure parameters are necessary to compute either with any accuracy.

Chiuso et al. [26] build upon the work of Azarbayejani and Pentland [25] and formalize a mathematical and geometric framework for recursive SFM. The work also presents an algorithm which is minimal and stable in the sense that the estimation error remains bounded throughout a sequence of arbitrary length. The algorithm is tested on monocular images with point features. Extending this idea, Favaro et al. [28] represent the scene with planar patches with orientation. Using EKF the surface normals associated to each patch is estimated. This gives us a richer representation of the scene structure.

It also takes into account the variation in lighting by normalizing the variance of the patches. The method also deals with landmarks which were occluded for sometime and then reappeared. These landmarks are re-observed and their structure estimate is updated. Feature tracking is performed in an innovative manner. The estimates of patches are projected back into the image and the search for features is performed in a region around the projection. This is a crude implementation of active search which is adopted in many subsequent works.

Instead of adopting EKF as in the previous approach, Nister et al. [29] uses a sliding-window bundle adjustment for monocular or a stereo rig. Each part of the algorithm is optimized for robustness and efficiency. Harris features (Harris and Stephens [30]) are tracked across the frames and outlier rejection is performed using the RANSAC 5-point algorithm (Nistér [31]). Position triangulation is performed using the earliest and the last observation as an approximation of the widest available baseline. Then the scale is recovered using the 3-point algorithm (Fischler and Bolles [32]). Finally, BA for structure and motion parameters over the three most recent frames provide a locally optimal estimate. This whole pipeline of estimating motion using only visual input was termed as visual odometry which has since become common terminology. The overall system showed impressive results with accurate motion recovery for long sequences and a real time performance (15 Hz). However it still suffers from the drift problem like all other recursive SFM methods. Unless the old landmarks are stored and re-observed later, drift will always be present while using noisy sensors.

2.2.3 SLAM using Vision

Unlike SFM methods, the works in this section maintain a more permanent representation of the structure (i.e. remembers the old landmarks) and uses this information to update the estimate when a previously explored region is revisited. We focus on methods which attempt to solve the SLAM problem using vision as the primary sensor.

Harris and Pike [33] present one of the first works which attempt at providing a drift-free solution to the recursive SFM problem. However the method lacked the insight regarding correlations provided by Smith et al. [4]. The landmarks chosen from feature points in a sequence of images were represented using a separate EKF as though they are independent of each other. As discussed by Castellanos et al. [6] and Dissanayake et al.

[7], the system cannot converge to the correct map since the correlations between the landmarks are ignored. However, the implementation showed two important properties. Firstly, the parametrization using disparity coordinates allowed representing uncertain landmark estimates in better way. The disparity coordinates were defined using the location of the landmark in the image along with the inverse depth of the landmark in the direction of camera view. Secondly, it presented one of the first implementations of the active search method. In active search approach, the current state estimate is used to determine a search area for further observations. Using an observation model a prediction is made as to where the landmarks should appear in the current image frame. As a result, the search for the landmark can be reduced to this area (typically described by an ellipse) instead of searching through the whole image.

Rahimi et al. [34] attempt to overcome the problem of drift in a differential tracking setting. This approach highlights the difference between recursive SFM and visual SLAM. Using a traditional differential tracking system, the pose is estimated using only frame to frame differential measurements. However this results in unbounded drift in the estimates. The paper provides a method for using past frames to obtain temporally non-local differential measurements. When a pose is sufficiently close to a previously saved state, tracking is performed with the newest frame and a set of past frames. A subset of all frames is preserved and updated continuously. Therefore the estimate of the current pose is modified to fit both the old and the new observations which puts a bound on the drift. Experimental results showed bounded error over time for head-tracking and camera ego-motion estimation

Neira et al. [35] implement a monocular visual SLAM algorithm using the stochastic map idea introduced by Smith et al. [4] and Moutarlier and Chatila [5]. The algorithm for a 2D SLAM for mobile robot equipped with odometry and a camera sensor is presented. Vertical edges are extracted from the image and tracked which are represented as 2D points in the plane of the robot's motion. The updates are performed using an EKF over the full state. Repeated observations of vertical edges reduce the uncertainty of the robot pose and thus closed trajectories are navigated without drift. This paper raises the important issue of data association which is the procedure by which sensor observations are associated with landmarks in the state. The edges are tracked by predicting their location using the current state estimates which in this case simplifies the data association problem. The idea is very similar to active search though the focus

in this work is not placed on image search while using prediction. The authors note that owing to the indirect nature of sensor observations as compared to the distance-bearing measurements provided by the range-finder sensor, a different approach for vision based SLAM is necessary. The experiments performed on a mobile robot mapping an indoor environment showed reasonable results with about 20 edge landmarks combined with odometry for a closed trajectory.

Se et al. [36] develop a visual SLAM system using trinocular rig which uses a different approach to address the data association problem. SIFT features (Lowe [37]) (scale invariant visual features) are extracted at each time step and matched to landmarks whose estimated projection is nearby in the image (an active search approach). The feature matching is confirmed using the orientation and scale associated to each SIFT feature. The overall state estimation is maintained using an EKF framework similar to Harris and Pike [33] where each landmark's estimate is considered independent of the other landmarks. However, the camera uncertainty is estimated at each time and propagated to the newly initialized landmarks. As a result of good odometry and the trinocular rig usable maps are generated, however in general this may not be the case as correlations between the landmarks are not considered. The distinctive nature of the SIFT features allow to solve the global localization problem and thereby addressing the kidnapped robot problem.

Davison and Murray [38], Davison and Kita [39] demonstrate a successful implementation of a SLAM system using vision as a primary sensor which provides an excellent baseline for the implementation of stochastic map based EKF SLAM using active search. The algorithm is tested for a mobile robot equipped with a stereo rig on a pan-tilt base and the only other input to the estimation is the odometry. An active search approach is employed to track the 3D point landmarks whose estimates are maintained using the EKF. The final map built by the system is purposefully sparse as it is primarily used for localization for which a dense representation is not required. Moreover, the quadratic time complexity of EKF methods makes using maps with a large number of maps infeasible for real-time performance. Despite the small number of landmarks since the map is carefully constructed, it can be used for accurate localization over small environments.

Extending this work, Davison [40] develops a real-time SLAM for a single hand-held camera. However, this time the problem is more challenging since there is no odometry

data available. In addition to the existing tools to maintain the EKF, a constant velocity model for the camera motion is incorporated which improves the stability of the system and makes active search feasible. One of the difficulties monocular SLAM faces is that just from one observation of a feature using a single camera the depth of landmark cannot be obtained. The problem of how the depth must be estimated and updated in the filter is called as initialization. One of the many approaches described in the structure from motion techniques can be used to recover depth in metric scale. Molton et al. [41] extended this system by including patch normals along with points as landmarks. The goal here was not to enrich the map representation but to increase the range over which active range for a given landmark is successful. Davison et al. [42] came out with an efficient and optimized version of the algorithm named MonoSLAM which runs at 30Hz with standard PC and camera hardware. The algorithm was demonstrated on a full-size humanoid robot and for live augmented reality using a hand-held camera.

2.2.4 Limitations Of EKF Based SLAM

The methods we have seen so far are based on the EKF framework. Although EKF provides a straightforward method for maintaining a stochastic map, it has many limitations. One of the main problems being the quadratic cost of the updates which constrains the number of landmarks used in the map if real-time performance is required. Moreover, even for small maps non-linear processes and observations can cause the filter to diverge. Both these problems have been addressed extensively in the SLAM literature. We only mention some of the key works without going into the details.

To deal with the computational issues of EKF, Dissanayake et al. [43] et al suggest map minimization where the idea is to limit the size of the map. They show that landmarks can be deleted form the state vector without compromising the consistency of the estimate. Whereas Csorba [44] recommends relative landmark formulation where instead of estimating landmark locations in a global frame, the filter estimates the relative transformations between pairs of landmarks. This representation allows EKF updates to be made using far fewer computations. Using this representation a Geometric Projection Filter (GPF) has been proposed by Newman and Durrant-Whyte [45]. Another idea was to exploit the fact that only a small subset of the state needs to be updated at any given time since the robot observes the landmarks in a area for a short amount of time

and then moves on. Based on this idea SLAM algorithms have been proposed by Knight et al. [46], Guivant and Nebot [47].

Even when the map is small such that the EKF is not limited by the computational demands, the estimation process can still give inconsistent results. In general, a consistent filter is one which given infinite number of observations converges to the truth. However with non-linear observation and dynamics model, EKF can diverge from the correct estimates. The consistency properties have been extensively studied by Julier and Uhlmann [48], Castellanos et al. [49], Bailey et al. [50].

As a result of the computational demands and consistency issues of EKF SLAM, divide-and-conquer methods for estimation on large maps have been developed. In these approaches, global maps are decomposed into many local maps of lesser complexity. Chong and Kleeman [51] suggest a simple method where they build a collection of local maps. When the uncertainty of robot pose increases beyond a limit, the robot saves the current map and starts creating a new map and also records the topological connection between the two. Leonard and Feder [52] present a sub-mapping strategy called as *Decoupled Stochastic Mapping* (DSM) where sub-map regions have predetermined geometric bounds in the global coordinate frame. Another method is proposed by Bailey [53] called *Network Coupled Feature Maps* (NCFM) where local maps (using local coordinates) are estimated using EKF which are joined by uncertain coordinate transformations resulting in a graph of local maps. More elaborate sub-mapping techniques have been discussed by Bosse et al. [11], Estrada et al. [54].

Instead of using sub-mapping techniques some SLAM implementations have been proposed taking advantage of the weak correlation between spatially and temporally distant state estimates. Guivant and Nebot [55], Thrun et al. [56] , Wang et al. [57] and Eustice et al. [58] propose several methods based on this idea of decorrelation.

2.2.5 Bundle Adjustment Based Implementations

The SLAM algorithms we have seen so far were based on filtering methods. Klein and Murray [59] propose a new system based on the batch optimization idea. In this approach, localization and mapping parts are split into two independent processes. They are processed on parallel threads on a dual-core computer where one thread deals with

the tracking of the camera pose and the other produces a 3D map of point features from previously observed frames. This splitting of the processes makes the computationally expensive batch optimization techniques possible to realize in real time. This system called as *Parallel Tracking and Mapping* is able to produce detailed maps with thousands of landmarks which can be tracked at frame rate. Experiments performed with a hand-held camera for a small workspace show accurate and robust performance of the system and matches the performance of other model-based system (such as an EKF based SLAM).

In order to keep the computations tractable and perform bundle optimization (BA) in real-time certain keyframes are selected which in some sense provide enough information without having to run the optimization process for every new frame. New keyframes are added based on criteria such as a minimal interval time between two consecutive keyframes (say 20-30 frames) and distance between the camera and the nearest landmark in the frame being greater than some distance etc. The BA techniques have the property of scale drift correction which means that it can converge to the correct solution (upto a scale) even if the initial values of the camera pose and feature positions are computed at a very different scale. So the idea of using BA on local maps and then joining these maps later provides an interesting approach to tackle the large scale monocular SLAM problem. Using this approach, Zhao et al. [60] implement a system in which local maps are built with SIFT features while using RANSAC at different levels to remove outliers and performing BA on them. Later these locals maps are put together by a joining algorithm to produce a global map. Strasdat et al. [61] address the issue of scale-drift with monocular SLAM and present a practical approach to deal with this drift effectively upon loop closures. Many interesting implementations using BA with vision as primary sensor have been presented by Konolige and Agrawal [62], Lim et al. [63], Mei et al. [64].

Sibley et al. [65] emphasize that the high computational complexity of BA is due to the choice of a single privileged coordinate frame and that the complexity could be reduced by adapting a relative approach. A new relative BA is described which instead of optimizing in single Euclidean space works in a metric space defined by a connected Riemannian manifold. Using an adaptive optimization algorithm strategy it was possible to solve the problem in constant time including at loop closure. The system was tested using an appearance based SLAM ([66]) which ran over 23,000 frames for a distance about 1 km with good accuracy.

Strasdat et al. [67] gives a detailed analysis of filtering versus keyframe based BA techniques. Filtering methods (such as the EKF) marginalize out past poses and summarize the information with a probability distribution whereas keyframe based methods retain the optimization idea of the global BA but must select a small number of past frames to process. The key question is to choose if it is necessary to maintain the probability distribution and propagate it over time or to discard some measurements in a way that repeated optimization from scratch becomes feasible. Rigorous tests were conducted to analyze the performance of the visual SLAM system using both approaches. The authors conclude that keyframe bundle adjustment outperforms filtering as it gives the most accuracy per unit of computing time. They found that increasing the number of observations (N) increases the accuracy while increasing the number of intermediate frames (M) only has a minor effect. Considering the cost of BA ($\mathcal{O}(N)$) to the cost of filtering ($\mathcal{O}(N^3)$), it is clear that keyframe based BA is more efficient especially if high accuracy is desired.

2.2.6 Dense Visual SLAM methods

In this section, we review SLAM algorithms which use complete image information instead of tracking some features over time. Though this results in high computational complexity, several methods have been proposed which provide a real-time performance.

Newcombe et al. [68] present a system named *Dense Tracking and Mapping* for camera tracking and reconstruction using every pixel methods. As compared to the sparse feature based methods, dense methods allow to better exploit the image data which leads to a higher pose accuracy. A detailed textured depth map at selected keyframes produce a surface patchwork with millions of vertices. The objective is defined in terms of photometric error over all pixels. The algorithm is parallelizable and achieves real-time performance using GPU grade hardware. Based on this technique several applications have been developed by Tanskanen et al. [69], Pradeep et al. [70], Prisacariu et al. [71].

With the advent of low-cost RGB-D cameras, many dense visual SLAM algorithms have been proposed. Kerl et al. [72] present such a system which minimizes both photometric and the depth error over all the pixels. They make use of a entropy based similarity measure for keyframe selection and for detecting loop closures. From all the successful matches, a graph is built which is optimized using the *g20* framework (Kummerle

et al. [73]). The approach was tested on various benchmark datasets and showed good performance including in scenes with low texture or having little structure. Similar approached using dense methods have been adopted by Newcombe et al. [74], Steinbrucker et al. [75], Whelan et al. [76].

2.3 Collaborative Visual SLAM

2.3.1 Localization of Multi-Robot Systems

Initial work in applying probabilistic techniques to multi-robot system was described by Fox et al. [77]. The main question this work attempts to answer is to what extent can cooperative multi-robot localization improve the localization quality when compared to a single robot localization. The paper presents a statistical algorithm for collaborative mobile robot (any-time) localization using a sample-based version (Monte Carlo localization) version of Markov localization. A detection model is defined which updates the belief when two robots see each other. When the team of robots localize themselves in a shared environment, each robot's belief is synchronized (updated) according to the detection model. Tests were performed on two mobile robots equipped with cameras and laser range finders for detecting the other robot. Supporting the hypothesis, the experimental results indeed showed a faster localization and a higher accuracy as compared to single robot localization.

Martinelli et al. [78] consider the problem of localizing simultaneously all the members of a team of robots. Each of the robot is equipped with a proprioceptive sensor (odometry) and an exteroceptive sensor (ex. camera) to sense other robots. The images captured provide relative bearing information between the robots. An extended Kalman filter is adopted which maintains the state containing configurations of all the robots. The experiments performed on two mobile robots suggest that the accuracy of the localization is strongly improved when relative bearing measurements are incorporated.

Furthering the idea of using bearing measurements by Martinelli et al. [78], Cognetti et al. [79] present a decentralized algorithm for estimation of mutual 3D poses in a group of robots such as a team of UAVs. The algorithm uses the bearing measurements which

are reconstructed from images captured by visual sensor and inertial measurements coming from the robot IMU. To avoid the cumbersome process of identifying a specific robot in the team, the bearing signals are simply assumed to be anonymous. While dealing with this lack of identity information, a probabilistic multiple registration of the measurements is performed where no global localization or distances are used. The results showed that this localization system using bearing only measurements exhibit similar accuracy as compared to the techniques using both bearing and distance information.

Achtelik et al. [80] propose a real time method to estimate the relative configuration of two robots navigating autonomously. The relative poses are recovered with absolute scale starting from an unknown initial configuration. A flexible stereo rig is formed by equipping each robot with a monocular camera. Further both the robots are installed with IMUs. The relative 6 DoF transformation between the two robots is estimated upto a scale using the feature correspondences in the shared field of view. Later the scale is recovered from the IMU measurements. All these measurements are fused using an EKF framework. This method was tested both experimentally (use two MAVs) and on simulation data showing results matching the ground truth with good accuracy.

2.3.2 Mapping With Multiple Robots

Rocha et al. [81] present a multi-robot system with the aim of cooperatively building 3-D maps of unknown environments. A distributed architecture for information sharing amongst the robots is described in a probabilistic framework for vision based 3D mapping. An entropy based information utility measure is defined which the robot uses for communicating with its team-mates. This measure helps in choosing the most useful measurements thus preventing the flooding of communication resources with redundant information. Experiments were conducted on two mobile robots with stereo vision sensor, sonar and having wireless communication capability. The results showed that with a cooperative setup greater number of measurements were incorporated per unit time and the overall time to accomplish the mapping task was less. The authors suggest that the time spent on communicating measurements to other robots, the time required for processing received measurements through communication and updating the map upon them are not be negligible and must be taken into account. The work also provides

guidelines regarding how communication schemes should be designed for multi-robot systems.

Howard [82] introduces a new method for representing two dimensional maps and shows how it can be used for SLAM problems involving multiple robots. The concept of a manifold map is explained where the idea is to transform planar maps into a two dimensional surface embedded in a higher dimensional space. This representation provides self-consistency which is lacking in the traditional planar maps. Also manifold graphs do not suffer from the crossover problem that planar maps commonly exhibit in environments containing loops. This self-consistency facilitates many desired capabilities such as robust retro-traverse, lazy loop closure, active loop closure using robot rendezvous and all of these features combined together provide excellent means of autonomous exploration. This multi-robot mapping approach was applied to a wide range of environments of varying size ($400m^2$-$900m^2$) and complexity using teams of two to four robots. The exploration resulted in the generation of good quality maps which could be used for higher level processes such as path planning or task allocation.

One of the main advantages of using a multi-robot system is the possibility of exploring larger areas which however brings a whole new set of challenges of its own. Vidal-Calleja et al. [83] describe a method to perform collaborative SLAM while dealing with the challenges emerging from a large semi-structured outdoor environments involving heterogeneous robots such as UAVs and ground mobile robots. A large-scale SLAM algorithm is designed in which a global graph maintains the relative relationships between a series of local sub-maps built by different robots. The robots use monocular camera as the primary sensor. The visual data from the camera is used to track a heterogeneous mix of visual landmarks to achieve effective cooperation. The system was tested for a team of robots consisting of a ground robot and a helicopter in a large outdoor environment. The distributed multi-map approach was able to cope up with large areas and handle communication failures. Moreover, the choice of heterogeneous landmarks allowed fusing of data from both the kinds of robots.

Forster et al. [84] present a centralized framework for collaborative monocular visual SLAM with multiple Micro Aerial Vehicles (MAV) in an unknown environment. It distributes the workload between the MAV's and the ground station while maintaining some autonomy with the MAV which ensures robust performance even case of communication

failures. Each MAV performs visual odometry on its on-board processor and estimates the motion. The group of robots act as a distributed system which send features from selected keyframes and relative pose estimates to the ground station. The ground station is responsible for estimating map of each robot and merges them whenever an overlap is detected. As a result each of the robot is able to localize itself in a global coordinate frame. The design of the data structures and processes play a significant role in the real time performance of the system. Moreover, the map building process is run in parallel threads allowing simultaneous reading and writing to the map. The system was implemented on two MAVs which showed good performance both in indoor($8m^2$) and outdoor environment($400m$ path).

Zou and Tan [85] attack the problem of visual SLAM using multiple cameras in a dynamic environment. A method is proposed which builds a 3D map of static objects in the environment and estimates the trajectory of the moving objects. Observations of the moving objects from multiple cameras are used to predict their trajectory. For this purpose an inter-camera pose estimation and mapping scheme is introduced. To facilitate the inter-camera operations, the cameras are clustered into groups based on their view overlap. Cameras having a significant overlap in their view are merged together in a group. As the cameras move and lose the overlap in view, the groups are split. Both the merging and splitting operations are performed in real-time. Experiment performed using upto 12 cameras showed that the system works robustly in highly dynamic environments and produces an accurate 3D map of the environment.

2.3.3 Decentralized Approach

In this section, we review some of the decentralized solutions which attempt at solving the problem of fusing data taken from multiple platforms.

Durrant-Whyte [86] show an early attempt at solving the data fusion problem in a distributed manner. The work describes a data fusion system consisting of a network of sensor nodes, each with its own processing unit which do not need any central fusion (or a centralized communication facility). Data fusion occurs locally at each node using local observations and the information is communicated to neighboring nodes. A decentralized Kalman filter based formulation is used to perform the data fusion. Demonstrations

using this system were carried out on group of four UAV's which successfully performed SLAM and were able to detect and track multiple ground targets.

Building upon this idea, Cunningham et al. [87] address the problem distributed SLAM using a decentralized data fusion approach where the goal is to achieve scalability within the available bandwidth. The proposed method is efficiently able to distribute map information across a team of robots and is robust to node failures and changes in network topology. It consists of a local optimization module which executes single robot SLAM, a communication module which propagates the local graphs to other robots and a neighborhood graph optimization module which combines all the local graphs into maps describing the neighborhood of a robot.

2.3.4 Cloud Based Architectures For Collaborative SLAM

The advantage of using a cloud based architecture is that the data intensive tasks can be offloaded from the on-board resources to a powerful back-end cluster system. Moreover as the back-end system can be shared by multiple clients, useful data (such as estimated maps) can be exchanged easily. In this section, we review some of the software frameworks which have been designed specifically for multi-robot collaboration.

Arumugam et al. [88] propose a software framework that provides the scalability and parallelism features of cloud computing for a team of robots operating in large environments. The framework is built around Hadoop [89] cluster using ROS (Robot Operating system) [90] as the messaging framework. Taking advantage of the *MapReduce* programming model, some of the robotics algorithms such as the FastSLAM has been implemented in a parallel fashion. As a result, performance gain in terms of time to build maps for large areas was achieved using a small eight-node Hadoop cluster. The global map generated can later be shared with new robots which are introduced in the scene using a *Software as a Service* model, thus reducing the burden of building the map from scratch again.

Hunziker et al. [91] present a cloud based architecture running parallel algorithms for collaborative 3D mapping using low-cost robots. Each of the robots run a dense visual odometry algorithm on the on-board processor (a smart-phone class ARM processor) which estimates the pose using both the color and depth images from the RGB-D sensor. In addition, certain keyframes are selected which in some way summarize the motion

of the robot. These keyframes are sent to the cloud for parallel optimization and then merged with the maps from other robots. Once the optimization process is complete, the updated poses of the local frames are sent back to the robots. The back-end is managed by a cloud robotics framework running in some commercial data center. The experimental setup included two low cost robots equipped with RGB-D cameras and are connected wirelessly to the cloud server. The results showed higher accuracy for visual odometry while using cloud supported global optimization as compared to odometry run without the optimization. The two robots were also successfully able to make a detailed point cloud map of a $40m$ long corridor.

Similar to the previous work, Riazuelo et al. [92] present a visual SLAM system based on a cloud framework where the expensive map optimization task is computed on the server while a simple camera tracking task is performed on the on-board processor. The experiments showed real time performance for collaborative SLAM using RGB-D cameras. The system provides an interface where a map can be built and stored, existing maps can be reused by other robots and each robot can estimate its individual map and fuse them together with other maps if an overlap is detected.

2.3.5 Coordination Strategies For Multi-Robot Exploration

How should a team of robots proceed towards exploration of an unknown environment is a key question in multi-robot systems. Coordination amongst team members plays an important role for a successful implementation of a collaborative task. Burgard et al. [93] present a strategy by appropriately choosing target points for individual team members such that the overall exploration time is minimized. This approach takes into account the cost of reaching the target and its utility which is given by the size of the unexplored area that a robot can cover using its sensor upon reaching that location. Both real and simulated experiments in various scenarios showed better results (in terms of total exploration time and area covered) as compared to a naive or a greedy approach.

Various techniques for exploring unknown environments have been presented by Cohen [94], Koenig et al. [95], Billard et al. [96], Balch et al. [97], Singh and Fujimura [98], Bender [99] . However, we do not detail these works here since our main focus lies in developing of a localization and mapping capability for a team of robots whereas coordination strategies deal with a higher level of planning and task management.

Chapter 3

Monocular Visual SLAM

3.1 Introduction

Given a sequence of images, a monocular SLAM algorithm aims to obtain the trajectory of the camera and the structure/model of the environment. Several techniques for real time monocular SLAM and 3D reconstruction have been developed recently [100],[101],[102]. They are being used for different application such as navigation of UAVs, virtual reality (VR) experiences etc.

The use of camera as the main perception sensor brings certain flexibility to the system. Due to its compact size and light weight nature, it could be mounted almost anywhere on the robot. Its particularly convenient for small sized robots like MAVs where the use of other perception sensors such as laser scanner is difficult. Moreover, cameras are able to provide rich data of the surroundings at relatively high rates. However, on the down-side there is a lack of direct 3D information from the camera. Therefore, the depth of the objects in the scene needs to explicitly computed somehow. Another disadvantage being that images captured by the camera are very sensitive to lighting conditions. This often places several restrictions on the type of workspace environments.

One of the main challenges of monocular SLAM is to deal with the inherent scale-ambiguity problem. To have a metric reconstruction of the world, the scale must be tracked explicitly. In practice, the scale estimate often drifts with time making the SLAM process to diverge. On the other hand, the system has the advantage of being used in differently scaled scenes in both indoor and outdoor environments. Typically,

this is not possible for other *scaled sensors* like depth or stereo cameras. These sensors have a limited range of depth for which they can provide reliable measurements.

In this chapter, we present a semi-dense monocular SLAM algorithm based on the work of Engel et al. [101]. In our collaborative SLAM scheme, each individual robot runs this algorithm on their on-board processor.

3.1.1 Contributions

This chapter describes a real-time monocular SLAM system for estimating the pose (including scale) of the camera and building a 3D map of the environment. The main contributions made in this chapter are:

- Adapting an existing semi-dense monocular SLAM approach (LSD-SLAM [101]) for our system.
- Evaluation of the monocular SLAM process on the live trials conducted.

First, section 3.2 describes the pose-based graph optimization procedure which will be used in the visual SLAM process. Then, section 3.3 describes a semi-dense SLAM scheme based on LSD-SLAM [101]. Finally, section 3.4 gives some results of the visual SLAM process from the live trials conducted in a semi-industrial indoor environment.

3.1.2 Related Work

There are two main approaches to solve the visual SLAM problem :(1) Filtering based approach (2) Keyframe-based optimization approach. In a filtering approach, measurements from all images are fused sequentially by updating the probability distributions over feature and camera pose parameters [42], [36], [103]. Whereas in keyframe-based approach an optimization over selected keyframes from the image stream is performed which gives a long-term drift free performance [59], [60], [62].

Another dimension along which visual SLAM algorithms are classified are feature-based versus direct methods. In feature based methods, first a set of features are extracted from the image. Then the camera position and the scene geometry is estimated as a function of these features. Using features to represent the image reduces the computational

complexity involved. However, as a result only the information that conforms to the feature type can be used. Whereas in a direct method, the whole image information is used. Typically, the camera pose is estimated by solving an optimization problem to minimize the photometric error between consecutive images. This enables the use of all the information contained in the image. Successful implementations of both feature-based ([100],[102]) and direct methods ([68], [104]) have been demonstrated.

Many other works related to monocular visual SLAM and their implementations can be found in chapter 2.

3.2 Pose-Graph Optimization

In this section, we describe a technique for pose-based graph optimization which helps in building a consistent map. In this approach, the SLAM problem is interpreted as a graph where the nodes correspond to the poses of the robot at different times and the edges represent constraints between the poses. These constraints are obtained from observations of the environment (sensor measurements) and movement of the robot (odometry). Once this graph is constructed, the map is obtained by finding the arrangement of nodes that is most consistent with measurements modeled by the edges.

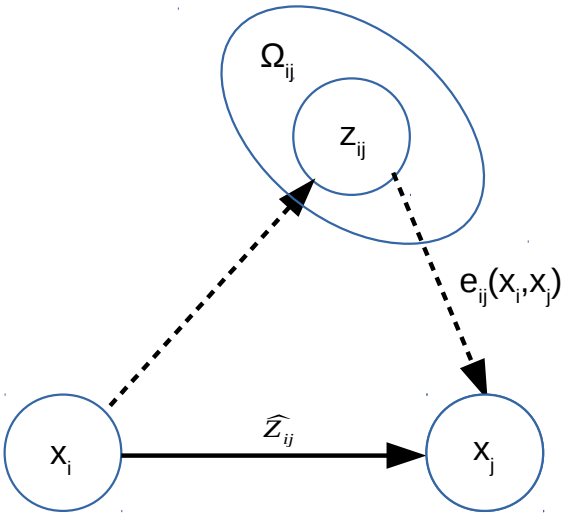


FIGURE 3.1: Portion of graph with node and edge definitions

Consider a graph with the nodes $x = \{x_1, \dots x_i, \dots, x_n\}$ where each node x_i represents the pose of the robot. Let $\mathcal{C}$ be the set containing pairs of indices $< i, j >$ for which a constraint exists in the graph .The measurement between node i and node j is represented by its mean z_{ij} and the information matrix Ω_{ij}. This measurement z_{ij} is basically the transformation that makes the observation from i maximally overlap with the measurement made at j. For a given configuration of x_i and x_j, $\hat{z}_{ij}$ is the predicted measurement. Finally, an error e_{ij} is defined as the difference between the actual measurement z_{ij} and the predicted measurement $\hat{z}_{ij}$.

$$e_{ij} = z_{ij} - \hat{z}_{ij} \tag{3.1}$$

In addition, the log-likehood l_{ij} of the measurement is given by

$$\begin{aligned} l_{ij} &= (z_{ij} - \hat{z}_{ij})^T \Omega_{ij} (z_{ij} - \hat{z}_{ij}) \\ &= e_{ij}^T \Omega_{ij} e_{ij} \end{aligned} \tag{3.2}$$

Given these definitions, our goal is to find out a configuration of the nodes that minimize the error introduced by the constraints. This can be done by minimizing the negative log-likelihood $F(x)$ of all the observations:

$$F(x) = \sum_{<i,j> \in \mathcal{C}} e_{ij}^T \Omega_{ij} e_{ij} \tag{3.3}$$

The solution to pose-graph optimization problem is given by:

$$x^* = \underset{x}{\operatorname{argmin}} F(x) \tag{3.4}$$

3.2.1 Optimization Procedure

Equation(3.4) can be solved using a Gauss-Newton minimization procedure. The solution is obtained by iteratively solving the linear system:

$$H \Delta x = -b \tag{3.5}$$

where the Hessian matrix $H = \sum_C J_{ij}^T \Omega_{ij} J_{ij}$, the coefficient vector $b = \sum_C J_{ij}^T \Omega_{ij} e_{ij}$ and J_{ij} is the Jacobian of the error function. At each step, the perturbation Δx is computed. This Δx is then applied to the previous x to obtain an improved solution.

For the SLAM problem, the Hessian H and the coefficient vector b have a special structure. This structure can be exploited to solve equation(3.5) efficiently. Note that H and b are obtained by summing up a set of matrices for each constraint in the graph. The structure of these terms depend on the Jacobian of the error function. As the error function e_{ij} depends only on node i and node j, the Jacobian J_{ij} can be written as:

$$J_{ij} = \begin{pmatrix} 0 & \dots & 0 & \underbrace{A_{ij}}_{\text{node } i} & 0 & \dots & 0 & \underbrace{B_{ij}}_{\text{node } j} & 0 \end{pmatrix} \tag{3.6}$$

where A_{ij} and B_{ij} are derivatives of the error function with respect to x_i and x_j respectively. Using equation(3.6), H and b can be expressed as:

$$\begin{aligned} H_{ij} &= \begin{pmatrix} \ddots & & & & \\ & A_{ij}^T \Omega_{ij} A_{ij} & \dots & A_{ij}^T \Omega_{ij} B_{ij} & \\ & \vdots & \ddots & \vdots & \\ & B_{ij}^T \Omega_{ij} A_{ij} & \dots & B_{ij}^T \Omega_{ij} B_{ij} & \\ & & & & \ddots \end{pmatrix} \\ b_{ij} &= \begin{pmatrix} \vdots \\ A_{ij}^T \Omega_{ij} e_{ij} \\ \vdots \\ B_{ij}^T \Omega_{ij} e_{ij} \\ \vdots \end{pmatrix} \end{aligned} \tag{3.7}$$

Algorithm(2) gives a summary of an iterative Gauss-Newton procedure to compute the optimal pose x^* (along with the information matrix H^*). As the matrix H is sparse, a memory efficient representation can be used to store it. Since the structure of H is known before-hand from the connectivity of the graph, its memory can be pre-allocated and then updated in place by looping over all edges when a new linearization is required. Note that each edge contributes only to the blocks $H_{[ii]}$, $H_{[ij]}$, $H_{[ji]}$, $H_{[jj]}$ and to the blocks $b_{[i]}$, $b_{[j]}$ of the coefficient vector.

Algorithm 2 Pose-Graph Optimization Using Gauss-Newton Algorithm

Input: Initial pose: x, Constraints: $\mathcal{C} = \{< e_{ij}, \Omega_{ij} >\}$

Output: Optimal pose: x^*, Final Information Matrix: H^*

while ¬converged **do**
  $H \leftarrow 0 \quad b \leftarrow 0$
  for all $< e_{ij}, \Omega_{ij} > \in \mathcal{C}$ **do**
    // Compute the Jacobian A_{ij} and B_{ij} of the error function
    $A_{ij} = \frac{\partial e_{ij}(x)}{\partial x_i} \quad B_{ij} = \frac{\partial e_{ij}(x)}{\partial x_j}$
    // Compute contribution due to constraint e_{ij} to the linear system
    $H_{[ii]} += A_{ij}^T \Omega_{ij} A_{ij} \quad H_{[ij]} += A_{ij}^T \Omega_{ij} B_{ij}$
    $H_{[ji]} += B_{ij}^T \Omega_{ij} A_{ij} \quad H_{[jj]} += B_{ij}^T \Omega_{ij} B_{ij}$
    $b_{[i]} += A_{ij}^T \Omega_{ij} e_{ij} \quad b_{[j]} += B_{ij}^T \Omega_{ij} e_{ij}$
  end for
  // Solve linear system using sparse Cholesky factorization
  $\Delta x \leftarrow \text{solve}(H \Delta x = -b)$
  // Update the pose
  $x = x + \Delta x$
end while
$x^* \leftarrow x$
$H^* \leftarrow H$
return x^*, H^*

3.3 Semi-Dense Monocular SLAM

In this section, we describe a semi-dense visual SLAM method using a direct(without features) approach. This method allows to estimate in real time the pose of the camera (including scale) and create a map of the environment as a pose-graph of keyframes. Each keyframe has an associated semi-dense depth map which is computed by performing stereo comparisons over consecutive image frames. Moreover, the technique is able to deal with large variations in scene depth by employing a *scale-drift aware* formulation .It is based on the LSD-SLAM implementation by Engel et al. [101]. The overall scheme is shown in Figure 3.2. It consists of the following main components:

- *Tracking:* This component tracks each new image from the camera. It computes the transformation $T \in se(3)$ between the new image and the previous keyframe image.

- *Depth Map Estimation:* This component computed the depth of the scene by making several stereo comparisons over consecutive image frames. The depth is computed only for a subset of the pixels in the image. Therefore, the resulting depth map can be referred to as semi-dense depth map.

- *Map Optimization:* This component performs an optimization procedure over the keyframe-graph as described in section 3.2. Loop closures and scale estimation is incorporated through the constraints in the graph.

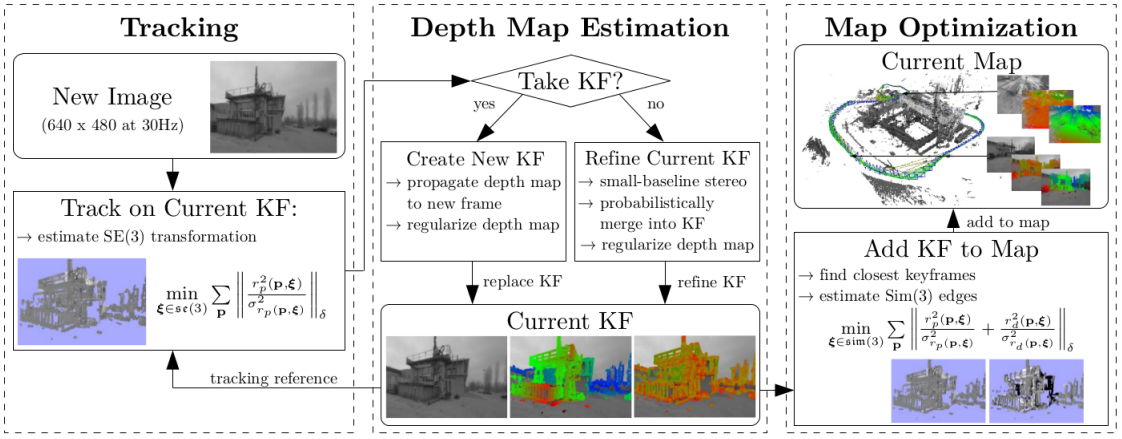


FIGURE 3.2: Overall scheme of LSD-SLAM Algorithm [101]

3.3.0.1 Tracking

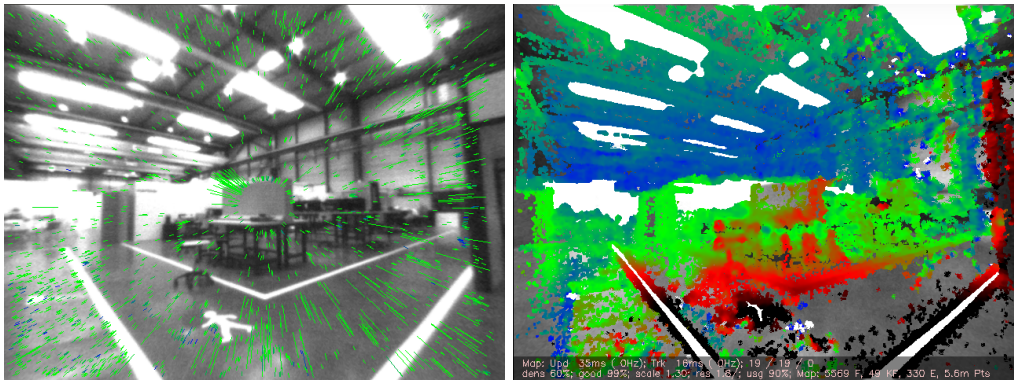
The camera pose $T \in se(3)$ is estimated with respect to the current keyframe K_i which consists of the image (I_i), the depth map (D_i) and the depth map variance V_i. For each new image I_j, the relative pose $T_{ji} \in se(3)$ is computed by minimizing the photometric error:

$$T_{ji} = \underset{T}{\operatorname{argmin}} \sum_{p} \left\| e_p^2(p, T_{ji}) \right\| \tag{3.8}$$

where the photometric residual $e_p(p, T_{ji}) = I_i(p) - I_j(\omega(p, D_i(p), T_{ji}))$ and ω is a warping function which computes the location of a pixel from the first image in the second image given the relative transformation T_{ji}. Note that in the actual implementation, a variance

normalized residual is minimized thereby implicitly including depth accuracy in the computation of T_{ji}. The optimization problem can be posed as a weighted least squares problem [105] which can be solved using the Gauss-Newton minimization method [73].

3.3.0.2 Depth Map Estimation



(A) Small Stereo Baseline Comparisons (B) Semi-dense Depth Map

FIGURE 3.3: Depth Map Estimation

A semi-dense inverse depth map is continuously estimated for each new frame. The depth map is computed by making several stereo comparisons of varying baseline over consecutive frames of the input video. The variable base line allows for accurate estimation of both near and far regions of the image. The method maintains a probabilistic depth hypothesis for each pixel modeled by a gaussian distribution which is continuously refined using an filtering approach described in [104]. Finally when the camera moves far from the current keyframe, a new keyframe is created and its depth map is initialized by projecting points from the previous keyframe on it. Figure 3.3a shows the small stereo baseline comparisons used to compute the depth of the scene. Figure 3.3b shows the resulting semi-dense depth map. The distance of the objects from the camera increases with the color changing from red to blue and then from blue to green.

3.3.0.3 Map Optimization

The frame to frame alignment method previously introduced in 3.3.0.1 inherently accumulates drift over time due to small errors in each estimate arising from sensor noise and other model inaccuracies.

To deal with this problem, the SLAM system maintains the map as a graph where each vertex is the pose of the keyframe and each edge represents the relative transformation between the corresponding keyframes. Each time a new keyframe is added to the map, new edges are created and finally when previously visited regions of the scene are encountered, additional edges (loop closures) are added which help in reducing the accumulated drift.

However in the case of monocular SLAM, the scale of the scene cannot be observed directly which over a long trajectory leads to a drift causing major errors in tracking. To take care of the scale parameter, the overall pose graph is constructed in a manner such that the mean inverse of each keyframe is one and instead the edges are represented as transformation $T_{ji} \in sim(3)$. This allows the integration of the scale parameter directly in the optimization problem. So, the scaled transformation between the keyframes is estimated by minimizing the error function:

$$E(T) = \sum_{p} \left\| e_p^2(p, T_{ji}) + e_d^2(p, T_{ji}) \right\| \tag{3.9}$$

where the depth residual is $e_d(p, T_{ji}) = X_{2,z} - D_2(\pi(X_2^{'}))$ and

- $X_2^{'}$:= Transformed 3D point obtained by applying T to X_1, i.e. $X_2^{'} = \left(X_{2,x}^{'} X_{2,y}^{'} X_{2,z}^{'}\right) = T.X_1$
- π := camera projection function which computes the pixel location (u, v) given the 3D point X using the camera calibration parameters.

Finally, the overall map consisting of keyframe poses as vertices and $sim(3)$ constraints as edges, is continuously optimized in parallel using a general graph optimization framework like g2o [73]. The pose-graph optimization procedure given in Algorithm 2 is used to compute the optimal solution. This optimization over the graph reduces the drift both in scale and pose estimates.

3.4 Results

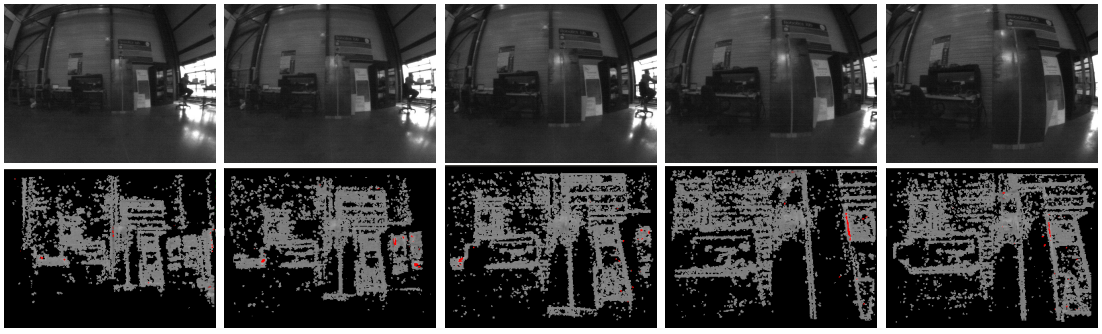
3.4.1 Experimental Setup

In order to test the visual SLAM algorithm, we use a hand-pushed cart as our mobile platform. The camera was attached on the front side of the cart as shown in Figure 5.2. A uEye camera with wide-angle lens (130 degrees Field of View) was used for the experiments. A core-2 duo laptop was used to do all the computations. The images were captured at 30 Hz with a resolution of 640 x 480 pixels and processed by the SLAM algorithm in real time. All the experiments were conducted in an industrial-like indoor environment approximately 20m x 20m.

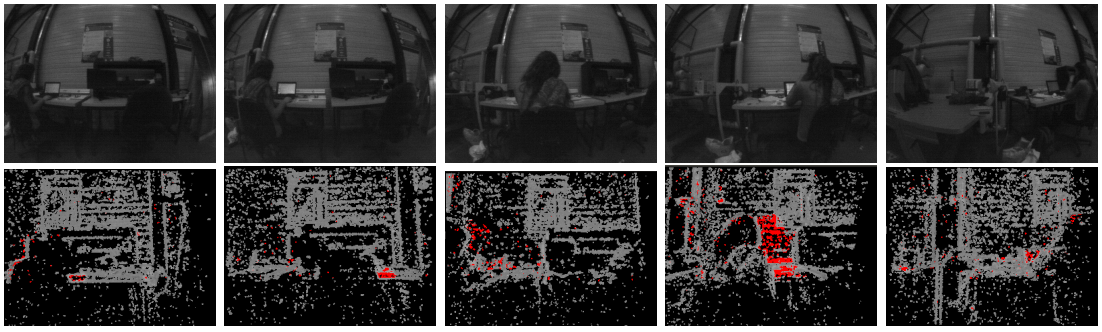
FIGURE 3.4: Hand-pushed cart with laptop at the top. The monocular camera is attached at the front for performing visual SLAM

3.4.2 Semi-dense monocular SLAM: Performance Evaluation

To illustrate the performance of the visual SLAM algorithm, we show results from a live test run conducted in out lab space (which is a semi-industrial indoor environment). Figure 3.5 shows the tracking performance of the algorithm. Two image sequences along with the corresponding tacking residual is shown. In Figure 3.6 , the depth estimation process is illustrated. Finally, the overall map of the environment in shown in Figure 3.7.

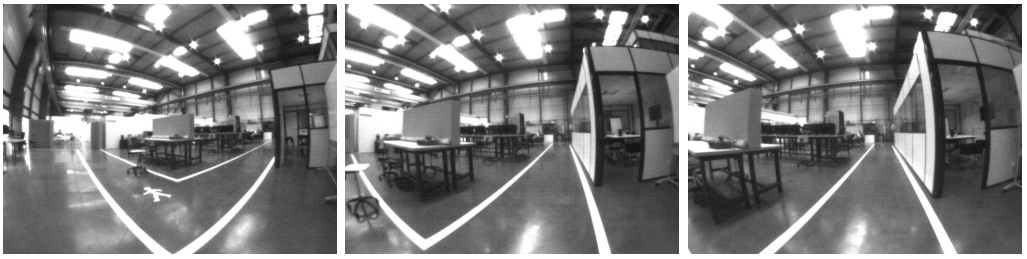


(A) Camera moving towards the wall



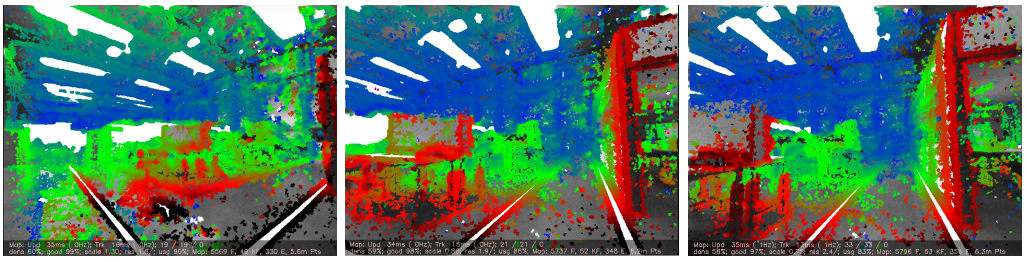
(B) Camera moving right to left across the scene

FIGURE 3.5: Tracking error between consecutive frames. The first row in each case shows the consecutive images of the sequence and the second row shows the corresponding pixels (gray) in each image which are tracked. These are the pixels with high image gradient. And the red regions represent pixels having high tracking error. In case (A), the camera moves towards the wall while in case (B), it moves across the scene from right to left. In both the sequences, we see very few pixels in red indicating that the overall tracking performance is good.



(A) Consecutive image frames

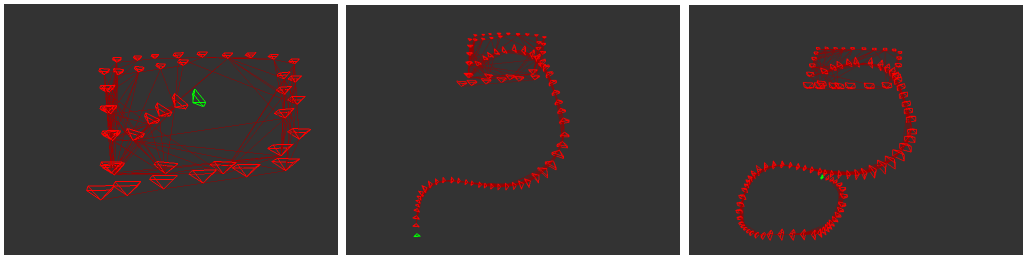
(B) Small baseline stereo comparisons



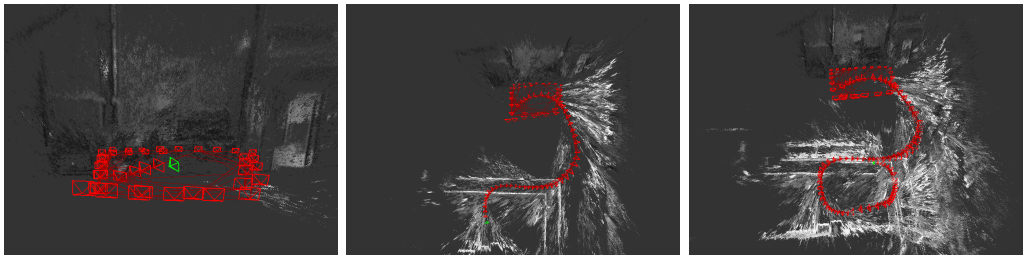
(C) Color coded depth map

FIGURE 3.6: Depth Map estimation between consecutive frames. The first row (A) shows images captured by the camera at three consecutive time instants. In the second row (B), the green lines in each frame shows the small stereo baseline comparisons made by the algorithm. These are used to compute the depth of the corresponding pixels over consecutive frames. Through a filtering process, these depth estimates are fused together to generate the depth map. (C) shows the depth for each frame in a color coded format. Here the color changes from red to green to blue as the distance of the objects from the camera increases.

(A) Images at three different instances



(B) Pose-graph of keyframes



(C) Semi-dense point cloud associated with each keyframe

FIGURE 3.7: Map of environment as a pose graph of keyframes. Snapshots are three different time instances of the visual SLAM process is depicted. Each column corresponds to an unique time instance. The first row (A) shows the camera view at these moments. The second row (B) shows the map that has been built until that instant. The map of the environment is represented a pose-graph of keyframes. The red pyramids in the image depict the position of the camera at instant when the keyframe is added to the graph. The green pyramid shows the current position of the camera. The lines between keyframes represent the constraints binding them. We can also see several loop-closures being made as the robot revisits old places. Finally, in (C) the semi-dense point clouds associated with each keyframe is shown.

Chapter 4

Scene Overlap Detection

4.1 Introduction

Scene overlap detection forms a vital component of the collaborative SLAM system for multiple robots. The overlap detection is necessary to find out if two robots have visited the same part of the environment. Once a scene overlap is detected between two robots, their relative positions in a global coordinate frame can be determined. Moreover, at the location of the detection, the maps generated by individual robots can be merged to form a combined global map.

For our intended application, the relative positions of the robots is not known at the beginning. This must be estimated by our collaborative SLAM system. At the time of an overlap detection, the only source of information about the environment is through the images captured by the camera. Therefore, the overlap between two robots must as well be detected using these images itself.

In this chapter, we consider the problem of detecting whether two scenes (also referred to as *places*) are similar based solely on the visual information in the images. This problem has been studied by the computer vision community as the problem of *Appearance-based Place Recognition*.

In summary, we are looking for an appearance-based technique which has the following characteristics:

- Recognize similar scenes in a fast (online) manner.

- Be robust to perceptual aliasing (i.e. discriminate between scenes which may look similar but do not correspond to the same place in the environment).
- Work without the need of any supervision.

A solution meeting all these criteria has been proposed by Cummins and Newman [106] called FAB-MAP (*Fast Appearance-based Mapping*). Traditionally this technique has been used for finding loop closures over long distances and performing global localization in a topological map. We adapt this technique to be able to detect similar scenes observed from multiple cameras.

4.1.1 Contributions

This chapter describes how to detect if the same place has been visited by two robots using visual information only. To this effect, an appearance-based approach is explained. The main contributions made in this chapter are:

- A scheme using appearance-based matching to detect scene overlap amongst multiple cameras.
- Adaptation of the existing FAB-MAP technique to accommodate input images from several cameras.
- Testing the system on real datasets captured at an industrial-like indoor environment.

4.1.2 Related Work

Several appearance-based methods for place recognition have been described in the context of detecting loop closures by Levin and Szeliski [107], Se et al. [108] and Ho and Newman [109]. These methods try to find out if the robot is visiting a previously mapped region by measuring similarity in their sensory information. This sensory similarity is computed independently of the robot's estimated metric position.

Efficient representation and matching of visual information are necessary for the successful implementation of an appearance-based system. Several works have been described

in this regard. Many of these are based on applying dimensionality reduction techniques to the input images. Ulrich and Nourbakhsh [110] represent the appearance of a place as a set of image histograms. Lamon et al. [111] do it using an ordered sequence of color and edge features. Another idea proposed by Torralba et al. [112] uses texture features to represent the scene. Kröse et al. [113] performs a *Principal Component Analysis* (PCA) to reduce the dimensionality of the input images. A Gaussian distribution is used to represent the place in a low-dimensional space. Finally, a probabilistic model is defined for appearance based robot localization. Ramos et al. [114] used a dimensionality reduction technique in a Bayesian framework where a generative model of the place is learned. However, both these techniques require significant supervised training to learn the generative place models.

All the methods mentioned so far use a single global descriptor for each image. These global descriptors are not very robust as they are sensitive to changes in lighting conditions, perspective changes and dynamic objects which cause parts of the image to change from observation to observation. To deal with these problems, many approaches use sets of local features to represent a place. Several local features have been developed which are robust to transformations such as rotation, scaling, some illumination changes and allow object recognition even when the scene is partially occluded. These features are computed using a region-of-interest (ROI) detector along with a descriptor of the local region. SIFT (Lowe [115]) and SURF(Bay et al. [116]) features are two widely used local features.

Several approaches for place recognition based on local invariant features have been proposed by Sim and Dudek [117], Wolf et al. [118] and Li and Kosecka [119]. Wang et al. [120] present a technique inspired by the text retrieval methods. It employs a visual vocabulary built using local feature descriptors as proposed by Sivic and Zisserman [121]. Each image is now represented as a *Bag of Words* (BoW) where a word corresponds to a particular feature in the space of the invariant descriptors. This representation allows a fast search between images using techniques developed for text retrieval.

Another challenging aspect of appearance-based matching is due to the problem of perceptual aliasing. This aliasing occurs very often as many different parts of the environment look similar. The situation is often aggravated due to repetitive features arising as a result of similar looking architectural structures or objects at different places. Several

ideas have been proposed to address this problem (Chen [122], Geodeme [123] and Kröse et al. [113]). Some other methods based on similarity matrices have been proposed by Levin and Szeliski [107], Zivkovic et al. [124] and Newman et al. [125]. In these appraoches, a pairwise similarity between all observations is computed to give a square matrix of matching probabilities. If the input images are given chronological order, then the off-diagonal elements represent previously visited places.

In our work, we use the FAB-MAP technique proposed by Cummins and Newman [106]. It provides a probabilistic approach for recognizing places based on their appearance. In our system, we adapt this technique to detect a scene overlap between multiple robots.

4.2 Appearance Based Place Recognition

In this section, we study a probabilistic approach for recognition of places based on their appearance. It is based on the FAB-MAP system developed by Cummins and Newman [106]. The system is able to compute a probability with which a given image matches any of the already seen images. In addition, it also computes the probability of the image being a new one (i.e. not seen before).

The approach is based on the *Bag of Words* (BoW) image retrieval system. Using the BoW data from the input images, a generative model is learnt. This generative model captures the fact that certain *visual words* occur together as they are generated by common objects in the environment. Moreover, a model of the common objects is learnt in an offline training step without any supervision. Such models improve the inference process by adding cues about which sets of features (visual words) are likely to appear or disappear together. The generative models also help in solving the problem of perceptual aliasing. As a result, the system has a much better ability to discriminate between scenes with many similarly appearing places.

We first describe the probabilistic framework for the appearance-based matching process. Later, we propose a scheme to use this technique in order to detect a scene overlap among the robots.

4.2.1 Appearance Based Matching: A Probabilistic Framework

We now describe a probabilistic framework for the appearance-based matching of images. In fact, the model we describe can be considered as a complete SLAM system in the space of appearance. This is because the system is not only able to match the current image (observation) to the previously seen places (Localization task) but also determine whether the observation comes from a new place which was not encountered before. Therefore, it augments the map with this new information (Mapping task).

The images are handled by the system using their BoW representations. For each incoming image, visual features (ex. SURF/SIFT) are computed which are view-point invariant. These features are then quantized with respect to a vocabulary to give a visual words description of the scene. The vocabulary used is generated in an offline step from a huge set of training data. This mapping of image features into visual words enables the use of fast matching and retrieval techniques.

A probabilistic model over the BoW representation of each scene is defined. At any given time k, the observation of the scene is denoted by $Z_k = \{z_1, \ldots, z_i, \ldots z_V\}$ where V is the number of visual words in the vocabulary. Each component of the observation z_i is a binary variable that takes the value 1 if the i^{th} visual word is present in the observation. $\mathcal{Z}^k$ is used to denote the set of all observations upto time k.

The map of the environment is represented as a discrete set of locations $\mathcal{L}_k = \{L_1, \ldots, L_{n_k}\}$. For each location L_i, an associated appearance model is defined by the set:

$$\mathcal{A}_i = \{p(e_1 = 1|L_i), p(e_2 = 1|L_i), \ldots, p(e_V = 1|L_i)\} \tag{4.1}$$

where the variables e_i models the feature existence, which is different from the previously defined z_i which models the feature observation. Both these variables, i.e. the existence e_i and feature observation z_i are related via the detector model $\mathcal{D}$ which defines the false positive and false negative word detections.

$$\mathcal{D} := \begin{cases} p(z_i = 1|e_i = 0) & \text{false positive probability} \\ p(z_i = 0|e_i = 1) & \text{false negative probability} \end{cases}$$

Normally, certain sets of visual words tend to occur together. This is because these words are generated due to the same object in the scene. For example, the words associated with the door handle and the door corners are likely to be observed simultaneously. These dependencies among the words are captured using a Chow-Liu Tree [126]. This tree structure provides an efficient way to represent the joint distribution of the word occurrences. Further, it also facilitates efficient learning and inference even for a very large vocabulary.

Now, given the probabilistic appearance model, a recursive Bayes estimation problem is solved to compute the distribution over the location L_i:

$$p(L_i|Z_k) = \frac{p(Z_k|L_i, \mathcal{Z}^{k-1})p(L_i|\mathcal{Z}^{k-1})}{p(Z_k|\mathcal{Z}^{k-1})} \tag{4.2}$$

where $p(Z_k|L_i, \mathcal{Z}^{k-1})$ is the observation likelihood, $p(L_i|\mathcal{Z}^{k-1})$ is the location prior, and $p(Z_k|\mathcal{Z}^{k-1})$ is the normalizing term.

The observation likelihood $p(Z_k|L_i, \mathcal{Z}^{k-1})$ is evaluated assuming an independence between the current and the past observations conditioned on the location. It is composed of two parts - a simple model for each location based on the existence variable e_i and a complex model which embeds the relations between the visual words . The simple location model is computed in an online manner using the detector model whereas the more difficult feature inter-dependence is captured using a Chow-Liu tree (which is computed in an offline training step). The prior belief over the location $p(L_i|Z_{k-1})$ is computed by transforming the previous position estimate through a appropriate motion model. While the normalizing term $p(Z_k|Z_{k-1})$ is computed in a particular manner that facilitates finding out if present image is a scene that has not been visited before. The complete derivations of these terms can be found in [106].

4.2.2 Overlap Detection Scheme

In this section, we present a scheme to detect overlap amongst multiple robots as shown in Figure(4.1). The scheme consists of the following main components:

1. *Input:* The input to the system are the images captured by the cameras on the robots. Since the robots move independently, they may visit the same places in the

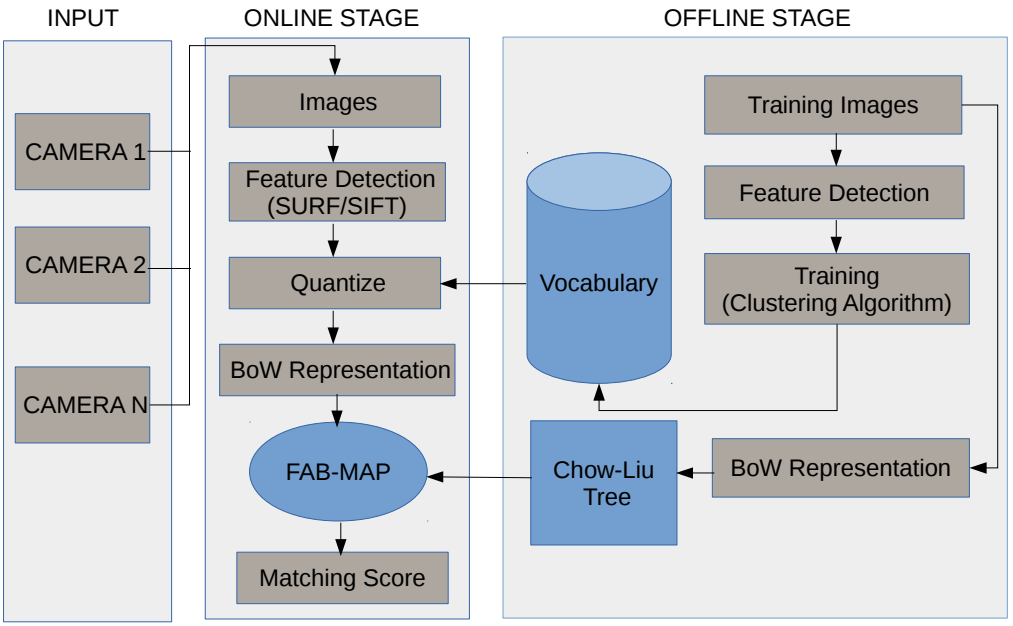


FIGURE 4.1: Overlap detection scheme using an appearance-based matching

environment at different times. This constraint must be considered while detecting overlaps.

2. *Offline Stage:* Appearance-based system requires a vocabulary of visual words and Chow-Liu tree to perform matching. For this, we need to

   (a) Collect Training Images: Large number of images (> 1000) from varied scenes have to be collected. Ideally, the training set should cover as many different types of objects as possible. Multiple images of the same place should not be taken as it can bias the inference process.

   (b) Build Vocabulary: To build a vocabulary, visual features such as SURF/SIFT are computed for the whole training set. Then a clustering algorithm (ex. K-Means) is run to train the vocabulary.

   (c) Build Chow-Liu Tree: A BoW description is extracted from the training images. Using the vocabulary, a Chow-Liu tree is built as described in [126]. We use the *OpenCV* implementation of the Chow-Liu tree for our application.

3. *Online Stage:* It involves the following steps:

(a) BoW representation of input data: For each incoming image, visual features like SURF/SIFT are computed. Then they are quantized with respect to the vocabulary to obtain the BoW description.The BoW representation corresponding to each image is stored in a list along with the corresponding camera ID. This would later allow computing matches between places which were visited by robots at different times.

(b) Comparison: Whenever a new image from any of the robots is received by the system, it compares the BoW representation of the present image with the entire list of other image BoW's using the FAB-MAP technique. This yields a matching score of the present image versus all other images. Finally, a positive match is declared, if the matching score is greater than a pre-defined threshold.

(c) Detection: Matches between each robot pair is monitored. An overlap is declared when two or three consecutive images show positive matches; This is done in order to avoid spurious detections.

4.3 Results

In this section, we present some results from the tests conducted to evaluate the performance of the overlap detection system.

4.3.1 Experimental Setup

The experiments were performed using images taken from a uEye camera fitted with a wide-angle lens (130 degrees FOV). The camera is mounted in top of a hand-pushed cart. It is the same setup which was used to test visual SLAM process. The experiments were conducted in an industrial like indoor environment. The environment consists of several rooms with different types of objects and slightly varying lighting conditions.

The place recognition algorithm requires a vocabulary of visual words and a Chow-Liu tree to perform matching. For this, we use a publicly available dataset having images from a mixed indoor/outdoor environment. This dataset is made available by QUT CyPhy Robotics Lab [127]. In addition the system requires certain parameters to be set which are listed in Table 4.1.

Parameter	Details
Feature detector	Type = SURF Hessian = 1000 Num of Octaves = 4 Num of octave layers = 2 Orientation = Rotated Description = 64D
Feature descriptor	SURF
Number of features per image	Min = 300 Max = 500
Detector Model	$p(z_i = 1\|e_i = 0) = 0.39$ $p(z_i = 0\|e_i = 1) = 0.00$
Other FAB-MAP options	New Place Model = *SAMPLED* Feature Dependency Model = *CHOW-LIU*

TABLE 4.1: Parameters for Appearance-Based Matching Algorithm

4.3.2 Place Recognition Tests

4.3.2.1 A simple scenario

For this experiment, a dataset of 100 images was collected by moving the cart through the environment. Later 5 randomly chosen places from this dataset were revisited. At these five places, new images were captured from slightly different perspectives. The appearance-based algorithm was run over the complete set of 105 images. The images which were visited two times are shown in Figure 4.2. The algorithm was able to match 4 out of 5 places correctly while the 5^{th} image was predicted as a new place. For these five places, the probability of matching with their corresponding ground truth image and probability of it being a new place is given in Table 4.2.

Image Num.	Prob. of matching wrt *ground truth*	Prob. of new place
1	0.991	0.001
2	0.085	0.910
3	0.922	0.002
4	0.991	0.001
5	0.911	0.131

TABLE 4.2: Simple Scenario: Matching probabilities of revisited places. For this case, the threshold for considering a match positive was set at 0.90.

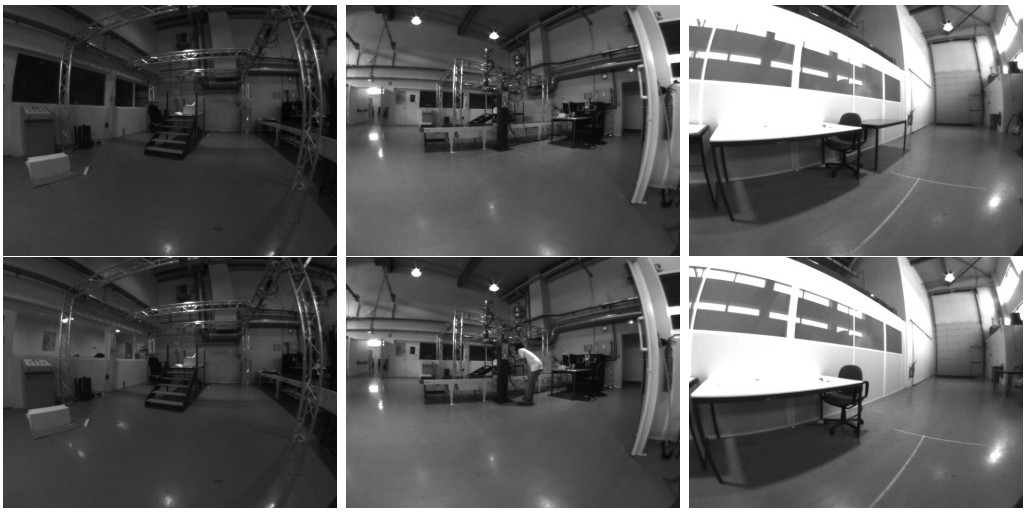
FIGURE 4.2: Places visited two times from slightly different perspectives.The top row shows the images of the places taken during the first trip. The bottom row shows the images of the same place but taken from a slightly different perspective. Four out of the five pairs were correctly matched by the algorithm.

4.3.2.2 With slight changes in scene appearance

Figure 4.3 shows three pairs of images taken at the same location however with some changes in the appearance of the scene. In image pair (a), the lighting conditions change slightly. In (b), a person appears in the second image who was not present before. Whereas in (c), some of the objects present in the first image are no more visible in second image. In all these cases, despite the changes in the appearance of the scene, the algorithm was able to find the matches correctly. The fact that the system was able to make these detections means that the information due to the missing features was considered indirectly during the matching process. This is probably due to the correlation of the missing features with the ones visible in the images which is modeled via the Chow-Liu tree.

4.3.2.3 Rejection of similarly looking images

Figure 4.4 illustrates a challenging case where the images appear to be very similar but do not correspond to the same place. Most features extracted from the corresponding pairs of images appear to be the same. In spite of this, the algorithm does not give a high matching probability to them. This is probably because the system has learnt that such kind of images are very common in the environment. Therefore their matching does not necessarily indicate that the images correspond to the same place. However, it must be noted that in the cases where the images actually do come from the same place, they would not be receiving high matching scores.



case (a) case (b) case (c)

FIGURE 4.3: Same location with slight difference in appearance. The top row shows three places which were visited during the first trip. The bottom row shows the same places visited again however with a slight change in the appearance of the scene. In (a) the illumination varies between the two visits. In (b) and (c), some artifacts are added or missing from the scene. In each of these cases, the algorithm was able to identify the correct matches.

4.3.2.4 Failure cases and their handling

Figure 4.5 illustrates two failure cases. In case (a), the two images coming from different places get high matching probability $p = 0.98$. This is possible due of the fact that training set contains no features of the objects present in the images. Therefore, due to the lack of feature correlation information, the images get incorrectly matched. A simple way to handle these kinds of false detections is to set a very high matching threshold $p > 0.99$. Though this may lead to the rejection of many of the true matches, for our application it is better to be confident about the match even at cost of losing some other close matches.

Instead in case (b), the images come from the same scene but do not receive a good matching probability. Again, this would not pose a big problem in our application since a match needs to be found amongst many images taken by the robots in a common area visited by them. It is very likely that some of these other images will result in positive matches.

case (a) case (b) case (c)

FIGURE 4.4: Similar looking images that do not originate from the same place. Each pair of images have very high resemblance but were taken from different parts of the environment. However, the algorithm does not give them very high matching probabilities. This is probably because the algorithm takes into consideration that such scenes are very common in the environment and may not really be coming from the same place despite high visual resemblance.

4.3.3 Discussion

In general, reasonable performance was recorded for the dataset of test images. However, these results was obtained after a lot of testing with different parameter settings. Our experiments on different datasets recorded showed varying performance ranging from acceptable to very poor in some cases. We suspect that following are the main sources of sub-optimal performance:

1. *Non-uniform feature distribution:* It was observed that the features extracted in many of the images were concentrated in one region of the image. Such a distribution gives an incorrect (biased) representation of the scene resulting in the computation of bad matches. The main reasons for bad feature distribution were (a) Different lighting conditions in different parts of the same scene (b) Blur in some parts of the image (c) Large texture-less portions in the scene. The feature distribution could be improved to some extent by using an adaptive feature detection technique. However, this comes at the cost of an increase in the computation time.

2. *Varying illumination conditions:* We observed that as the camera moves from a bright place to a darker place, the auto-exposure functionality of the camera

case (a) case (b)

FIGURE 4.5: Failure cases: False positive and False negative examples. In case(a), the two images get a high matching probability in spite of coming from different places in the environment. In case(b), the two images correspond to the same place but does not receive a good matching score.

does not adapt very well. This often results in bad quality images which then leads to bad feature detection. To ensure a consistent quality in image feed, a robust exposure control needs to be implemented at the hardware level. Some improvements can also be obtained at the software level by processing the images before the feature detection step.

3. *Visual Vocabulary:* Although the system is designed to work well with any kind of vocabulary, we suspect that the pre-built vocabulary we are using is a source of reduced performance. Generally speaking, more the number of words in the vocabulary, greater is the specificity obtained by the algorithm. The number of words in the vocabulary should be chosen based on the size and variance in the appearance of the environment. A vocabulary generated specifically for our working environment could improve the overall matching performance.

Chapter 5

3D Similarity Transform Estimation

5.1 Introduction

When a scene overlap is detected by the place recognition system, we need to compute the relative transformation between the two views. This would later be required to merge the local maps generated by the individual robots into a common global map. The relation between two views can be described as a 3D similarity transformation T :

$$T = \begin{pmatrix} sR & t \\ 0 & 1 \end{pmatrix} \quad \text{with } R \in SO(3),\ t \in \mathbb{R}^3 \text{ and } s \in \mathbb{R}^+ \tag{5.1}$$

For our application, we must remember that the transformation must be computed only using the image and corresponding depth map. We will see two types of techniques for estimating the transformation based on the kind of image information used. The first type where image features from the two views are used. The second type where the complete image and depth information from the two views is used. These type of methods using whole image information are referred to as *dense methods*.

In this chapter, we discuss three main techniques for 3D similarity transform estimation. The first technique is based on a closed-form solution proposed by Horn [128]. It uses feature correspondences from the two views to compute the transformation. A robust

RANSAC version of the algorithm is presented which performs well even in the presence of bad feature correspondences.The second technique uses an optimization method based on direct image alignment. It uses the complete image information and can be considered as a dense method. Finally, the third technique comes from the pointcloud registration literature. Here we discuss the *Iterative Closest Point (ICP)* algorithm. In particular, we focus on a variant of the ICP algorithm which includes the surface normal and tangent information.

Each of these techniques can be used individually to estimate the transformation. They can be used in a standalone manner with reasonable accuracy. However, these techniques can also be put together in a chain to have higher precision. For this purpose, we propose a three-stage procedure to obtain the final transformation. In the first stage, an initial estimate is given by the Horn's algorithm. This estimate is then improved by the optimization method based on direct image tracking. Finally, in the third stage, the previous estimate is further refined using a pointcloud registration technique.

5.1.1 Contributions

This chapter describes how to compute the 3D similarity transformation between two views with a visual overlap. The contributions made in this chapter are as following:

- A three-stage procedure for accurate computation of similarity transformation between two scenes with an overlapping view.
- Implementation of a robust version of the Horn's closed-form solution using RANSAC.
- Adaptation of existing direct image alignment and ICP based techniques for our system.
- Evaluation of these techniques on both synthetic and real datasets.

5.1.2 Related Work

Estimating transformation between two coordinate systems using corresponding measurements of the points in the two systems have been dealt extensively in the photogrammetry literature. Numerous solutions based on empirical, graphical and numerical iterative methods have been presented in [129], [130]. Schut [131] proposes a technique

based on unit quaternions which computes the transformation as a solution to system of linear equations. A solution based on least squares formulation has been described by Oswal H.L. [132]. However, the final solution is computed numerically in an iterative manner. In this chapter, we focus mainly on the technique proposed by Horn [128]. He derives a closed-form solution to the least squares formulation in the case where exact correspondence between points in two systems are known.

Another set of techniques for computing the similarity transform are based on direct image alignment. These are dense methods as complete image information is used. Here the solution is obtained by solving an optimization problem which minimizes the overall photometric (and depth) error in the two views. These techniques have been well established for RGB-D and stereo sensors [133], [72]. More recently, these techniques have been extended to the monocular camera case. These techniques include an extra step where the depth of the scene must be computed first. Newcombe et al. [68], Pizzoli et al. [134] and Stühmer et al. [135] propose various algorithms based on a variational formulation for the computation of depth maps. These depth maps are then used for tracking the camera pose between consecutive frames. These techniques form the basis for a monocular direct visual odometry (VO) scheme. However, the steps involved are very computationally demanding and require a powerful GPU for real-time performance. To deal with this complexity, Forster et al. [102] propose an algorithm which combines direct tracking with keypoints able to work at high-frame rate even on embedded processors. In our work, we adopt the *semi-dense* approach proposed by Engel et al. [104] to deal with the high number of computations involved. In this approach only those parts of the image which have high intensity gradient are used for estimating the similarity transformation. This technique facilitates real-time performance on a standard CPU processor.

The third set of techniques we consider in this chapter come from the pointcloud registration literature known as the Iterative Closest Point (ICP) algorithms. The fundamental idea here is to compute a transformation which minimizes the distance of a set of corresponding points in the two clouds. The original algorithm was proposed by Besl and McKay [136] in 1992. Since then several variants of the ICP have been proposed which address the shortcomings of the basic ICP algorithm. Magnusson et al. [137] propose a variant called as *Normal Distribution Transform* (NDT) where local surface information is added using Gaussians in the neighbourhood of a point. Another variant was

proposed by Segal et al. [138] called *Generalized ICP* (GICP) where the surface information around a point is described using planar patches. This information is then used to improve the transformation estimate during the optimization process. In our work, we use the variant proposed by Serafin and Grisetti [139] where the surface normal and surface tangent information is used in addition to the standard distance metric.

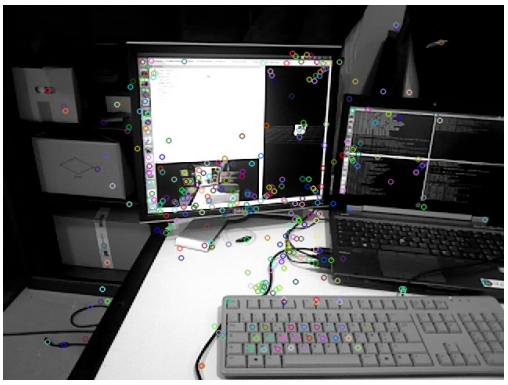
5.2 Horn's Method (Closed Form Solution)

Horn [128] presents a closed form solution to recover the 3D similarity transformation (also known as the absolute orientation in the photogrammetry literature). It computes the transformation between two coordinate systems using pairs of corresponding points measured in their local reference frames. This solution is the optimal solution in the least-squares sense given three or more point correspondences.

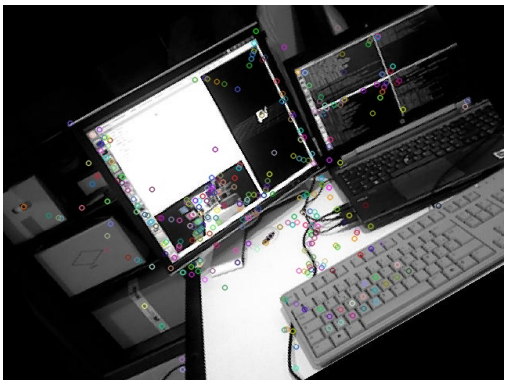
The transformation T is computed by finding the rotation R, translation t and the scale s individually. The rotation estimate R is computed as a unit quaternion which is the eigen vector corresponding to the most positive eigen value of a 4×4 symmetric matrix. This symmetric matrix is composed of elements which are combinations of different sums of products of corresponding coordinates of the points. The translation estimate t is the difference between the centroid of the coordinates of one system and the rotated-scaled centroid coordinates in the other system. The scale estimate is the root mean square of the deviation of the coordinates in the two systems from their respective centroids. The exact derivations of these results are given in [128]. In this section, we outline the overall algorithm for the computation of the 3D similarity transformation.

5.2.1 Input Data

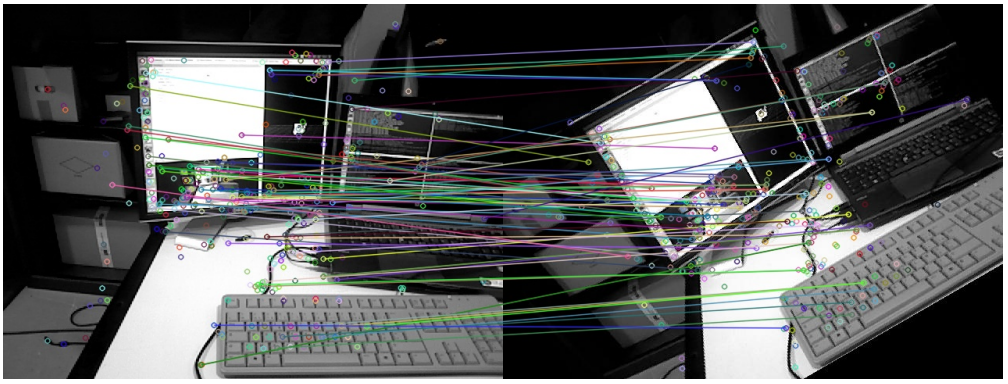
For our application, we need to compute the transformation from visual information (image and corresponding depth information) only. This transformation needs to be computed when a scene overlap is detected among two camera views. At this moment, we only have available with us the image and the corresponding depth map from the two views. In order to use the Horn's algorithm, we must have the corresponding 3D points from the two views. To get these 3D point correspondences, we perform the following steps:



(A) SURF features in the first image



(B) SURF features in the second image



(C) Feature matches computed using Flann matcher

FIGURE 5.1: Image features and matches between the two views. Images (A) & (B) show the SURF features extracted from both the images. (C) shows the feature matches between the two views computed using a Flann Matcher.

1. Compute features (for example SURF) in the images from both the views (Figure 5.1a and Figure 5.1b).

2. Compute feature matches between two images. These matches can be computed in a brute force manner or using Flann based Matcher (Figure 5.1c). These matches provide the correspondences between points from the two views.

3. Compute the 3D points corresponding to 2D features previously computed in the two images. This is done by taking the depth at the 2D feature locations from the corresponding depth map.

5.2.2 Horn's Algorithm: Basic Version

The input to the Horn's algorithm is a set of n 3D point correspondences from the two coordinate systems. Starting from this input, the steps to compute R, t and s is summarized in Algorithm 3.

We first find the centroids C_1 & C_2 from the inputs X_1 & X_2 in their respective coordinate frames. Then we find the relative coordinates for each set by subtracting coordinates of each point from their respective centroids [Step(1)]. In order to compute the rotation R, we need to compute two important matrices M & N [Step(2) and Step(3)]. The elements of these matrices are obtained by simple combinations of the products of the point coordinates from the two systems. All the information neccessary to compute R is present in the matrix N. Once N is computed, the rotational estimate R is obtained as a unit quaternion by taking the eigen vector corresponding to the most positive eigen value of N [Step(4)]. This unit quaternion can then be converted into a 3×3 orthonormal matrix using the formula given in [128]. Next, we compute the scale s as described in Step(5). Once R and s are known the translation t is computed as the difference between centroid C_2 and scale-rotated C_1 [Step(6)]. Finally, we compose R, t, s as shown in Step(7) to give the 4×4 similarity transformation matrix T.

Algorithm 3 Horn's Algorithm For Computation of Similarity Transformation

Input: X_1, X_2, n

Output: R, t, s

1: Compute Centroid and Relative Coordinates:

$C_1 = \frac{1}{2}\sum_{i=1}^{n} X_{1,i}, C_2 = \frac{1}{2}\sum_{i=1}^{n} X_{2,i}$

$\hat{X}_1 = X_{1,i} - C_1, \hat{X}_2 = X_{2,i} - C_2$

2: Compute M:

$M = \sum_{i=1}^{n} \hat{X}_{1,i}\hat{X}'_{2,i}$

$$M = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix}$$

Where $S_{xx} = \sum_{i=1}^{n} \hat{x}_{1,i}\hat{x}_{2,i}, \quad S_{xy} = \sum_{i=1}^{n} \hat{x}_{1,i}\hat{y}_{2,i}$ as so on.

3: Compute N:

$$N = \begin{bmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & -S_{xx}+S_{yy}-S_{zz} & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & -S_{xx}-S_{yy}+S_{zz} \end{bmatrix}$$

4: Rotation Estimate:

$q = \text{argmax} \quad \dot{q}N\dot{q}$

$= \text{EigenVec}\ [max\ (\text{EigenVal}(N))]$

$R = \text{UnitQuaternionToRotMat}(q)$

5: Scale Estimate:

$$s = \left(\frac{\sum_{i=1}^{n} \left\| \hat{X}_{2,i} \right\|^2}{\sum_{i=1}^{n} \left\| \hat{X}_{1,i} \right\|^2} \right)^{\frac{1}{2}}$$

6: Translation Estimate:

$t = C_2 - sRC_1$

7: Compute Transformation Matrix:

$$T = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} \frac{1}{s}R' & -\frac{1}{s}R't \\ 0 & 1 \end{bmatrix}$$

8: **return** R,t,s

5.2.3 Horn's Algorithm: Robust Version

The Horn's algorithm gives the optimal transformation provided that the correspondences between points is accurate. However, if there are some bad correspondences (outliers), the estimate becomes biased very easily.

As we see in Figure 5.1, we have many bad correspondences while matching features between two images. To deal with this problem, we need a robust mechanism to estimate the transformation in the presence of outliers. In this section, we present a RANSAC based Horn's algorithm which can work well even with 30-40% bad correspondences.

A summary of the RANSAC based Horn's algorithm is given in Algorithm 4. The input consists of n 3D point correspondences from two coordinate systems. However, many of these correspondences could be bad. The algorithm uses an iterative method to give as output a best estimate of the rotation R_{best}, translation t_{best} and scale s_{best}. Initially, we set the maximum number of iterations either manually or based on the desired accuracy. For each iteration, we randomly select three point correspondences and compute the current estimate R, t, s using the Horn's method as describe earlier in Algorithm 3. The current estimate R, t, s form a hypothesis over the observations i.e. the point correspondences. We check how good the hypothesis is by computing the error over the entire data. At each iteration, we compare the error obtained with the best error so far. If the current error is less than this best error, the current hypothesis R, t, s is set as the best hypothesis $R_{best}, t_{best}, s_{best}$. When maximum iterations is reached, the best estimate is obtained. Optionally, a refined final estimate can be obtained by using all the inliers to compute the final transformation T.

Algorithm 4 RANSAC Version of Horn's Algorithm

Input: X_1, X_2, n

Output: $R_{best}, t_{best}, s_{best}$

Initialization: MaxIterations = $min(1000, log(1-p)/log(1-\epsilon^n))$ where p = prob. of a correct result , $\epsilon = min_{inliers}/n$, $err_{best} = \infty$

for i = 1 : MaxIterations **do**
  // Choose 3 Point Correspondences Randomly
  $rand\_idx = rand(1, n, 3)$
  $X_{1r} = X_1(rand\_idx)$
  $X_{2r} = X_2(rand\_idx)$
  // Compute R, t, s using Horn's Algorithm
  $[R, t, s]$ = Horn_Algorithm(X_{1r}, X_{2r})
  // Compute Error:
  err = ComputeError$(X_1, X_2, [R, t, s])$
  $Inliers$ = findInliers$(X_1, X_2, [R, t, s])$
  // Update Model Parameters:
  if $err < err_{best}$ **then**
    $R_{best} = R$, $t_{best} = t$, $s_{best} = s$
  end if
end for
// Refine Estimate Using All Inlier Data:
$X_1^{in} = X_1(Inliers)$
$X_2^{in} = X_2(Inliers)$
$[R_{best}, t_{best}, s_{best}]$ = Horn_Algorithm(X_1^{in}, X_2^{in})
// Compute Transformation Matrix:

$$T = \begin{bmatrix} s_{best}R_{best} & t_{best} \\ 0 & 1 \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} \frac{1}{s_{best}}R'_{best} & -\frac{1}{s_{best}}R'_{best}t \\ 0 & 1 \end{bmatrix}$$

return R_{best}, t_{best}, s_{best}

5.3 Direct Image Alignment Method

In this section, we study a technique for estimating the similarity transformation using the direct image alignment concept. The basic idea is to find the transformation which reduces a combined photometric and depth error. The error is computed as the difference between the image intensities and depth maps from the two views. Instead of computing features from the images (which was the case in the previous technique using Horn's algorithm), this technique uses the whole image information to compare the two views. Therefore, it can be referred to as a direct method.

An iterative solution to the direct image alignment problem is presented. At each step, an improved transformation estimate is computed starting from the estimate of the previous step. Finally, the goal is to find the transformation with the minimum possible error. While computing the error, we use only those parts of the image which have image intensity gradients. This helps in keeping the computations tractable while including the most informative parts of the image. This technique can therefore be classified as a semi-dense approach. (As opposed to a dense approach where the complete image information is used).

5.3.1 Problem Formulation

The direct image alignment technique poses the estimation of the similarity transformation $T \in sim(3)$ as an optimization problem. The cost function is defined as a combination of photometric and depth error. The solution to this optimization problem is the transformation which minimizes this combined error.

The input data available are the images (I_1, I_2) and depth maps (D_1, D_2) from the two views. From the given depth map, the 3D coordinates(X_1, X_2) corresponding to each pixel in the image can be computed using the camera calibration parameters. Given this data, the aim is to compute the transformation which aligns them in the best possible manner. The total error in alignment is computed as the sum of the photometric and depth error .

The photometric error (e_p) is defined as:

$$e_p(p, T) = I_1(p) - I_2(\omega(p, D_1(p), T)) \tag{5.2}$$

where

- $p :=$ location of the pixel (u, v) in the image frame.
- $T :=$ Current transformation estimate between the two views.
- $\omega :=$ warping function which computes the location of a pixel from the first image in the second image given the relative transformation between the two views.

The depth error (e_p) is defined as:

$$e_d(p, T) = X_{2,z} - D_2(\pi(X_2^{'})) \tag{5.3}$$

where

- $X_2^{'} :=$ Transformed 3D point obtained by applying T to X_1, i.e. $X_2^{'} = \left(X_{2,x}^{'} X_{2,y}^{'} X_{2,z}^{'}\right) = T.X_1$
- $\pi :=$ camera projection function which computes the pixel location (u, v) given the 3D point X using the camera calibration parameters.

Using these two errors, i.e. the photometric error (e_p) and the depth error (e_d), the overall cost function is defined as:

$$E(T) = \sum_{p \in I_S} \left\| e_p^2(p, T) + e_d^2(p, T) \right\| \tag{5.4}$$

Note that the summation of the error is taken only on a subset of image pixels I_S, namely those which have high image intensity gradient. The solution to the direct image alignment problem is given by the transformation T^*:

$$T^* = \underset{T}{\operatorname{argmin}}\, E(T) \tag{5.5}$$

5.3.2 Optimization Procedure

The cost function in equation(5.4) can be rewritten as:

$$E(T) = \sum e_t^T \Omega(T) e_t \tag{5.6}$$

where

- $e_t(T) :=$ Total error consisting of both photometric and depth error.
- $\Omega_t(T) :=$ Information matrix consisting of the weights which decide the importance of the error term.

The objective function in equation(5.6) is minimized using a least squares procedure. The optimization procedure aims to find the transformation T^* that minimizes the objective function. T^* can be computed using a Gauss-Newton minimization procedure. The overall procedure is summarized in Algorithm 5.

Algorithm 5 Gauss Newton Solution To Direct Image Alignment Problem

Input: I_1, I_2, D_1, D_2
Output: $T^* \in sim(3)$
Initialization: $T^{(0)}$
while $\neg$ converged **do**
  // Compute photometric error e_p
  $e_p(p,T) = I_1(p) - I_2(\omega(p, D_1(p), T))$
  // Compute depth error e_d
  $e_d(p,T) = X_{2,z} - D_2(\pi(X_2^{'}))$
  // Compute the combined error e_t
  $e_t \leftarrow Combine(e_d, e_p)$
  // Compute Jacobian of the error function
  $J = \delta e_t(T^{(n)} + \Delta T)$
  // Compute $\Delta T^{(n)}$
  $\Delta T^{(n)} = (-J^T J)^{-1} J^T e_t$
  // Update Transformation estimate
  $T^{(n+1)} = T^{(n)} + \Delta T^{(n)}$
end while
$T^* \leftarrow T$
return T^*

5.4 Iterative Closest Point (ICP) Methods

In this section, we see a technique from the pointcloud registration literature called as the Iterative Closest Point (ICP) algorithm. ICP tries to find a transformation by minimizing the distance between the corresponding points in the two clouds. At each iteration, ICP performs a search and optimization step alternatively. In the search step, starting from the current transformation, corresponding points in the two pointclouds are found using a heuristic such as a nearest neighborhood search. Whereas in the

optimization step, the transformation is computed which minimizes the distance between the corresponding points found during the search step.

The basic idea behind ICP is that at each iteration, a better transformation estimate is computed. The search heuristic uses this new transformation estimate as an initial guess to find correspondences in the two point clouds. As a result, better associations can be determined in the next iteration.

The optimization and the correspondence search step are dependent on each other. An optimization procedure which is smooth and robust to outliers, has better chances of finding an improved solution at the next step. Similarly, a better starting point for the search procedure will give better correspondences at the next step. In this manner, an improved solution is obtained at each iteration until it converges to a good solution.

One of the challenges in ICP is to find a heuristic that provides *good* correspondences. In the original implementation, the idea was to pick up the closest points. This idea has been extended to consider features, curvature and other characteristics of a point. However, the heuristic used must be efficient and fast to compute. Since the optimization procedure is only linear in the number of correspondences, the major computational bottleneck is due to the heuristic to find these correspondences.

Several variants of the ICP have been proposed to address the lacunae in the original solution. For example, in the original formulation ICP assumes that the points on the surfaces captured from two views are exactly the same. However, this is not true since the points are generated by sampling of the surface as observed by the sensor from the two views. Their chance of being exactly the same points is very low. To address this issue, Magnusson et al. [137] propose a method called *Normal Distribution Transform* (NDT) where the surface is approximated with a set of Gaussians to capture the local statistics of the surface in the neighborhood of a point. During the correspondence search step, it uses the Mahalanobis distance instead of the Euclidean distance as the similarity metric. Similarly, another solution was proposed by Segal et al. [138] called *Generalized ICP* (GICP). The idea in this approach was to consider the shape of the surface around a point by approximating it with a planar patch. Later during optimization, the corresponding patches from the two point clouds are aligned with each other. It has been shown that both these methods show better convergence than the original ICP formulation.

In our work, we consider another variant of ICP proposed by Serafin and Grisetti [139]. Here, in addition to the relative point distances, the differences between surface normals and tangents are also considered.

5.4.1 ICP

ICP aims to find the transformation which maximizes the overlap between two point-clouds. Given two sets of points X_1 & X_2, we want to find the transformation T^* that minimizes the distance between the corresponding points in the two clouds:

$$T^* = \underset{T}{\operatorname{argmin}} \sum_{C} (X_{1,i} - T.X_{2,j})^T \Omega_{ij} (X_{1,i} - T.X_{2,j}) \tag{5.7}$$

where

- T := Transformation estimate that is updated at each iteration using the one found at the previous step $k-1$.
- $C :=< i, j >_{1:n}$ Set of correspondences in the two point clouds.
- Ω_{ij} := Information matrix to take into account the noise properties of the sensor and the confidence about the correspondences.

At the beginning, the correspondences between the two point clouds are not known. However, if a reasonable initial transformation is known, the correspondences can be found using a heuristic like the nearest neighborhood search. The ICP iteratively improves the initial transform T by first searching for correspondences and then finding a solution to equation(5.7). The estimated solution is then used as a new starting point for finding the correspondences. The information matrix Ω_{ij} is a diagonal matrix which reflects the confidence in the correspondence between $X_{1,i}$ & $X_{2,j}$.

Note that with an increase in the dimensionality of points, the whole system becomes more observable. As a result, fewer correspondences will be required to compute the transformation. Keeping this in mind, an extended definition of the point including the surface normal and tangent is proposed by Serafin and Grisetti [139].

5.4.2 Extended ICP

In this section, an extended version of ICP taking into account the surface normals and tangent is presented. For each point X_i, a normal N_i is defined. Similarly, a tangent T_i is defined if the point X_i belongs to an edge. Now T^* is defined as the transformation which minimizes an extended cost function:

$$\begin{aligned} T^* = \underset{T}{\operatorname{argmin}} & \sum_{C} (X_{1,i} - T.X_{2,j})^T \Omega_{ij} (X_{1,i} - T.X_{2,j}) \\ & + \sum_{C} (N_{1,i} - R.N_{2,j})^T \Omega_{ij}^N (N_{1,i} - R.N_{2,j}) \\ & + \sum_{C} (\mathrm{T}_{1,i} - R.\mathrm{T}_{2,j})^T \Omega_{ij}^{\mathrm{T}} (X_{1,i} - T.\mathrm{T}_{2,j}) \end{aligned} \tag{5.8}$$

where

- $N_{1,i}$ & $N_{2,j}$ are normals of the points $X_{1,i}$ & $X_{2,j}$ respectively.
- $\mathrm{T}_{1,i}$ & $\mathrm{T}_{2,j}$ are tangents of the points $X_{1,i}$ & $X_{2,j}$ respectively.
- Ω_{ij}^N and Ω_{ij}^{T} are the corresponding information matrices for normals and tangents respectively.

Note that the normal and tangent terms are directional vectors. Therefore they are multiplied only by the rotation matrix R instead of the whole transformation matrix T .

5.4.3 Optimization Procedure

The objective function in equation(5.8) is minimized using a least squares procedure. The input to the algorithm are the two sets of points X_1 and X_2, the correspondences $C =< i, j >_{1:n}$ and the information matrices Ω_{ij}. The computation of the information matrix is given in [139]. The optimization procedure aims to find the transformation T^* that minimizes the objective function:

$$T^* = \underset{T}{\operatorname{argmin}} \sum_{C} (\tilde{X}_{1,i} - T.\tilde{X}_{2,j})^T \tilde{\Omega}_{ij} (\tilde{X}_{1,i} - T.\tilde{X}_{2,j}) \tag{5.9}$$

$$= \underset{T}{\operatorname{argmin}} \sum_{C} (\tilde{e}_{ij})^T \tilde{\Omega}_{ij} (\tilde{e}_{ij}) \tag{5.10}$$

which is written by aggregating the terms from equation(5.8). T^* can be computed using a Gauss-Newton minimization procedure. The solution is obtained by iteratively solving the linear system:

$$H\Delta T = -b \tag{5.11}$$

where the Hessian matrix $H = \sum_C J_{ij}^T \tilde{\Omega}_{ij} J_{ij}$, the coefficient vector $b = \sum_C J_{ij}^T \tilde{\Omega}_{ij} \tilde{e_{ij}}$ and J_{ij} is the Jacobian of the error function. By solving equation(5.11), the perturbation ΔT is computed. This ΔT is applied to the previous T to obtain the improved estimate. The overall procedure is summarized in Algorithm 6.

Algorithm 6 Estimation Of Transformation Using ICP

Input: X_1, X_2, C

Output: T

Initialization: $T^{(0)}$

while ¬ Converged **do**

  // Compute Information Matrix $\tilde{\Omega}_{ij}$.

  // Compute Error Vector $\tilde{e_{ij}}$:

  $\tilde{e_{ij}} = \tilde{X}_{1,i} - T.\tilde{X}_{2,j}$

  // Compute Jacobian J_{ij}:

  $J_{ij} = \dfrac{\delta e_{ij}(T + \Delta T^{(k-1)})}{\delta \Delta T}$

  // Compute Hessian H and Coeff. Vector b:

  $H = \sum_C J_{ij}^T \tilde{\Omega}_{ij} J_{ij}$, $b = \sum_C J_{ij}^T \tilde{\Omega}_{ij} \tilde{e}_{ij}$

  // Solve System Of Linear Eqs To Find $\Delta T^{(k)}$

  $\Delta T^{(k)} \leftarrow \text{Solve}(H\Delta T = -b)$

  // Update Current Transformation Estimate:

  $T^{(k)} = T^{(k-1)} + \Delta T^{(k)}$

end while

$T^* \leftarrow T$

return T^*

5.5 Results

5.5.1 Experimental Setup

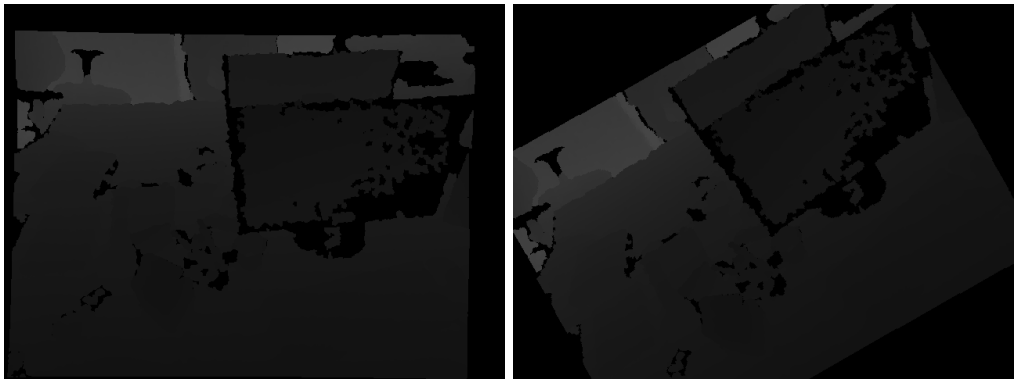
We test the performance of the similarity transform estimation algorithms using data captured from an *ASUS Xtion Pro* sensor. It provides a feed of images (RGB) along with dense depth information. We have intentionally chosen to use a depth sensor for testing transformation estimation module. This would allow us to assess the performance independently of the quality of depth map computed by the visual SLAM algorithm. In the following experiments, we show results obtained from two datasets. The first one is a benchmark dataset for RGB-D images provided by TUM Vision Lab [140]. The second dataset is recorded in our lab environment using a Asus Xtion depth sensor.

FIGURE 5.2: Depth Sensor: Asus Xtion

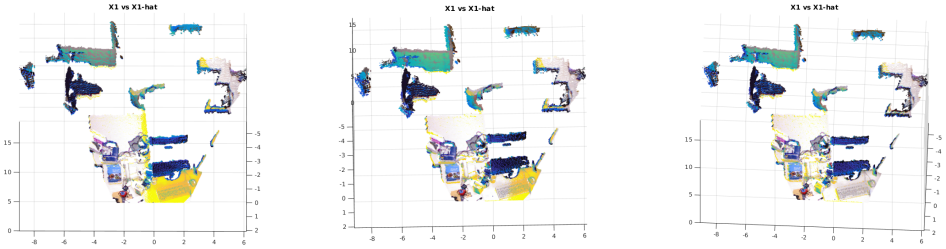
5.5.2 Tests: Synthetic and Real Datasets

To illustrate the performance of our estimation algorithms, we show results from three experiments. The first two experiments are performed on images from the TUM benchmark dataset. We synthetically modify these images with known transformations in order to understand the precision of the techniques. For the first test, we rotate the original image by 30 degrees to obtain the new image. Whereas in the second image, we translate the image by 50 pixels in both x and y directions. The aim here was to assess the performance for both rotational and translational changes in the two views. Finally, in the third experiment, we capture a pair of measurements with free movement of the camera. This illustrate how the algorithms perform in the presence of mixed movements having both rotational and translational components. The results from the three experiments are shown in Figures 5.3, 5.4 and 5.5.

(A) First Image (B) Second Image



(C) Depth map for First Image (D) Depth map for Second Image



(E) Horn's method (F) Direct image tracking (G) ICP technique

FIGURE 5.3: Images(A) and (B) show the images for which the similarity transformation T has to be computed. The image is taken from the TUM RGB-D Benchmark dataset. The second image is synthetically obtained by rotating the first image by 30 degrees. (C), (D) show the corresponding depth maps of the two images. Finally, (E),(F) and (G) show the two depth maps after applying the transformation computed using Horn's method, Direct image tracking method and ICP respectively

5.5.3 Discussion

In general, satisfactory results were obtained for majority of the image pairs from both the datasets. During the experimentation process, we encountered some failure cases as well. We observed the main failure cases for the three techniques as the following:

1. Horn's Method: In the cases where the feature detection and matching performance went down (due to blurring, texture-less regions etc.), the accuracy of this

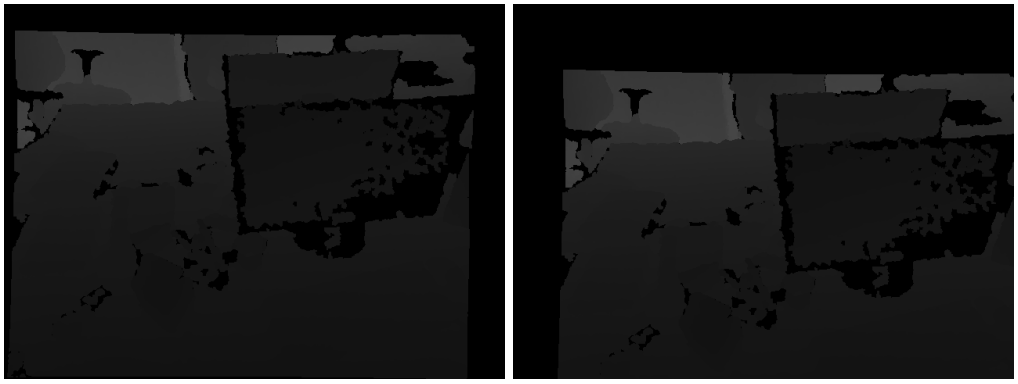
technique also went down. In some severe cases with more than 60% bad correspondences, the estimate computed is incorrect.

2. Direct image tracking: It was found that the initial estimate T^0 plays an important role in the convergence of the algorithm. In general, given a reasonable (upto 20 degrees error in rotation) this technique was found to be very robust and accurate. It also showed good performance in the presence of a scale change (upto the order of two to three times).

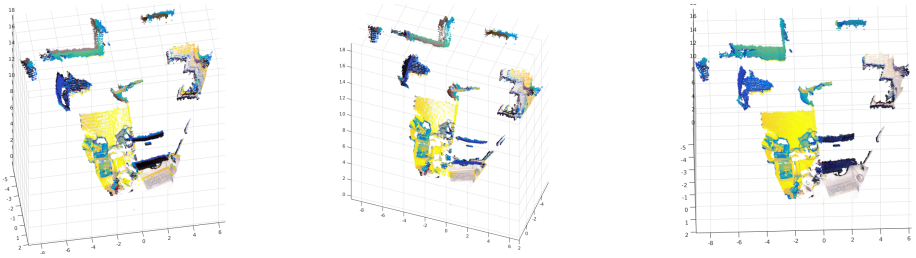
3. ICP technique: This method was found to be the most accurate among the three techniques. However, it also diverged for some cases if the initial estimate was bad (particularly the scale factor) .

However, it is important to note that many of the failure cases were avoided when we used the three-stage procedure to compute the transformation. The downside being that the overall computation time becomes high. For our application, this should not pose a big problem since we do not expect that the merge situation happens very often.

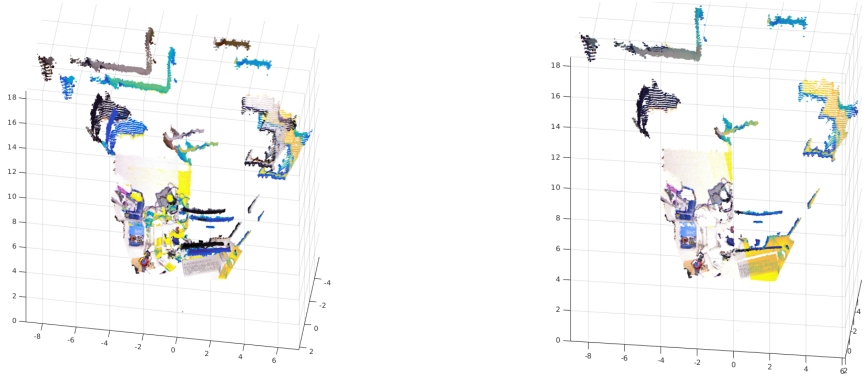
(A) First Image (B) Second Image



(C) Depth map for first image (D) Depth map for second image



(E) Horn's method (F) Direct image tracking (G) ICP technique

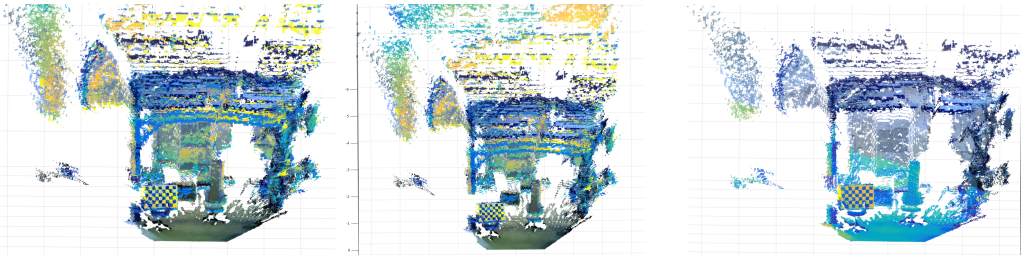


(H) Before Applying Transformation (I) After Applying Transformation

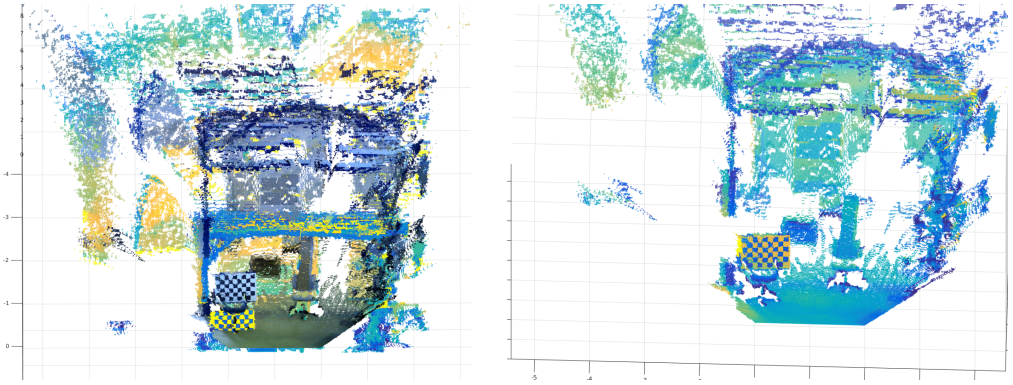
FIGURE 5.4: Images(A) and (B) show the images for which the similarity transformation T has to be computed. The image is taken from the TUM RGB-D Benchmark dataset. The second image is synthetically obtained by shifting the first image by 50 pixels in both x & y direction. (C), (D) show the corresponding depth maps of the two images. Figures (E),(F) and (G) show the two depth maps after applying the transformation computed using Horn's method, Direct image tracking method and ICP respectively. (H) and (I) shows the two depths overlaid on each other, before and after applying the transformation respectively. Note that (I) was obtained by applying the three techniques in series one after the other.

(A) First Image (B) Second Image

(C) Depth map for first image (D) Depth map for second image



(E) Horn's method (F) Direct image tracking (G) ICP technique



(H) Before Applying Transformation (I) After Applying Transformation

FIGURE 5.5: Images(A) and (B) show the images for which the similarity transformation T has to be computed. Both the images were taken from the Asus Xtion sensor in our lab environment. (C), (D) show the corresponding depth maps of the two images. Figures (E),(F) and (G) show the two depth maps after applying the transformation computed using Horn's method, Direct image tracking method and ICP respectively. (H) and (I) shows the two depths overlaid on each other, before and after applying the transformation respectively. The figure (I) was obtained through the three-stage estimation process.

Chapter 6

Collaborative SLAM System

6.1 Introduction

In this chapter we present a framework for collaborative visual SLAM using monocular cameras for a multi-robot system.The objective is to develop a multi-robot system with the capability to autonomously navigate and build a detailed map in an unstructured environment. The individual robots must be able to collaborate with other robots in the team to create a global map. The measurements captured by one robot can aid other robots improve their localisation and also give information about the regions of the environment which they might not have visited themselves.

Throughout the chapter, we deal with a team of mobile robots each equipped with a monocular camera to perform SLAM. Using a monocular camera gives the advantage for the system to be used both for indoor/outdoor applications and also for scenes with large variations in depth. Typically these conditions impose severe restrictions on other vision sensors such as RGB-D cameras and stereo pairs. However, using a monocular camera comes with its own challenges. In particular, the scale of the scene is unknown and needs to be continuously estimated as the depth information is not directly available.

6.1.1 Contributions

In this thesis, we propose a framework for collaborative visual SLAM using a centralized approach (as shown in Fig. 6.1) where:

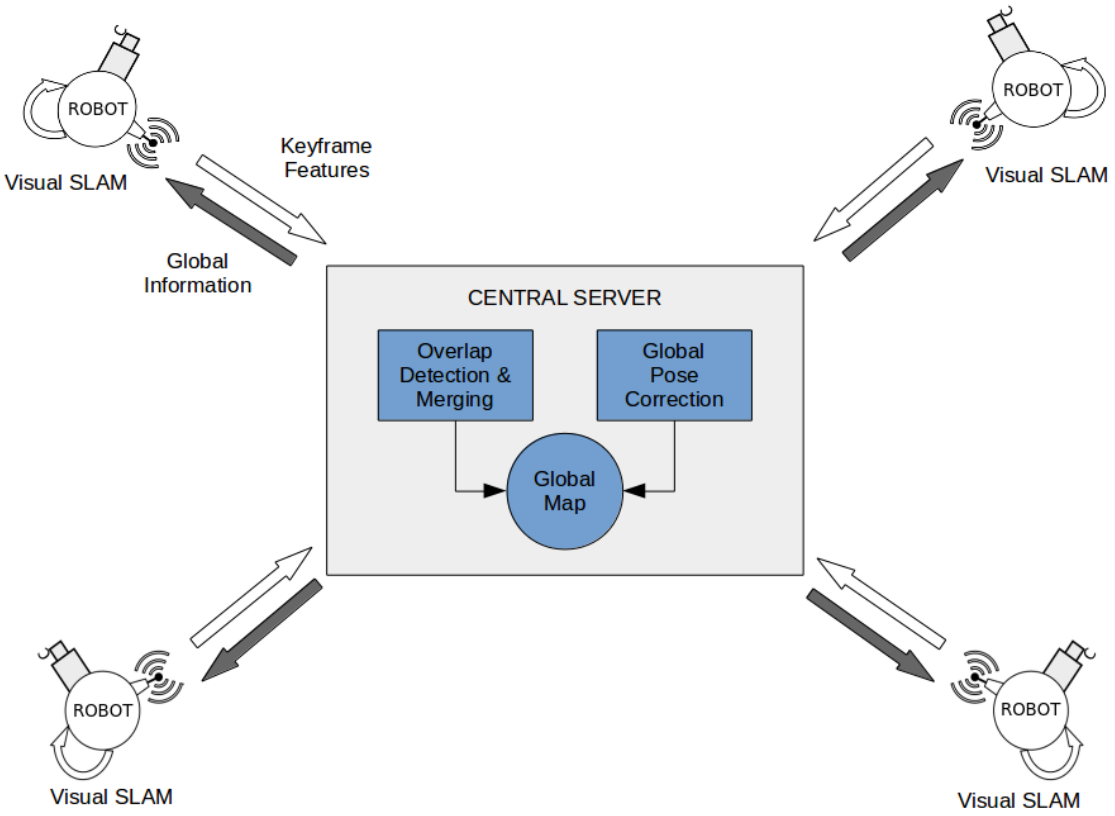


FIGURE 6.1: System Architecture: Collaborative Visual SLAM

- Each individual robot performs monocular visual SLAM and sends local keyframe information to a central server.
- The central server merges this information to create a global map and performs a pose correction using bundle adjustment.
- The updated pose is communicated back to individual robots as feedback thereby improving the local map and the localization estimate of the individual robots.

6.1.2 Related Work

Traditionally, SLAM has been performed using range sensors like laser scanners, sonars or using stereo vision [141]. More recently, monocular cameras (bearing only sensors) are also being used as the primary vision sensor like in [100],[101],[42]. In the multi-robot context, this problem has been studied under the banner of multi-camera structure from motion (SfM) [142] or multi-camera SLAM [143].

In [77], the authors analyse the improvement in localization quality of cooperative multi-robot localization over single robot localization. [78] demonstrates that by incorporating relative bearing information of the cameras, the overall accuracy of the localization is strongly improved. In this approach, an Extended Kalman Filter (EKF) is adopted to maintain the state containing configurations of all robots. In [80], the authors propose an interesting idea where two UAVs with monocular cameras act as a flexible stereo rig. With additional input from IMUs the relative poses are recoverd with absolute scale starting from an unknown initial configuration.

[83] deals with large-scale collaborative SLAM in an outdoor environment involving heterogeneous robots such as UAVs and ground mobile robots equipped with stereo cameras. It employs a global graph which maintains the relative relationships between a series of submaps built by each robot. The links between each submaps are created by events like robot rendezvous, scene feature matches or absolute localization information provided by GPS etc. These constraints allow the correction of the position estimates of submaps with respect to each other. [85] uses a multi-camera system to estimate the trajectory of moving objects in the scene along with building a 3D map of static objects. However, the system requires the image streams from all the cameras to be synchronized making it impractical to be used for real-time applications.

Several decentralized solutions have been proposed where data fusion is performed using only robots which are in direct communication range of each other. [87] proposes a method to efficiently distribute map information across a team of robots which is robust to node failures and changes in network topology. The proposed scheme consists of a local optimization module which executes single robot SLAM, a communication module which propagates the local graphs to other robots and a neighbourhood graph optimization module which combines all the local graphs into maps describing the neighbourhood of a robot. On the other hand, recently many centralized cloud based architectures for collaborative SLAM have been designed where the data intensive tasks can be mitigated to a powerful back-end cluster system [88], [91], [92]. This allows the use of small and energy efficient on-board processor to be placed on the robots while offloading major computations to the cloud.

In [84], the authors propose a centralized framework for a group of MAVs equipped with monocular cameras. Each MAV performs visual odometry on its on-board processor

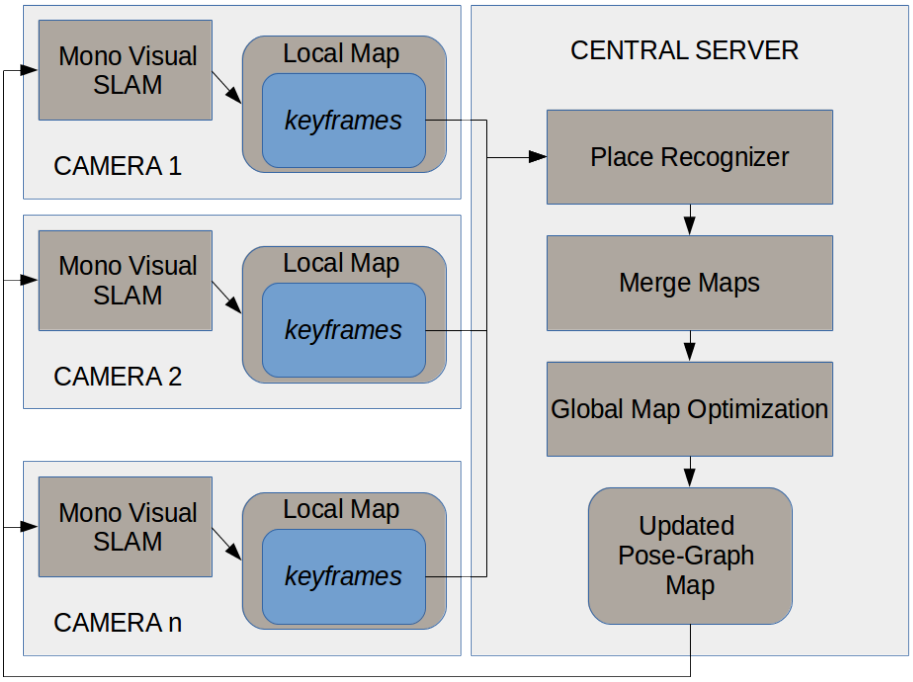


FIGURE 6.2: Overall scheme of our collaborative SLAM system

and sends keyframe information to a ground server where it is merged to realize a global map. In this chapter, we present a similar framework however each robot is capable of performing complete SLAM individually using full image information instead of using only features. In addition, a feedback mechanism is put in place which corrects the local estimates continuously. Also the framework allows the robots to join asynchronously and no prior relative information is required.

The organization of the rest of the chapter is as following. The next section presents a general overview of the system. The following sections on Monocular Visual SLAM, Place Recognition System and Map Merge explain each of the module in detail. The experimental results related to each module are explained. Finally, the overall system in described and the results from real-world experiments are presented.

6.2 System Overview

Figure 6.2 illustrates the overall scheme of our collaborative SLAM system. Each mobile robot equipped with a monocular camera performs visual SLAM using its on-board

computer. This provides each robot with an estimate of its pose and a 3D map of the environment in their respective co-ordinate frames. In our approach, we use a monocular SLAM algorithm based on direct image alignment which is able to estimate the seven DoF's including the scale of the scene. The map of the environment is stored as a pose-graph of keyframes each consisting of a semi-dense depth map of the corresponding view. This function is based on LSD SLAM [101].

Each robot sends its keyframe information including its pose in the local co-ordinate frame to the central server. Here the *place recognizer* function constantly monitors all the keyframes to detect overlapping scenes from different robots. The overlap detection is performed in the appearance space by extracting visual features from each keyframe and comparing them in a fast manner using Bag of Words (BoW) technique [121].

Once an overlap is detected between two cameras, the map merging sequence is initiated. It involves computing an initial transformation estimate between the matched keyframes by using a RANSAC version of the traditional Horn's algorithm [128]. This estimate is used as a starting point to run an optimization algorithm which estimates the similarity transformation between the two matched keyframes. Finally, this estimate is refined by performing an iterative closest point algorithm as described in [139].

After computation of this transformation, the two corresponding maps are merged into a global map and a new constraint is added between the two matched keyframes.In parallel, a bundle adjustment procedure is run over the global graph and the updated poses of the keyframe graph are communicated back to the individual robots as feedback. This information is in turn used by each robot to improve its localization estimate and the local map.

6.3 Methodology

In this section we detail each individual function in Fig. 6.2.

6.3.1 Visual SLAM

Each mobile robot performs an on-board monocular SLAM process which is able in real time to estimate its pose and create the map of the environment as a pose-graph

of keyframes . The problem of scale drift is addressed by implicitly including it as a parameter in the overall optimization procedure. This function is based on LSD-SLAM [101]. The overall method consists of the following main components:

6.3.1.1 Tracking

The camera pose $T \in se(3)$ is estimated with respect to the current keyframe K_i which consists of the image (I_i), the depth map (D_i) and the depth map variance V_i. For each new image I_j, the relative pose $T_{ji} \in se(3)$ is computed by minimizing the photometric error:

$$T_{ji} = \underset{T}{\operatorname{argmin}} \sum_{p} \left\| e_p^2(p, T_{ji}) \right\| \tag{6.1}$$

where the photometric residual $e_p(p, T_{ji}) = I_i(p) - I_j(\omega(p, D_i(p), T_{ji}))$ and ω is a warping function which computes the location of a pixel from the first image in the second image given the relative transformation T_{ji}. Note that in the actual implementation, a variance normalized residual is minimized thereby implicitly including depth accuracy in the computation of T_{ji}. The optimization problem can be posed as a weighted least squares problem [105] which can be solved using the Gauss-Newton minimization method [73].

6.3.1.2 Depth Map Estimation

A semi-dense inverse depth map is continuously estimated for each new frame. The depth map is computed by making several stereo comparisons of varying baseline over consecutive frames of the input video. The variable base line allows for accurate estimation of both near and far regions of the image. The method maintains a probabilistic depth hypothesis for each pixel modelled by a gaussian distribution which is continuously refined using an filtering approach described in [104]. Finally when the camera moves far from the current keyframe, a new keyframe is created and its depth map is initialized by projecting points from the previous keyframe on it.

6.3.1.3 Map Management and Optimization

The frame to frame alignment method previously introduced in 6.3.1.1 inherently accumulates drift over time due to small errors in each estimate arising from sensor noise and other model inaccuracies.

To deal with this problem, the SLAM system maintains the map as a graph where each vertex is the pose of the keyframe and each edge represents the relative transformation between the corresponding keyframes. Each time a new keyframe is added to the map, new edges are created and finally when previously visited regions of the scene are encountered, additional edges (loop closures) are added which help in reducing the accumulated drift.

However in the case of monocular SLAM, the scale of the scene cannot be observed directly which over a long trajectory leads to a drift causing major errors in tracking. To take care of the scale parameter, the overall pose graph is constructed in a manner such that the mean inverse of each keyframe is one and instead the edges are represented as transformation $T_{ji} \in sim(3)$. This allows the integration of the scale parameter directly in the optimization problem. So, the scaled transformation between the keyframes is estimated by minimizing the error function:

$$E(T) = \sum_{p} \left\| e_p^2(p, T_{ji}) + e_d^2(p, T_{ji}) \right\| \tag{6.2}$$

where the depth residual is $e_d(p, T_{ji}) = X_{2,z} - D_2(\pi(X_2^{'}))$ and

- $X_2^{'}$:= Transformed 3D point obtained by applying T to X_1, i.e. $X_2^{'} = \left(X_{2,x}^{'} X_{2,y}^{'} X_{2,z}^{'}\right) = T.X_1$
- π := camera projection function which computes the pixel location (u, v) given the 3D point X using the camera calibration parameters.

Finally, the overall map consisting of keyframe poses as vertices and $sim(3)$ constraints as edges, is continuously optimized in parallel using a general graph optimization framework like g2o [73]. This optimization over the graph reduces the drift both in scale and pose estimates.

6.3.2 Place Recognizer

This module runs continuously on the central server and is responsible to find scene overlap between different robots. Since the relative position of each robot is not known in the global coordinate frame at the beginning, the overlap is detected using the appearance space information only.

For every new keyframe image, visual features (e.g. SURF [116]) are computed which are view-point invariant. These features are then quantized with respect to a vocabulary and the resulting *visual words* description is stored. This *bag of words* (BoW) technique allows the scene to be represented as a collection of words which facilitates fast comparisons of feature descriptors.

We use the FAB-MAP method [106] to detect a scene overlap. This algorithm takes as input the BoW description of each image, compares it against all previously seen images. It gives as output the probability with which the current image matches any of the previously seen images. Moreover, it also computes the probability of the current image being a new one. These probabilities are calculated by solving a recursive Bayes estimation problem:

$$p(L_i|Z_k) = \frac{p(Z_k|L_i, Z_{k-1})p(L_i|Z_{k-1})}{p(Z_k|Z_{k-1})} \tag{6.3}$$

where L_i is a scene (location) in the world, Z_k is an observation (visual words) at time k.

The observation likelihood $p(Z_k|L_i, Z_k - 1)$ embeds the information that similar scenes will have same features correlated in a particular manner. This inter-dependence between visual features is captured using a Chow-Liu Tree [126] which is computed in a offline training step. $p(L_i|Z_{k-1})$ is the prior belief over the location which can be computed using an appropriate motion model. While $p(Z_k|Z_{k-1})$ is the normalizing term which is computed in a particular manner that facilitates finding out if present image is a scene not encountered before. [106]

Traditionally, the FAB-MAP technique has been used to find loop closures over long trajectories. Instead in our application, we use it to find if a place has been visited by other robots in the team and in-effect creating a *virtual loop closure*. Finally, in order to

avoid spurious matches, we only proceed for map merging if FAB-MAP reports a match over two or three consecutive images.

6.3.3 Map Merge

When the place recognizer module detects a scene overlap between robots and indicates a match point, the map merging procedure is initiated. The transformation between matching frames is done in three steps followed by an update to the global map.

6.3.3.1 Initial Transformation Estimate Using Horn's Method

For each keyframe image arriving at the central server, SURF features are computed and stored. Note that the same features were also used to compute a BoW representation required as an input to FAB-MAP.

The Horn's method describes a closed form solution to compute the scaled transformation given three 3D point correspondences using unit quaternions [128]. From the matching keyframe candidates proposed by FAB-MAP, 2D feature correspondences are extracted. Later the depth of each 2D feature is computed by taking an average over the keyframe depth in the descriptor neighbourhood. Finally, in order to deal with bad matches between features, we implement a robust RANSAC based version of the Horn's algorithm.

6.3.3.2 Refining Estimate Using $Sim3$ Tracker

As a second step, we use the tracking method based on minimizing the cost function as described in equation(6.2) to find an improved estimate for the scaled transformation. The estimate provided by Horn's method is used as a starting point for the tracker.

6.3.3.3 Correction using ICP

A final correction is made using the iterative closest point (ICP) algorithm, a technique from point cloud registration literature, which tries to find a transformation that minimizes the distance between a set of corresponding points in two clouds. We use an

augmented version of ICP which also includes surface normal and tangent information to improve the estimate as described in [139].

6.3.3.4 Global Map Update

Once the transformation is computed, new similarity constraints are added between the matching keyframes. The corresponding local maps are transformed into the global coordinate frame considering one of the two as reference (if its the first map merge) or using the existing reference otherwise. After the new constraints have been added a bundle adjustment technique is performed over the merged graph.

6.3.4 Overall Feedback System

Each time different robots visit the same place in the environment, new constraints are created in the global graph. While the mobile robots move in the environment, they may cross each others path multiple times resulting in *virtual loop closures*. These loop closure constraints help in reducing the overall drift.

Finally, the central server communicates the updated pose graph to individual robots which can then use this information to improve their localization estimate and the local maps.

This overall feedback mechanism facilitates the extension of sensing capability of an individual robot beyond the direct reach of their respective on-board sensors. In a sense, each robot in the team is able to "look" beyond what they can directly see and thus taking advantage of the collaborative system.

6.4 Experimental Results

In this section we present preliminary results with the aim of validating the concepts presented before. The experiments presented are not intended to be particularly challenging examples, they are simply used to take the reader through the functionality of the system.

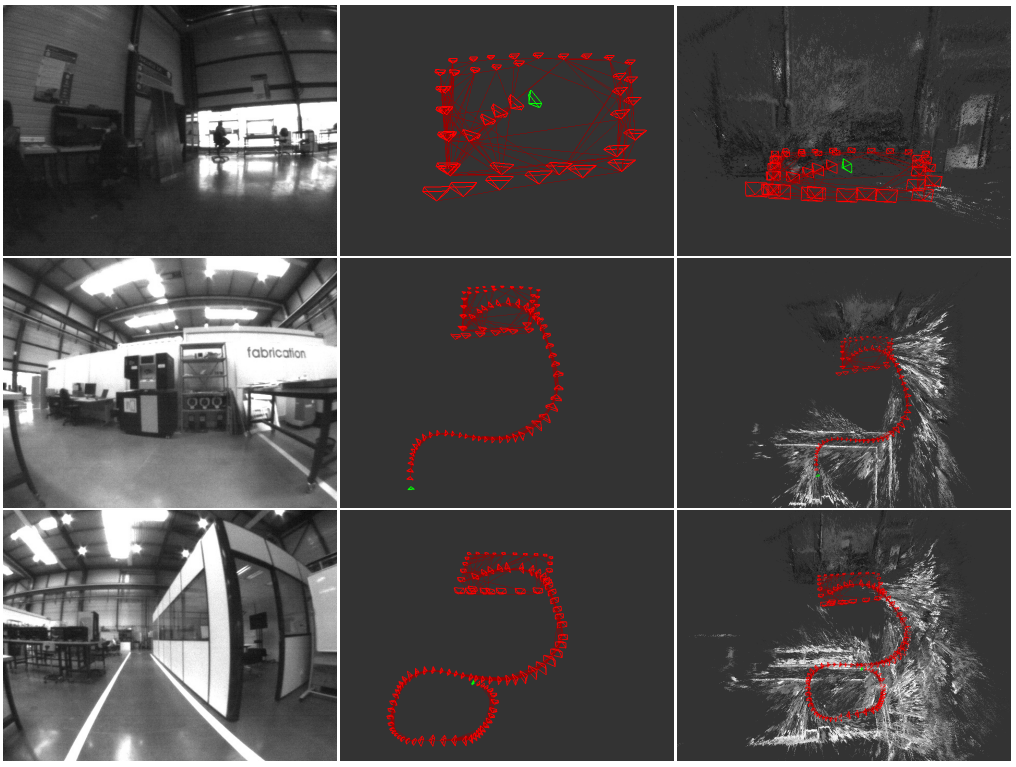


FIGURE 6.3: Monocular SLAM Process on robot $R1$ at three different instances. Left: Image captured by the camera. Middle:Trajectory built by the robot. Right: Map built by the robot.

The experiments were performed using two *Turtlebots*, each equipped with a *uEye* monocular camera attached with a wide-angle lens (∼130 degrees Field of View) and a Core 2 Duo laptop. The images are captured at 30 Hz with a resolution of 640 x 480 pixels. The experiments were conducted in an industrial-like indoor environment approximately 20m x 20m. The two robots start exploring the environment asynchronously. Moreover, their starting positions are not known to the central server. The robots traverse through regions with large variations in scene scales. The depth of theses scenes range from 1m to 15m. Finally, the communication between the robots and the central server is carried through the standard Wi-Fi protocol.

Figures 6.3 and 6.4 shows results from the monocular SLAM process running on the local computers of the two robots $R1$ and $R2$ respectively. The three columns in these figures show the images captured by the camera, the trajectory of the robot and the corresponding map built at three different instances (corresponding to the three rows). The trajectory of the robot $R1$ is illustrated in red and that of robot $R2$ in blue. Each pyramid in the map represents a keyframe location and the lines joining these keyframes

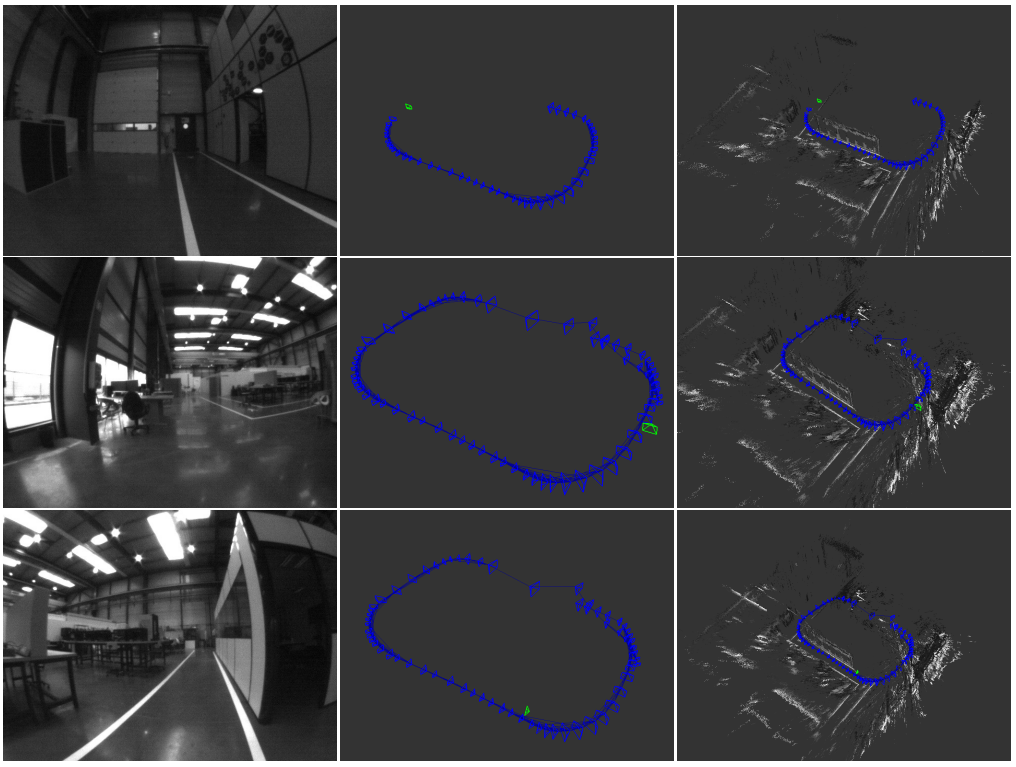


FIGURE 6.4: Monocular SLAM Process on robot $R2$ at three different instances. Left: Image captured by the camera. Middle:Trajectory built by the robot. Right: Map built by the robot.

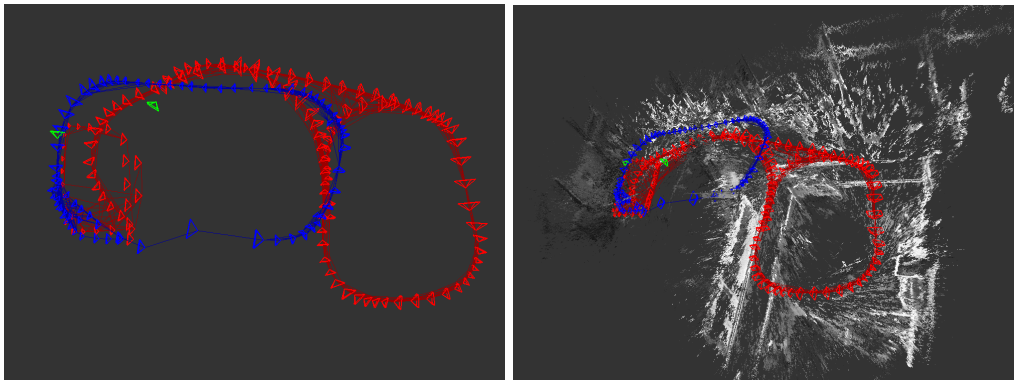


FIGURE 6.5: Global map computed at the central server . *Left:* Trajectory of the two robots in the merged map. *Right:* Depth map associated with the keyframes from both robots.

represent the constraints. The green pyramid depicts the current keyframe for both the cameras.

At instance 1, we see that robot $R1$ has completed a small square loop. During this trajectory, it makes some loop closures as well. Later robot $R2$ starts exploring some other part of the environment. At instance 2, the robot $R2$ completes a loop closure. As a result, we see that the overall trajectory of robot $R2$ has been optimized and the scale

factor is corrected as well. At instance 3, robot $R1$ enters a region previously visited by robot $R2$. At this time, the visual place recognition system triggers a merge between the two maps. The relative transformation between these two views is computed in three stages as described in section 6.3.

In this example, robot $R1$'s origin is considered as the reference coordinate frame. All the keyframes of robot $R2$ are transformed with respect to this origin. The global trajectory of the two robots and the joint map is shown in figure 6.5 . Finally, after searching for additional constraints and optimizing the global map using bundle adjustment, the updated keyframe poses are sent back to the two robots. This updated keyframe pose is then used by robots $R1$ and $R2$ to correct the localization estimate and the local map.

6.5 Conclusions

In this chapter, we presented a framework for collaborative visual SLAM for a team of mobile robots using a centralized approach. Each robot is able to individually perform monocular SLAM using its camera and on-board computer. The central server continuously receives local keyframe information from individual robots over Wi-Fi. Keyframes from all the robots are merged at this server and an optimization procedure is followed which minimizes the overall pose and mapping error. The updated pose information is sent back to the individual robots incorporating a feedback mechanism. No prior information regarding the relative position of the robots or the initial configuration is required. The system allows the robots to join and leave the team asynchronously.

The framework can be extended to work with different type of cameras by including the corresponding projection functions. In addition to the centralized framework, it would be interesting to add robot-to-robot communication. In this case, the robots can also exchange pose and map information with each other when they are in direct communication range.

Chapter 7

Conclusions

7.1 Conclusions

Multi-robot systems are increasingly being used for challenging real-world applications. These applications range from assisting first responders in cases of fire or a post-earthquake scenario to cleaning up toxic waste in industrial hazards. For such applications, the ability to localize and map the environment (SLAM) becomes the most essential requirement. This task can be performed with greater effectiveness if various team members collaborate with each other. A collaborative system extends the sensing capability of an individual robot by allowing them to *look* beyond the direct reach of their on-board sensors.

In this thesis, we present a flexible framework for collaborative visual SLAM using monocular cameras for a team of mobile robots. The main components of the framework include (a) A monocular SLAM process for each robot, (b) Detection of scene overlap amongst several robots, (c) Computation of the global map fusing measurements from all team members. The framework was tested on a team of two ground robots navigating in an indoor environment. With few modifications, the framework can be extend in a modular manner to accommodate different kinds of sensors, robots and operating environments.

The key contribution of the thesis is in the integration of several existing solutions to develop a flexible framework for collaborative Visual SLAM. The work in this thesis has been submitted as a paper titled "Collaborative Visual SLAM framework for a

Multi-Robot System" to the *7th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, Hamburg, 2015.*

7.2 Future Work

The collaborative visual SLAM framework developed during the thesis project has a huge scope for improvements. Some ideas in this direction could be:

- Extending the communication channels in the framework by introducing direct robot to robot communication. This could bring in advantages of a decentralized system such as computational load sharing among various team members and increased robustness in case of failures.
- Validating the framework in terms of localization accuracy and map quality in a rigorous manner. Qualitative analysis of the system performance could be done by comparing the results against the ground truth. The ground truth may be obtained by a motion capture system in an indoor environment or by using a RTK-GPS in outdoor scenarios.
- Extending the framework to work with different types of camera lens such as omni-directional, fish-eye etc. This can be done by including the corresponding projection function.
- Use of a cloud based architecture for back-end computations instead of a local centralized server. This would help in scaling the framework to cater to larger number of team members.
- Adapting and testing the framework with a hybrid team of robots such as a mixture of UAVs and ground mobile robots.
- Build upon the framework and design coordination strategies to efficiently explore the environment in a collaborative manner.

Bibliography

[1] H. Kuntze, C.W. Frey, I. Tchouchenkov, B. Staehle, E. Rome, K. Pfeiffer, A. Wenzel, and J. Wollenstein. Seneka - sensor network with mobile robots for disaster management. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 406–410, Nov 2012. doi: 10.1109/THS.2012.6459883.

[2] G.-J.M. Kruijff, V. Tretyakov, T. Linder, F. Pirri, M. Gianni, P. Papadakis, M. Pizzoli, A. Sinha, E. Pianese, S. Corrao, F. Priori, S. Febrini, and S. Angeletti. Rescue robots at earthquake-hit mirandola, italy: A field report. In *Safety, Security, and Rescue Robotics (SSRR), 2012 IEEE International Symposium on*, pages 1–8, Nov 2012. doi: 10.1109/SSRR.2012.6523866.

[3] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[4] Randall Smith, Matthew Self, and Peter Cheeseman. A stochastic map for uncertain spatial relationships. *International Journal of Robotics*, 1988.

[5] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *International Symposium of Robotics Research*, 1989.

[6] J.A. Castellanos, J.D. Tardos, and G. Schmidt. Building a global map of the environment of a mobile robot: the importance of correlations. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 2, pages 1053–1059 vol.2, Apr 1997. doi: 10.1109/ROBOT.1997.614274.

[7] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam)

problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, Jun 2001. ISSN 1042-296X. doi: 10.1109/70.938381.

[8] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *AUTONOMOUS ROBOTS*, 4:333–349, 1997.

[9] M. Golfarelli, D. Maio, and S. Rizzi. Elastic correction of dead-reckoning errors in map building. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 2, pages 905–911 vol.2, Oct 1998. doi: 10.1109/IROS.1998.727315.

[10] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[11] Michael Bosse, Paul Newman, John Leonard, and Seth Teller. Slam in large-scale cyclic environments using the atlas framework. *International Journal of Robotics Research*, pages 1113–1139, 2004.

[12] T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 4, pages 3841–3846 vol.4, 2000. doi: 10.1109/ROBOT.2000.845330.

[13] Sebastian Thrun and Michael Montemerlo. The graphslam algorithm with applications to large-scale mapping of urban structures. *INTERNATIONAL JOURNAL ON ROBOTICS RESEARCH*, 25(5):403–430, 2006.

[14] Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 176–183. Morgan Kaufmann Publishers Inc., 2000.

[15] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using fastslam. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1985–1991 vol.2, Sept 2003. doi: 10.1109/ROBOT.2003.1241885.

[16] C. Tomasi and T. Kanade. Factoring image sequences into shape and motion. In *Visual Motion, 1991., Proceedings of the IEEE Workshop on*, pages 21–28, Oct 1991. doi: 10.1109/WVM.1991.212792.

[17] C.J. Taylor, D. Kriegman, and P. Anandan. Structure and motion in two dimensions from multiple images: a least squares approach. In *Visual Motion, 1991., Proceedings of the IEEE Workshop on*, pages 242–248, Oct 1991. doi: 10.1109/WVM.1991.212801.

[18] R. Szeliski and Sing Bing Kang. Recovering 3d shape and motion from image streams using nonlinear least squares. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, pages 752–753, Jun 1993. doi: 10.1109/CVPR.1993.341157.

[19] P.F. McLauchlan and D.W. Murray. A unifying framework for structure and motion recovery from image sequences. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 314–320, Jun 1995. doi: 10.1109/ICCV.1995.466923.

[20] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Computer Vision, 1998. Sixth International Conference on*, pages 90–95, Jan 1998. doi: 10.1109/ICCV.1998.710705.

[21] Olivier Faugeras. Stratification of 3-d vision: projective, affine, and metric representations. *Journal of the Optical Society of America A*, 12:46548–4, 1995.

[22] Andrew W. Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In Hans Burkhardt and Bernd Neumann, editors, *ECCV (1)*, volume 1406 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 1998. ISBN 3-540-64569-1. URL http://dblp.uni-trier.de/db/conf/eccv/eccv1998-1.html#FitzgibbonZ98.

[23] Bill Triggs, P McLauchlan, Richard Hartley, and A Fitzgibbon. Bundle Adjustment – A Modern Synthesis. In B Triggs, A Zisserman, and R Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883, pages 298–372, 2000. URL http://lear.inrialpes.fr/pubs/2000/TMHF00.

[24] T.J. Broida, S. Chandrashekhar, and R. Chellappa. Recursive 3-d motion estimation from a monocular image sequence. *Aerospace and Electronic Systems, IEEE Transactions on*, 26(4):639–656, Jul 1990. ISSN 0018-9251. doi: 10.1109/7.55557.

[25] A. Azarbayejani and A.P. Pentland. Recursive estimation of motion, structure, and focal length. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(6):562–575, Jun 1995. ISSN 0162-8828. doi: 10.1109/34.387503.

[26] A. Chiuso, P. Favaro, Hailin Jin, and S. Soatto. Structure from motion causally integrated over time. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):523–535, Apr 2002. ISSN 0162-8828. doi: 10.1109/34.993559.

[27] J.-Y. Bouguet and P. Perona. Visual navigation using a single camera. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 645–652, Jun 1995. doi: 10.1109/ICCV.1995.466877.

[28] P. Favaro, Hailin Jin, and S. Soatto. A semi-direct approach to structure from motion. In *Image Analysis and Processing, 2001. Proceedings. 11th International Conference on*, pages 250–255, Sep 2001. doi: 10.1109/ICIAP.2001.957017.

[29] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–652–I–659 Vol.1, June 2004. doi: 10.1109/CVPR.2004.1315094.

[30] C Harris and M Stephens. A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conf.*, pages 147–151, 1988.

[31] David Nistér. Preemptive ransac for live structure and motion estimation. *Mach. Vis. Appl.*, 16(5):321–329, 2005. URL http://dblp.uni-trier.de/db/journals/mva/mva16.html#Nister05.

[32] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[33] Christopher G. Harris and J. M. Pike. 3d positional integration from image sequences. *Image Vision Comput.*, 6(2):87–90, 1988. URL http://dblp.uni-trier.de/db/journals/ivc/ivc6.html#HarrisP88.

[34] Ali Rahimi, Louis-Philippe Morency, and Trevor Darrell. Reducing drift in parametric motion tracking. In *ICCV*, pages 315–322, 2001. URL http://dblp.uni-trier.de/db/conf/iccv/iccv2001-1.html#RahimiMD01.

[35] Jose Neira, María Isabel Ribeiro, Juan Domingo Tardós, and Centro Politecnico Superior (cps. Mobile robot localization and map building using monocular vision. In *IN THE 5TH SYMPOSIUM FOR INTELLIGENT ROBOTICS SYSTEMS*, pages 275–284, 1997.

[36] S. Se, D. Lowe, and J. Little. Local and global localization for mobile robots using visual landmarks. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 414–420 vol.1, 2001. doi: 10.1109/IROS.2001.973392.

[37] DavidG. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. ISSN 0920-5691.

[38] A.J. Davison and D.W. Murray. Simultaneous localization and map-building using active vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):865–880, Jul 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1017615.

[39] A.J. Davison and N. Kita. 3d simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–384–I–391 vol.1, 2001. doi: 10.1109/CVPR.2001.990501.

[40] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410 vol.2, Oct 2003. doi: 10.1109/ICCV.2003.1238654.

[41] N. D. Molton, A. J. Davison, and I. D. Reid. Locally planar patch features for real-time structure from motion. In *Proc. British Machine Vision Conference*. BMVC, September 2004. (To appear).

[42] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, June 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1049.

[43] G. Dissanayake, H. Durrant-Whyte, and Tim Bailey. A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 1009–1014 vol.2, 2000. doi: 10.1109/ROBOT.2000.844732.

[44] M. Csorba. Simultaneous localisation and map building. Ph.D. thesis, university of oxford, uk, 1997.

[45] Paul M. Newman and Hugh F. Durrant-Whyte. Geometric projection filter: an efficient solution to the slam problem. volume 4571, pages 22–33, 2001. doi: 10.1117/12.444166. URL http://dx.doi.org/10.1117/12.444166.

[46] J. Knight, A. Davison, and I. Reid. Towards constant time slam using postponement. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 405–413 vol.1, 2001. doi: 10.1109/IROS.2001.973391.

[47] J.E. Guivant and E.M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *Robotics and Automation, IEEE Transactions on*, 17(3):242–257, Jun 2001. ISSN 1042-296X. doi: 10.1109/70.938382.

[48] S.J. Julier and J.K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4238–4243 vol.4, 2001. doi: 10.1109/ROBOT.2001.933280.

[49] J.A. Castellanos, J. Neira, and J.D. Tardós. Limits to the consistency of ekf-based slam. In *5th IFAC Symp. on Intelligent Autonomous Vehicles, IAV'04*, Lisbon, Portugal, 2004. ISBN 978-0-08-044237-2. URL http://webdiis.unizar.es/~jdtardos/papers/Castellanos_IAV_2004.pdf.

[50] Tim Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the ekf-slam algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3562–3568, Oct 2006. doi: 10.1109/IROS.2006.281644.

[51] Kok Seng Chong and Lindsay Kleeman. Feature-based mapping in real, large scale environments using an ultrasonic array, 1999.

[52] J.J. Leonard and H.J.S. Feder. Decoupled stochastic mapping [for mobile robot amp; auv navigation]. *Oceanic Engineering, IEEE Journal of*, 26(4):561–571, Oct 2001. ISSN 0364-9059. doi: 10.1109/48.972094.

[53] T. Bailey. Mobile robot localisation and mapping in extensive outdoor environments. ph.d. thesis, university of sydney, australia, 2002.

[54] C. Estrada, J. Neira, and J.D. Tardos. Hierarchical slam: Real-time accurate mapping of large environments. *Robotics, IEEE Transactions on*, 21(4):588–596, Aug 2005. ISSN 1552-3098. doi: 10.1109/TRO.2005.844673.

[55] J. Guivant and E. Nebot. Improving computational and memory requirements of simultaneous localization and map building algorithms. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 3, pages 2731–2736, 2002. doi: 10.1109/ROBOT.2002.1013645.

[56] Sebastian Thrun, Daphne Koller, Zoubin Ghahramani, Hugh Durrant-Whyte, and AndrewY. Ng. Simultaneous mapping and localization with sparse extended information filters: Theory and initial results. In Jean-Daniel Boissonnat, Joel Burdick, Ken Goldberg, and Seth Hutchinson, editors, *Algorithmic Foundations of Robotics V*, volume 7 of *Springer Tracts in Advanced Robotics*, pages 363–380. Springer Berlin Heidelberg, 2004. ISBN 978-3-642-07341-0. doi: 10.1007/978-3-540-45058-0_22. URL http://dx.doi.org/10.1007/978-3-540-45058-0_22.

[57] Zhan Wang, Shoudong Huang, and G. Dissanayake. Decoupling localization and mapping in slam using compact relative maps. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3336–3341, Aug 2005. doi: 10.1109/IROS.2005.1545117.

[58] R. Eustice, M. Walter, and J. Leonard. Sparse extended information filters: insights into sparsification. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3281–3288, Aug 2005. doi: 10.1109/IROS.2005.1545053.

[59] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234, Nov 2007. doi: 10.1109/ISMAR.2007.4538852.

[60] Liang Zhao, Shoudong Huang, Lei Yan, J.J. Wang, G. Hu, and G. Dissanayake. Large-scale monocular slam by local bundle adjustment and map joining. In *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pages 431–436, Dec 2010. doi: 10.1109/ICARCV.2010.5707820.

[61] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Scale drift-aware large scale monocular slam. In Yoky Matsuoka, Hugh F. Durrant-Whyte, and José Neira, editors, *Robotics: Science and Systems*. The MIT Press, 2010. ISBN 978-0-262-51681-5. URL http://dblp.uni-trier.de/db/conf/rss/rss2010.html#StrasdatMD10.

[62] K. Konolige and M. Agrawal. Frameslam: From bundle adjustment to real-time visual mapping. *Robotics, IEEE Transactions on*, 24(5):1066–1077, Oct 2008. ISSN 1552-3098. doi: 10.1109/TRO.2008.2004832.

[63] Jongwoo Lim, J.-M. Frahm, and M. Pollefeys. Online environment mapping. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3489–3496, June 2011. doi: 10.1109/CVPR.2011.5995511.

[64] Christopher Mei, Gabe Sibley, Mark Cummins, Paul Newman, and Ian Reid. Rslam: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, 94(2):198–214, 2011. ISSN 0920-5691. doi: 10.1007/s11263-010-0361-7. URL http://dx.doi.org/10.1007/s11263-010-0361-7.

[65] Gabe Sibley, Christopher Mei, Ian Reid, and Paul Newman. Adaptive relative bundle adjustment. In *Robotics Science and Systems (RSS)*, Seattle, USA, June 2009.

[66] M. Cummins and P. Newman. Probabilistic appearance based navigation and loop closing. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2042–2048, April 2007. doi: 10.1109/ROBOT.2007.363622.

[67] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Editors choice article: Visual slam: Why filter? *Image Vision Comput.*, 30(2):65–77, February 2012. ISSN 0262-8856. doi: 10.1016/j.imavis.2012.02.009. URL http://dx.doi.org/10.1016/j.imavis.2012.02.009.

[68] Richard A. Newcombe, S.J. Lovegrove, and A.J. Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327, Nov 2011. doi: 10.1109/ICCV.2011.6126513.

[69] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live metric 3d reconstruction on mobile phones. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 65–72, Dec 2013. doi: 10.1109/ICCV.2013.15.

[70] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche. Monofusion: Real-time 3d reconstruction of small scenes with a single web camera. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 83–88, Oct 2013. doi: 10.1109/ISMAR.2013.6671767.

[71] V.A. Prisacariu, O. Kahler, D.W. Murray, and I.D. Reid. Simultaneous 3d tracking and reconstruction on a mobile phone. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 89–98, Oct 2013. doi: 10.1109/ISMAR.2013.6671768.

[72] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2100–2106, Nov 2013. doi: 10.1109/IROS.2013.6696650.

[73] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613, May 2011. doi: 10.1109/ICRA.2011.5979949.

[74] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 127–136, Oct 2011. doi: 10.1109/ISMAR.2011.6092378.

[75] F. Steinbrucker, J. Sturm, and D. Cremers. Real-time visual odometry from dense rgb-d images. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 719–722, Nov 2011. doi: 10.1109/ICCVW.2011.6130321.

[76] T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard, and J. McDonald. Robust real-time visual odometry for dense rgb-d mapping. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5724–5731, May 2013. doi: 10.1109/ICRA.2013.6631400.

[77] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi-robot localization. *Auton. Robots*, 8(3):325–344, 2000. URL http://dblp.uni-trier.de/db/journals/arobots/arobots8.html#FoxBKT00.

[78] A. Martinelli, F. Pont, and R. Siegwart. Multi-robot localization using relative observations. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2797–2802, April 2005. doi: 10.1109/ROBOT.2005.1570537.

[79] M. Cognetti, P. Stegagno, A. Franchi, G. Oriolo, and H.H. Bulthoff. 3-d mutual localization with anonymous bearing measurements. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 791–798, May 2012. doi: 10.1109/ICRA.2012.6225288.

[80] Markus W. Achtelik, S. Weiss, M. Chli, F. Dellaert, and R. Siegwart. Collaborative stereo. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2242–2248, Sept 2011. doi: 10.1109/IROS.2011.6094866.

[81] R. Rocha, J. Dias, and A. Carvalho. Cooperative multi-robot systems a study of vision-based 3-d mapping using information theory. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 384–389, April 2005. doi: 10.1109/ROBOT.2005.1570149.

[82] A. Howard. Multi-robot mapping using manifold representations. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 4198–4203 Vol.4, April 2004. doi: 10.1109/ROBOT.2004.1308933.

[83] Teresa A. Vidal-Calleja, Cyrille Berger, Joan Solà, and Simon Lacroix. Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain. *Robotics and Autonomous Systems*, 59(9):654 – 674, 2011. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2011.05.008. URL http://www.sciencedirect.com/science/article/pii/S0921889011000923.

[84] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza. Collaborative monocular slam with multiple micro aerial vehicles. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3962–3970, Nov 2013. doi: 10.1109/IROS.2013.6696923.

[85] Danping Zou and Ping Tan. Coslam: Collaborative visual slam in dynamic environments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35 (2):354–366, Feb 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2012.104.

[86] H. Durrant-Whyte. Data fusion in sensor networks. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 2–, April 2005. doi: 10.1109/IPSN.2005.1440882.

[87] Alexander Cunningham, Balamanohar Paluri, and Frank Dellaert. Ddf-sam: Fully distributed slam using constrained factor graphs. In *IROS*, pages 3025–3030. IEEE, 2010. ISBN 978-1-4244-6674-0. URL http://dblp.uni-trier.de/db/conf/iros/iros2010.html#CunninghamPD10.

[88] R. Arumugam, V.R. Enti, Liu Bingbing, Wu Xiaojun, K. Baskaran, Foong Foo Kong, A.S. Kumar, Kang Dee Meng, and Goh Wai Kit. Davinci: A cloud computing framework for service robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3084–3089, May 2010. doi: 10.1109/ROBOT.2010.5509469.

[89] Apache Hadoop. URL http://hadoop.apache.org/.

[90] Willow Garage's Robot Operating System (ROS). URL http://www.ros.org/.

[91] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea. Rapyuta: The roboearth cloud engine. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 438–444, May 2013. doi: 10.1109/ICRA.2013.6630612.

[92] L. Riazuelo, Javier Civera, and J.M.M. Montiel. C2tam: A cloud framework for cooperative tracking and mapping. *Robotics and Autonomous Systems*, 62(4):401 – 413, 2014. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2013.11.007. URL http://www.sciencedirect.com/science/article/pii/S0921889013002248.

[93] Wolfram Burgard, Mark Moors, and Frank E. Schneider. Collaborative exploration of unknown environments with teams of mobile robots. In Michael Beetz, Joachim Hertzberg, Malik Ghallab, and Martha E. Pollack, editors, *Advances in Plan-Based Control of Robotic Agents*, volume 2466 of *Lecture Notes in Computer Science*, pages 52–70. Springer, 2001. ISBN 3-540-00168-9. URL http://dblp.uni-trier.de/db/conf/dagstuhl/robagents2001.html#BurgardMS01.

[94] William W. Cohen. Adaptive mapping and navigation by teams of simple robots. *Robotics and Autonomous Systems*, 18(4):411–434, 1996. URL http://dblp.uni-trier.de/db/journals/ras/ras18.html#Cohen96.

[95] Sven Koenig, Boleslaw K. Szymanski, and Yaxin Liu. Efficient and inefficient ant coverage methods. *Ann. Math. Artif. Intell.*, 31(1-4):41–76, 2001. URL http://dblp.uni-trier.de/db/journals/amai/amai31.html#KoenigSL01.

[96] Aude Billard, Auke Jan Ijspeert, and Alcherio Martinoli. A multi-robot system for adaptive exploration of a fast changing environment: Probabilistic modeling and experimental study, 1999.

[97] Tucker Balch, Ronald, and C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1:27–52, 1994.

[98] K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 254–259 vol.2, May 1993. doi: 10.1109/ROBOT.1993.292155.

[99] Michael A. Bender. The power of team exploration: Two robots can learn unlabeled directed graphs. In *In Proceedings of the Thirty Fifth Annual Symposium on Foundations of Computer Science*, pages 75–85, 1994.

[100] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *CoRR*, abs/1502.00956, 2015.

[101] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014.

[102] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 15–22, 2014. doi: 10.1109/ICRA.2014.6906584. URL http://dx.doi.org/10.1109/ICRA.2014.6906584.

[103] Cyril Roussillon, Aurélien Gonzalez, Joan Solà, Jean-Marie Codol, Nicolas Mansard, Simon Lacroix, and Michel Devy. Rt-slam: A generic and real-time visual slam implementation. *CoRR*, abs/1201.5450, 2012. URL http://dblp.uni-trier.de/db/journals/corr/corr1201.html#abs-1201-5450.

[104] Jakob Engel, Jürgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 1449–1456, 2013. doi: 10.1109/ICCV.2013.183. URL http://dx.doi.org/10.1109/ICCV.2013.183.

[105] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Robust odometry estimation for RGB-D cameras. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pages 3748–3754, 2013.

[106] Mark Cummins and Paul Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008. doi: 10.1177/0278364908090961.

[107] Anat Levin and Richard Szeliski. Visual odometry and map correlation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, volume I, pages 611–618, Washington, DC, June 2004. IEEE Computer Society. URL http://research.microsoft.com/apps/pubs/default.aspx?id=75590.

[108] S. Se, D.G. Lowe, and J.J. Little. Vision-based global localization and mapping for mobile robots. *Robotics, IEEE Transactions on*, 21(3):364–375, June 2005. ISSN 1552-3098. doi: 10.1109/TRO.2004.839228.

[109] Kin Leong Ho and Paul Newman. Detecting loop closure with scene sequences. *Int. J. Comput. Vision*, 74(3):261–286, September 2007. ISSN 0920-5691. doi: 10.1007/s11263-006-0020-1. URL http://dx.doi.org/10.1007/s11263-006-0020-1.

[110] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 1023–1029 vol.2, 2000. doi: 10.1109/ROBOT.2000.844734.

[111] Pierre Lamon, Illah Nourbakhsh, Bjorn Jensen, and Roland Siegwart. Deriving and matching image fingerprint sequences for mobile robot localization. In *Proceedings of ICRA 2001*, May 2001.

[112] Antonio Torralba, Kevin P. Murphy, William T. Freeman, and Mark A. Rubin. Context-based vision system for place and object recognition. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 273–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1950-4. URL http://dl.acm.org/citation.cfm?id=946247.946665.

[113] B.J.A Kröse, N. Vlassis, R. Bunschoten, and Y. Motomura. A probabilistic model for appearance-based robot localization. In *In First European Symposium on Ambience Intelligence (EUSAI*, pages 264–274. Springer, 2000.

[114] Fabio Ramos, Ben Upcroft, Suresh Kumar, and Hugh Durrant-Whyte. A bayesian approach for place recognition. *Robotics and Autonomous Systems*, 60(4):487 – 497, 2012. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2011.11.002. URL http://www.sciencedirect.com/science/article/pii/S0921889011002053.

[115] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0164-8. URL http://dl.acm.org/citation.cfm?id=850924.851523.

[116] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[117] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 2, pages 1060–1065 vol.2, Oct 1998. doi: 10.1109/IROS.1998.727439.

[118] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization by combining an image-retrieval system with monte carlo localization. *Robotics, IEEE Transactions on*, 21(2):208–216, April 2005. ISSN 1552-3098. doi: 10.1109/TRO.2004.835453.

[119] Fayin Li and J. Kosecka. Probabilistic location recognition using reduced feature set. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3405–3410, May 2006. doi: 10.1109/ROBOT.2006.1642222.

[120] Junqiu Wang, R. Cipolla, and Hongbin Zha. Vision-based global localization using a visual vocabulary. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 4230–4235, April 2005. doi: 10.1109/ROBOT.2005.1570770.

[121] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 1470–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1950-4. URL http://dl.acm.org/citation.cfm?id=946247.946751.

[122] Han Wang Chen, Cheng. Appearance-based topological bayesian inference for loop-closing detection in a cross-country environment. *I. J. Robotic Res.*, 25(10):953–983, 2006. URL http://dblp.uni-trier.de/db/journals/ijrr/ijrr25.html#ChenW06.

[123] T. Geodeme. Visual navigation Ph.D. thesis,katholieke universitiet leuven, 1997.

[124] Z. Zivkovic, B. Bakker, and B. Krose. Hierarchical map building using visual landmarks and geometric constraints. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2480–2485, Aug 2005. doi: 10.1109/IROS.2005.1544951.

[125] P. Newman, D. Cole, and K. Ho. Outdoor slam using visual appearance and laser ranging. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1180–1187, May 2006. doi: 10.1109/ROBOT.2006.1641869.

[126] Marina Meila. An accelerated chow and liu algorithm: Fitting tree distributions to high-dimensional sparse data. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 249–257, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-612-2. URL http://dl.acm.org/citation.cfm?id=645528.657640.

[127] Cyphy visual slam, qut brisbane. URL http://code.google.com/p/cyphy-vis-slam/.

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

[129] American Society of Photogrammetry, C.C. Slama, C. Theurer, and S.W. Henriksen. *Manual of photogrammetry*. American Society of Photogrammetry, 1980. ISBN 9780937294017. URL https://books.google.fr/books?id=1MoYAQAAIAAJ.

[130] Bon A. DeWitt and Paul R. Wolf. *Elements of Photogrammetry(with Applications in GIS)*. McGraw-Hill Higher Education, 3rd edition, 2000. ISBN 0072924543.

[131] Schut. *On exact linear equations for the computation of the rotational elements of absolute orientation*. Photogrammetria, 1960.

[132] Balasubramanian S. Oswal H.L. *An exact solution of absolute orientation*. Photogrammetria, 1968.

[133] A.I. Comport, E. Malis, and P. Rives. Accurate Quadri-focal Tracking for Robust 3D Visual Odometry. In *IEEE International Conference on Robotics and Automation, ICRA'07*, Rome, Italy, April 2007. doi: http://dx.doi.org/110.1177/0278364909356601. URL http://www.i3s.unice.fr/~comport/publications/2007_icra_comport.pdf.

[134] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. REMODE: probabilistic, monocular dense reconstruction in real time. In *2014 IEEE International*

Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014, pages 2609–2616, 2014. doi: 10.1109/ICRA.2014.6907233. URL http://dx.doi.org/10.1109/ICRA.2014.6907233.

[135] Jan Stühmer, Stefan Gumhold, and Daniel Cremers. Real-time dense geometry from a handheld camera. In *DAGM-Symposium'10*, pages 11–20, 2010.

[136] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, Feb 1992. ISSN 0162-8828. doi: 10.1109/34.121791.

[137] Martin Magnusson, Tom Duckett, and Achim J. Lilienthal. Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10): 803–827, Oct 24 2007. ISSN 1556-4959 (Print), 1556-4967 (Online).

[138] Aleksandr Segal, Dirk Hähnel, and Sebastian Thrun. Generalized-icp. In Jeff Trinkle, Yoky Matsuoka, and José A. Castellanos, editors, *Robotics: Science and Systems*. The MIT Press, 2009. ISBN 978-0-262-51463-7. URL http://dblp.uni-trier.de/db/conf/rss/rss2009.html#SegalHT09.

[139] Jacopo Serafin and Giorgio Grisetti. Using augmented measurements to improve the convergence of icp. In *Proc. of the 4th International Conference on Simulation, Modeling and Programming for Autonomous Robots. (SIMPAR 2014)*, 2014. doi: 10.1007/978-3-319-11900-7_48.

[140] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[141] Sebastian Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.

[142] Michael Kaess and Frank Dellaert. Probabilistic structure matching for visual slam with a multi-camera rig. *Comput. Vis. Image Underst.*, 114(2):286–296, 2010. ISSN 1077-3142.

[143] J. Sola, A. Monin, and M. Devy. Bicamslam : Two times mono is more than stereo. In *IEEE Int. Conf. on Robotics Automation*, 2007.