Towards Reliability and Scalability in Feature Based Simultaneous Localization and Mapping

Gibson Hu

Submitted in fulfillment of the requirement
for the degree of Doctor of Philosophy

2014

The Faculty of Engineering and Information Technology
Centre for Autonomous Systems, University of Technology Sydney

www.uts.edu.au

Supervisor	:	A/Prof. Shoudong Huang
Co-Supervisor	:	Dr. Alen Alempijevic
Second Co-Supervisor	:	Prof. Gamini Dissanayake

Certificate

I, Gibson Hu, declare that this thesis entitled *Towards Reliability and Scalability in Feature Based Simultaneous Localization and Mapping* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself or jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Production Note:
Signature removed prior to publication.

Date: 01/07/2014

Acknowledgements

Having come to the end of my PhD candidature, it has been a wonderful experience working with some of the best and brightest at the Centre for Autonomous Systems, UTS. Four years has passed by very quickly with much learning and research in the exciting field of Robotics. The road has been filled with many challenges, from late nights spent coding to traveling overseas for competition and conferences.

I have been very privileged to work with Associate Professor Shoudong Huang, Dr Alen Alempijevic and Professor Gamini Dissanayake, who have not only been inspirational but role models to me. I would also like to thank the post docs, Liang Zhao, Zhang Wang, Jack Wang, Gavin Paul and the students, Minjie Liu, Lei Shi, Andrew To, Kasra Khosoussi, Mohammad Norouzi, and Freek De Bruijn.

Finally, I would like to thank my friends and family for their unending support.

Abstract

Simultaneous Localization and Mapping (SLAM) has always been an attractive topic in the vibrant field of robotics. Feature based representations of the problem can be seen as one of the most common definitions. In recent years, many SLAM researchers have realized some limitations of filtering based methods and started to focus more on optimization based SLAM techniques. However, this raises several questions surrounding convergence reliability and, similar to filtering, algorithm scalability.

In SLAM, sensor noise and non-linearity often causes the problem to become difficult. Converging towards the global minimum in a non-linear least squares formulation is by no means easy. Typically, one would need to start from a good initial estimate, preferably already inside the basin of attraction of the global minimum. In this thesis, we introduce a technique called Iterative Re-Weighted Least Squares bootstrapping to achieve a good initial estimate even when the noise is exceptionally large.

As a robot continues to traverse through its environment the complexity of SLAM tends to scale badly with the cumulative nature of graph nodes and edges. To solve large SLAM problems within a reasonable time scale one must also take into consideration elements of accuracy and consistency. In this thesis, we propose two alternative algorithms to handle complexity, Sparse Map Joining and Pose Graph Representation. Both of which contain unique advantages for handling the diverse scenarios within SLAM.

A series of quantitative analyses are performed on a number of challenging datasets, both real and simulated. In addition to this we perform a comprehensive case study on a specific type of feature based SLAM problem, RGB-D SLAM. This demonstrates how our technique is capable of avoiding inaccuracies and failure scenarios that is otherwise common in other RGB-D SLAM algorithms.

Contents

List of Tables

List of Figures

Nomenclature

Formatting Style

$\hat{x}$	Measured
$\bar{x}$	Estimated
$\tilde{x}$	Actual

Subscript

m	features index	i, j	pose index
t	time index		

Superscript

$\mathcal{M}$	marginalized	P	pose set
F	feature set	O	odometry set
K	pose subset	L	local map
G	global map	S	sensor

Notations

$\sim \mathcal{N}$	normally distributed	$\underset{X}{\text{argmin}}$	minimizer
$[.]$	vector elements	$\|\|.\|\|$	Euclidean norm
X^*	the optimum	$X^{(0)}$	initial estimate

Variables

$\mathcal{G}$	undirected graph	$\mathcal{V}$	graph vertices
$\mathcal{E}$	graph edges	E	essential matrix
Z	measurement vector	X	state vector
Σ	covariance matrix	Ω, Λ	information matrix
χ^2	chi squared value	ν	degree of freedom
$\mathcal{P}$	optimisation problem	w	weight scalar
F	fundamental matrix	S	scale
K	calibration matrix	T	transformation matrix

Functions

$g()$	pose to pose	$h()$	pose to feature
$b()$	generalized model function	$\rho()$	m-estimator
$Rot()$	rotation matrix	$Proj()$	projection matrix
$Horn()$	Horn's method	$\psi()$	influence function

Abbreviations

SLAM	Simultaneous Localisation and Mapping
ML	Maximum Likelihood
GN	Gauss-Newton
GD	Gradient Decent
PDL	Powell's Dog-Leg
LM	Levenberg-Marquardt
STD	Standard Deviation
SGD	Stochastic Gradient Descent
SBA	Sparse Bundle Adjustment
iSAM	Incremental Smoothing and Mapping
g^2o	General Graph Optimization
ParallaxBA	Parallax Angle Bundle Adjustment
Alg	Algorithm
RMSE	Root Mean Squared Error
NEES	Normalized Estimation Error Squared

IRLS	Iterative Re-weighted Least Squares
GT	Ground Truth
IMU	Inertial Measurement Unit
RPE	Relative Pose Error
ATE	Absolute Trajectory Error
SIFT	Scale Invariant Feature Transform
SURF	Speeded-Up Robust Features
I-SLSJF	Iterated Sparse Local Submap Joining Filter
EIF	Extended Information Filter
EKF	Extended Kalman Filter
SMJ	Sparse Map Joining
BO	Batch Optimization
SO	Sequential Optimization
DCO	Divide and Conquer Optimization
LAGO	Linear Approximation for Graph Optimization
TORO	Tree based netwORk Optimizer
MO	Multi Observation method
SO	Single Observation method
PGR	Pose Graph Representation
MAP	Maximum a Posteriori

DBN	Dynamic Bayesian Network
ICP	Iterative Closest Point
RE-RANSAC	Re-projection Error RANdom SAmpling Consensus
EM-RANSAC	Essential Matrix RANdom SAmpling Consensus
VO	Visual Odometry
FABMAP	Fast Appearance Based Mapping
IR	Infrared
M-Estimator	Maximum likelihood-type Estimator
GPS	Global Positioning System

Chapter 1

Introduction

Robotics is an exciting field of research which tries to build and understand many of the tasks human perform in everyday life. While many robots have in the past been restricted by known locations such as assembly line in a factory, nowadays, a wide range of solutions in non-fixed environments are made possible through the aid of robotics. The major challenge that exists is when these robots are faced with new unstructured and unknown environments.

Even for humans confronted with the same problem, the first typical course of action is to represent this new environment as a map. However, in doing so one must also have some knowledge of ones location. This leads to the famous *causality dilemma* in robotics called Simultaneous Localization and Mapping (SLAM), the process of building the map around the localized position whilst at the same time localizing oneself given a map.

Knowing one's location and having a well established understanding of the environment is critical for many high level operations. Some key competencies of an advanced robotic system should include navigation, exploration, obstacle avoidance and path planning, all of which require some information gained from SLAM.

The SLAM problem was first raised in a seminal paper by Leonard and Durrant-Whyte [1]. They asked, whether it was possible to perform SLAM using only the sensors on board a robot, without the aid of any external global positioning systems. Since no sensor is perfect, the robot can never make an exact judgment on what will happen next or where

it will be in the next time step. However, Leonard and Durrant-Whyte convey that it is possible to make an informed estimate on the robot's location and map with a measured degree of certainty.

1.0.1 SLAM Applications

What makes SLAM useful is the large number of real world applications that exist, whereby there is no aid given from any external referencing systems (such as GPS). In this situation a robot must rely solely on its own measurement sensors to build up an understanding of its surroundings. Below are two such applications.

Home Robotics

(a) `Willow Garage PR2` (b) `UTS RobotAssist`

Figure 1.1: Home Robots

For a robot to confidently operate in a home environment, it must be robust to changing environments, clutter and sometimes dynamic scenes. Maps should therefore be very descriptive and regularly updated as the robot continues to operate. The robot must also be spatially aware of its surrounding such that tasks can be smoothly executed. Typical on-board sensors found on a home robot include RGB-D cameras, lasers, sonars, and inertial measurement units (IMU). Either a single sensor or combination of sensors is used in the building of SLAM maps for the home environment.

Search and Rescue

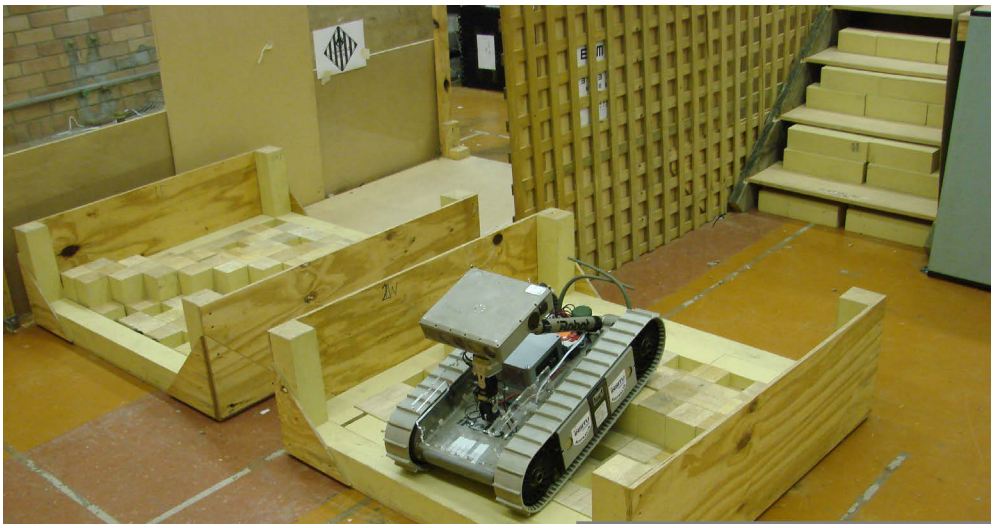


Figure 1.2: Search and Rescue Robot

In a search and rescue scenario, robots are built for their stability and rigidity. An example of a platform is the **Packbot**, seen in Figure 1.2. On top of platform structure, due to the unpredictable nature and complexities in the environment, robots are also expected to build very high fidelity 3D maps. Not only does this provide the robot with awareness about its surroundings but allows rescue workers to pin point exact locations of potential victims.

In many situations, the robot will also lose communication with its user and must rely solely on its own autonomy to navigate safely. Therefore it is extremely critical that the robot knows accurately about its own location uncertainty.

1.0.2 Brief History of SLAM

The original problem of SLAM has been traditionally formulated as a pose and feature based Bayesian network, solved by Bayesian filters such as the Extended Kalman Filter (EKF). However filtering does not scale well with larger feature sizes and can gradually become inconsistent over time, resulting in the solution becoming corrupted due to linearization errors [2]. Particle Filters [3] have also been researched with some increasingly good results, but yet again do not scale well to higher dimensions and suffer from degen-

eracy and depletion [4]. Current state of the art SLAM approach have reverted back to the original idea of optimization, first proposed in the seminal paper by Lu and Milios [5]. Due to the increased computation power of modern computers and sensor, SLAM maps of high quality have been produced for very large scale environments. In many cases, optimization approaches [6–8] are proven to be the superior method.

1.1 Motivation

Current trends have led the SLAM problem into many new directions. In this thesis we will be focusing on one of the most common and concise representations, known as feature based SLAM. This is when the joint probability of the posterior is estimated for both robot position and landmark location using the observations of landmarks and vehicle motion. When landmarks can be easily expressed as point features (absolute Euclidean position), this SLAM variant is by far the most precise way to represent the problem.

Many modern methods have chosen to solve feature based SLAM using optimization, [6, 9] [10] [11, 12]. With the success of these algorithms, also come major draw-backs. First is the problem of local minima. The optimization step involves solving a high dimensional non-linear least squares problem. This means that the global minimum of the cost function can be hard to find through conventional iterative methods. Some initial evaluations has been made in this field [13, 14]; however, more reliable methods should be further investigated.

The second is the problem of scalability. This is when the manageability of internal states and information correlation become overly large due to the length of trajectory or the total number of landmarks. One scenario is that of a large environment, where the robot must either perform a long trajectory to explore the whole map, and/or observe many features to get an accurate estimate of its location.

1.2 Contributions

This thesis is divided up into a series of contributions made over the course of my PhD candidature. The principle contributions of this thesis are as follows:

- Developing a reliable algorithm that will achieve a high *success rate* for the global minimum of a SLAM problem. The contributions include a proposal for the general framework towards reliability and a technique called Iterative Re-weighted Least Squares (IRLS) bootstrapping for the realization of this framework.
- For large scale maps, the computational complexity of maximum likelihood will grow over time. Sparse Map Joining (SMJ) counters this problem through sub-mapping and marginalization of robot poses. This approach is supported, by experimental evidence, to have *good* consistency for both 2D and 3D feature based SLAM problems. Emphasizes is also put on the evaluation, where we have compared the different sub-map joining methods (Batch, Sequential and Divide & Conquer).
- The next contribution demonstrates how a Pose Graph Representation (PGR) of feature based SLAM has the capability of also greatly improving the efficiency. The major issues tackled are information reuse and the degree of efficiency. These have been addressed through the *Multi Observation Model* and the *Key Poses* method.
- Our final contribution is the development of a fully-fledged SLAM system that is capable of solving RGB-D SLAM. By applying (IRLS), (SMJ), and (PGR), our algorithm becomes highly reliable and efficient. Due to the poor depth observability of the sensor, we also propose a new hybrid mapping strategy that allows us to map environments where other RGB-D SLAM strategies would have otherwise failed.

1.3 Publications

The following is a list of publications supporting the work presented in this dissertation.

- Hu. G, Khosoussi. K and Huang. S, "Towards a reliable SLAM back-end", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp. 37-43, Tokyo, Japan, 2013.
- Hu. G, Huang. S, Zhao. L, Alempijevic. A and Dissanayake. G, "A robust RGB-D SLAM algorithm", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp. 1714-1719, Vilamoura, Portugal, 2012.
- Hu. G, Huang. S and Dissanayake. G, "Evaluation of Pose Only SLAM", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp. 3732-3737, Taipei, Taiwan, 2010.
- Hu. G, Huang. S, and Dissanayake, G., "3D I-SLSJF: A consistent sparse local submap joining algorithm for building large-scale 3D Map", *Proceedings of the IEEE International Conference on Decision and Control (CDC) held jointly with the Chinese Control Conference (CCC)* pp. 6040-6045, Shanghai, China, 2009.

- Wang. H, Hu. G, Huang. S and Dissanayake. G, "On the Structure of Nonlinearities in Pose Graph SLAM", *Proceedings of Robotics: Science and Systems, Sydney*, Australia, 2012.
- Zhao. L, Huang. S, Yan. L, Wang. J and Hu. G, "Large-scale monocular SLAM by local bundle adjustment and map joining", *Proceedings of the IEEE International Conference on Control Automation Robotics & Vision (ICARCV)* pp. 431-436, Singapore, 2010.
- Wang. J, Hu. G, Huang. S and Dissanayake. G, "3D Landmarks extraction from a Ranger Imager Data for SLAM", *Proceedings of Australasian Conference on Robotics and Automation (ACRA)*, Sydney, Australia, 2009.
- Kirchner, N, Alempijevic. A, Caraian. S, Fitch. R, Hordern. D, Hu. G, Paul. G, Richards. D, Singh. S and Webb. S, "Robot assist-a platform for human robot interaction research", *Proceedings of Australasian Conference on Robotics and Automation (ACRA)* pp. 1-10, Sydney, Australia, 2009.

1.4 Thesis Outline

- **Chapter 2** gives the reader a preliminary understanding into the major concepts expressed in this thesis. These include how to solve SLAM through optimization and how to properly evaluate different SLAM algorithms. There will also be a discussion on related literature surrounding the contributions.
- **Chapter 3** details the first contribution, **Reliable Optimization**. In this chapter we introduce the general framework for reliability and propose Iterative Re-weighted Least Squares bootstrapping (IRLS). A comparison is made against other popular bootstrapping algorithms in SLAM literature.
- **Chapter 4** introduces a method called **Sparse Map Joining**. Through the process of pose marginalization and sub-map joining, the overall computational complexity of the problem is reduced. The evaluation of this approach is conducted on both simulated and real dataset.
- **Chapter 5** promotes the idea of **Pose Graph Representation** where only robot poses are estimated and observations are inferred through relative pose constraints. This type of representation exhibit some unique properties that will be further investigated and discussed.
- **Chapter 6** is a case study that incorporates the contributions in Chapters 3-5. Two SLAM algorithms are developed to solve the problem of RGB-D SLAM. The first will tackle the ideal conditions and the other will handle the problematic case of poor depth observability.
- **Chapter 7** concludes this thesis with a discussion on possible future work.

Chapter 2

Preliminaries

2.1 Extended Kalman Filtering for SLAM

Traditionally SLAM has been solved using filtering based methods such as the Extended Kalman Filter (EKF). The main problem arises from the non-linearity of the observation and motion models. Constantly marginalizing and linearizing at every iteration will quickly introduce optimistic estimates and lead to inconsistency. EKF SLAM has a special monotonic property where the determinant of map covariance will eventually become zero due to the landmarks becoming fully correlated [15]. Individual landmark variance will converge towards a lower bound determined only by the robot location error that existed when the first observation was made. The inconsistency of EKF results is an estimated covariance which may be less than the true covariance [2]. In addition to these properties, the computation complexity is $\mathcal{O}(N^2)$ where N is the number of features. Given that the covariance matrix is dense, the complexity quickly escalates as more and more features are added.

Many of the pitfalls in filtering are caused by the marginalization of robot poses. The more stable alternative is to keep all the poses and use an optimization based framework. The next section provides preliminary knowledge into how optimization can be formulated for SLAM.

2.2 Graph Based SLAM

A popular way to represent the SLAM problem is through graphical modeling. These techniques are often referred to as "Graph Based" or "Network Based" approaches. In literature there are many ways to representation SLAM graphically, (e.g., Factor Graphs [12, 16, 17], Bayes Trees [18] or Markov Random Fields [19]). All of which are fundamentally derived from the Dynamic Bayesian Network (DBN) [20, 21], a much generalized expression derived from probability theories, see Figure 2.1.

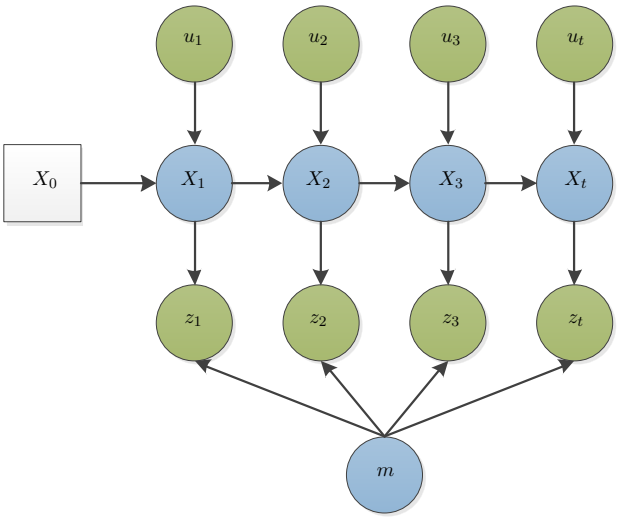


Figure 2.1: In this Dynamic Bayesian Network, the robot poses and map are hidden variables (blue), and measurements are known variables (green). The connectivity follows a recurring pattern given the transitional models defined by odometry u and observation z.

In DBN, the transition model is defined by $p(X_t|X_{t-1}, u_t)$, describing how the robot position is updated through the odometry. The map (m) probability is then updated given the robot location X_t. In DBN, only the temporal structure of the SLAM problem is being described. More modern graphical models actually highlight the spatial structure as well. In the follow section, we will solve a specific type of SLAM problem known as **Feature Based SLAM**, using a graphical model.

2.2.1 Odometry and Observation Information

A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is used to denote our graphical representation of SLAM. Where the vertices $\mathcal{V} = X^P \cup X^F$ correspond to robot and feature positions. The edges $\mathcal{E} = \hat{Z}^F \cup \hat{Z}^O$ represent the relative transforms/measurements between vertices.

The nodes or states are often represented by their absolute positions in a global coordinate frame. In the 2D case, x, y denote the position and θ is the orientation.

$$X_i^P = \begin{bmatrix} x_i^P \\ y_i^P \\ \theta_i^P \end{bmatrix} \ , \ X_m^F = \begin{bmatrix} x_m^F \\ y_m^F \end{bmatrix} \tag{2.1}$$

Here, $X_i^P \in X^P$ and $X_m^F \in X^F$.

Edges in the graph encapsulates the dependences that exist between node variables, mapped directly to the sensor data. There are multiple ways one can define the mapping, and in this thesis we will be using a very generalized form

$$\begin{aligned} \hat{Z}_{ij}^O &= g_{ij}(X_i^P, X_j^P) + \omega_{ij}^O \\ \hat{Z}_{im}^F &= h_{im}(X_i^P, X_m^F) + \omega_{im}^F \end{aligned} \tag{2.2}$$

In this expression, $\hat{Z}_{ij}^O \in \hat{Z}^O$ and $\hat{Z}_{im}^F \in \hat{Z}^F$, measurement functions $g_{ij}(\cdot,\cdot)$ and $h_{im}(\cdot,\cdot)$ are non-linear and $\omega_{ij}^O \sim \mathcal{N}(\mathbf{0}, \hat{\Sigma}_{ij}^O)$ and $\omega_{im}^F \sim \mathcal{N}(\mathbf{0}, \hat{\Sigma}_{im})$ denotes the measurement noise. The noise is assumed to be a multivariant Gaussian with covariance matrices.

$$\hat{\Sigma}_{ij}^O = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} & \Sigma_{x\theta} \\ \Sigma_{yy} & \Sigma_{yy} & \Sigma_{y\theta} \\ \Sigma_{\theta x} & \Sigma_{\theta y} & \Sigma_{\theta\theta} \end{bmatrix} \ , \ \hat{\Sigma}_{im}^F = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \tag{2.3}$$

The measurement functions describing the transformation for 2D (pose to pose and pose to feature) are

$$g_{ij}(X_i^P, X_j^P) = \begin{bmatrix} \cos(\theta_i^P)(x_j^P - x_i^P) + \sin(\theta_i^P)(y_j^P - y_i^P) \\ -\sin(\theta_i^P)(x_j^P - x_i^P) + \cos(\theta_i^P)(y_j^P - y_i^P) \\ \theta_j^P - \theta_i^P \end{bmatrix} \tag{2.4}$$

$$h_{im}(X_i^P, X_m^F) = \begin{bmatrix} \cos(\theta_i^P)(x_m^F - x_i^P) + \sin(\theta_i^P)(y_m^F - y_i^P) \\ -\sin(\theta_i^P)(x_m^F - x_i^P) + \cos(\theta_i^P)(y_m^F - y_i^P) \end{bmatrix} \tag{2.5}$$

Remark. *In feature based SLAM, $j = i+1$ for all g_{ij}. However, in this thesis we will also be looking at the pose graph problem, in which case, $\hat{Z}^F$ and X^F do not exist. The graphs become $\mathcal{E} = Z^P \cup \hat{Z}^O$ and $\mathcal{V} = X^P$, where a new edge in Z^P is also a relative pose between X_i^P to X_j^P and $j \neq i+1$. Typically Z^P is not a measured edge so $\hat{}$ is not assigned.*

Transforming Sensor Measurements

Different sensors produce different sensor models based on their operating principles. Often these models are non-linear in their state variables. Nevertheless, some of these can be transformed into the generalized form, defined by Equations (2.4), (2.5).

$$\hat{Z}_{im}^F = s(\hat{Z}_{im}^S) \tag{2.6}$$

For a range ($\mathcal{R}$) and bearing (ϕ) sensor (e.g., laser)

$$\hat{Z}_{im}^S = \begin{bmatrix} \mathcal{R}_{im} \\ \phi_{im} \end{bmatrix}, \; s(\hat{Z}_{im}^S) = \begin{bmatrix} \mathcal{R}_{im}\cos(\phi_{im}) \\ \mathcal{R}_{im}\sin(\phi_{im}) \end{bmatrix} \tag{2.7}$$

To transform the covariance we must linearize about the measured point by taking the partial derivative (J^S), Jacobian of function s and obtain

$$\hat{\Sigma}_{im}^F = (J^S)\hat{\Sigma}_{im}^S(J^S)^T \tag{2.8}$$

Gaussian Assumption

A common assumption made for SLAM problems, is that the measurement noise is approximately Gaussian. This assumption is both common and appropriate for many robotic sensors. By doing this, one can exploit various statistical approaches for solving and analysing optimization problems. Note, the flaw of this assumption is that, even if the noise of a sensor is exactly Gaussian, the non-linearity of various function (e.g., motion model) can cause the transformed result to be no longer Gaussian [16]. Therefore, it is important to realize that the Gaussian assumption is only ever a *good* approximation.

The other major flaw is on the influence of outliers. When an outlier is present in the graph, the Gaussian distribution can become skewed, causing the final solution to become corrupted by the outlier. In practice a 3 Sigma bound should always be imposed on the noise values during simulations. However, when dealing with real sensor data, this restriction, in many instances is difficult to enforce.

2.2.2 Linear Least Squares

Once we have a representation for all the nodes and edges of the graph, the next objective is to find the most likely configuration of robot poses and/or features such that the error between the measurements and the estimates are minimized. In terms of probability, this can be expressed as a Maximum Likelihood (ML) or Maximum a Posteriori (MAP). In optimization, we can treat this as being a weighted least squares problem under the assumption of independent Gaussian noise and for MAP, also independent Gaussian prior.

A common analogy for least squares, theorized by Golfarelli et al [22], expresses the graph as a springs and masses model (edges are springs and vertices are masses). The higher the uncertainty existing between two vertices, the weaker the springs become. Trying to minimize the energy required for the springs to hold all the masses together is the objective.

First, the general form of the optimization problem can be posed in the following manner

$$X^\star = \underset{X}{\operatorname{argmin}} F(X) \tag{2.9}$$

When $F(X)$ is in a particular format, then the least squares problem is formed. The aim now is to minimize the sum of the squares of all the residuals in the cost function $F(X)$. Residuals arise from the error between the theoretical measurements $\bar{Z}$ given by the estimated $X^\star$ and the actual measurements $\hat{Z}$.

The objective function is described by (Note, the following series of formulations are all derived in their general forms)

$$F(X) = \left\| \mathbf{f}(X) - Z \right\|^2 \tag{2.10}$$

A linear least squares problem exists when the function $\mathbf{f}$ can be constructed from linear combinations

$$\mathbf{f}(X) = \begin{bmatrix} f_1(X) \\ f_2(X) \\ \vdots \\ f_n(X) \end{bmatrix} \tag{2.11}$$

$$f_i(X) = a_{i,1}x_1 + a_{i,2}x_2 + \cdots + a_{i,n}x_n \tag{2.12}$$

Collecting all the co-efficient a_{ij} into matrix A, the equation becomes

$$F(X) = \left\| AX - Z \right\|^2 \tag{2.13}$$

Now to find the minimum point of function F, we will take the derivative and make it equal to zero.

$$\begin{aligned} \left\| AX - Z \right\|^2 &= (Z - AX)^T (Z - AX) \\ &= Z^T Z - Z^T AX - X^T A^T Z + X^T A^T AX \end{aligned} \tag{2.14}$$

$$\frac{\partial}{\partial X}(Z^T Z - Z^T AX - X^T A^T Z + X^T A^T AX) = 0 \tag{2.15}$$

The extrema exist when

$$(A^T A)X = A^T Z \tag{2.16}$$

Equation (2.16) is a linear equation to solve for X. One can see that the matrix $(A^T A)$ must be invertible. Directly computing $(A^T A)$ is sometimes computationally expensive to do in SLAM; however, given that the matrix is symmetric and positive definite, Cholesky or QR Decomposition [7] may be used to quickly solve $A^T AX = A^T Z$. In Cholesky decomposition $A^T A$ is converted into $L^T L$ where L is a lower triangular matrix with positive diagonal elements. QR decomposition replaces A with QR, where Q is an orthogonal matrix and R is an upper triangular matrix of A. Although QR is considered more stable, the faster of the two is Cholesky and will be the preferred approach in this thesis.

2.2.3 Non-Linear Least Squares

In linear least squares, the problem is convex, meaning that there is only ever one unique solution that exists. In fact, the functions $\mathbf{f}(X)$ when associated with SLAM is non-linear, and cannot simply form a linear combination (2.12). In our 2D measurement models (2.4), this arises from the trigonometric functions *cosine* and *sine*. However, many other complex non-linarites also exist. e.g. 3D parameterizations in Section 4.3.1 and Bundle Adjustment in Section 4.3.2.

A popular way is to use iterative approaches to solve non-linear least squares. The idea is that the solution of the maximum likelihood can be found through a series of iterations, each time re-linearizing at the point of estimation. Simply put, the state vector

$X^n \rightarrow X^{n+1}$ is continually updated until convergence (n specifying the iteration number).

An important issue surrounding this method is that one cannot guarantee the global minimum of the problem (in many cases the iterations converge to the local minimum for general non-linear least squares problems). The intuition is that the objective function for SLAM has some special properties [13] that makes the convergence relatively stable. In Chapter 3, we introduce a way to further improve the convergence.

Finding the step size associated with each iteration of non-linear least squares is a critical issue in SLAM and there exist multiple options. The most popular of which are Gradient Descent, Newton Methods and Gauss Seidel [23]. In Newton Method's, the step size is calculated based on the second order derivative, exploiting the curvature of the objective function. The most popular of the Newton Methods is Gauss-Newton, where the second derivative is not explicitly calculated but approximated by the Taylor expansion. The main reason is that the actual second order derivative can be exceptionally difficult to determine. In the other two methods, Gradient Descent and Gauss Seidel, the step size is adjusted manually at every iteration to fulfil the gradient condition. Due to this fact the convergence rate becomes very slow when it gets close to a minimum but is overall more stable. Gauss-Newton is by far the fastest of the three methods; Figure 2.2 demonstrates an simple example.

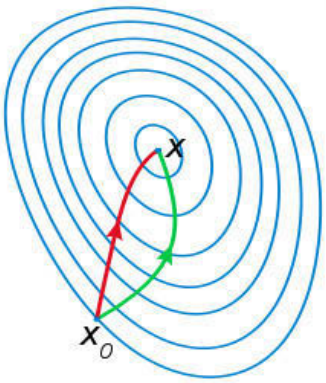


Figure 2.2: Gauss-Newton (Red) compared to Gradient Descent (Green) on a simple objective function[1].

[1]Image courtesy of http://en.wikipedia.org/wiki/Newton's_method_in_optimization

Gauss-Newton (GN)

The formulation of Gauss-Newton is as follows

$$F(X) = (Z - \mathbf{f}(X))^T (Z - \mathbf{f}(X)) \tag{2.17}$$

$$\mathbf{f}(X) \approx \mathbf{f}(X^{n-1}) + J(X - X^{n-1}) \tag{2.18}$$

First we need to linearize the non-linear function on the state X by applying the Taylor Expansion. Here X^{n-1} is the previous estimate (X^0 being the initial estimate).

The following is an example of the Jacobian matrix in SLAM when two poses X_1^P, X_2^P are observing two features X_1^F, X_2^F. In reality, this matrix is typically larger, more complex and sparser. J_i represents the partial derivatives of P (pose to pose equations) and F (pose to feature equations) with i being either pose or feature indecies.

$$J = \begin{bmatrix} J_1^P & J_2^P & 0 & 0 & 0 \\ 0 & J_2^P & J_3^P & 0 & 0 \\ J_1^F & 0 & 0 & J_1^F & 0 \\ 0 & J_2^F & 0 & 0 & J_2^F \end{bmatrix} \tag{2.19}$$

The last step is to substitute Equation (2.18) back into (2.13).

$$X^\star = \underset{X}{\operatorname{argmin}} \left\| Z - \mathbf{f}(X^{n-1}) - J(X - X^{n-1}) \right\|^2 \tag{2.20}$$

$$X^n = (J^T J)^{-1} J^T (Z - \mathbf{f}(X^{n-1}) + J X^{n-1}) \tag{2.21}$$

Often in literature, this expression is rewritten as $X^n = X^{n-1} + \Delta X$ such that only ΔX is being calculated at each iteration.

$$\begin{aligned} X^n - X^{n-1} &= (J^T J)^{-1} J^T (Z - \mathbf{f}(X^{n-1})) \\ &= \Delta X \end{aligned} \tag{2.22}$$

As we can see, one step of Gauss-Newton is very similar to linear least squares but $X^\star$ is derived iteratively until the change in $|X^{n+1} - X^n| \to 0$. However, it is not guaranteed that the matrix $J^T J$ is capable of always forming a positive definite one. Often, it will be dependent on the current linearization point X^{n-1}. Moreover, Newton methods will only converge quickly when the estimate is already close to the minimum or else the chosen descent step will be uncertain.

Hybrid Methods

To counteract the issues mentioned above, algorithms such as Levenberg-Marquardt [24] or Powell's Dog-Leg [25] were introduced. In LM, a *damping factor* λ is selected such that the matrix is always positive definite.

$$(J^T J + \lambda I) \tag{2.23}$$

This way, the λ value allows the algorithm to switch between Gradient Descent and Gauss-Newton in a gradual way. The selection of λ can be found in [26].

Powell's Dog-Leg is an alternative approach to this problem, first employed in SLAM by Rosen et al [27]. This method also combines the rapid convergence of Gauss-Newton and the stability of Gradient Descent but maintains a trust region and applies a gradual switch. Rosen conveys that the performance of Powell's Dog-Leg is better than Levenberg-Marquardt in some instances.

2.2.4 Weighted Non-Linear Least Squares

In many optimization problems, such as SLAM, there is often an uncertainty $\hat{\Sigma}$ associated with the measurement $\hat{Z}$. The additional information can then be translated into least squares by weighting each individual function. Ultimately, resulting in some edges having greater influence over others. Weighted least squares are extremely important in SLAM due to sensor noises being not independent and different sensors having different noise models.

$$X^{\star} = \underset{X}{\operatorname{argmin}} \left\| \mathbf{f}(X) - Z \right\|^2_{\Sigma^{-1}} \tag{2.24}$$

When (2.24)[2] is passed through equations (2.18) to (2.21), the resulting expression for non-linear least squares becomes

$$X^n = (J^T \Sigma^{-1} \ J)^{-1} J^T \Sigma^{-1} (Z - f(X^{n-1}) + JX^{n-1}) \tag{2.25}$$

Often in SLAM, it is much easier to express the covariance matrix in information form (measurement information Ω and graph information Λ). The relationships are given by

$$\begin{aligned} \Omega &= \Sigma^{-1} \\ \Lambda &= J^T \Omega J \end{aligned} \tag{2.26}$$

Why the information form is much more efficient will become clearer in later chapters. The information form has a major drawback associated with the recovery of the graph covariance. Due to the poor scaling of the Λ inversion, the true covariance is almost impossible to recover for larger sizes of Λ. However various tricks in literature have been proposed to approximate some marginal covariance [28–30].

[2]The notation $||e||^2_{\Sigma^{-1}}$ means $e^T \Sigma^{-1} e$

2.2.5 Least Squares for SLAM

Now we will formulate weighted non-linear least squares for SLAM. In feature based SLAM, the weighted minimization problem (2.24) is replaced by

$$X^\star = \underset{X}{\operatorname{argmin}} \sum_{ij} \left\| \hat{Z}_{ij}^O - g_{ij}(X_i^P, X_j^P) \right\|_{\Omega_{ij}}^2 + \sum_{im} \left\| \hat{Z}_{im}^F - h_{im}(X_i^P, X_m^F) \right\|_{\Omega_{im}}^2 \tag{2.27}$$

The pose graph SLAM problem is expressed in a similar fashion.

$$X^\star = \underset{X}{\operatorname{argmin}} \sum_{ij} \left\| \hat{Z}_{ij}^O - g_{ij}(X_i^P, X_j^P) \right\|_{\Omega_{ij}}^2 + \sum_{ij} \left\| Z_{ij}^P - g_{ij}(X_i^P, X_j^P) \right\|_{\Omega_{ij}}^2 \tag{2.28}$$

Finally, we will introduce a general form of these equations represented by generic edges and nodes.

$$X^\star = \underset{X}{\operatorname{argmin}} \sum_{ij} \left\| Z_{ij} - b_{ij}(X_i, X_j) \right\|_{\Omega_{ij}}^2 \tag{2.29}$$

This equations grants the possibility that each node can carry any parameterization and any edges, and function b_{ij} can define that relation between node X_i and X_j through Z_{ij}.

In SLAM, there is always an anchor point that prevents the Hessian matrix ($J^T J$) from becoming rank deficient. This is intuitive, since the map must be always fixed onto a specific coordinate system or else the map becomes ambiguous. In our case, X_0^P is typically referred to as the anchor point.

2.3 Front-End and Back-End

In the previous section, a formulation of SLAM is proposed, but in fact we have only solved SLAM from the perspective of a **Back-End**. Recently, researcher have been separating the SLAM problem into two individual parts, **Front-End** and **Back-End**, with supporting literature targeting either.

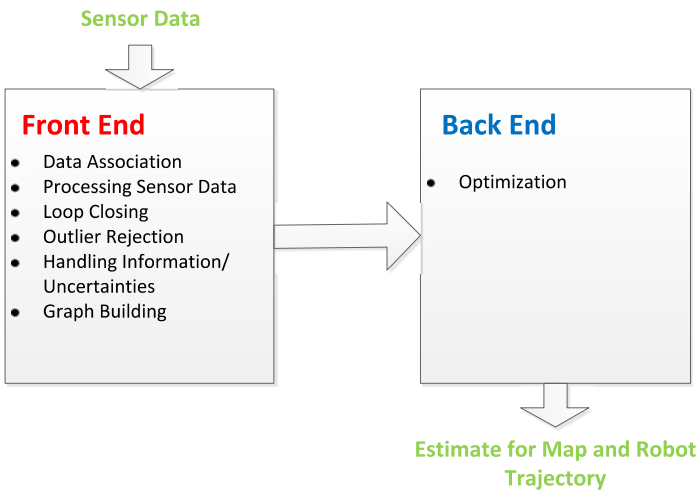


Figure 2.3: Processes involved in a Front-End and Back-End SLAM algorithm

In the Front-End, the graph structure connection are constructed. In doing so, the Front-End is very sensor dependent, making it very specific to different SLAM problems. The most difficult of all the Front-End processes is that of *Data Association*. This is where correlations must be made about the information in the sensor data. Some examples include scan matching, feature matching or place recognition. A sub category of *Data Association* is that of *Loop Closure*. This is when associations must be made after the robot has traversed back to similar locations along its trajectory. The information gained from *Loop Closure* is critical in correcting the error in both the map and robot trajectory.

An algorithm to guarantee perfect data association does not exist. Many factors corrupt the accuracy, such as ambiguity, dynamics, occlusion and structure, just to name a few.

In this thesis, our main contributions are towards the Back-End elements of SLAM. The

major advantage is that it allows us to make more generalizations across multiple SLAM problems. As for the Front-End, we have relied on the absolute certainty of simulated datasets when supporting our results. When dealing with real data, one cannot completely trust the reliability of any given Front-End estimation process.

2.4 Evaluating a SLAM algorithm

Any newly developed SLAM Back-End must be evaluated in an appropriate way. In this section, we propose some fundamental ideas and methods to determine the **Reliability**, **Consistency** and **Accuracy** of a SLAM solution. These are the χ^2 value, Normalized χ^2, Normalized Estimation Error Squared and χ^2 Ratio.

2.4.1 Chi Squared (χ^2)

The χ^2 value allows us to verify any solution obtained for a least squares problem. This number relates to the standard deviations of error between estimated and measured values. It is well-known that for the measurement model defined in (2.29), if the measurement function $b_{ij}(\cdot,\cdot)$ are linear in X_i, X_j , then

$$F^\star \triangleq \left\| \mathbf{f}(X^\star) \right\|_\Omega^2 \tag{2.30}$$

In a linear problem $F^\star \sim \chi^2_\nu$ with ν denoting the number for the degrees of freedom[3]. Therefore, the expected value and variance of $F^\star$ (over all possible measurements) are equal to ν and 2ν, respectively. It is also common to extend this result to non-linear measurement functions and use ν as the *approximate* expected value of $F^\star$ by linearizing the non-linear models around $X^\star$ [31].

The natural interpretation of χ^2 is that of the squared Mahalanobis distance (geometric distance). In our case, χ^2 lets us test varies properties of a SLAM algorithm.

[3] In general $\nu = \texttt{dim}(z) - \texttt{dim}(x)$. Here z denotes the vector of all measurements and $\texttt{dim}(\cdot)$ returns the size of the given vector [31]

A common usage of χ^2 is to monitor if Gauss-Newton has converged. This is done by examining at the χ^2 change between the current and previous iterations of non-linear least squares. In a perfect system, we should wait until the χ^2 change is zero. However, due to numerical errors, we have found that using a fixed threshold is an appropriate alternative.

$$|\chi^2_{k-1} - \chi^2_k| < \tau \tag{2.31}$$

Here k is the iteration of Gauss-Newton. From our experiments, $\tau = 0.001$ is shown to be sufficient.

2.4.2 Expected Value of χ^2 / Normalized χ^2

An approximated expected value can be used to test and/or validate the assumptions (e.g., distribution of the noise). To do this, one could compare the theoretical approximate expected value to the obtained minimum in order to verify that the obtained solution is in fact $X^\star$, if the assumptions of the problem can be trusted.

The value of ν depends on the number of edges and vertices of the network. Therefore, to evaluate the performance across different datasets, it is more convenient to normalize $F^\star$ and report the value of $\frac{F^\star}{\nu}$, commonly referred to as the **Normalized** χ^2.

If $\frac{F^\star}{\nu}$ is close to 1, then it would confirm to us that the Maximum Likelihood solution has been reached. In this thesis, the term **Reliability** is given to express how often a method can achieve the Maximum Likelihood solution, measured by the percentage of success (*success rate*).

2.4.3 χ^2 Ratio

One can also quantitatively assess the error between two χ^2 values obtained from different SLAM methods. This thesis introduces the χ^2 ratio as a measure of accuracy or *closeness* towards the maximum likelihood solution (Best possible SLAM solution).

$$\chi^2_{(\texttt{Alg})} = \left\| \mathbf{f}(X_{(\texttt{Alg})}) - Z \right\|^2_{\Omega_{(\texttt{ML})}} \tag{2.32}$$

$$\chi^2_{\texttt{ratio}} = \frac{\chi^2_{(\texttt{Alg})}}{\chi^2_{(\texttt{ML})}} \tag{2.33}$$

Where $\chi^2_{(\texttt{ML})}$ is the χ^2 at the global minimum and $\chi^2_{(\texttt{Alg})}$ is the χ^2 assessed at the final estimate of the algorithm $X_{(\texttt{Alg})}$. The closer the ratio is to 1, the less the approximation is made between the Maximum Likelihood and the proposed algorithm. In later chapters, we will see that some pre-processing is required to obtain $X_{(\texttt{Alg})}$ from the actual algorithm estimate.

It is important to recognize that the objective of a SLAM Back-End is not to obtain the ground truth but the best estimate of maximum likelihood given the measurements. The relationship between the ground truth and maximum likelihood is somewhat related to the graph structure. Olson [32] demonstrates that increasing the average node degree brings the χ^2 ratio closer to 1 as does recent works by Carlone [14] investigating other relationships (number of spanning trees, graph cycles, etc). Evaluation criterias such as Root Mean Squared Error (RMSE) or State Square Error (SSE) [33] are good indications of map quality but they should not be mistaken with the accuracy of an optimized solution which is the χ^2 ratio.

2.4.4 Normalized Estimation Error NEES

χ^2 alone cannot judge the overall consistency (how well the uncertainties encapsulates the ground truth) of the algorithm. Consistency is often considered one of the most important factors in judging how well a SLAM algorithm is behaving. An inconsistent result will be

too conservative or optimistic in its estimate.

A common check for map consistency is to evaluate the Normalized Estimation Error Squared (NEES) [34] when the ground truth is known. This insures that the solution lies within the 95 or 99 % probability region of the χ^2 distribution. The lower the NEES value, the closer the estimate trajectory is to the ground truth, in the Mahalanobis sense. A single NEES is defined by

$$NEES \triangleq (\tilde{X} - X)^T \Sigma^{-1} (\tilde{X} - X) \tag{2.34}$$

Where $\tilde{X}$ is the ground truth and NEES is χ^2 distributed with $\texttt{dim}(X)$ degree of freedom.

To be totally accurate, the term consistency actually refers to the quality of the final estimate based over multiple Monte Carlo simulations[4]. If N is the number of Monte Carlo experiments, then given that the covariance is exactly Gaussian when N approaches infinity the average $\overline{NEES}$ will tend towards the true dimension of the state.

The $\overline{NEES}$ is calculated using

$$\overline{NEES} = \frac{1}{N} \sum_{l=1}^{N} \left\| \tilde{X} - (X^\star)_l \right\|^2_{\Lambda_l^\star} \tag{2.35}$$

Where l is a particular noise seed, $\Lambda^\star$ is the final graph information obtained by the algorithm itself (2.26).

A $\overline{NEES}$ check refers to an error *match* between the covariance given by the Cramer-Rao upper bound and the estimate. For a successful check, the $\overline{NEES}$ would need to be less than a 95% or 99% χ^2 gate [34].

The drawback to NEES is that evaluations are only made on the macro level, given that a single NEES consists of all elements of the state. Frese [35] proposed, using generalized eigenvalues, to evaluate each state term individually. However, Huang et al [6] proves that

[4] One single run of NEES does not correlate the error together and does not follow the χ^2 distribution. In the book by Bar-Shalom [34], there is a quote stating, "thorough examination of a nonlinear filter is needed via (multiple) Monte Carlo runs to find out if it is consistent"

the two results are roughly equal. Even though NEES is taken on average over multiple Monte Carlo Simulations and the errors are completely different, the covariance Σ should remain roughly the same.

2.5 Related Works

Over the recent years, the standard feature based SLAM problem has evolved in many ways. We have separated the related work into two categories. First, is to improve the reliability of SLAM and the second is to obtain scalability.

2.5.1 Improving Reliability

Often the nonlinearity in SLAM can cause Gauss-Newton [36] iterations to either diverge or coverage onto a local minimum rather than finding the maximum likelihood solution (global minimum). The following are various methods proposed to improve reliability in SLAM.

Gauss Seidel

The idea of solving the non-linear problem through *relaxation* was first introduced by Duckett et al [37]. The idea is to only solve for one node at a time until each node in the graph has been updated. For unknown orientations, Duckett's formulation uses fine-tuning to change the state estimates such that the global minimum may be achieved. However, this approach is unfeasible for large datasets. Frese et al [11], extends Duckett's work to improve the efficiency, by introducing multi-level relaxation (MLR), which essentially applies Gauss Seidel relaxation at different resolutions.

Stochastic Gradient Decent

Olson [33, 38] introduced another iterative method for optimizing pose-graphs based on the Stochastic Gradient Decent (SGD). The method considers the single cost of each individual constraint and adds a dynamic learning rate which evolves during the optimization process.

The method will systematically iterate rather than randomly choose which edges to process. An effective way to control the learn rate α has been suggested by Robbins and Monro [39]. However, Olson has acknowledged that this way may be sub optimum.

Grisetti et al in TORO [40] further improved on the efficiency of SGD by exploiting the topology of the pose graph by introducing a tree based parameterization. The algorithm has also been extended into 3D, correctly handling the learning rate issue for 3D rotations. TORO is currently considered the most widely employed SGD algorithms in SLAM.

According to [33], due to the approximations involved, these two methods are unable to completely converge onto the exact maximum likelihood estimate.

Linear Approximation Graph Optimisation

Carlone et al [41] suggested that the poor maximum likelihood estimate is mainly connected to the angles and its representations, which make the problem non-linear and non-convex. Each pose-to-pose constraint in pose-graphs consists of a relative position part (x and y components in 2D) and a relative orientation part. The latter part is a linear function. Therefore, by ignoring the effect of x, y for each constraint, a suboptimal estimate for robot orientations may be obtained by solving a linear least squares problem. After obtaining an estimate of robot orientations, observations become linear in x and y and they can be approximated by solving another linear problem.

By exploiting the structure of 2D pose-graphs, LAGO is very quick and can produce a good approximate for the initial guess of Gauss-Newton. However, for the same reason, it is closely dependent on the problem formulation, and any extension to other SLAM variants, such as feature based or 3D, seems very difficult.

Furthermore, the covariance matrix of measurement (both loop closing and odometry) noise must be block diagonal. The quality of LAGO's solution depends on the ratio between the variances of the x and y parts, and the variance of the orientation part in the measurement noise. If this ratio is not close to 1, then it is impossible to ignore the effect on x and y.

Spanning Tree

A spanning tree is a connected spanning subgraph where no cycles are formed. We can consider odometry to be a special case where the edge sequences are defined by the robot motion. The spanning tree is a relatively simple concept which has become very popular in predicting initial poses in pose graphs [8]. In spanning trees a breadth-first search is performed on the poses where the root of the tree starts from the anchored pose.

A standard way to find the spanning tree is to employ a Dijkstra [42] algorithm starting from the anchor pose. While doing this search, the graph should be considered un-directed, such that measurement from one node to another may be inverted. The cost is then propagated by 1 at each edge until all nodes have been visited.

2.5.2 Overcoming Computational Complexity

In literature, many techniques have been proposed to solve the SLAM problem in an efficient way. These are further categorized into, approximate or non-approximate. Non-approximate solutions exploit the structure of the problem to simplify the mathematical operations. e.g., factorization, re-ordering, tree-parameterization. Approximate methods change the nature of the problem itself to directly reduce the complexity; however, the final solution is often sub-optimal.

Graph SLAM

This technique was first proposed by Thrun and Montemerlo in [9] and now has many variations [43–46]. The key insight is that landmarks/features can be marginalized and the reduction lets the algorithm solve complex SLAM problems where the number of features is exceptionally large.

Although this is a non-approximation approach, (the final result is equivalent to the ML), the problem must be solved iteratively, meaning that at each iteration marginalization and recovery of features must also occur. In which case, the time taken to perform these calculations may greatly devalue the effectiveness of the algorithm. Graph SLAM does not

explicitly discard any information which also suggests that the sparseness is not necessarily reduced. Therefore, during factorization, there is no guarantee of any speed improvement.

Thrun and Montemerlo [9] discuss that under the constraint of local features and a small number of iterations, Graph SLAM can be the most effective.

Smoothing and Mapping (SAM)

This technique designed by Dellaert et al [12], takes advantage of the sparsity of the information matrix. Instead of using the full information matrix, they keep the matrix in the square root form, which can be applied in conjunction with QR Factorization. Doing so, the algorithm avoids calculating the matrix $J^T J$, greatly reducing complexity. As a result of the algorithm solving the Maximum a Posteriori, it is also solving a full SLAM problem. However, considering that smoothing is performed on the entire trajectory and features, the overall complexity will still grow over long periods of time.

To overcome growing complexity, Kaess et al [30] introduces *incremental smoothing and mapping* (iSAM), further exploring SAM by incrementally building the map, without rebuilding the data structures or constantly re-linearizing. Another advantage of iSAM is that it allows access to the marginal covariance matrix for data association given that the full covariance is otherwise impossible to recover.

Sub-mapping algorithms have also been explored through the SAM framework, (Tectonic SLAM) [47]. Each sub-map is solved locally using SAM and then each relative pose between sub-maps are optimized individually. Linearization is only performed on the sub-map level so the solution is an approximation on ML.

TreeMaps

Tree Maps reduce the dimensionality by affecting the edges of a graphical tree, essentially pruning the edges to achieve an approximate solution.

Paskin et al, Thin Junction Tree Filter (TJTF) [48], tries to maintain a tree where the overall probability distribution is kept within a bound by marginalizing distributions

along the edges. This method is based on a filtering framework which we know can be inconsistent over time.

Frese et al (TreeMap) [49], divides the environment up into local and sub regions through the concept of binary trees. However, this representation has in-consistency associated with the propagation of linearization points. The result is poor uncertainty estimates and artifacts in the final solution.

2.6 Summary

In this chapter, we have provided a formulation for feature based SLAM. Starting from a graphical model, we finally arrive at an optimization problem solved through weighted non-linear least squares. The equations that have been presented provide some fundamental concepts needed for subsequent chapters.

Furthermore, we have built a formal understanding for evaluating a SLAM algorithm through Normalized χ^2, Normalized Estimation Error Squared, and χ^2 Ratio.

In the second part of this chapter, some related works were discussed. This involved the two major issues investigated in this thesis. Improving the reliability of SLAM (Section 2.5.1) and achieving scalability by reducing computational complexity (Section 2.5.2).

In the following chapters, we will be proposing new ways to manage these issues, and this will form the main contributions of this thesis.

Chapter 3

Reliable Optimization

3.1 Introduction

When a robot needs to traverse through its environment and plan tasks, the accuracy of the environment map and robot pose is critical. The best estimate one can obtain is that of the maximum likelihood, mentioned in Section 2.2. However, due to the existence of non-linearity in the measurement functions, there is no guarantee of the global convergence when using Gauss-Newton. It is common for the Back-End to converge onto a local minima or even worse diverge. One key insight is the importance of the initial estimate ($X^{(0)}$) (see Section 2.2.3). A naive approach would be to use the robot odometry and concatenate each relative measurement to estimate an initial state. e.g., *dead reckoning*. However, the sensor noise will accumulate error very quickly causing the final estimate to be very far from the actual ML solution. The pursuit of finding a good initial estimate has always been an attractive research topic in SLAM, commonly recognised in literature as *bootstrapping*.

In general, any method which obtains an initial estimate for non-linear least squares can be considered a bootstrapper, even *dead reckoning*. In this chapter we will demonstrate the effectiveness of Iterative Re-weighted Least Squares as an idea for bootstrapping.

Through stringent evaluations, and comparing to other existing bootstrappers, the initial guess for our method has a greater *success rate* of being inside the basin of attraction of the global minimum. Also, not only is this bootstrapped solution easy to apply, but easily

generalized across many different SLAM variants.

3.2 A General Framework for Reliable Optimization

As mentioned earlier, to solve the SLAM problems (2.29) with Gauss-Newton, it is crucial to have an initial estimate that is *sufficiently close* to the global minimum. The basin of attraction may depend on various factors such as the noise level or graph structure [14]. Our main motivation comes from a very intuitive idea to smooth the optimization process such that the final solution is guaranteed to converge.

The idea is to solve a sequence of intermediate optimization problems $\mathcal{P}_1, \ldots, \mathcal{P}_N$ such that:

(**C1**) The initial guess obtained from a spanning tree is within the basin of attraction of the global minimum of $\mathcal{P}_1$.

(**C2**) The solution of each problem $\mathcal{P}_k$ is within the basin of attraction of the global minimum of the next problem $\mathcal{P}_{k+1}$.

(**C3**) The solution of the final problem $\mathcal{P}_N$ is within the basin of attraction of the global minimum of Equation 2.29 (Bootstrapping).

The existence of the spanning tree in **C1** relates to the fact that the objective function value for that subgraph is exactly zero, since there are no cycles present in this graph, there are also no residual errors. Therefore, each spanning tree has only one unique solution. Using the spanning tree as the initial guess for $\mathcal{P}_1$ and using the solution of $\mathcal{P}_k$ as the initial estimate in $\mathcal{P}_{k+1}$, we will eventually arrive at the maximum likelihood estimate. **C2** must then be continually satisfied until the initial value is within the basin of attraction of the original least squares problem, condition **C3**.

Graduated Non-Convexity and Simulated Annealing

This idea is somewhat related closely to Graduated Non-Convexity [50], shown in Figure 3.1. Starting from a convex problem, it is possible to define a series of sub-problems which become progressively and smoothly more non-convex, until finally arriving at the original problem. The region where the global optimum lies, for a current problem, must include the point where the global optimum is from a previous problem. The use of graduated non-convexity is common practice in image processing (e.g., image blurring [51]).

Simulated Annealing [52] also has close relationships with our idea. Rather than smoothing the function, the algorithm randomly perturbs the current estimate by a fixed amount, arriving at the same result. The disadvantage is that the randomness of sampling can greatly increase the complexity.

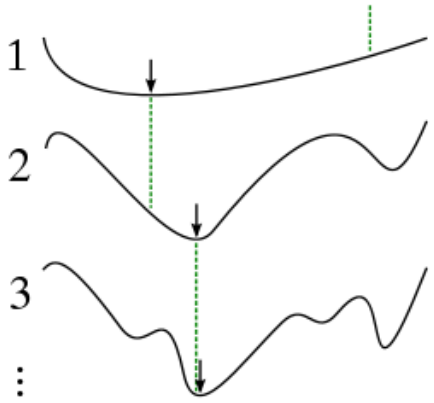


Figure 3.1: Example of graduated non-convexity

In our framework, we are not smoothing or adding randomness to the problem but rather finding individual problems where the global minimums are easy to find. The smoothness in each function may come naturally but is not the primary focus.

3.2.1 Defining a Sequence

Designing a finite sequence $\{\mathcal{P}_k\}_{k=1}^{N}$ according to general framework (3.2), can be very difficult. The reason being that there are currently no techniques available to determine if an estimate is inside the basin of convergence. However, similar to graduated non-convexity the algorithm does not have to strictly meet all three criterias to yield good results. Therefore, it is possible to design our own sequence $\{\mathcal{P}_k\}_{k=1}^{N}$ based on some approximate methods and support the heuristic principles with extensive Monte Carlo studies. It is also critical to test for a broad range of possible scenarios, which are realistic in terms of noise, graph structure and graph types.

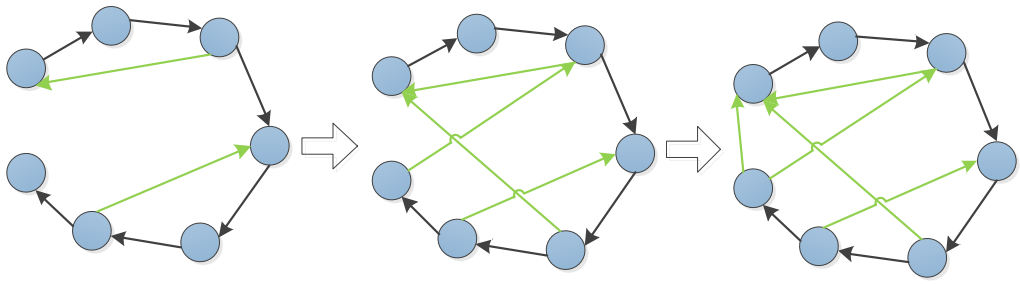


Figure 3.2: A basic sequence of edges which slowly increases the difficulty of the problem. The blue circles represent the pose nodes and the green lines are the loop closure edges.

Let us first start by analyzing a basic sequence $\{\mathcal{P}_k\}_{k=1}^{N}$: $\mathcal{P}_k$, of subgraphs, $\mathcal{G}_k = (\mathcal{V}, \mathcal{E}_k)$ (i.e., $\mathcal{E}_k \subset \mathcal{E}$). Here each graph has a subset of edges from the original graph. We can then enforce an addition condition, that for any point in the sequence k we have $\mathcal{E}_k \subset \mathcal{E}_{k+1}$. Subgraphs are solved using the formulations specified in Equation (2.29). Figure 3.2 shows an example of such a sequence.

From an intuitive standpoint, each new edge introduced in $\mathcal{E}_k$ (i.e., $\mathcal{E}_k \setminus \mathcal{E}_{k-1}$) should be *sufficiently consistent* with the edges in $\mathcal{E}_{k-1}$. In other words, closing a (new) big loop in $\mathcal{E}_k$ might violate (**C2**), and consequently increase the *risk* of converging to a local minima. On the other hand, if we start to ignore measurements, then we will never reach an optimum subgraph which will satisfy (**C3**). A well rounded algorithm must somehow *gradually* incorporate edges by controlling sudden *influence* but also produce a final solution which

exists to satisfy the third condition.

Alternative Sequences

Many alternative algorithms can be viewed as special (or extreme) cases of our general framework. For example, incremental methods such as iSAM [30] fit well within this framework: the $\mathcal{P}_k$ in incremental approaches is the non-linear least squares problem that arises in the process of obtaining the maximum likelihood estimate, at time step k (using all the available edges up to that time). Sub-mapping techniques such as the one mentioned in the Chapter 4, are also an example of this. The sequence is defined by the joining sequence of the local maps.

3.2.2 Iterative Re-weighted Least Squares

Our proposal is to apply a heuristic approach, which is to assign (additional) weights $w_{ij}^{(k)}$ to each individual edge.

$$X_{(k)}^{\star} = \underset{x}{\operatorname{argmin}} \sum_{i,j} w_{ij}^{(k)} \left\| (\hat{Z}_{ij} - b_{ij}(X_i, X_j)) \right\|_{\Omega_{ij}}^2 \tag{3.1}$$

It is important to note the difference in this new approach compared to the aforementioned ideas posed in the previous section. Now, all the edges are being used in the optimization process (3.1) rather than only a subset. Therefore, the influence of edges follow a specific continuous function compared to being equal or having no influence. Doing so, we are able to achieve this *gradual* transition. Figures 3.3 shows the new weighted sequence.

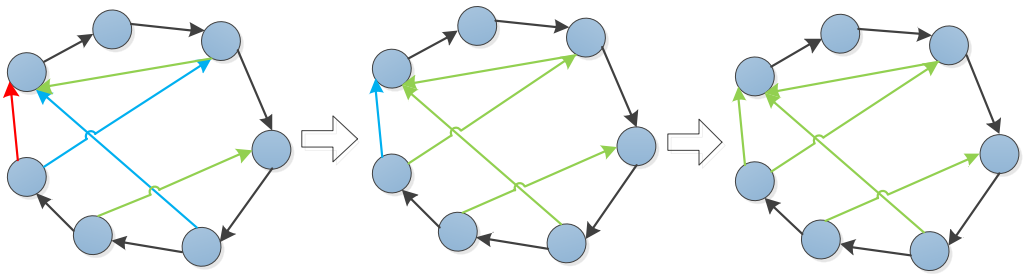


Figure 3.3: A weighted sequence. Red: low weight, Blue: moderate weight, Green: high weight.

One way to add weights is to apply Iterative Re-weighted Least Squares (IRLS) in M-estimation. Typically this technique has been used in removing outliers in optimization problems where large residuals are considered *potential* outliers. In our case we recognize large residuals as being inconsistent with the initial guess within each intermediate optimization problem. The real advantage of IRLS is in its simplicity, which does not deviate much from the original Gauss-Newton equations.

3.2.3 Formulation for IRLS

Starting from the original non-linear least squares problem and using the square root of $\Omega_{ij} = \Omega_{ij}^{\frac{1}{2}}\Omega_{ij}^{\frac{1}{2}}$ we can define the normalized error vector as

$$e_{ij} \triangleq \Omega_{ij}^{\frac{1}{2}}(\hat{Z}_{ij} - b_{ij}(X_i, X_j)) \tag{3.2}$$

Then we propose to use the following $X^{(0)}_{(\texttt{irls})}$ as the initial estimate

$$X^{(0)}_{\text{irls}} = \underset{x}{\operatorname{argmin}} \sum_{i,j} \rho(r_{ij}) \tag{3.3}$$

Where $r_{ij} \triangleq \|e_{ij}\|^2$ denotes the ℓ_2-norm of e_{ij}. $\rho(\cdot)$ is a new cost function which will have different convergence properties. This function is known as the M-estimator and will be further discussed in Section 3.2.4.

Because of this new function $\rho(\cdot)$, the optimization problem in Equation (3.3) may seem difficult to solve. However, it can be reformulated into IRLS and solved with only slight modifications to Gauss-Newton.

Using the notations proposed in [53] we start by computing the gradient of the objective function in (3.3) w.r.t. x, setting this to zero

$$\sum_{i,j} \psi(r_{ij}) \frac{\partial r_{ij}}{\partial x_t} = \mathbf{0}, \quad \text{for } t = 1, \ldots, n \tag{3.4}$$

Where $\psi(r) \triangleq d\rho(r)/dr$ is known as the *influence* function of the M-estimator (The importance of this will be explained in the following section).

Now by defining the *weight* function $w(r) \triangleq \psi(r)/r$ we can rewrite (3.4) as

$$\sum_{i,j} w(r_{ij})\, r_{ij} \frac{\partial r_{ij}}{\partial x_t} = \mathbf{0}, \quad \text{for } t = 1, \ldots, n \tag{3.5}$$

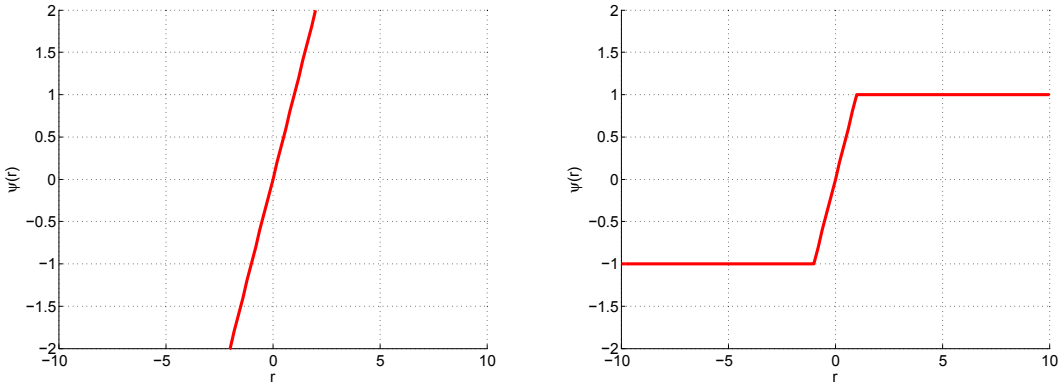
The LHS of (3.5) can be inferred as the gradient of the cost function in the k^{th} iteration and is similar to the following IRLS problem

$$\text{minimize} \sum_{i,j} w(r_{ij}^{(k-1)}) r_{ij}^2 \tag{3.6}$$

Where $r_{ij}^{(k-1)}$ is the residual computed using the latest estimate. For each iteration k we need to re-compute new weights according to the residuals and solve (3.6) using Gauss-Newton (iteratively). Ideally, weights should only be updated after Gauss-Newton has converged but in practice, we observed that comparable results may be obtained by performing only a single Gauss-Newton iteration for a fixed set of weights.

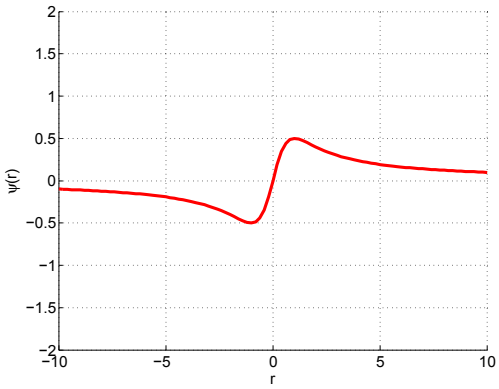
3.2.4 M-Estimators

It should be clear that the M-estimator function $\rho(\cdot)$ is a critical part of our algorithm. Furthermore, the influence function $\psi(\cdot)$ is directly linked to the convergence of IRLS and the final result. Therefore, it is of utmost importance to understand its properties. Figure 3.4 and Figure 3.5 depict the influence and weights functions of some popular M-estimators.

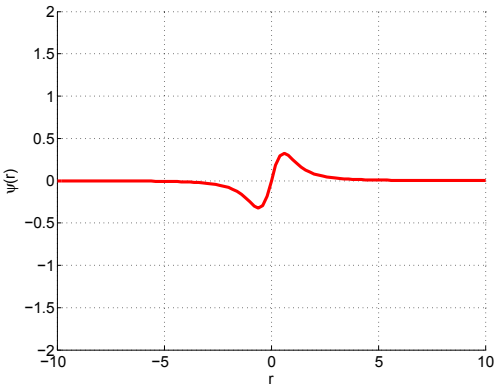


(a) `L2 (Least Squares)`

(b) `Huber`

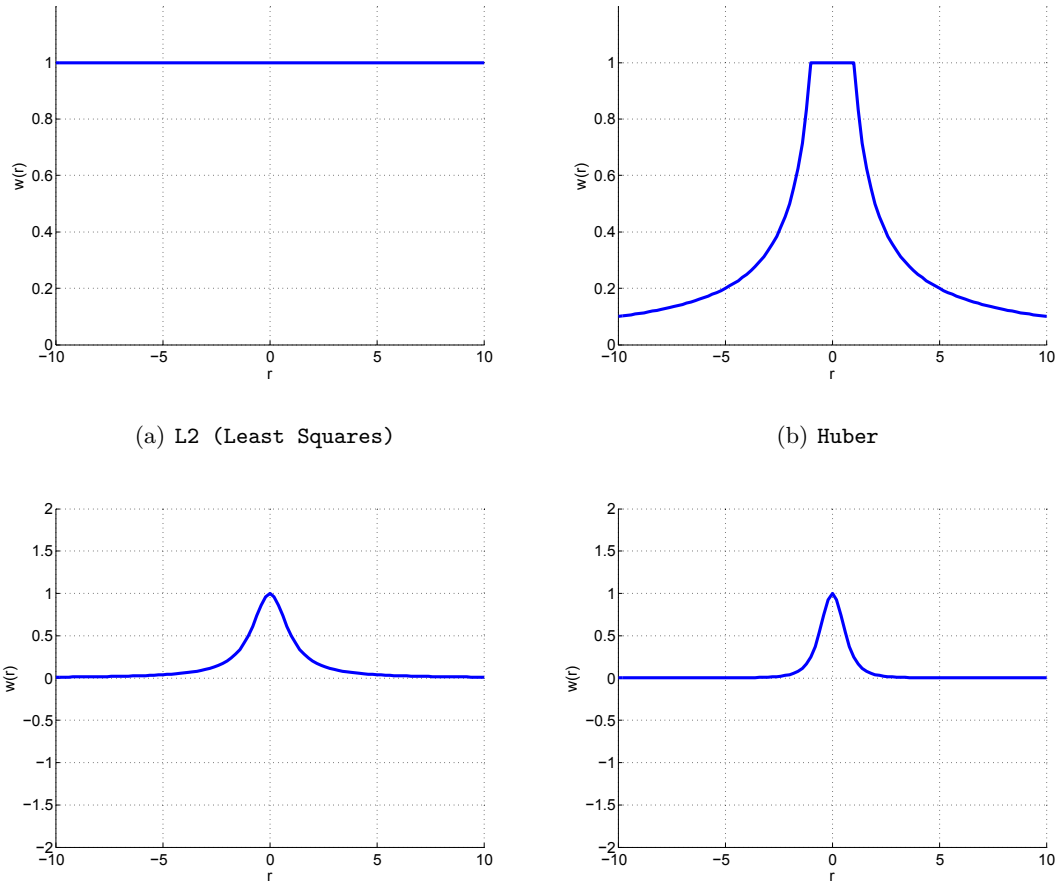


(c) `Cauchy`



(d) `Geman-McClure`

Figure 3.4: Influence functions



(a) `L2 (Least Squares)`

(b) `Huber`

(c) `Cauchy`

(d) `Geman-McClure`

Figure 3.5: Weight functions

Firstly M-estimators can be categorized based on their influence function. From the influence, a weight value is assigned to each graph edge, following the formations given in Section 3.2.3.

We can see that the influence function in normal least squares is un-bounded, which is the reason why outliers have large and detrimental influence on the result or in our case with no outliers, exhibit poor convergence properties. The other M-estimators can be split into two categories, re-descending or non-re-descending.

Huber is a non-re-descending estimator. The influence keeps increasing until it reaches a fixed value $\lim_{|r|\to\infty} \psi(r) = \pm\alpha$. Re-descending estimators such as Cauchy and Geman-McClure has tails which descend towards zero $\lim_{|r|\to\infty} \psi(r) = 0$ causing large residuals

to be de-emphasized. Intuitively speaking the re-descending influence functions fit better with the ideas posed in the general framework in Section 3.2. The reason being that some edges may be de-emphasized to a point where their influence has little to no effect on the overall problem.

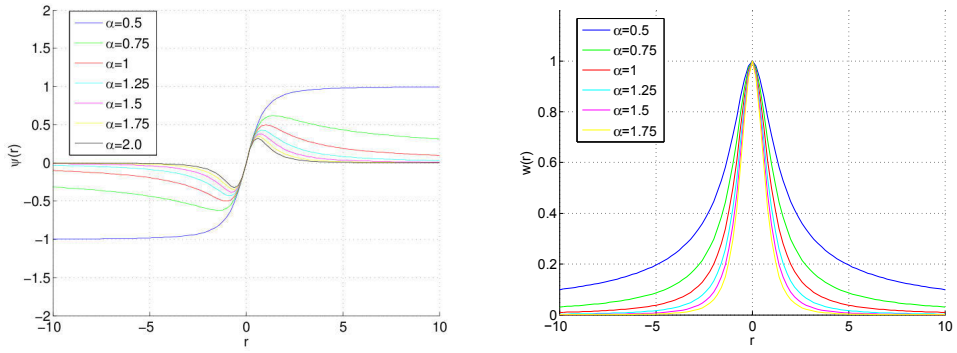
The descent rate is another important factor in selecting the right M-estimator. As can be seen in Figure 3.4, the Geman-McClure influence function descends much faster than the Cauchy influence function. If the descent phase is too quick, then loop-closing edges might not have a chance to affect the bootstrap solution and we might violate (**C3**). On the other hand, descending too slowly will reduce the reliability and violate (**C2**).

3.2.5 Generalized Influence Function

To find the most suitable influence function, we define the following family of re-descending influence functions which incorporates both descent rate and the idea of re-decadence into one equation.

$$\psi_\alpha(r) \triangleq \frac{r}{(1+r^2)^\alpha} \quad , \quad w_\alpha(r) \triangleq \frac{1}{(1+r^2)^\alpha} \tag{3.7}$$

When the variable $\alpha = 1$, we will achieve the same influence function as the Cauchy. If we increase α, then the tail will re-descend faster, becoming exactly Geman-McClure [53] at $\alpha = 2$. Going the other way we start to see that the function no longer becomes re-descending. At $\alpha = 0.5$, we arrive at a similar influence function to Huber, and by reducing α even further to zero, this is equivalent to solving a normal least squares (ℓ_2-norm). We have represented this visually in Figure 3.6.



(a) `Influence Function` (b) `Weight Function`

Figure 3.6: Generalized influence and weight function

In general, the *optimal* value for α is dependent on the nature of the measurements and the noise level. We conducted a series of Monte Carlo simulations to test the behavior of M-estimators for different values of α. In these experiments the final solution of the M-estimator is bootstrapped to the Gauss-Newton algorithm. The *success rate* in Table 3.1 depicts how often the final value of the (least squares) objective function is less or equal to the best achievable value if we start from the ground truth $\tilde{X}$.

Table 3.1: Success Rate (%) of $\psi_\alpha(\cdot)$ in 100 Monte Carlo Simulations for Different α Values for the Manhattan Dataset

STD $(\sqrt{\Sigma_{xx}}, \sqrt{\Sigma_{yy}}, \sqrt{\Sigma_{\theta\theta}})$	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 1$	$\alpha = 1.25$	$\alpha = 1.5$	$\alpha = 1.75$	$\alpha = 2$
(0.1,0.1,0.1)	54%	94%	100%	98%	78%	14%	4%
(0.2,0.2,0.2)	16%	94%	98%	88%	22%	0%	0%
(0.3,0.3,0.3)	0%	58%	74%	60%	2%	0%	0%

Table 3.1 is the result of 100 Monte Carlo simulations of the Manhattan dataset at three noise levels. According to our results, the best performance happens when α close to 1 (i.e., Cauchy M-estimator [53] with constant $c = 1$). Smaller α values tend to only perform poorly under larger noises and large values of α are shown to be very unreliable. Note that $\alpha = 0$ is not tested since this is equivalent to the original non-linear least squares problem. In the rest of this thesis, when we have referred to IRLS then the default α value is 1.

3.2.6 Initial Influence

Although the Cauchy M-estimator has the best performance, simply applying a fixed influence works well on **C2** and **C3** but does not easily address **C1**. From our experiments, we have found that the number of iterations needed to converge from the first initial set of weights to be greater than 5. This suggests that the initial value is not necessarily close to the first global minimum.

To solve this problem, we propose a set of initial weighted steps that will change the value of α. We begin by setting $\alpha = 2$ (exactly that of Geman-McClure) then decrease the parameter by 0.5, apply a single iteration of IRLS at each α level, and finally stop when $\alpha = 1$ (our original IRLS function).

It is important to note that this has no bearing on solving a new M-estimator problem since only one iteration of IRLS is performed for each new weight set. This can only be considered a smooth way to find a *better* start point for IRLS.

3.2.7 Stopping Condition for IRLS

It can be difficult to determine at which iteration point the current IRLS iteration is *close* enough to satisfy condition (**C3**). The only way is to identify if the solution is fully converged.

A typical way to identify convergence of an iterative algorithm is to check the changes in the objective function values. However, it can be difficult to set fixed thresholds for a non-normalized value. As a result, an unnecessary number of iterations would be performed if the threshold is too conservative. A better way would be to monitor the weight values directly, given that our M-estimator function have already been normalized between 0 to 1. The following equation will be used as our stopping criteria

$$\frac{1}{m}\sum_{i,j}\left[w(r_{ij}^{(k)}) - w(r_{ij}^{(k-1)})\right]^2 < \epsilon \tag{3.8}$$

Where m is the total number of edges and ϵ is the threshold value. We have found $\epsilon = 0.01$ to work well for our datasets.

3.2.8 Summary of IRLS algorithm

The algorithm that summarizes the approach of IRLS bootstrapping, is described in detail in Appendix B.

3.3 Evaluation Criteria

To evaluate IRLS, first we must find a reasonable way to evaluate for *reliability*. The follow section explains our approach.

3.3.1 Benchmark Solution

Gauss-Newton(GN) can be considered the standard approach in SLAM Back-Ends, although other techniques such as gradient descent, LM or PDL [25] might perform better under some conditions. However, we have found that GN is sufficient for the complexities in our parameterizations.

A popular way to find the global minimum is to run a Gauss-Newton algorithm initiated from the ground truth `GT+GN`. The `+GN` postfix implies that the result of the bootstrapper has been used as the initial value of GN. This method is arguably the most reliable way to obtain the global minimum if the ground truth is available. Therefore, if a bootstrapper were to achieve the same or lower value as `GT+GN`, then its solution can be considered as the ML estimate $X^\star$.

The values used in our comparisons are the normalized χ^2, described in Section 2.4.4.

3.3.2 Noise Conditions

In simulation the noise level is critical in determining the difficulty of the problem. Noise levels were chosen carefully to cover all of the possible cases. The noise in the covariance matrices are tested for:

1. A scalar multiple of the identity matrix.

2. Diagonal and $\Sigma_{xx} = \Sigma_{yy} \neq \Sigma_{\theta\theta}$.

3. Full with correlation coefficients:
   $\frac{\Sigma_{xy}}{\sqrt{\Sigma_{xx}\Sigma_{yy}}} = \frac{\Sigma_{x\theta}}{\sqrt{\Sigma_{x\theta}\Sigma_{\theta\theta}}} = \frac{\Sigma_{y\theta}}{\sqrt{\Sigma_{yy}\Sigma_{\theta\theta}}} = 0.5.$

3.3.3 Monte Carlo Evaluation

It is of utmost importance to conduct a Monte Carlo study when evaluating SLAM algorithms. Failure to do so may result in drawing wrong conclusions about the reliability of those methods. A single successful realization of noise can be misleading from an overall perspective. In other words, without a proper Monte Carlo study, one is not able to fully take into account the possible failures of a SLAM algorithm.

3.3.4 Success Rate

The success rate refers to the number of Monte Carlo simulations in which the bootstrapper was able to obtain the benchmark solution, see Section 2.4.4. In addition, we can also look at the average normalized χ^2 to assess if there are large deviations in the final result. If the algorithms were not able to achieve the global minimum, the χ^2 value will give us a sense of how close the converged local minimum is from the global minimum in terms of objective function.

3.4 Experiment and Results

The following is a detailed list of experiments performed to test the reliability of our IRLS bootstrapping.

For our simulation we have generated 50 Monte Carlo noise instances. The units used in Tables 3.2 and 3.3 for the standard deviation of noise are in metres ($\sqrt{\Sigma_{xx}}$, $\sqrt{\Sigma_{yy}}$) and radian ($\sqrt{\Sigma_{\theta\theta}}$). For each case we report the *success rate* of different methods. Additionally, we report (in parenthesis) the average of the obtained normalized χ^2 over Monte Carlo simulations to verify if the obtained solution is in fact the global minimum $X^\star$.

3.4.1 IRLS for Feature Based Graphs

First, we will test the success rate for a simulated feature based graph. The initial pose estimates are obtained by concatenating the odometry. The initial position of features are define by the first/initial observation to that feature, transformed from their respective pose estimate.

The odometry covariance Σ^O and features observation covariance Σ^F are set to similar values. The simulation trajectory is that of Loop 2D. For simulation details see Appendix A.

Table 3.2: Success Rate (%) for 50 Monte Carlo Simulations (Average normalized χ^2)

STD ($\sqrt{\Sigma^O_{xx}}, \sqrt{\Sigma^O_{yy}}, \sqrt{\Sigma^O_{\theta\theta}}$) ($\sqrt{\Sigma^F_{xx}}, \sqrt{\Sigma^F_{yy}}$)	`Odometry+GN`	`IRLS+GN`	`GT+GN`
(0.05,0.05,0.05) (0.05,0.05)	100 (0.985)	100 (0.985)	(0.985)
(0.2,0.2,0.2) (0.2,0.2)	70 (1.211)	98 (1.006)	(0.988)
(0.3,0.3,0.3) (0.3,0.3)	34 (1.256)	82 (1.033)	(0.986)
(0.1,0.1,0.1) (0.1,0.1) (correlated)	90 (1.508)	100 (0.984)	(0.984)
(0.2,0.2,0.2) (0.2,0.2) (correlated)	70 (1.306)	98 (0.999)	(0.989)

From Table 3.2, we can see a significant increase in success rate when applying our *IRLS+GN* algorithm. Odometry as an initial value quickly becomes unreliable as the

noise value increases. However, the local minimum is not very far from the global minimum according to the normalized χ^2.

3.4.2 IRLS on Pose Graphs

There exists bootstrapping algorithms to compare against if we re-formulate IRLS for pose graphs. This allows us to conduct a much more rigorous evaluation.

We have chosen the following three popular bootstrappers.

1. `TORO+GN`, A popular implementation of tree based parameterization adapted from the concept of SGD [1].
2. `LAGO+GN`, Linear Approximation, solving a linear problem for angles first [2].
3. `ST+GN`, Spanning Tree, the initial guess is obtained from a Dijkstra breadth first search [8]

To make the evaluation fair, initial values have all been set to be the odometry. For Tree-based Network Optimizer, **TORO** and Linear Approximation for Pose Graph Optimisation, **LAGO**, we have used the author's original implementations. Rather than generating our own dataset we choosen to use two popular datasets, Manhattan3500 (**MAN**) by Olson [33], and City10000 (**CITY**) by Grisetti [40], that have been tested in various pose graph SLAM papers. The number of Monte Carlo trails is kept constant and noise is chosen such that the variance of odometry Σ^O and loop closure constraints Σ^P are kept the same.

[1] http://www.openslam.org/toro.html
[2] http://www.lucacarlone.com/index.php/resources/software

Table 3.3: Manhattan3500 Dataset, Success Rate (%) for 50 Monte Carlo Simulations (Average normalized χ^2)

STD $(\sqrt{\Sigma_{xx}}, \sqrt{\Sigma_{yy}}, \sqrt{\Sigma_{\theta\theta}})^{O,P}$	IRLS+GN	LAGO+GN	TORO+GN	ST+GN	Odometry+GN	GT+GN
0.05,0.05,0.05	100 (1.00)	50 (6.64)	100 (1.00)	100 (1.00)	50 (5.63)	(1.00)
0.1,0.1,0.1	100 (1.00)	2 (1.29e+4)	100 (1.00)	90 (1.07)	2 (5.04e+5)	(1.00)
0.2,0.2,0.2	98 (0.99)	0 (1.28e+4)	70 (1.14)	10 (2.11e+5)	0 (2.05e+3)	(0.99)
0.3,0.3,0.3	80 (90.03)	0 (4.60e+3)	40 (2.0e+2)	0 (1.05e+4)	0 (1.07e+3)	(0.99)
0.05,0.05,0.2	96 (1.04)	0 (1.16e+4)	74 (1.33e+2)	28 (3.21e+5)	0 (6.32e+5)	(1.00)
0.2,0.2,0.05	100 (1.00)	46 (1.62e+3)	100 (1.00)	100 (1.00)	0 (3.81e+2)	(1.00)
0.1,0.1,0.1 (correlated)	86 (1.15e+3)	4 (1.63e+6)	0 (3.80e+3)	90 (1.03)	0 (8.28e+6)	(1.00)
0.2,0.2,0.2 (correlated)	78 (1.008)	0 (1.81e+4)	0 (2.15e+3)	12 (6.51e+5)	0 (1.26e+6)	(0.99)

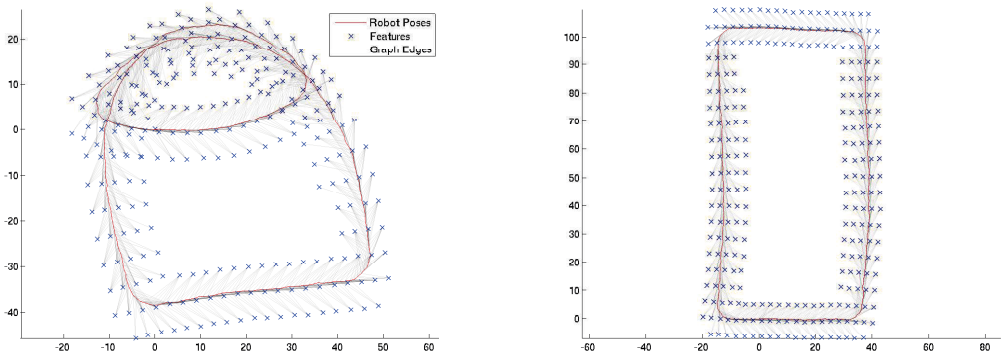
Table 3.4: City10000 Dataset, Success Rate (%) for 50 Monte Carlo Simulations (Average normalized χ^2)

STD $(\sqrt{\Sigma_{xx}}, \sqrt{\Sigma_{yy}}, \sqrt{\Sigma_{\theta\theta}})^{O,P}$	IRLS+GN	LAGO+GN	TORO+GN	ST+GN	Odometry+GN	GT+GN
0.05,0.05,0.05	96 (1.04)	2 (54.74)	94 (1.11)	100 (1.00)	0 (2.78e+2)	(1.00)
0.1,0.1,0.1	100 (1.00)	0 (22.16)	82 (1.069)	94 (1.01)	0 (68.52)	(1.00)
0.2,0.2,0.2	98 (1.00)	0 (7.50)	8 (1.23)	0 (1.29)	0 (19.3)	(1.00)
0.3,0.3,0.3	92 (1.00)	0 (4.49)	0 (1.30)	0 (1.57)	0 (8.82)	(1.00)
0.05,0.05,0.2	70 (1.31)	0 (16.44)	20 (8.17)	4 (2.59)	0 (50.86)	(1.00)
0.2,0.2,0.05	100 (1.00)	0 (96.90)	94 (1.026)	100 (1.00)	0 (361.32)	(1.00)
0.1,0.1,0.1 (correlated)	92 (1.03)	0 (32.05)	0 (43.99)	96 (1.01)	0 (112.65)	(1.00)
0.2,0.2,0.2 (correlated)	90 (1.00)	0 (8.32)	0 (40.95)	0 (1.43)	0 (24.60)	(1.00)

It is clear that even for pose graphs the *success rate* for IRLS+GN is high. In fact, according to Table 3.3 and 3.4, it is the only algorithm capable of handling all noise levels. Except for two cases, the average of the normalized χ^2 for IRLS+GN is always close to 1, although its *success rate* might be lower than 100%. For the other methods, ST+GN has a high *success rate* only when the noise is small, while TORO+GN completely fails when the noise components are correlated. LAGO+GN, in general, performs poorly especially if the noise components are correlated and/or $\Sigma_{\theta\theta}$ is larger than Σ_{xx} and Σ_{yy} (this behavior was predicted in Section 2.5.1). Finally, the total failure of Odometry+GN underlines the fact that a reliable bootstrapper is important in SLAM.

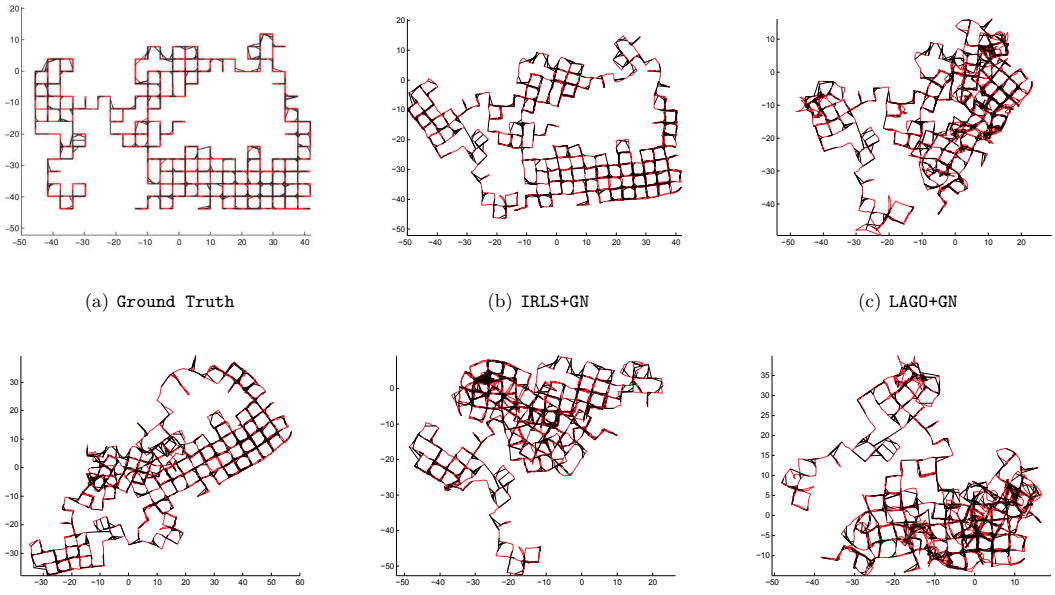
3.4.3 Resulting Maps

Now let us take a look at the resulting maps for a given noise instance



(a) `Odometry+GN` (b) `IRLS+GN`

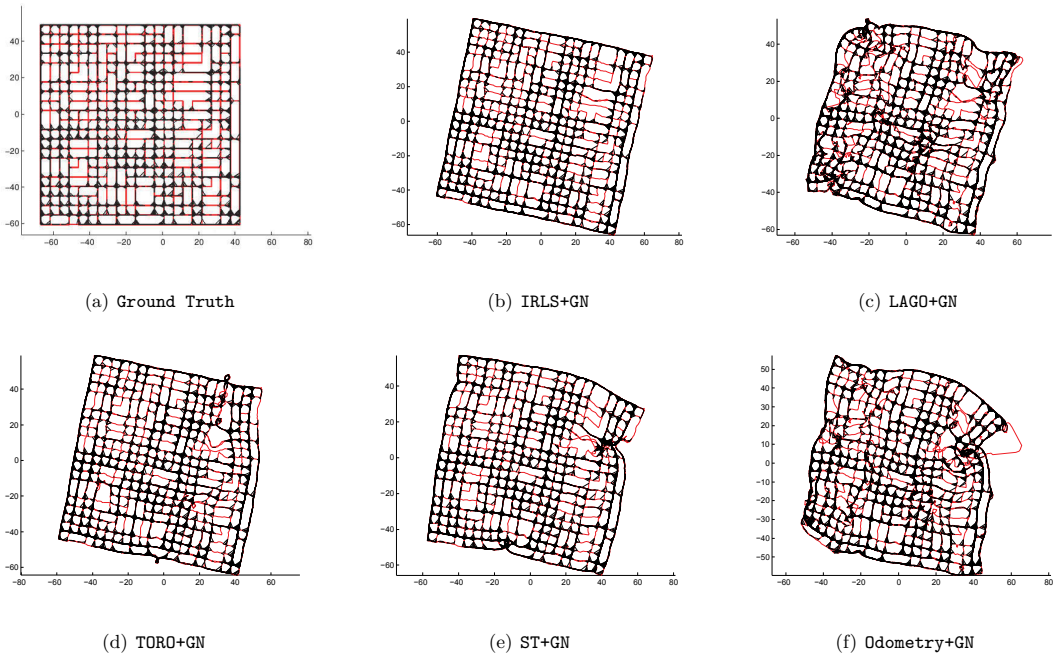
Figure 3.7: A single run at noise level (0.2,0.2,0.2) with correlated noise Loop 2D



(a) `Ground Truth` (b) `IRLS+GN` (c) `LAGO+GN`

(d) `TORO+GN` (e) `ST+GN` (f) `Odometry+GN`

Figure 3.8: A single run at noise level (0.2,0.2,0.2) with correlated noise Manhattan3500



(a) `Ground Truth` (b) `IRLS+GN` (c) `LAGO+GN`

(d) `TORO+GN` (e) `ST+GN` (f) `Odometry+GN`

Figure 3.9: A single run at noise level (0.2,0.2,0.2) with correlated noise City10000

When visually inspecting Figures 3.7, 3.8 and 3.9, it is apparent that our method produced the most consistent maps. The obtained solutions by `IRLS+GN` are exactly the maximum likelihood estimates. Note that for a fixed noise level, the usefulness of the maps may depend on graph connectivity. As can be seen in City 10000, if the graph connectivity is better (e.g., higher average node degree), then the maximum likelihood estimate would actually be closer to the ground truth [32].

3.5 Discussion

3.5.1 Importance of Noise Correlation

In many practical scenarios, the noise components may be correlated. For instance, if feature matching is used to obtain a relative pose measurement or if the motion model belongs to a non-holonomic vehicle, then the correlation will naturally exist between the x, y and θ components of the noise.

3.5.2 Computation Time

The computation time of each technique for a single run is reported in Table 3.5. For a fair comparison of computation time between different techniques we have to make sure that all of the algorithms are converging to the true maximum likelihood estimate $X^\star$. It is very difficult to generate a series of (realistic) Monte Carlo simulations with this property. Therefore, we decided to report the computation time of each method for a single run. Unlike Table 3.3, here we do not generate our noise samples; instead we use the original (noisy) datasets.

Table 3.5: Computation Time for a Single Run of Each Bootstrapper on an Intel corei5-2400 Running at 3.10GHz

Dataset	Bootstrapper	# Iterations	Time(s)	# +GN Iterations	GN Time (s)	Total Time (s)
MAN	IRLS (single GN)	9	0.19	3	0.07	0.26
	IRLS	48	0.95	3	0.07	1.03
	TORO	100	2.56	3	0.07	2.63
	LAGO	-	0.06	3	0.07	0.13
	ST	-	≈ 0	4	0.09	0.09
	Odometry	-	≈ 0	6	0.13	0.13
CITY	IRLS (single GN)	11	1.62	3	0.48	2.10
	IRLS	64	9.14	3	0.48	9.62
	TORO	100	11.95	3	0.48	12.43
	LAGO	-	0.35	2	0.33	0.68
	ST	-	≈ 0	4	0.63	0.63
	Odometry	-	≈ 0	7	1.07	1.07

Empty entries in Table 3.5 denote N/A cases (e.g., number of bootstrapping iterations in `Odometry`). In Table 3.5, "`IRLS` (single `GN`)" refers to the case in which, for each set of weights, instead of solving that intermediate non-linear least squares fully using GN, we only perform a single GN step. As mentioned in Section 3.2.3, in practice the performance of "`IRLS` (single `GN`)" is close to the original `IRLS` method, while being much faster. According to Table 3.5, our proposed method does not increase the computational complexity per iteration of a standard SLAM Back-End (adding the weights computation step); only that the total number of iterations will change. The total computation time of our proposed method is comparable to that of alternative methods.

3.6 Summary

In this chapter we have demonstrated that the IRLS can be considered as a reliable bootstrapping technique. This idea is also supported by the framework detailed in Section 3.2. After discussing a number of heuristic realizations for that *ideal* framework, we illustrated the connection between this problem and robustness against outliers. Then we decided to use M-estimators with re-descending influence functions as our IRLS bootstrapper.

Extensive Monte Carlo studies revealed that the proposed method can outperform the existing methods with comparable computation time. Furthermore, unlike the alternative methods, the proposed algorithm is capable of handling different noise levels, graph types and formulations. From our findings, we can now say that the search for the global minimum using IRLS bootstrapping is much better than that of standard Gauss-Newton with odometry as the initial estimate. The next problem lies in the scalability of the graph which causes the optimization problem to become inefficient. In the next two chapters we will introduce two techniques that attempt to resolve this issue.

Chapter 4

Sparse Map Joining

4.1 Introduction

In the previous chapter it was demonstrated that a range of general SLAM optimization problems can be improved through Iterative Re-Weighted Least Squares (IRLS), the next attractive topic of research is to solve large scale SLAM problem more efficiently. In many applications of feature based SLAM, the number of poses and features can grow considerably *large*, due to the size and complexity of the environment in which the robot must operate. If we were to be optimizing for all poses and features, then the process of optimization would not scale particularly well. The problem lies within the computation burdens associated with re-linearization at each iteration of non-linear least squares and the number of iterations needed for convergence. It is possible to reduce the number of iterations with IRLS, but the time per iteration increases as more states are added to the problem.

Our first proposed idea, for managing scalability, is to marginalize robot poses through a method called Sparse Map Joining (SMJ). The original idea stems from previous work done by Huang et al [54], Iterative-Sparse Local Submap Joining Filters (I-SLSJF). The main contribution in this chapter, is the added extension into 3D and the ability to handle multiple observation models.

In this chapter we will first provide a methodology of SMJ and 3D SMJ in Sections 4.2

and 4.3. Then, in Section 4.4 we will conduct a series of evaluations to provide evidence for the three major evaluation criterias: consistency; accuracy; and efficiency. Finally, in Section 4.5, we pose a few discussion topics on the properties of SMJ.

4.2 Sparse Map Joining

The main idea behind SMJ is that a series of consistent sub/local maps are first obtainable from the original feature based data set. Then, after marginalizing out the poses, the process of joining the maps is a simple solution to a standard Gauss-Newton optimization. In the process of marginalization, the final result will only be an approximate estimate on the original feature based SLAM problem (see Equation (2.27)).

The interesting aspect about this approach is that not all the poses are completely removed causing the information matrix to remain sparse. In which case, selecting an optimum number of local maps can significantly improve the computation time. Furthermore, the way in which one solves the map joining problem, can have an effect on the convergence properties of non-linear least squares.

The steps involved in SMJ are local map building, marginalization, map fusing and optimization.

4.2.1 Building Local Maps

It is important to understand that there are currently no simple methods available to recover lost information once the local maps have been joined. Therefore, we are under the assumption that each local map is already consistent before the joining process. To better meet this assumption, our local maps are built using standard ML, detailed in Section 2.2.5, rather than EKF or EIF, as suggested by I-SLSJF. The convergence of non-linear least squares optimization in our local maps are further improved by reliable optimization (Refer to Appendix B), $X^{L(0)}_{\texttt{irls}} \rightarrow (X^L)$.

Below is the formulation for optimizing a local map, where F is given by Equation

(2.27).

$$(X^L)^\star = \underset{X^L}{\operatorname{argmin}} F(X^L) \tag{4.1}$$

The state vector and measurements vectors are encapsulated by

$$\begin{aligned} X^L &= (X^P \cup X^F) \\ Z^L &= (\hat{Z}^O \cup \hat{Z}^F) \end{aligned} \tag{4.2}$$

Odometry and observation functions are kept standard, according to equations (2.4) and (2.5). Once each local map is optimized, the map is represented by its final state vector X^L and its associated information matrix Λ^L.

When splitting up the data into local maps, the structure (edges and nodes) of the local maps should also determine the sizes and regions of each map. We have taken the basic approach, dividing the overall map up equally. The techniques takes trajectory size and splits it based on the required number of local maps n. Features that lie within each division of the trajectory are segregated accordingly. In sub-mapping literature [55], researchers have typically created local maps based on metric separation (e.g., splitting at every 5 metres along the trajectory). However, we feel that an optimal splitting strategy should also take into consideration reliability of local map convergence as well as graph structure. In our evaluation, we are only relatively assessing the quality of SMJ under differing numbers of local maps, justifying our simplistic splitting approach.

4.2.2 Marginalization

The next step involves marginalizing the poses out of each local map. First we define the new state representation

$$X^{L(\mathcal{M})} = [X_k^P, X_1^F \cdots X_m^F]^T \tag{4.3}$$

$X^{L(\mathcal{M})}$ now encapsulates all the features and the last poses X_k^P for each local map. To

marginalize the corresponding information matrix $\Lambda^L \rightarrow \Lambda^{L(\mathcal{M})}$, Schur Complement is applied (See Appendix E). Note, the process of marginalization only applies on the original set of local maps. This is a pivotal concept for later methods with additional local map subsets (sub-maps).

The final stage of marginalization is to treat all our local maps as local observations $\hat{Z}^{L(\mathcal{M})}$ from the anchor pose of each local maps. The measurements are assumed to have a zero-mean Gaussian noises with information matrix $\Omega^{L(\mathcal{M})}$ from the marginalized local map information matrix $\Lambda^{L(\mathcal{M})}$.

$$\begin{aligned} Z^{L(\mathcal{M})} &= X^{L(\mathcal{M})} \\ \Omega^{L(\mathcal{M})} &= \Lambda^{L(\mathcal{M})} \end{aligned} \tag{4.4}$$

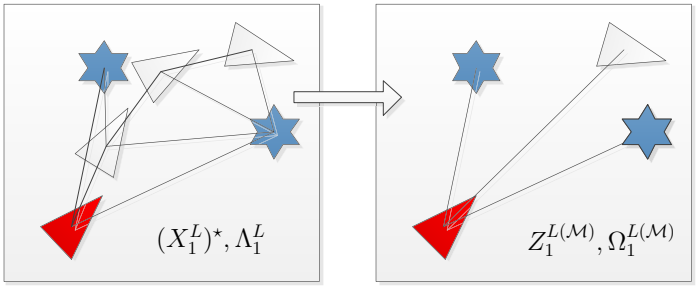


Figure 4.1: Marginalization of a local map. Blue: Features, Red: Anchor pose

4.2.3 Fusing Local Maps

In fusing, the global optimization problem represented is in the *global map* space, defined by $X^G, Z^G, \Omega^G, \Lambda^G$. The observations Z^G are formed by combining the global locations of the local maps X^L. Due to the correlations that exist within each sub map, IRLS bootstrapping is no longer possible. Therefore, we have resorted to using standard Gauss-Newton (2.27) for optimization. Consequently, the initial estimate now becomes more important for convergence. Figure 4.2 illustrates the graph setup for the process of map fusion.

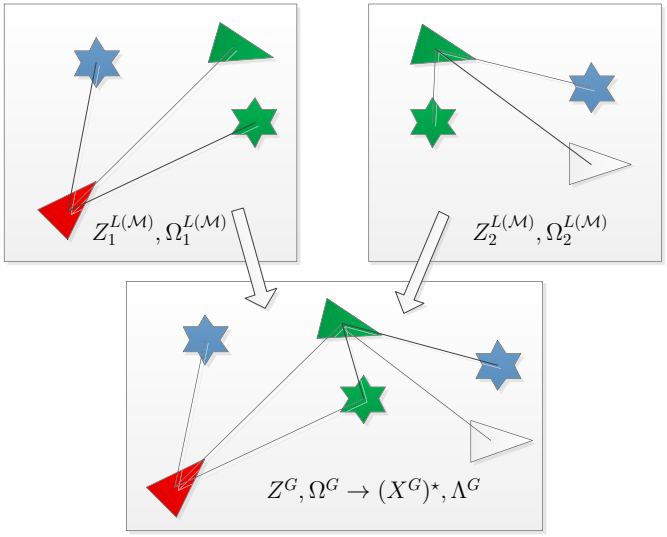


Figure 4.2: Two local maps are joined by applying Gauss-Newton on a global map. Green: Common nodes

The behavior of the joining algorithm is affected by the order in which local maps are fused. In the next sections we will be introducing three ways in which the order can be changed. We call these map joining methods.

Batch Optimization (BO)

Batch optimization combines all local maps as a whole, by concatenating the end maps at the points where they join (last pose of each local map). In the end only one optimization step is needed. Given that the number of local maps stays the same, out of all the map joining methods, BO makes the least approximations.

The downside is that BO can suffer from poor initial estimates and is prone to failure like any non-linear least squares algorithm. By solving multiple optimization problems we can try to alleviate this issue. See Appendix C.1 for the algorithm.

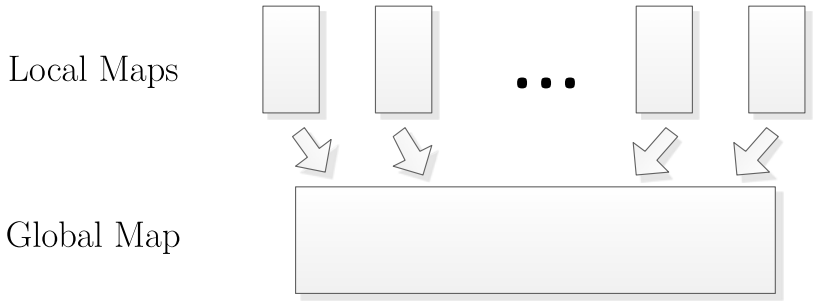


Figure 4.3: Batch Optimization joins all local maps in one step

Sequential Optimization (SO)

In sequential optimization, there is only ever one *global map* being solved at a time, which means a new local map is added after each optimization step. The idea is to link each local map sequentially according to the trajectory of the robot. The current global maps and its associated variables are re-calculated after each addition of a new local map.

Now that intermediate optimization are being solved along the trajectory, a better estimate of the current states is obtained after every joining phase. This is somewhat tied to the reliability condition introduced in Chapter 3, Section 3.2 and similar to the concepts of incremental methods such as iSAM [30]. Intuitively, SO should be less prone to convergence errors, but no means immune. See Appendix C.2 for the Sequential Optimization algorithm.

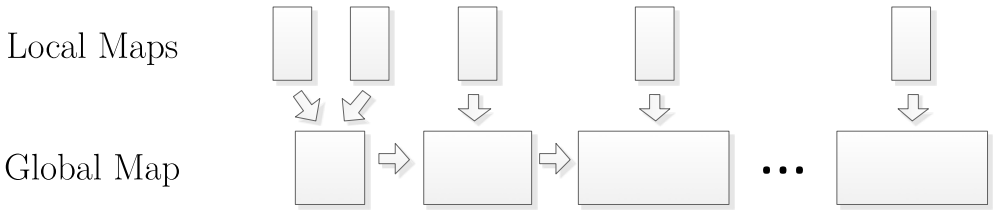


Figure 4.4: Sequential Optimization links each local map to a growing global map. Arrows indicate the joining of two local maps

Divide & Conquer Optimization (DCO)

Divide & Conquer is an alternative ordering for map joining. Commonly used in EKF SLAM [56], the idea is that neighboring local maps are joined in separate optimization steps and slowly merge into one globally consistent map. This can be done in a tree like fashion, where the intermediate sub maps are a set of smaller optimization problems with each new level in the tree improving the initial estimate for the next level. See Appendix C.3 for the Divide & Conquer algorithm.

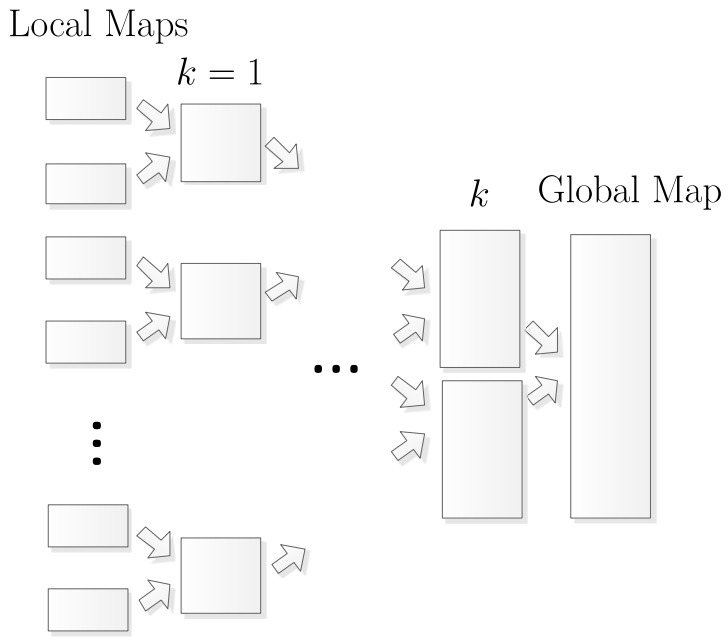


Figure 4.5: Divide & Conquer Optimization combines the initial local maps into sub maps in a tree like fashion. Then tree eventually ends with a single global map. k indicates the depth of the tree

4.2.4 SMJ Algorithm

To better understand the SMJ on a whole, we have provided a top level algorithm to go from a feature based graph SLAM problem into a SMJ solution. See Appendix C.4.

4.3 3D Sparse Map Joining

SMJ is further extended for 3D SLAM problems by expanding the state and redefining the measurement functions to cater for new parameterizations.

4.3.1 Standard 3D Range and Bearing

For range and bearing models, the modifications are simple. Poses and features are represented by

$$X_i^P = \begin{bmatrix} x_i^P \\ y_i^P \\ z_i^P \\ \alpha_i^P \\ \beta_i^P \\ \gamma_i^P \end{bmatrix} \qquad X_m^F = \begin{bmatrix} x_m^F \\ y_m^F \\ z_m^F \end{bmatrix} \tag{4.5}$$

Where (x, y, z) are positions and (α, β, γ) are Euler angles for rotation. Now we can define the new measurement function $h^{(3d)}$ as

$$h_{im}^{(3d)}(X_i^P, X_m^F) = Rot(\alpha_i^P, \beta_i^P, \gamma_i^P) \begin{bmatrix} x_m^P - x_i^F \\ y_m^P - y_i^F \\ z_m^P - z_i^F \end{bmatrix} \tag{4.6}$$

Where Rot is the rotation matrix consisting of the three angles in $SO(3)$. In addition to this we also have a new odometry function, $g_{ij}^{(3d)}$ defined as

$$g_{ij}^{(3d)}(X_i^P, X_j^P) = \left\{ \delta x_{ij}, \delta y_{ij}, \delta z_{ij}, \delta \alpha_{ij}, \delta \beta_{ij}, \delta \gamma_{ij} \right\} \tag{4.7}$$

The function $h^{(3d)}$ is reused for the x, y, z components of $g_{ij}^{(3d)}$ and the rotations $\delta\alpha_{ij}, \delta\beta_{ij}, \delta\gamma_{ij}$ are obtained from the rotation matrix relationship, given that $Rot(\cdot, \cdot, \cdot)$ is orthogonal

$$\left\{\delta x_{ij}, \delta y_{ij}, \delta z_{ij}\right\} = h^{(3d)}(X_i^P, X_j^P) \tag{4.8}$$

$$Rot\left\{\delta\alpha_{ij}, \delta\beta_{ij}, \delta\gamma_{ij}\right\} = Rot(\alpha_i^P, \beta_i^P, \gamma_i^P)Rot(\alpha_j^P, \beta_j^P, \gamma_j^P)^T \tag{4.9}$$

Decomposing the Euler angles from a rotation matrix $Rot(\cdot,\cdot,\cdot)$ is very standard, for more detail see Appendix F.

When solving non-linear least squares, we simply refer back to (2.27) and replace $h_{im} \to h^{(3d)}$ and $g_{ij} \to g^{(3d)}$.

Remark. *Our decision to use Euler parameterization over Manifold methods [57], comes from the fact that the observation information matrix is no longer block diagonal. In Manifold, as defined by Hertzberg [58], there is an assumption that information can be normalized into an identity matrix by changing the measurement value. However, in the case of SMJ this cannot be performed when matrices are correlated. Although we are aware that using Euler parameterization can lead to singularity in rotation for the 3D feature based graph SLAM, the occurrences are uncommon in our problems.*

4.3.2 SMJ for Bundle Adjustment(BA)

Many feature based SLAM problems exist where there is no odometry and the sensor is only capable of measuring the bearings to features. Such problems are seen in computer vision, commonly referred to as Bundle Adjustment [26]. In BA formulation, the state variable stays the same while the observation function $h_{im}^{(3d)}$ changes. This new function is typically defined by a re-projection equation

$$h_{im}^{(3d)}(X_i^P, X_m^F) = Proj(X_i^P, X_m^F, K) \tag{4.10}$$

Where K represents the camera calibration matrix. The residual is now the re-projection error between the observed pixel location $\hat{Z} = [u_{im}, v_{im}]^T$ and the theoretical value $\bar{Z} =$

$h_{im}^{(3d)}(X_i^P, X_m^F)$. When applying SMJ to this problem, each local map is only correct up to a scale. Unless the scale is corrected for all local maps, the final solution can become consistent to a single scale. Zhao et al supported this point in [59]. Through the process of map joining, the scale drift can also be minimized.

Local maps in BA must be solved using LM over GN due to the conditioning of bearing only being very sensitive to the linearization point. Sparse Bundle Adjustment (SBA) [60] is a popular algorithm that applies BA while exploiting the sparse block structure to achieve efficiency.

4.3.3 Dual-Observation Model Joining

An interesting idea for SMJ, is not to only focus on a fixed observation model, but switch between multiple ones. An instance of this is to combine a range and bearing local map with bearing only local map. One such application is given in Chapter 6 Section 6.5, (RGB-D SLAM).

The equation itself is only a slight modification on (4.6)

$$h_{im}^{(3d)}(X_i^P, X_m^F) = S(Rot(\alpha_i^P, \beta_i^P, \gamma_i^P) \begin{bmatrix} x_m^P - x_i^F \\ y_m^F - y_i^P \\ z_m^P - z_i^F \end{bmatrix}) \tag{4.11}$$

where S represents a new scale variable. Considering scale does not affect rotation, only the translation part of $h_{im}^{(3d)}$ is modified. The same is repeated for $g_{ij}^{(3d)}$. The graphical representation of the dual-observation joining is represented in Figure 4.6.

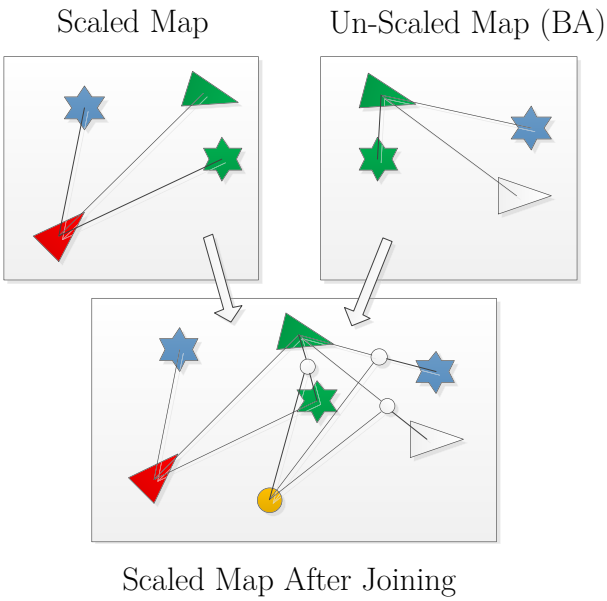


Figure 4.6: An additional variable (Orange: Scale) is introduced onto the marginalized local maps, unlike the other variables in the graph, scale is just a scalar value without spatial context. The Jacobian and observation functions are modified respectively. This variable is then jointly optimized with the rest of the poses and features.

As a side note, the properties of the global map must be considered when using SO. The scale of the global map is not recoverable until a metrically accurate map is joined.

4.4 Evaluation

Due to marginalization in local maps, SMJ is an approximate solution to the original Maximum Likelihood problem. We will assess the information loss through a series of experiments, evaluating according to the ideas outline in Section 2.4.

Rather than consider noise level as a possible changing parameter, we want to focus on the primary variables in our algorithm, which is the number of local maps n and the joining method. Therefore, we propose to only use a series of fixed simulated scenarios with fixed noise uncertainties in our evaluations. These are detailed in Appendix A. The following is a summary of noise variances that were chosen and a summary of the simulated datasets. Notice that our simulations are targeted towards both scale and graph connectivity. The generated noises are bounded by 3σ to prevent outliers.

Table 4.1: Simulation Noise

Dimension	Odometry Noise STD	Observation Noise STD
2D	$(\sqrt{\Sigma_{xx}}, \sqrt{\Sigma_{yy}}, \sqrt{\Sigma_{\theta\theta}})$= (0.05,0.05,0.05)	$(\sqrt{\Sigma_{xx}}, \sqrt{\Sigma_{yy}})$ = (0.05,0.05)
3D	$(\sqrt{\Sigma_{xx}}, \sqrt{\Sigma_{yy}}, \sqrt{\Sigma_{zz}}, \sqrt{\Sigma_{\alpha\alpha}}, \sqrt{\Sigma_{\beta\beta}}, \sqrt{\Sigma_{\gamma\gamma}})$ = (0.05,0.05,0.05,0.05,0.05,0.05)	$(\sqrt{\Sigma_{xx}}, \sqrt{\Sigma_{yy}}, \sqrt{\Sigma_{zz}})$ = (0.05,0.05,0.05)

Table 4.2: Summary of Datasets

Dimension	Dataset	no. Poses	no. Features	no. Feature Observations	Average Node Degree
2D	Circle	61	36	2196	44.8
	Loop	301	76	785	4.15
	Manhattan	3500	36	3871	2.18
3D	Circle	61	64	3904	61.96
	Loop	301	381	3096	9.06
	Sphere	701	149	4416	29.6

4.4.1 Consistency using NEES

When we consider consistency, the Normalized Estimation Error Squared (NEES) is tested, refer back to Section 2.4.4.

NEES is typically performed on the algorithm in question, which in our case is Sparse

Map Joining. Therefore, to do a comparison with Maximum Likelihood, we must first *deduce* the state vector $X_{(\texttt{ML})}$ and information matrix $\Lambda_{(\texttt{ML})}$ into the same state represented as SMJ.

First, the non-associated states are removed from the ML solution (states that do not exist in the SMJ formulation). The information matrix is updated using the Schur Complement (Appendix E), so to be consistent with the reduced state vector $X^{(\mathcal{M})}_{(\texttt{ML})}$.

$$\begin{aligned} X_{(\texttt{ML})} &\rightarrow X^{(\mathcal{M})}_{(\texttt{ML})} \\ \Lambda_{(\texttt{ML})} &\rightarrow \Lambda^{(\mathcal{M})}_{(\texttt{ML})} \end{aligned} \tag{4.12}$$

By applying marginalization, a NEES of ML and SMJ can be directly compared to the 95% χ^2 Gate calculated at $\texttt{dim}(X)$ degree of freedom, refer to Section 2.4.4. Note, during a NEES test it is wise to reject any Monte Carlo simulations that are above the NEES value for any given ML result, $(X_{(\texttt{ML})}, \Lambda_{(\texttt{ML})})$ on Equation (2.35). The reason being that SMJ is only an approximation and can never achieve a better solution to ML, so it would be pointless to test SMJ on an already inconsistent ML result.

One issue faced when calculating NEES in 3D is the handling of Euler angles due to their non-uniqueness. There exists several rotation sequences of the 3 angles to achieve a specific 3D rotation. One cannot simply take the difference between the ground truth angles and the estimated angles to calculate the NEES. Also transforming angles into a unique definition such as quaternions would require further linearization. As a result, for 3D problems, we have decided to ignore checking of consistency in rotation and only focused on the x, y, z elements. This is simply done by applying an addition marginalization step to remove the Euler angles from the state.

$$\begin{aligned} X^{3D(\mathcal{M})}_{(\texttt{ML}(xyz))} &\rightarrow X^{3D(\mathcal{M})}_{(\texttt{ML})} \\ \Lambda^{3D(\mathcal{M})}_{(\texttt{ML}(xyz))} &\rightarrow \Lambda^{3D(\mathcal{M})}_{(\texttt{ML})} \end{aligned} \tag{4.13}$$

Experiments

The experiments are conducted for all the datasets mentioned in Appendix A. The changing parameter is the number of local maps that the trajectory is divided up into, $n = 4, 10, 20$. The number in parentheses corresponds to the percentage of successful Monte Carlo runs where the NEES value lie within the theoretical 95% χ^2 Gate.

Table 4.3: 2D NEES Test on 50 Monte Carlo Runs (Percentage of Runs Within 95% χ^2 Gate)

MAP	Fusing Method	$n = 4$	$n = 10$	$n = 20$
CIRCLE	Gate	**106.39**	**126.57**	**159.81**
	BO	79.09(100)	96.25(100)	124.63(98)
	SO	79.09(100)	96.25(100)	124.63(98)
	DCO	79.09(100)	96.25(100)	124.63(98)
	ML	79.09(100)	96.25(100)	124.63(98)
LOOP	Gate	**194.8**	**214.47**	**246.96**
	BO	235.88(44)	191.64(84)	215.39(90)
	SO	485.79(30)	1862(30)	2882(5)
	DCO	424.66(42)	360.04(44)	421.26(39)
	ML	166.79(98)	182.54(100)	210.42(98)
MANHATTAN	Gate	**106.39**	**126.57**	**159.8135**
	BO	127.92(64)	191.78(40)	332.84(10)
	SO	161.89(56)	196.58(33)	466.49(6)
	DCO	161.91(56)	598.27(6)	1354(2)
	ML	82.24(96)	100.63(96)	132.44(94)

Table 4.4: 3D NEES Test on 50 Monte Carlo Runs (Percentage of Runs Within 95% χ^2 Gate)

MAP	Fusing Method	$n = 4$	$n = 10$	$n = 20$
CIRCLE	Gate	**238.32**	**257.75**	**290.03**
	BO	196.60(98)	212.03(98)	242.48(100)
	SO	196.60(98)	212.03(98)	242.48(100)
	DCO	196.60(98)	212.03(98)	242.48(100)
	ML	196.60(98)	212.03(98)	242.49(100)
LOOP	Gate	**1247.6**	**1284.8**	**1346.8**
	BO	1165.9(96)	1194.4(100)	1250(96)
	SO	1466.6(47)	1541(46)	1832.3(36)
	DCO	1217(98)	1245.1(83)	1308.7(81)
	ML	1155.2(98)	1190.6(98)	1249.1(96)
SPHERE	Gate	**509.94**	**528.91**	**560.48**
	BO	460.98(98)	475.69(100)	504.57(100)
	SO	536.99(88)	632.79(94)	504.59(100)
	DCO	469.04(94)	521.75(98)	504.68(100)
	ML	456.57(100)	474.39(100)	503.16(100)

From Tables 4.3 we can observe that the consistency between SMJ and ML is identical when the graph is fully connected (Circle). For less connected graphs, increasing the number of local maps can either improve (Loop) or degrade (Manhattan) the probability of consistency (number in parenthesis). The trend is that by increasing n there is an overall reduction in the consistency, especially when we analyze SO and DCO methods. We can associate this to the approximations imposed by the additional linearization at each joining step.

It is safe to state that BO can produce the most consistent results for SMJ and by increasing the total number of local maps inconsistency may occur depending on how well connected the graph is.

4.4.2 Accuracy using χ^2 Ratio

To measure the accuracy, we will be using the χ^2 ratio test mentioned in Section 2.4.3. This is where we evaluate the *closeness* of the new proposed algorithm (SMJ) to Maximum Likelihood.

χ^2 ratio is compared in reference to the ML estimate, after all, this is the best obtainable solution for feature based SLAM.

State Recovery

For a given SLAM algorithm that has its states marginalized from the original problem, the missing states can be recovered by fixing the known states associated with the results from the given algorithm. First, re-define the original ML problem then solve for the remaining variables.

For the case of SMJ, the fixed states are all the features and remaining poses in the joined graph. The final step is to calculate the new $\chi^2_{(\texttt{SMJ})}$ with the recovered state estimate using Equation (2.32) and then apply the ratio equation

$$\chi^2 ratio = \frac{\chi^2_{(\texttt{SMJ})}}{\chi^2_{(\texttt{ML})}} \tag{4.14}$$

Experiments

The χ^2 ratio test is performed on the same datasets employed in the NEES experiments in Section 4.4.1.

Table 4.5: 2D χ^2 Ratio on 50 Monte Carlo Trails (Standard Deviation)

MAP	Fusing Method	$n = 4$	$n = 10$	$n = 20$
CIRCLE	BO	1.0($\approx$ 0)	1.0($\approx$ 0)	1.0($\approx$ 0)
	SO	1.0($\approx$ 0)	1.0($\approx$ 0)	1.0($\approx$ 0)
	DCO	1.0($\approx$ 0)	1.0($\approx$ 0)	1.0($\approx$ 0)
LOOP	BO	1.02(0.01)	1.00(0.003)	1.00(0.001)
	SO	1.17(0.30)	1.65(1.28)	1.91(1.55)
	DCO	1.13(0.29)	1.07(0.13)	1.09(0.15)
MANHATTAN	BO	1.00($\approx$ 0)	1.00(0.003)	1.00(0.002)
	SO	1.00($\approx$ 0)	1.00(0.003)	1.00(0.001)
	DCO	1.00($\approx$ 0)	1.00(0.005)	1.00(0.002)

Table 4.6: 3D χ^2 Ratio on 50 Monte Carlo Trails (Standard Deviation)

MAP	Fusing Method	$n = 4$	$n = 10$	$n = 20$
CIRCLE	BO	1.0($\approx$ 0)	1.0($\approx$ 0)	1.0($\approx$ 0)
	SO	1.0($\approx$ 0)	1.0($\approx$ 0)	1.0($\approx$ 0)
	DCO	1.0($\approx$ 0)	1.0($\approx$ 0)	1.0($\approx$ 0)
LOOP	BO	1.00($\approx$ 0)	1.0($\approx$ 0)	1.0($\approx$ 0)
	SO	1.01(0.012)	1.02(0.03)	1.03(0.05)
	DCO	1.00(0.007)	1.0(0.005)	1.0(0.005)
SPHERE	BO	1.017(0.06)	1.00(0.002)	1.0($\approx$ 0)
	SO	25.74(174.74)	7.90(48.8)	1.00($\approx$ 0)
	DCO	1.01(0.017)	1.40(2.88)	1.00($\approx$ 0)

The results from Tables 4.5 and 4.6 indicate that, after the poses are recovered from the estimate, the overall χ^2 ratio is very close to 1. This suggests that the SMJ results have come very *close* to the actual ML solution. Only in the instances of SO is the average large. We can associate this with possible outliers in the results due to poor convergence. Overall, DCO can be seen as a better approximation. Also, increasing the number of local maps does not have any profound change on the accuracy.

4.4.3 Computation Time

To measure the computation time of SMJ, we must also compare the different optimization approaches. The key to an efficient map joining algorithm depends on the time in which the local maps are built and the time taken for joining them.

When assessing the computation time, we have taken into consideration the IRLS bootstrapping time also. Given that the trajectories chosen are complex, we believe that applying IRLS is critical to obtaining consistent results. In our tests, the times for IRLS are included in the parentheses, and thresholds are set according to Sections 3.2.7 and 2.4.1.

For the test, the focus is on a single run of the algorithms. SMJ is currently implemented in Matlab and compared with the Matlab implementations of Maximum Likelihood and IRLS. Unlike Chapter 3, there is no easy way to implement SMJ into g^2o due to correlations within the observation information matrix. Therefore, it would not be fair to compare the ML method in Chapter 3 with our current SMJ implementation. However, in actual SMJ usage, we have taken advantage of g^2o for building local maps.

Table 4.7: Computation Time For a Single SMJ Run on an Intel corei5-2400 Running at 3.10GHz, (IRLS Bootstrapping Time)

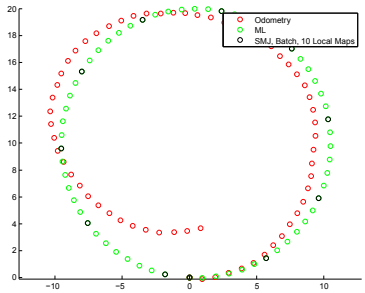
Dimensions	Local Maps	Local Maps(s)	SO(s)	BO(s)	DCO(s)	Total Time(s)
Manhattan	$n = 4$	2.66+(19.75)	0.99	1.19	0.99	23.39/23.60/23.40
	$n = 10$	2.23+(13.18)	0.74	0.48	0.59	16.15/15.90/16.00
	$n = 20$	2.10+(10.81)	1.08	0.99	0.71	14.00/13.90/13.62
	ML					11.44+(17.56)=29
Sphere	$n = 4$	3.44+(29.48)	1.30	1.15	1.13	34.21/34.07/34.04
	$n = 10$	3.08+(20.54)	2.45	1.38	2.11	26.07/25.01/25.74
	$n = 20$	2.96+(14.03)	5.08	2.11	3.28	22.07/19.11/20.28
	ML					4.63+(79.93)=84.57

Table 4.7 is an computation time analysis on the Manhattan and Sphere datasets. It is obvious that SMJ is much more efficient when compared to regular ML. The results are even more evident for 3D, where the overall time reduction is substantial (76%). We can see that the majority of the time is spent on solving the local maps when the number of local maps n is small. Only when we increase the number of local maps does the time taken for joining start to influence the overall time. BO is not always the most efficient method, but for the sphere trajectory it is considerably better than DCO and SO.

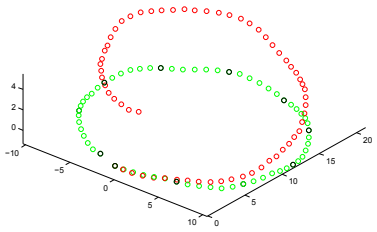
The number of local maps chosen has the greatest impact on the efficiency. Note, although the larger n value gives lower computation times, from the results in Section 4.4.1, the results may also become in-consistent (especially in the Manhattan case). Therefore, choosing the best n value will be critical in obtaining an optimal SMJ algorithm.

4.4.4 Resulting Maps

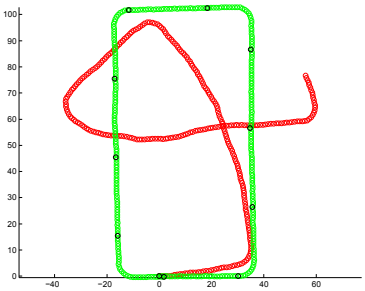
It is better to visualize SMJ once the marginalized poses are recovered. The following are the maps of a single Monte Carlo instance for Manhattan and Sphere. In view of the large number of features, the features have been omitted from the figures to provide clarity.



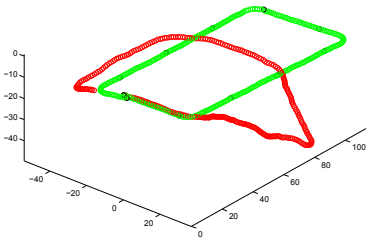
(a) `Circle 2D Result`



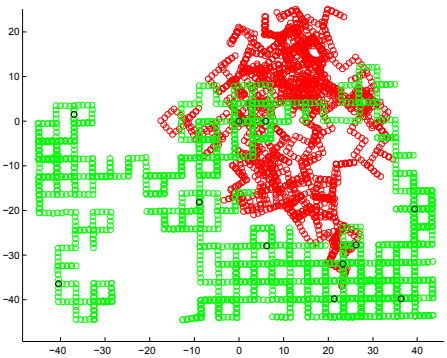
(b) `Circle 3D Result`



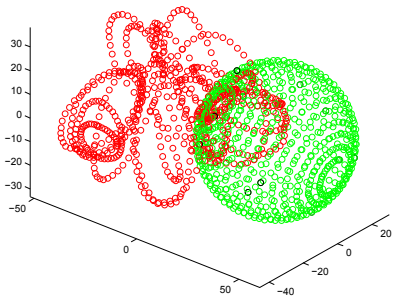
(c) `Loop 2D Result`



(d) `Loop 3D Result`



(e) `Manhattan Result`



(f) `Sphere Result`

Figure 4.7: A single run comparing ML to Batch Optimization at $n = 10$.

It is a common misconception to compare the results of a optimized solution visually. SLAM results from very noisy data can result in extremely poor looking maps. However, this does not mean the solution is less optimal than a poorly converged SLAM result with less noise.

4.4.5 Real Datasets

To better verify SMJ, we run our algorithm on two real datasets. The DLR-Spatial-Cognition dataset [61] is collected using artificial landmarks (white/black circles) placed on the ground. This data contains both odometry and landmarks with good uncertainty measurements. Pre-processing of data has been performed with known data associations. There are a total of 3296 poses, 539 features, 14163 observation and average node degree of 25.83.[1]

The other benchmark dataset is Victoria Park [62]. The data is collected by using a laser sensor to detect natural features in an outdoor environment coupled with vehicle odometry. Data associated is provided using an EKF based SLAM system. There are a total of 6899 poses, 299 features, 45390 observation and average node degree of 151.80.[2]

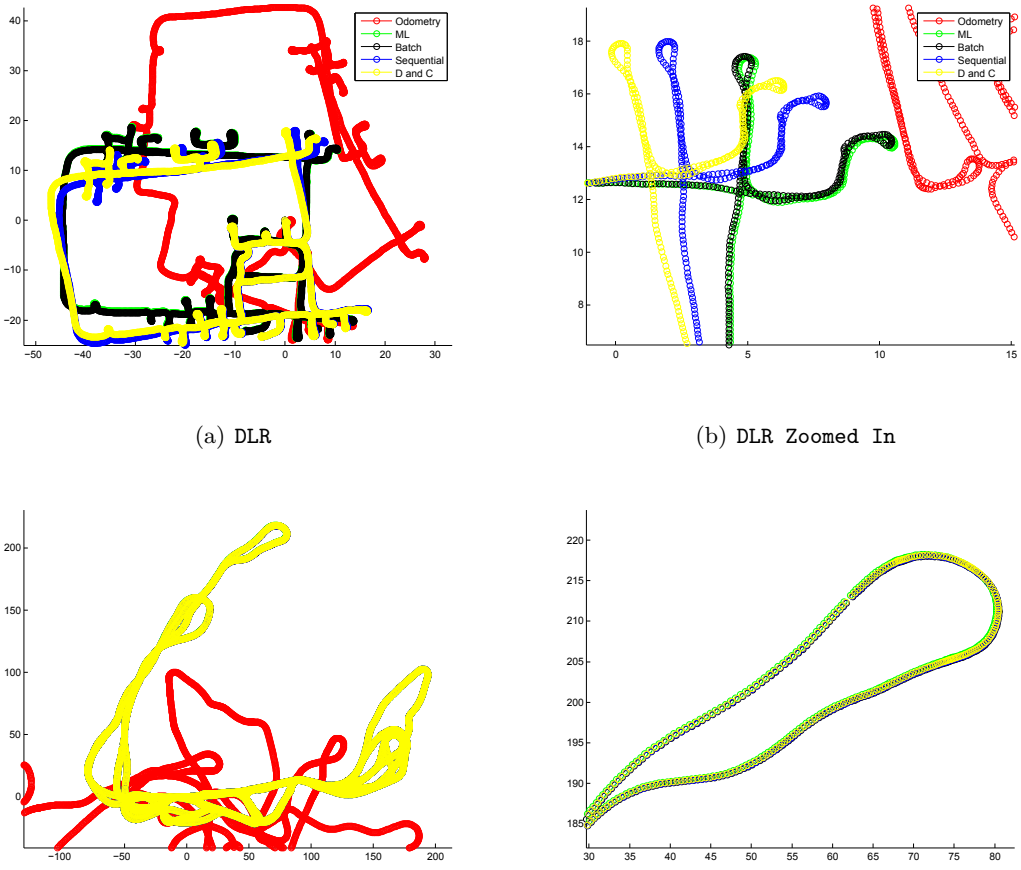
Both datasets have passed the expected value test in Section 2.4.4, meaning that the Gaussian assumption is being satisfied within the measurements.

The results are summarized in Table 4.8, and a visual comparison is provided in Figure 5.8.

Table 4.8: χ^2 Ratios and Time taken for 20 local maps

Dataset	SO	BO	DCO	ML time	Best SMJ time
DLR	1.02	1.00	1.13	317.41	30.18
Victoria Park	1.00	1.00	1.00	1390.70	134.56

[1] https://svn.openslam.org/data/svn/2d-i-slsjf
[2] http://www-personal.acfr.usyd.edu.au/nebot/victoria_park.htm



(a) `DLR`

(b) `DLR Zoomed In`

(c) `Victoria Park`

(d) `Victoria Park Zoomed In`

Figure 4.8: Results from the DLR dataset and the Victoria Park dataset.The SMJ algrotihms applied are Batch, Sequential and D&C.

These results prove that SMJ is also very effective on real world scenarios, with a significant improvement on the efficiency over standard ML. The χ^2 ratios are also very good suggesting that there is very little information loss in the SMJ approximation. In DLR there is a global rotation on DCO and SO methods which is indicative of the lower connectivity in the dataset.

4.5 Discussion

The following is a series of discussions about the SMJ algorithm.

Information Matrix Sparsity

We should recognize that some of the the efficiency of SMJ comes from the sparsity of the information matrix. This sparsity is a result of keeping the last poses of each local map within the final state vector. If we were to marginalize every pose except the current ones, the result would be equivalent to an EKF based method becoming very inefficient as the feature size grows [63]. Through additional marginalization of poses, SMJ will reduce the overall number of states but still keep a certain degree of sparsity. Considering that local maps only contain the information about its *nearby features*, the sparsity of the final information matrix Λ_{SMJ} is determined by how the map is split and the total number of local maps.

Improving the Convergence

From our results, it is evident that SMJ can sometimes diverge due to linearization issues. We have found that the node degree (number of times a feature is observed) plays an important role in improving convergence, consistency and accuracy of the result. With a fully connect graph such as **Circle**, there is almost no quality lost due to the approximations; however, for a more realistic SLAM scenario, many factors can impact the outcome of SMJ.

The main assumption of the SMJ algorithm is that local maps are consistent. On top of applying IRLS bootstrapping, selecting the number of local maps also helps with better convergence properties, such that the assumption is satisfied. We have found that smaller local maps are typically simpler non-linear least squares problems that can converge very quickly.

From the results, increasing the number of local maps can improves the overall computation time of the algorithm, see Table 4.7. One major drawback is that the optimal method,

BO, can sometimes diverge due to bad initial estimates (concatenating the ends of several local maps). By using SO or DCO, the convergence is improved to some degree given that a better initial value is achieved at each joining step. However, these methods may impose additional approximations and will be even further away from the maximum likelihood solution. To avoid these approximations one could apply SO and DCO as bootstrappers to solving a Batch SMJ problem.

Alternatively, if the dataset is able to be improved directly (e.g., Active SLAM), one should aim to improve the average or minimum node degree of the dataset. Our recommendation is to either find more features, increasing the overall density of features, or improve the sensing capabilities to observe more features from a single pose. A small example of this can be seen in the real datasets - Victoria park has a much larger sensor range compared to DLR and respectively also has a better χ^2 ratios (Table 4.8).

SMJ as a Bootstrapper

In many cases, the estimates achieved by SMJ are very *close* to the actual Maximum Likelihood estimate. Tables 4.5 and 4.6 provide evidence for this. If we were to solve ML, starting from the SMJ estimate, our chances of converging to the global minimum is much better than that of odometry. Therefore, SMJ also makes a good candidate for the reliability framework described in Section 3.2, and can be categorized as a bootstrapper for feature based SLAM. According to the framework, the intermediate optimization steps **C2** can be treated as the SO or DCO joining of local maps and **C3** can be satisfied from the final BO solution.

4.6 Summary

In this chapter we have demonstrated how to solve the SLAM Back-End, through 2D and 3D SMJ for the efficient building of large scale maps. The overall dimensionality is reduced by the marginalization of poses through the local map building and joining process.

We have proposed three methods to solve the map joining problem: Batch Optimization; Sequential Optimization; and Divide & Conquer Optimization. Batch Optimization is demonstrated to be the most consistent out of the three. The other two methods are in-fact approximations of BO and should only be considered if BO has or is likely to converge onto a local minimum. Overall, computation time can be improved by increasing the number of local maps to some extent. However, we must take care that the consistency is not also being compromised.

SMJ is an effective way of handling scalability for the feature based SLAM problems mentioned in this chapter. Unfortunately, pose marginalization is still not enough when the number of features are exceptionally large. This is especially true in vision based SLAM. In the next chapter we will propose *Pose Graphs Representation* - a way to effectively remove the features from the map and handle the scalability issue in a completely different way.

Chapter 5

Pose Graph Representation

5.1 Introduction

One way to reduce state dimensionality is to remove all features from a graph and use a purely pose based representation of the data. As discussed in Chapter 3, pose graphs can be a simpler SLAM problem to solve, since all vertices ($\mathcal{V}$) share the same dimension and all edges ($\mathcal{E}$) can be expressed using the same measurement equations. Consequently, pose graphs have very strong properties that link with many important concepts in graph theory. Many works in pose graphs [38, 40, 41, 43] have found ways to exploit these properties to improve on the issues faced in SLAM. Although the growing trend in the SLAM community has been towards pose graph; the process and affects in converting a feature based problem into a pose graph representation has remained somewhat unexplored.

Intuitively, one ought to expect a reduction in computation time when features are no longer being estimated. What is unknown are the associated approximations and inconsistencies that arise when changing the feature observations into relative poses. The important fact to consider is that a single observation may correlate many poses once the least squares is solved, if they are to be approximated, some of these correlations may become lost.

The critical question posed in this chapter is whether the original problem of **Features based** SLAM can be represented as a **Pose Graph**, is it possible to achieve a substantial

gain in SLAM efficiency while still minimizing the amount of information loss? In addition to this, are there any added advantages to this type of representation?

The chapter is comprised of the following sections. Section 5.2 gives a brief introduction into pose graphs. Section 5.3 details the steps involved in converting a feature based SLAM problem into a pose graph representation. Section 5.4 is our evaluation and Section 5.5 poses some discussion topics.

5.2 Pose Graphs

In addition to the related work mentioned in Section 2.5, the pose graph representation (PGR) has also been very effective in solving many large scale mapping applications for both 2D [8, 30] and 3D environments [64]. In recent developments, researchers have even found ways to remove the influence of bad data association and outliers [16, 65].

In pose graphs, raw sensor data is replaced with the so called *"virtual measurements"* conditioned by the original measurements. In other words relative poses are formed by the most likely transform that fits the feature observations. Standard estimation techniques are made possible, if the *"virtual measurements"* are also assumed to be Gaussian. However, in reality the observation model is multi modal, meaning that removing single features observation may lead to multiple relative poses and the connectivity needs to be described as a probability distribution. Trying to deal with multi modality is highly complex and can sometimes be infeasible to solve [66]. Therefore, researchers have resorted to approximate methods like the one proposed in this chapter.

Figure 5.1 gives a simple example of creating a pose graph. Robot poses are estimated independently from the map/features once in the pose graph form. It is also possible to recover the features once the best configuration of the poses is known.

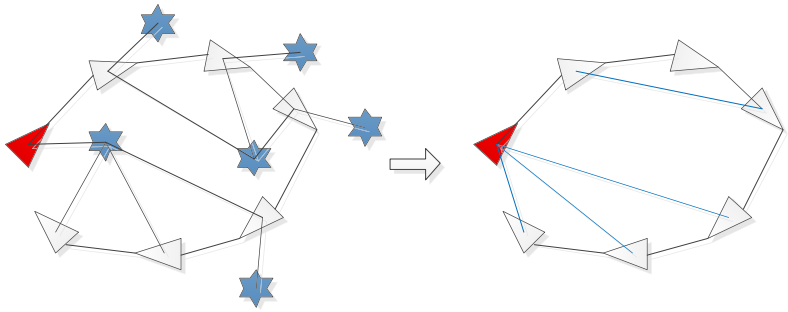


Figure 5.1: Example of a feature based graph represented as a pose only graph.

5.3 Pose Graph Representation of Feature Based SLAM

In many robotic systems the idea of extracting observation vectors from sensor measurements might be non-trivial. Given that we are able to use a simple point feature based model (g_{im}), described in (2.2): the following sections are devoted to the methodologies for extracting relative pose constraints whilst at the same time preserving the overall consistency of the problem.

5.3.1 Obtaining Relative Pose

Using very standard methods, it is possible to derive a relative pose Z_{ij}^P from two sets of point feature observations $\hat{Z}^F$ from poses i and j. The associated covariance/information matrix Ω_{ij}^P is either approximated or directly obtained through the least squares algorithm. The formulation is as follows

$$Z_{ij}^P = \begin{bmatrix} \delta x_{ij}^P \\ \delta y_{ij}^P \\ \delta \theta_{ij}^P \end{bmatrix} \sim N(0, \Sigma_{ij}^P) \tag{5.1}$$

To obtain this relative pose, Z_{ij}^P (with Gaussian noise), one way is to assume data

association and optimize for T.

$$T_{ij}^{\star} = \underset{T_{ij}}{\operatorname{argmin}} \sum_{m} \left\| T_{ij}(\hat{Z}_{im}^{F}) - (\hat{Z}_{jm}^{F}) \right\|^{2} \tag{5.2}$$

A better way is to reformulate Equation (5.2) into a two pose SLAM optimization problem defined by the functions expressed in Section 2.2.1. This way features are also being jointly optimized.

$$X^{\star} = \underset{X}{\operatorname{argmin}} \sum_{m} \left\| (\hat{Z}_{im}^{F} - X_{m}^{F}) \right\|_{\Omega_{im}}^{2} + \sum_{m} \left\| (\hat{Z}_{jm}^{F} - h_{jm}(X_{j}^{P}, X_{m}^{F})) \right\|_{\Omega_{jm}}^{2} \tag{5.3}$$

Assuming that $X_{i}^{P} = \{0, 0, 0\}$ is an anchor pose, then the optimized state $(X_{j}^{P})^{\star} = T_{ij}^{\star}$. From the least squares formulation we can obtain Ω_{ij}^{P} after marginalizing out the feature part of the states.

$$\begin{aligned} \Lambda &= J(X^{\star})^{T} \Omega^{F} J(X^{\star}) \\ \Omega_{ij}^{P} &= \Lambda_{j}^{P(\mathcal{M})} \\ Z_{ij}^{P} &= (X_{j}^{P})^{\star} \end{aligned} \tag{5.4}$$

A simple example of this process is conveyed below.

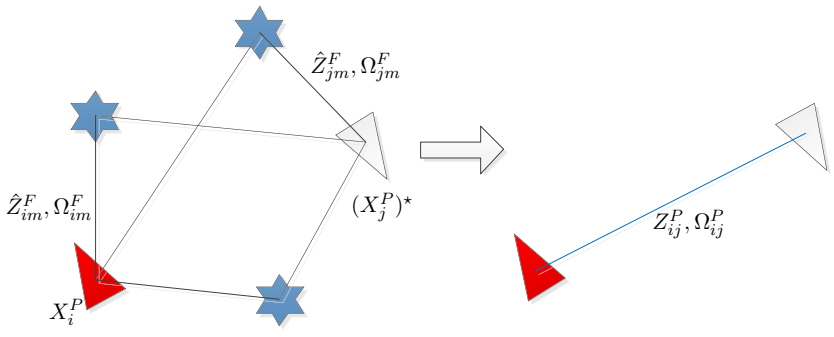


Figure 5.2: In this example, the relative pose measurement Z_{ij}^{P} is obtained by solving a non-linear least squares using the observations $\hat{Z}_{im}^{F}, \hat{Z}_{jm}^{F}$.

In regression, the initial estimate is a critical factor for convergence. Horn [67] proposes a closed form way of finding T_{ij} using the centroids of the coordinate system. This technique

is both efficient and accurate for 2D and 3D transformations.

$$T_{ij}^{(0)} = \texttt{Horn}(\hat{Z}_{im}^{F}, \hat{Z}_{jm}^{F}) \tag{5.5}$$

To avoid solving a full non-linear least squares problem, one could directly apply the result of Horn and approximate the covariance by linearizing about the estimate. However, this would not lead to the most accurate estimate or covariance of the relative pose. For further discussion see Section 5.5.1.

Minimum Number of Observations ($\texttt{Obs}_{\texttt{min}}$)

When obtaining a relative pose, one important condition is the minimum number ($\texttt{Obs}_{\texttt{min}}$) of feature observations needed. This number is equal to the degrees of freedom in the system, (For our 2D models it is 2, and 3D it is 3). If $\texttt{Obs} > \texttt{Obs}_{\texttt{min}}$, less uncertain and more accurate Z_{ij}^{P} will be obtained but may also result in less relative poses. The minimum number will be an important factor in the search for relative poses.

5.3.2 Information Reuse

Information reuse, when computing relative poses, is a critical factor to consider when trying to avoid inconsistency. Simply stated, the uncertainties in a feature observations $\hat{\Sigma}_{ij}^{F}$, when applied in its current form, will lead to overconfidence given that the information itself is being reused in several relative pose calculations.

In this section, we will express two ways for overcoming information reuse.

Single Observation Method, PGR(SO)

Avoiding information reuse all together is a popular approach. This is done by only ever applying information once. Which means, when an observation is selected to compute Z^P_{ij}, it can no longer be employed in any subsequent calculations for relative poses. To do this, one must first distribute the observations, such that the maximum number of relative poses are obtained. In the process, the minimum observation condition, Obs_{min}, should also be taken into account.

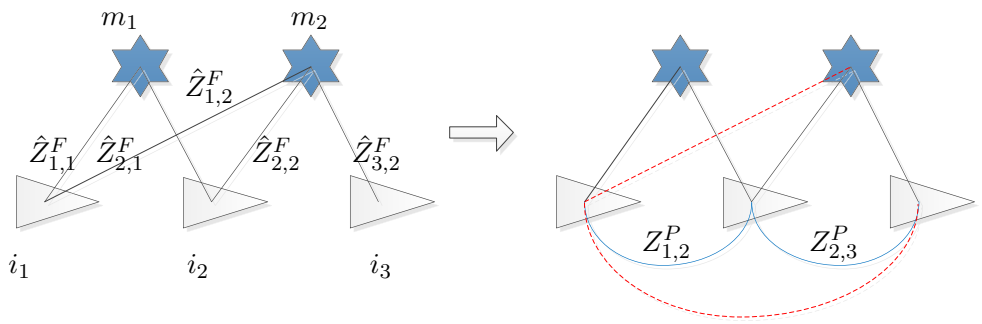


Figure 5.3: Given a simple 1D problem where $Obs_{min} = 1$, $(\hat{Z}^F_{1,1}, \hat{Z}^F_{2,1})$ are used to calculate $Z^P_{1,2}$ and $(\hat{Z}^F_{2,2}, \hat{Z}^F_{3,2})$ for $Z^P_{2,3}$. If we wanted to add an additional relative pose between i_1 and i_3 (red dotted lines), then the constraint $\hat{Z}^F_{1,2}$ would need to be used in conjunction with observation $\hat{Z}^F_{3,2}$, violating the **PGR(SO)** criteria. Therefore, the observation $\hat{Z}^F_{1,2}$ is ignored (red dotted lines)

PGR(SO) does not impose any alterations to the information, but is only effective when the number of features are exceptionally large. If this is not the case, then very few relative poses can be obtained, resulting in significant information loss.

Multi Observation Method, PGR(MO)

The Multi Observation Method, **PGR(MO)**, reconciles the problem of information reuse by evenly dividing information heuristically. The intuition is that, an observation with measurement $\hat{Z}_{ij}$ with information matrix Ω_{ij} is equivalent to k number of the same observations each with the information $\frac{1}{k}\Omega_{ij}$. Simply put, depending on the number of times an observation is applied to calculate a relative pose, its corresponding information

will then be divided up equally. Figure 5.4 conveys an example of this. The objective of the Multi Observation Method is to find an optimium solution to PGR, that is most likely close to the solution of feature based SLAM.

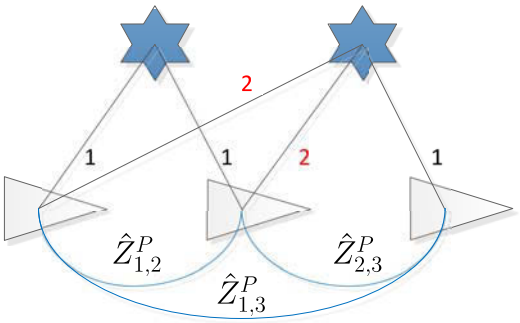
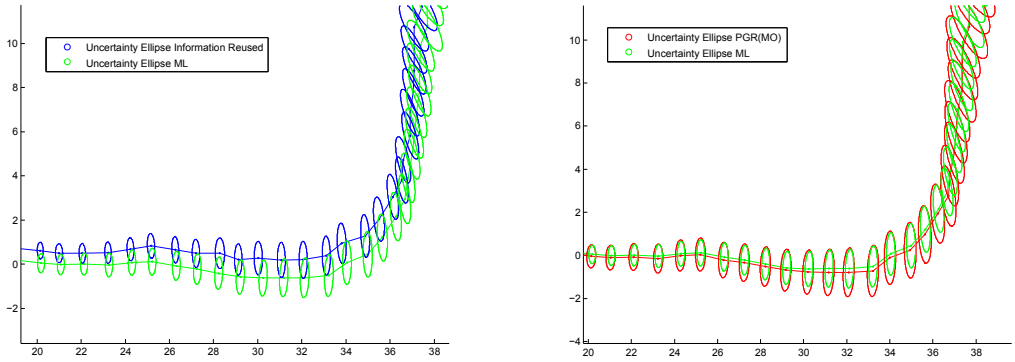


Figure 5.4: Following the same example posed in Figure 5.2, If we apply all the information, calculating $\hat{Z}^F_{1,1}$ requires $(\hat{Z}^P_{1,2}, \hat{Z}^F_{1,2}, \hat{Z}^F_{2,1}, \hat{Z}^F_{2,2})$. $\hat{Z}^P_{2,3}$ requires $(\hat{Z}^F_{2,2}, \hat{Z}^F_{3,2})$ and $\hat{Z}^P_{1,3}$ requires $(\hat{Z}^F_{1,2}, \hat{Z}^F_{3,2})$. It is clear that $\hat{Z}^F_{1,2}$ and $\hat{Z}^F_{2,2}$ are used exactly twice. According to **PGR(MO)**, the information matrices are modified to be $\Omega^F_{1,2}/2, \Omega^F_{2,2}/2$. The k values are represented by the number at each edge.

If we compare the uncertainties of **MO** opposed to ignoring information reuse (i.e., calculating the maximum number of relative poses without changing the information matrix), we see from Figure 5.5 that our estimate is a much better approximate on the actual uncertainty.



(a) `Information Reused` (b) `Multi Observation Method (MO)`

Figure 5.5: When information reuse is not handled, the uncertainties in x, y are overconfident. For MO, the resulting ellipse is much closer to that of the maximum likelihood. The trajectory is a zoomed section of Loop 2D.

Effectiveness of PGR

Depending on the graph structure, PGR is not necessarily faster at solving a feature based problem. This is due to the scaling on the number of edges that is obtainable. The following is a simple condition to quickly determine which methods (PGR or ML) are more efficient based on the total number of edges in each graph.

$$3n \ll 2m \tag{5.6}$$

Equation (5.6) describes the efficiency condition only for a 2D problem, given that the number of equations for an edge in pose-pose is 3 and pose-feature is 2. Thus, the number of feature observations m must be significantly larger than the number of obtainable relative poses n, such that steps involved in pose graph optimization is more efficient.

In PGR there exist two pre-processing steps needed before applying optimization on the pose graph, Equation (2.28). The first involves the searching of possible relative poses and the second is the calculation of the found relative poses. The first step will scale quadratically based on the number of poses in the search space and the second is linear over the number of relative poses found. Step two is often more computationally expensive

since it involves solving multiple non-linear least squares problems. If the condition (5.6) is not sufficiently satisfied, then additional computation burdens are added to the pre-processing steps as well, making the overall algorithm even slower.

Reducing Complexity Via Key Poses

Understanding that our main concern is computation time, one way to better meet condition (5.6), is to further approximate the pose graph. The proposed method, *Key Poses*, limits the number of relative poses that the algorithm can make, by only searching through a subset of the original pose set. $X^K \subset X^P_{(\texttt{ML})}$. From the search process, we can obtain a new n.

Finding the optimal subset for achieving a *good* approximation while also improving the efficiency is difficult. A *good* subset has to be both consistent and accurate. Our technique is heuristic, in that we define a *Key Pose* from every k number of poses along the length of the trajectory. If $k = 1$, then the maximum number of relative poses will be searched for. As k increases, the search space becomes smaller, in turn increasing our chance to satisfy (5.6) while applying a greater approximation on the optimal pose graph.

The intuition for this method is that neighboring poses will not add much influence when improving the overall quality of the graph and only lower quality information is lost when ignoring some of these during a search. Another interpretation is that, feature observations tend to occur in clusters, meaning that a series of consecutive poses will most likely observe the same features. The chances of our search missing important relative poses for loop closures are low, unless k is set too large.

To find the optimal subset X^K, the simplicity and computational overheads of *Key Poses* makes our method very attractive. Although more sophisticated techniques should be pursued to make **PGR(MO)** even better. We wish seek to investigate this idea further in future work.

5.3.3 Algorithm

Once the pose graph is created, we can resort to our standard *IRLS+GN* approach, mentioned in Section 3.4.2, to solve the optimization problem. The algorithm for **PGR** is summarized in Appendix D.

5.4 Evaluation

Similar to Chapter 4, the same set of techniques as in Section 4.4 are employed for evaluation. Here the concern is placed on the performance of **PGR(MO)** under differing k values. Since the efficiency of pose graphs comes from the removal of features, the datasets (Appendix A) utilized in SMJ have been slightly modified. The modification is to increase overall feature density while keeping all other parameters constant, (e.g., Sensor Model, Trajectory and Noise).

Below is the summary of the modified datasets. The trajectory size of Manhattan has been halved so that ML is able to finish without suffering from memory issues due to the increased number of feature observations.

Table 5.1: Summary of Datasets

Dimension	Dataset	no. Poses	no. Features	no. Feature Observations	Average Node Degree
2D	Circle	61	64	3904	62
	Loop	301	324	3482	11.12
	Manhattan	1750	172126	3871	94.16
3D	Circle	61	64	3904	62
	Loop	301	1369	11189	13.39
	Sphere	701	363	10435	19.5

Again we will be comparing against the benchmark solution, Maximum likelihood for feature based SLAM, checking for NEES and χ^2 ratios.

5.4.1 Consistency using NEES

This time only poses are checked for NEES, in which case all the features have to be marginalized from $X_{(\texttt{ML})}$ and $\Lambda_{(\texttt{ML})}$.

$$X_{(\texttt{PGR})} = [X^{P^1}, \cdots, X^{P^n}]^T \tag{5.7}$$

Once again this is done by imposing the Shur Complements (5.8). For NEES equations refer back to (2.35).[1]

$$\begin{aligned} X_{(\texttt{ML})} &\rightarrow X_{(\texttt{ML})}^{(\mathcal{M})} \\ \Lambda_{(\texttt{ML})} &\rightarrow \Lambda_{(\texttt{ML})}^{(\mathcal{M})} \end{aligned} \tag{5.8}$$

Experiments

The results from NEES tests are detailed in the following two tables:

Table 5.2: 2D NEES Test on 50 Monte Carlo Trails (Percentage of Runs Within 95% χ^2 Gate)

MAP	Gate	$k=1$	$k=2$	$k=3$	$k=5$
CIRCLE	215.56	99.14(100)	81.14(100)	88.10(100)	92.94(100)
LOOP	974.02	572.46(100)	549.32(100)	585.07(100)	647.187(100)
MANHATTAN	5419.70	2924.5(100)	2883.70(100)	3912.3(100)	4224.40(100)
MAP	Gate	$k=8$	**PGR(SO)**	ML	
CIRCLE	215.56	96.17(100)	198.00(82)	181.72(90)	
LOOP	974.02	698.168(100)	904.55(90)	891.04(100)	
MANHATTAN	5419.70	4623.00(100)	4780.40(88)	5179.90(100)	

Table 5.3: 3D NEES Test on 50 Monte Carlo Trails (Percentage of Runs Within 95% χ^2 Gate)

MAP	Gate	$k=1$	$k=2$	$k=3$	$k=5$
CIRCLE	215.56	99.44(100)	81.14(100)	88.10(100)	92.94(100)
LOOP	974.02	569.96(100)	550.24(98)	579.19(100)	721.35(100)
SPHERE	2210.80	1367.00(100)	1415.90(100)	1467.40(100)	1640.50(100)
MAP	Gate	$k=8$	**PGR(SO)**	ML	
CIRCLE	215.56	96.17(100)	198.00(82)	181.72(90)	
LOOP	974.02	1300.70(30)	988.21(36)	901.62(96)	
SPHERE	2210.80	1833.50(100)	2183.40(60)	2075.40(100)	

[1] Refer to Sections 2.4.4 and 4.4.1 for more information regarding this evaluation technique

The NEES results provide evidence that the consistency of our algorithm is comparable to ML. **PGR** is overall consistent, even after reducing the search space significantly with **PGR(MO)**. Only at Loop 3D $k = 8$ does the NEES become unstable, which is related to a combination of poor connectivity and high k value. When compared to **PGR(SO)**, at reasonable k values **(1-3) PGR(MO)** results are always better, concluding that handling information reuse is far more effective than trying to avoid it.

For 3D trajectories the results confirm similar trends; however, 3D parameterization for pose graphs can become unstable due to ambiguities associated with Euler angles parameterization. Because of this we have started from ground truth to guarantee convergence and stability.

5.4.2 Accuracy using χ^2 Ratio

Once again accuracy is evaluated by the χ^2 ratio, refer to Section 4.4.2. This time, recovery of the unknown states is obtained by fixing the robot poses (estimate of PGR) in place and optimizing only features. For pose graphs, the recovery of features via this technique may be regarded as the most optimal way to obtain the map once a pose graph is solved. Although the problem is still non-linear, we have found the convergence to be, at most times, reliable.

The χ^2 ratio equation is defined by

$$\chi^2 ratio = \frac{\chi^2_{(\texttt{PGR})}}{\chi^2_{(\texttt{ML})}} \tag{5.9}$$

Experiments

The same simulations as in Section 5.4.1 are repeated for the χ^2 Ratio test:

Table 5.4: 2D χ^2 Ratio on 50 Monte Carlo Trails (Standard Deviation)

MAP	**PGR(MO)** $k=1$	$k=2$	$k=3$	$k=5$	$k=8$	**PGR(SO)**
CIRCLE	1.00(≈ 0)	1.03(0.01)	1.03(0.01)	1.02(0.01)	1.33(1.33)	1.29(0.46)
LOOP	1.00(≈ 0)	1.06(≈ 0)	1.32(0.26)	3.75(1.16)	8.23(1.85)	1.22(0.01)
MANHATTAN	1.00(≈ 0)	1.02 (≈ 0)	1.08(0.15)	2.95(0.95)	7.53(1.64)	1.10(0.02)

Table 5.5: 3D χ^2 Ratio on 50 Monte Carlo Trails (Standard Deviation)

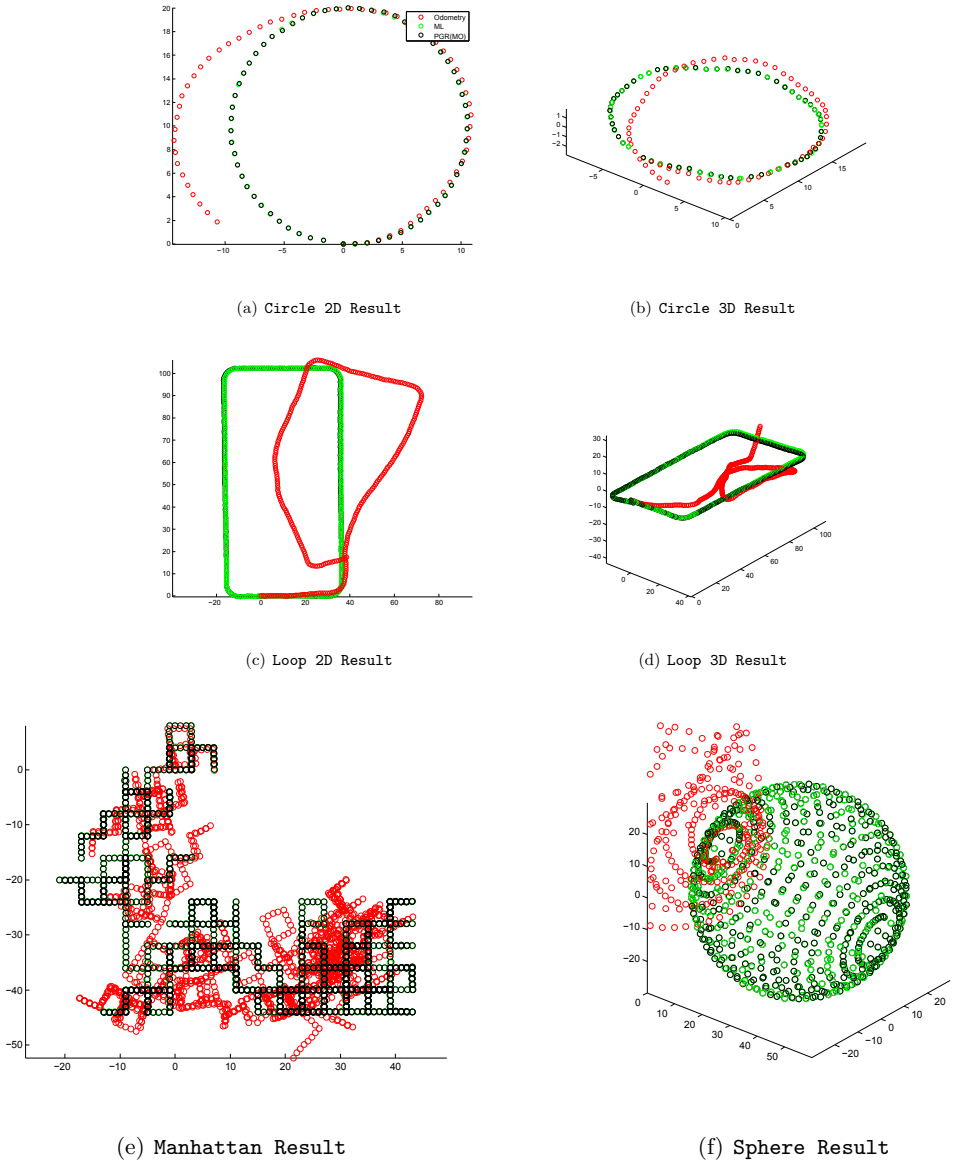
MAP	**PGR(MO)** $k=1$	$k=2$	$k=3$	$k=5$	$k=8$	**PGR(SO)**
CIRCLE	1.00(≈ 0)	1.03(≈ 0)	1.03(0.01)	1.02(0.01)	1.32(1.33)	1.29(0.46)
LOOP	1.00(≈ 0)	1.02(≈ 0)	2.44(0.57)	22.38(2.18)	56.51(6.75)	1.24(0.01)
SPHERE	1.49(3.37)	2.11(4.78)	3.27(5.85)	22.55(9.95)	53.83(12.56)	8.98(10.39)

Although the NEES test demonstrates much consistency in all cases of **PGR(MO)**, only under some k values is the estimate accurate or *close* to ML. The reasons are linked to the connectivity of the graph or pose graph structure. As k increases, even small errors incurred from pose estimation will translate to large errors in the re-estimation of feature location. If the graph is already poorly connected in the pose feature graph, then it is unlikely that a good χ^2 ratio can be obtained. In the 3D datasets, the translation error is even higher. Even at the lowest k, Sphere has a substantial difference in χ^2 ratio.

The χ^2 ratio tells us that we might lose significant information in the pose graph representation and this is amplified when trying to reduce the computation time with high values of k. As a consequence, unlike SMJ, it would be unwise to use PGR unless both accuracy and efficiency can be maintained. The situation will be highly dependent on the nature of the problem whereby pose and feature configurations should satisfy good connectivity and the efficiency condition (5.6).

5.4.3 Resulting Maps

Here are the outcomes of a single run of PGR. Again features have been omitted for clarity.



(a) `Circle 2D Result` (b) `Circle 3D Result`

(c) `Loop 2D Result` (d) `Loop 3D Result`

(e) `Manhattan Result` (f) `Sphere Result`

Figure 5.6: A single run comparing ML to PGR(MO) $k = 3$.

5.4.4 Computation Time

The algorithm, in terms of computation time, is split into the following sub-categories: pre-processing - searching for and calculating relative poses. and solving - optimization

using *IRLS+GN*. From this we can compare the total time for standard ML and PGR. The results are given for a single Monte Carlo simulation of Manhattan and Sphere. Again we have decided to evaluate all our results in Matlab for a fair comparison.

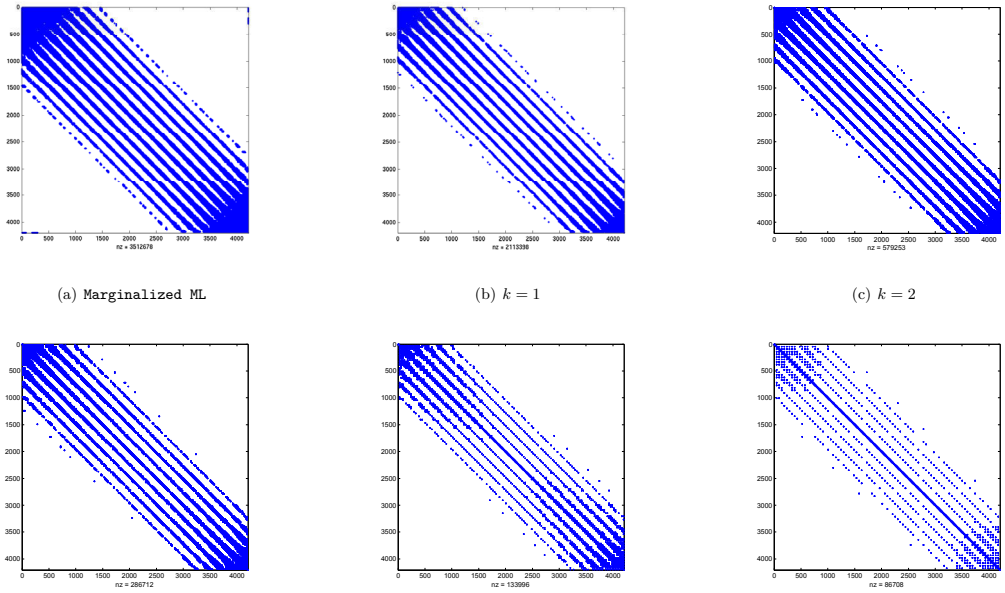
Table 5.6: Computation Time in Seconds, on an Intel Corei5-2400 running at 3.10GHz, (IRLS time)

Dataset	k	Search Time	Computing Relative Pose	Solving Pose Graph	Total Time	$3n \ll 2m$
Manhattan	1	1054.9	14526	4157+(28060)	47798	1127613,125416
	2	112.9	2171.2	1502.8+(2704.0)	6490.9	291177,125416
	3	46.8	903.2	205.4+(409.2)	1564.6	124449,125416
	5	16.3	295.7	37.4+(86.5)	435.9	44184,125416
	8	6.5	131.3	12.1+(37.2)	187.1	18963,125416
	SO	397.5	368.1	173.2+(117.3)	1056.2	34038,125416
	ML				1190.2+(394.4)=1584.6	
Dataset	k	Search Time	Computing Relative Pose	Solving Pose Graph	Total Time	$6n \ll 3m$
Sphere	1	26.8	842.9	1361.4+(2210.8)	4441.9	87510,20870
	2	5.2	210.1	156.0+(219.8)	591.3	21882,20870
	3	2.3	105.0	43.0+(66.3)	216.8	9618,20870
	5	0.9	32.4	12.8+(38.0)	84.3	3198,20870
	8	0.4	16.1	21.7+(18.4)	56.7	1194,20870
	SO	24.2	45.1	29.41+(46.9)	145.7	2355,20870
	ML				94.7+(185.4)=280.2	

We see that only when k is equal or greater than 3, is the efficiency condition satisfied. Seeing that searching takes up substantial computation time at lower values of k, we cannot simply keep applying the searching step until the condition is met. By either approximating the search or finding a better way to structure variables, it would be possible to obtain the n relative poses much faster. Currently the best k can only be obtainable from Monte Carlo testing.

When solving the pose graph, the efficiency can be observed from the sparsity of the information matrices. The sparsity will relate directly to the time taken to apply the factorization step at each iteration point. As the value of k grows so does the sparsity, which is obvious because less relative poses are being constructed. Figure 6.1 displays the change in the information matrix as we increase k. We compare this to a marginalized ML information matrix without the features. Even at $k = 1$, the matrix is a lot sparser, suggesting that pose graphs are quicker to factorize when compared to marginalized feature

based SLAM problems. However, the additional overhead associated with calculating Jacobians, residuals, and pre-processing diminish the overall computation time.



(a) `Marginalized ML` (b) $k = 1$ (c) $k = 2$

(d) $k = 3$ (e) $k = 5$ (f) $k = 8$

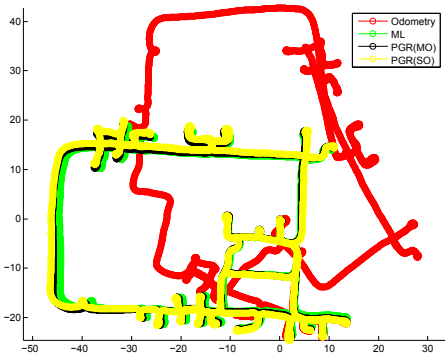
Figure 5.7: Information matrix sparsity of differing k values on the 3D Sphere dataset

5.4.5 Real Datasets

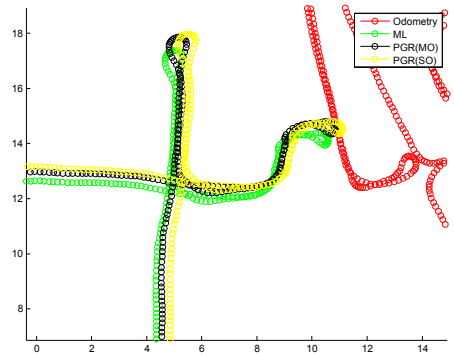
PGR(MO) and **PGR(SO)** are tested on real datasets introduced in Section 4.4.5. k values are adjusted accordingly such that $3n \ll 2m$.

Table 5.7: χ^2 Ratios and Time taken

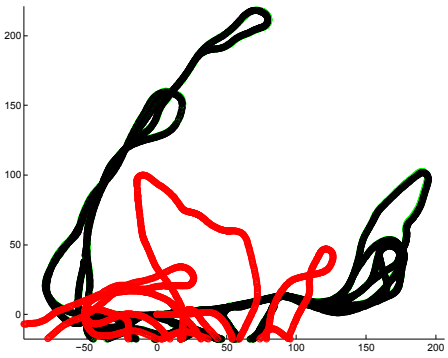
Dataset	k	$3n \ll 2m$	**PGR(MO)**	**PGR(SO)**	**PGR(MO)** time	**PGR(SO)** time	ML time
DLR	3	21786,28368,	1.40	2.03	240.60	377.89	317.41
Victoria Park	11	52683,90780	1.83	-	700.09	-	1390.70



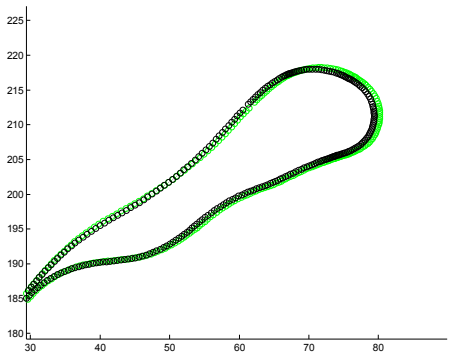
(a) `DLR Results`



(b) `DLR Results Zoomed In`



(c) `Victoria Park Results`



(d) `Victoria Park Results Zoomed In`

Figure 5.8: Results of DLR dataset and Victoria Park dataset.

The reason why we have omitted **PGR(SO)** for Victoria Park is that $3n$ is exceptionally large. The k value must be set high for Victoria Park due to the poses completely outnum-

bering features, resulting in the same feature being observed multiple times. Considering that features are not especially large in the datasets, using PGR is not as effective when compared to SMJ as shown in Table 4.8. Evidently we will also not be able to achieve a better χ^2 ratio given the large k. It is clear that for DLR and Victoria park, the precedent for estimating features is much higher than that of poses. From the results, we realize that PGR is a very sub-optimum approach for these types of problems.

5.5 Discussion

Now we will discuss some interesting properties of PGR.

5.5.1 Further Improving Efficiency

As mentioned in Section 5.3.1, one way to avoid solving a full least squares problem is to use the closed form Horn function [67] to approximate relative poses Z_{ij}^P. Then recover the covariance through marginalization of the resulting information $\Lambda \rightarrow \Omega_{ij}^{\mathcal{M}}$. In some SLAM problems it may be very expensive to use full optimization methods, given the number of feature observations and features.

If the solution from Horn is *close* to the actual maximum likelihood, then the consistency of the results should be preserved. The following is a preliminary test for Horn.

Table 5.8: Horn vs. Least Squares on Sphere Dataset, $k = 3$

MAP	Gate	NEES	Time to Computing Relative Poses (sec)
Least Squares	2210.8	1786.7	105.0
HORN	2210.8	1795.3	61.7

Table 5.8 indicates a 40% decrease in the time taken to resolve all the relative poses. Also a very small change is seen in the NEES values. Ultimately more investigation is required but from preliminary testing, this type of approximation can still lead to a consistent solution.

5.5.2 Euler Angle Parameterization

In three dimensions, the singularities of Euler angles and over parametrization of quaternions can become problematic in defining the correct state space. Grisetti et al [57] provides evidence that the singularities of Euler angle parametrization can easily lead to divergence in Gauss-Newton methods.

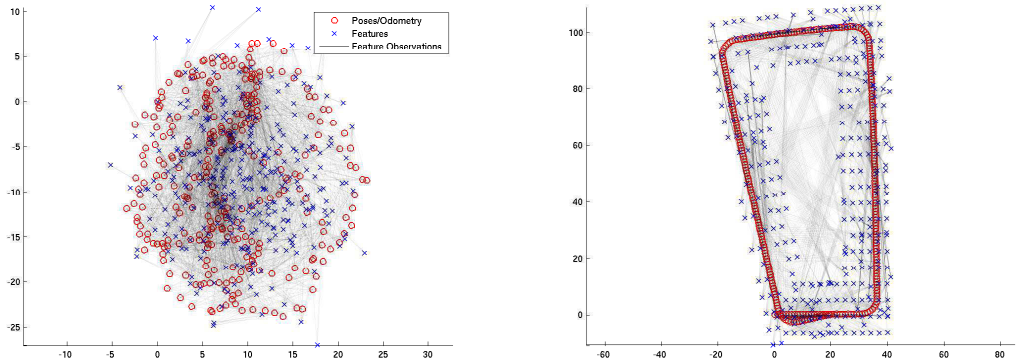
The manifolds representation proposed by [58] defines the underlying space as manifolds and applies a special operator $X^0 \boxplus \Delta X$ to replace the original $X^0 + \Delta X$. The special operator encapsulates the transition of incremental state vector ΔX into the new estimate $X^\star$. During minimization, each update is done in small increments on the local Euclidean space and the accumulated result is represented globally in non-Euclidean space. Doing so avoids Euler angles becoming trapped into singular configuration during updates steps and also the associated ambiguities. In future work, we wish to investigate this better representation of 3D rotation.

5.5.3 Outliers in Feature Observations

In filtering based SLAM, outliers are handled through data association given known covariances [68]. However, in full SLAM one must perform an expensive inverse on the information or calculate marginals to get covariances. Therefore, data association is typically left up to the Front-End to solve, where data association is never guaranteed and outliers are inevitable. If they are passed through to ML, then the result is often extremely poor or completely unusable.

Robust estimation methods have often been used to counter these issues by reducing the influence of large residuals (outliers), refer to Chapter 3. The major problem associated with these methods is that robustness is lost once the number of outliers is too large.

Figure 5.9(a) illustrates what happens to ML when only a small number of observation outliers are introduced. Figure 5.9(b) demonstrates the affects in applying a Robust M-estimator, Cauchy, for only 5% feature outliers.



(a) `ML with` 5% `outliers` (b) `Cauchy M estimator with` 5% `outliers`

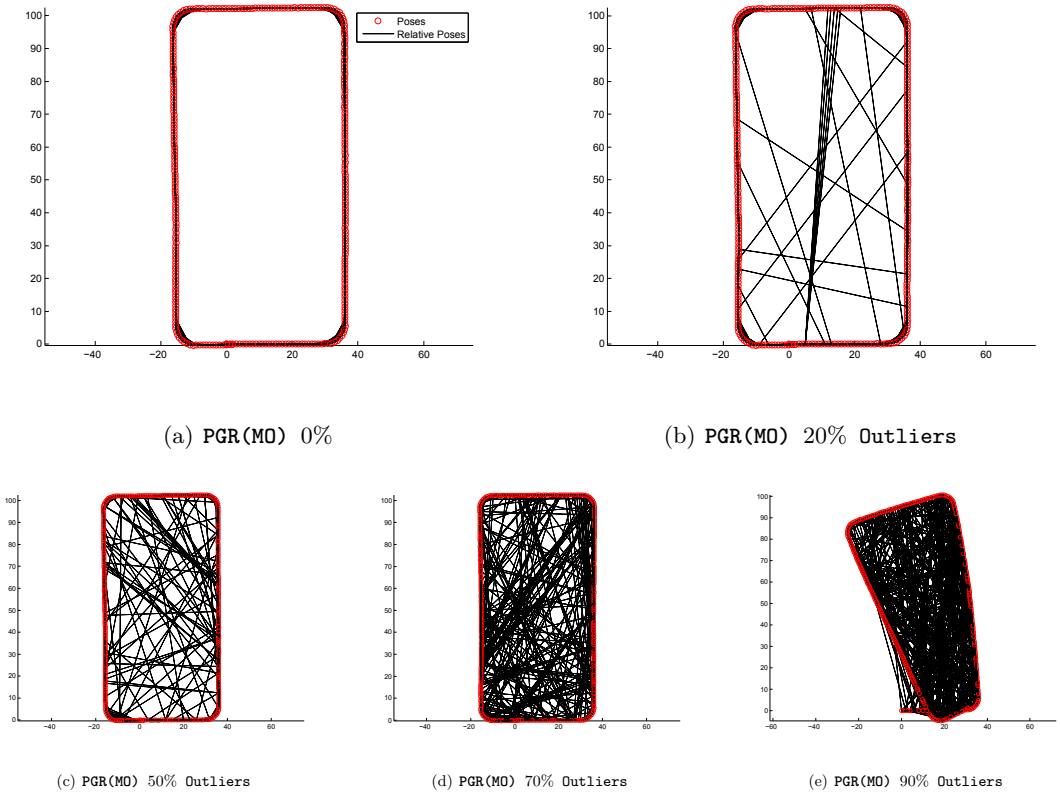
Figure 5.9: ML in the presents of outliers (outliers are randomly generated on observations in no specific order)

It is clear that robust methods do work but not particularly well for solving pose feature graphs when there are too many outliers.

Our proposal is to employ **PGR(MO)** first to find relative poses and then apply a robust method on the pose graph to damping the influence of outliers in the poses graph space. $k = 1$ is imposed in our test case because we are not concerned about computation time.

The algorithm will return a result for the M-estimator problem which cannot be directly compared to ML using existing strategies. Instead we choose to take a look at the root mean square error (RMSE).

$$\texttt{RMSE} = \sqrt{\frac{\left\| \tilde{X^P} - (X^P)^{\star}_{(\texttt{Alg})} \right\|^2}{\texttt{dim}(X^P)}} \tag{5.10}$$



(a) PGR(MO) 0%

(b) PGR(MO) 20% Outliers

(c) PGR(MO) 50% Outliers

(d) PGR(MO) 70% Outliers

(e) PGR(MO) 90% Outliers

Figure 5.10: The following list of figures depicts the final estimate from using **PGR(MO)** at $k = 1$. Solving for a feature based SLAM problem with outliers. The RMSE's are respectively for, 0% = 0.1151, 5%=0.0987, 20%= 0.1448, 50%=0.0925, 70%=0.3035, 90%= 8.4773. ML pose graphs RMSE = 0.1,

Figure 5.10 displays a single run of **Loop** trajectory. The default set of relative poses can be seen from Figure 5.10(a), additional relative poses in subsequent figures are in fact outliers. The RMSE remains close to the outlier free RMSE until around 70 percent. Only when the percentage become extremely large (90%) does the outliers start to influence the data and our method fails. PGR with robust estimation conveys an overall robustness towards observation outliers.

The primary cause of this impressive outcome comes from a number of factors. First, PGR naturally imposes a pre-filter on outliers by ignoring constraints that fail to converge onto the expected value, see Section 2.4.4. Second, the in-consistencies from the outliers, when they appear in pose graph form, are easily picked up by the M-estimator. Finally and most importantly, the odometry has greater influence on the structure of the graph

itself and is the dominating factor during convergence. Unlike graphs with features, where the observations play a more dominate role. Some of these factors may explain why many state of the art robust techniques have worked so well [16, 65, 69].

Our technique is very simple in nature but has not undergone rigorous evaluations that are expected from a robust SLAM Back-End. However we feel there is great value in investigating this idea further.

5.6 Summary

In this chapter we have set out to improve the efficiency of feature based SLAM by reducing the feature observations into pose graph representation **PGR**. To tackle the issue of information reuse we have proposed a technique known as Multi Observation **PGR(MO)**, dividing up the observation information heuristically. From experimental results we have demonstrated that not only is our method consistent but also made computation efficient through the concept of *Key Poses*.

The drawback to this additional approximation is that **PGR(MO)** becomes less accurate as information of features are slowly lost to coincide with the computation time reduction. Although poses are consistent, as supported by Section 5.4.1 the χ^2 ratio deteriorates as the variable k is increased.

We conclude that, it is only advisable to apply **PGR** when the feature densities are high or the efficiency condition (5.6) are sufficiently satisfied at lower values of k. Or else it is more advisable to use Sparse Map Joining.

In the final chapter, to investigate the two scalability strategies further, we will apply them to a real world scenario. Starting from the sensor data, we will arrive at a final representation of the completed map and estimate for pose locations. The aim is to build a complete framework which will encapsulate both the Front-End and Back-End elements of SLAM.

Chapter 6

Case Study: RGB-D SLAM

6.1 Introduction

Having demonstrated novel and improved ways of handling **Reliability** and **Scalability** in feature based SLAM, in this chapter we apply these concepts to a real world problem, RGB-D SLAM. This is when a single RGB-D camera is employed to map an indoor environment without the aid of any auxiliary sensors.

Recently, RGB-D sensors have become very common in the field of SLAM. The popularity stems from the fact that rich 3D and color information can be obtained at relatively low cost. This means that sufficient information is available to utilize a single RGB-D camera to perform SLAM. Unfortunately, the sensor quality, limited range and narrow field of view, restrict current SLAM algorithms to only operate under specific environmental conditions. We feel that the full potential of the RGB-D camera has yet to be explored and more optimal strategies should be developed.

Our aim is to develop a general framework for RGB-D SLAM which will handle multiple aspects of both the Front-End and Back-End. The final outcomes are to build accurate and consistent 3D SLAM maps even when faced with challenging indoor environments.

6.1.1 Related Work

A significant body of work in SLAM has gone into utilizing range and bearing cameras models. These can be linked to the pioneering work done by Nister et al [70], combining feature extraction and RANSAC matching to obtain accurate pose estimation using stereo cameras.

Newcombe et al [71] tackled SLAM by splitting tracking and mapping into separate problems. In this work the camera is tracked over a global model of the environment through an ICP algorithm. The environment model is made denser and refined over several observations such that tracking is improved. Unfortunately ICP is often unreliable causing the algorithm to lose track of the model being constructed. Furthermore, keeping track of a dense model can be computationally expensive. This is why current implementations of Newcombe's algorithm have been limited to operating inside very small and structured environments.

Henry et al [72] proposes a method to jointly optimize visual features. This approach is simplistic in nature but able to work very intuitively with the RGB-D camera models. During optimization, Sparse Bundle Adjustment (SBA) is applied to obtain the final maps. However, SBA assumes the errors in u, v and d are equally weighted which is incorrect given that depth is being measured separately from the RGB image and has its own unique uncertainty. Hence, Henry's method will not be able to accurately encapsulate the true uncertainty of features.

Other works into RGB-D SLAM have generally formulated themselves around the pose graph SLAM model [73, 74]. Nevertheless, we feel that the formal representation should be in fact feature based, which will ultimately lead us to more consistent and accurate solutions.

6.1.2 Motivation

First, we want to develop a well-rounded feature based SLAM system (Front-End and Back-End) for solving RGB-D SLAM. Once we have a good framework, injecting concepts such as, Reliable Optimization, Sparse Map Joining, or Pose Graph Representation becomes trivial. Given the extensive work done in computer vision to handle many of the critical issues in data association, the Front-End itself is not the major contribution in this chapter. Rather, our primary goal is to setup a *good* quality graphical model of poses and features, for a flexible Back-End.

Second, for all the methods mentioned in Section 6.1.1, sufficient depth information must be provided at all times to continue their functionality. Otherwise a failure state will be reached whereby the algorithms cannot recover. Rather than treating this as a sensor limitation, we feel the RGB element of the sensor allows us to develop more advance techniques to overcome this problem.

6.1.3 Chapter Overview

In Section 6.2, we will talk about RGB-D cameras and their associated sensor models. Section 6.3 lists the techniques used in the SLAM Front-End, including a description as to how they can be applied specifically to RGB-D sensors. In our SLAM framework, we propose two algorithms. The first is to solve RGB-D SLAM for the ideal condition (Section 6.4). The second algorithm is a hybrid mapping scheme to handle depth observability problems (Section 6.5). A final discussion and summary is given in Section 6.6.

6.2 RGB-D Cameras

One of the most recent developments in sensing technology has come in the form of 3D range cameras. The most popular of which include the Microsoft Kinect and Asus Xtion Pro, both derived from PrimeSense's patented technology [75]. The sensors are categorized as RGB-D cameras, due to the two image types that are outputted. A primary **RGB** based coloured image, represented by the three color channels $(Red, Green, Blue)$ and a secondary **Depth** image, where each pixel is representative of a range measurement. The camera can be further classified under active sensors [76] since the camera must constantly project a structure light pattern to calculate the depth values. Other popular RGB-D sensors include the Swiss Ranger, Kinect 2.0 and PMD, which work using time of flight principles.

In this thesis we have opted to use the Microsoft Kinect sensor, for both its accuracy and availability (low cost).

(a) Microsoft Kinect (b) Asus Xtion

(c) PMD Camera (d) Microsoft Kinect 2.0 (e) Swissranger

Figure 6.1: Range of 3D cameras

RGB-D Sensor Model

From our observations, we found that the range values in the Kinect becomes heavily discretized and sparse after 4 metres. The overall depth uncertainty grows exponentially and become very unreliable past this operating range. As a result, we have opted to truncate the depth image at the limit to avoid erroneous measurements corrupting the results. Another limiting factor in these cameras is the small field of view, 60 ° in contrast with many 3D lasers which are typically 180 °.

The RGB-D camera follows a very standard sensor model. We start by transforming the sensor into Euclidean space using the projection equation.

$$\hat{Z}^S_{im} = \begin{bmatrix} u_{im} \\ v_{im} \\ d_{im} \end{bmatrix} \tag{6.1}$$

$$\hat{Z}^F_{im} = \begin{bmatrix} \delta x^F_{im} \\ \delta y^F_{im} \\ \delta z^F_{im} \end{bmatrix} = \begin{bmatrix} (u_{im} - c_x) * d_{im}/f_x \\ (v_{im} - c_y) * d_{im}/f_y \\ d_{im} \end{bmatrix} \tag{6.2}$$

Here c_x, c_y denotes the principle points and f_x, f_y are the focal lengths. All elements can be found in the camera calibration matrix (K) obtained from the pre-calibration of the RGB camera. Then the sensor model is then defined by

$$\hat{Z}^F_{im} = h_{im}(X^F_m, X^P_i) + \omega \tag{6.3}$$

The noise is assumed to be Gaussian $\omega \sim \mathcal{N}(\mathbf{0}, \Sigma^F_{im})$ and the precision of the depth is roughly 3cm at 3m. The covariance in the sensor readings have been estimated to be

$$\hat{\Sigma^S}_{im} = \begin{bmatrix} 1^2 & 0 & 0 \\ 0 & 1^2 & 0 \\ 0 & 0 & 0.03^2 \end{bmatrix} \tag{6.4}$$

Using the Equation (2.8), we can successfully obtain the associated covariance/information matrix $\hat{\Sigma}^F_{im}$ of all feature measurements defined by the transform Equation (6.2).

Luckily the correlations between the RGB image and Depth image $(u, v)_{rgb} \rightarrow (u, v)_{depth}$ have already been provided by the *OPENNI* drivers[1]. Therefore, all sensor measurements are already fixed to the RGB camera reference frame. Given that we do not have any additional transforms to robot or other sensors, the *poses* are now representative of the RGB camera location.

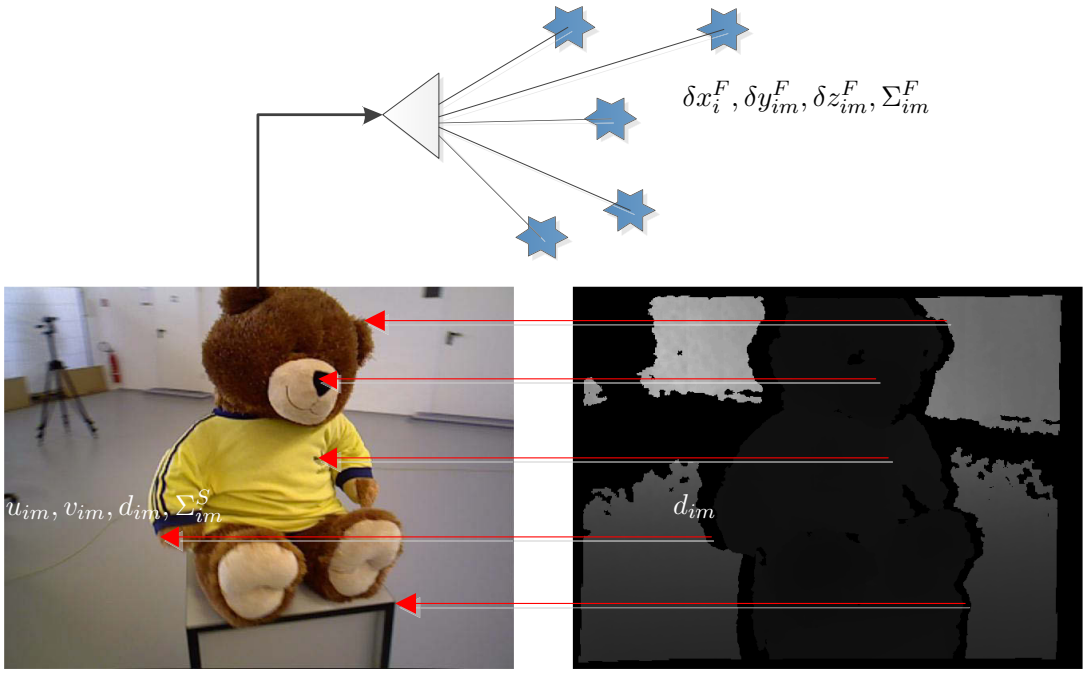


Figure 6.2: Camera model of the Microsoft Kinect Camera, (Left: RGB Image Right: Depth Image). Red Arrows illustrates the one to one pixel correlation of d values. u, v, d which are then projected into Euclidean space through the projection equation. The grayscale intensity is representative of the depth value of each pixel.

[1] see openni (under registered depth)

6.3 Handling the RGB-D SLAM Front-End

In previous chapters, limited information is provided about the SLAM Front-End, due to this aspect being very sensor and problem specific. For our RGB-D SLAM Front-End we have adopted some well-known and state of the art techniques in computer vision and range imaging. The following section details these techniques.

6.3.1 Feature Selection

In selecting the feature type, we have opted to use appearance based over geometric. The reason is that more characterizations are available in image appearance given the fidelity of edges, corners and areas of intensity change. Geometric features are often lost inside the noisy measurements as a result of depth image inaccuracies. In the general case it is far easier to extract more appearance features than that of geometric features in one given scene.

The robustness of an appearance based extractors often depend on its scale and rotation invariance. Without going into specifics, below is a table of key properties for three popular extractors.

Table 6.1: Feature Extractors

Feature	Scale Invariant	Rotation Invariant	Feature Density	Efficiency
Harris Corners [77]	No	No	High	High
SIFT [78]	Yes	Yes	High	Low
SURF [79]	Yes	Yes	Moderate	High

Due to the simplicity of descriptors, Harris Corners lack the robustness when matching scene scale or rotation invariance. This is especially critical, as we will see later, in solving the loop closure problem. Algorithms such as SURF or SIFT assign multi-dimensional feature vectors describing the orientations of surrounding pixels. SURF features are very efficient to compute; however, better quality maps can be achieved if we maximize the total number of obtainable features. Thus our preferred feature type is SIFT.

6.3.2 Feature Matching

Once features are identified we must now find a method to associate them across multiple camera frames.

Descriptor Matching

Before performing robust matching and outlier removal, a heuristic matching scheme can be applied to directly match feature descriptors. K Nearest Neighbor [80] is a popular choice in this case, matching based on the Euclidean distance of descriptors. Matches are rejected if the distance is greater than a fixed threshold (0.5m). Although this may still lead to many wrong matches, using the information of descriptors can greatly reduce the search space for more rigorous algorithms.

Random Sample Consensus

Random Sample Consensus, or RANSAC, is a well-known method in computer vision. First introduced by Fischler and Bolles [81], the idea is that at each iteration of this algorithm a set of *random* points are sampled, defining the overall model (transform) of all the remaining points. Several hypotheses are tested until the transform with the lowest error residual or the largest number of inliers is obtained. In RANSAC theory it is widely known that the minimum number of iterations (k) can be found up to a given probability represented by

$$k = \frac{log(1-p)}{log(1-(1-v)^s)} \tag{6.5}$$

Where p is the probability of the RANSAC algorithm to only select inliers from the dataset. v is the probability that, within a sampled number of points (s), a single match can be an outlier. Common parameter values to set are $p = 0.99$ and $v = 0.7$.

An additional condition is to stop when we have reached an acceptable number of inliers (T). A rule of thumb is to terminate based on the proportion of assumed outliers.

$$T = N(1 - \epsilon) \tag{6.6}$$

Here, N is the total number of matches. A conservative value for the probability ϵ is 0.2.

Once the inliers are found an optimal model function can be calculated by properly formulating least squares over the inliers. This process is helpful when a more accurate initial estimate is needed to solve the SLAM optimization problem. However, we have found that a closed form estimate from the inliers to be adequate.

There are two RANSAC algorithms applied, each catered towards a specific data association scenario.

Re-projection Error (RE-RANSAC)

Re-projecting 3D data into the 2D projection space is common in feature matching [72, 82] Re-projection Error RANSAC involves finding the distance squared error between two set of 3D points reprojected into camera space. This technique only works when image features also have associated depth information. First $\hat{Z}^F_{im} = (u_{im}, v_{im}, d_{im})$ is kept in the projection space. Then the function f maps the observed features from pose i to the theoretical measurement $\bar{Z}^F_{jm}$, observed from pose j. This occurs all in Euclidean space.

$$\bar{Z}^F_{jm} = f(T_{ij}, \hat{Z}^F_{jm}, \hat{Z}^F_{im}) \tag{6.7}$$

One can easily obtain transform T_{ij}, from image i to j, by combining Horn [67] with the re-projection Equation (6.2). First we project the measurements into Euclidean space and find T_{ij} with $\texttt{Horn}(\hat{Z}^F_{im}, \hat{Z}^F_{jm})$. Then $\hat{Z}^F_{im}$ is transformed by T_{ij} and re-project back into the camera space to obtain $\bar{Z}^S_{jm}$. Equal weighting is assumed to simplify the problem and hasten the calculations. To find the inliers, the residual r (in image space) of RE-RANSAC is given by the sum squared error for each feature m, which is exactly the re-projection error.

$$r = \sum_{m} \left\| \bar{Z}_{jm}^{S} - \hat{Z}_{jm}^{S} \right\|^{2} \tag{6.8}$$

Essential Matrix RANSAC (EM-RANSAC)

To find the inlier matches given two monocular images, Nister [83] proposed a 5 point method to find the solution of an essential matrix. Since the essential matrix (E) is directly related to the fundamental matrix (F), if the calibration matrix (K) is pre-determined, optimizing for E will also lead to the optimum F.

The algorithm computes a 10th degree polynomial to find the roots of the function in a closed form manner. One important factor that this algorithm considers is the enforcement on the calibration parameters. In doing so it is able to overcome degeneracy from planar or near planer scenes.

Without explicitly detecting the degeneracy, the 5 point algorithm assumes that the plane structure is finite and has a cheirality constraint [83] whereby there exists a unique solution for the essential matrix. Another method was proposed by Torr et al [84] which switches between a homography model and a fundamental matrix model to handle degeneracy. We feel that the 5 point algorithm provides a smoother transition given that the calibration matrix is known.

The camera motion is estimated by two corresponding homogeneous image point sets $\hat{Z}_{im}^{S} = (u_{im}, v_{im}, 1), \hat{Z}_{jm}^{S} = (u_{jm}, v_{jm}, 1)$ related by the fundamental matrix F.

$$(\hat{Z}_{jm}^{S})^{T} F \hat{Z}_{im}^{S} = 0 \tag{6.9}$$

Given the calibration, the essential matrix E has the relationship

$$K^{-T} E K^{-1} = F \tag{6.10}$$

The essential matrix has the property to represent both translation and rotation up to a

scale. So to be a valid matrix, it must also satisfy the following condition

$$2EE^TE - tr(EE^T)E = 0 \tag{6.11}$$

Given the Equations (6.9) and (6.11), the essential matrix is then solved through Nister's 5 point algorithm [83]. To find the inliers, we will need to calculate the residuals of any given sample of E. Simple algebraic distance has no geometric significance, and thus the Sampson distance [26] is used instead. This distance is a first order distance error approximation on the perpendicular distance to the epipoles.

6.3.3 Iterative Closest Point (ICP)

Since there is no odometry when using only a single RGB-D camera, we propose a "virtual odometry" constructed by Iterative Closest Point (ICP).

ICP attempts to finds the minimum difference between two point clouds and optimizes using a cost function expressed by the *closeness of points and/or planes*. The algorithm itself is iterative and stops once the solution converges. For our ICP algorithm we employ the state of the art Generalised ICP [85], which is a hybrid implementation between point-to-point and point-to-plane. We also found that point clouds can be down sampled by a factor of 3 to speed up processing without general loss of accuracy [86]. The Generalised ICP algorithm is publicly available [2].

A major limitation of ICP, is the assumption of *good* environmental structure. However, the limited field of view in the sensor can easily cause problems. For very planar observations the sliding effect [87] is a known phenomenon and the reason why scenes that have clutter are often considered extremely beneficial. Another limitation of ICP is its inability to provide an accurate estimate of uncertainty. Unlike wheel encoders or IMU systems, this odometry has no direct bearing on real world sensor data, rather it is an estimate of the most probable configuration given all the information in two depth images.

In our idea, ICP is restricted to only odometry estimates. To make sure if the "virtual

[2] http://www.robots.ox.ac.uk/ avsegal/generalized_icp.html

odometry" can be trusted, we compare the ICP result to the best transform found by RE-RANSAC. If the difference is large, we make the assumption that ICP has most likely failed and proceed to ignore the measure for odometry at that pose.

6.3.4 RGB Visual Odometry

Visual odometry in monocular SLAM is only essential for the initial estimate of the states. This is because pose locations are inferred from the features themselves through the optimization of re-projection error in Bundle Adjustment (BA).

To obtain the visual odometry from EM-RANSAC, one must decompose the essential matrices down to its fundamental elements of rotation R and translation T. SVD is the most common approach for this decomposition [26]. Note that E is only seen as a projective transform and is assigned an arbitrary scaling. Scale consistency between poses and features is only obtained after BA.

6.3.5 Initializing a New Pose

New poses are initialized by either ICP or Visual Odometry. However, using every pose of each camera frame can rapidly increase the size of the state vector. Therefore we apply a common technique to only update the pose state when the camera has moved or rotated a fixed distance. This distance travelled is simply tested by analyzing the ICP changes between the current frame and the previous pose frame. In visual odometry, given that scale is arbitrary, we instead apply Optical Flow [88] to judge the camera motion.

6.3.6 Initializing a New Feature

Although features are important in defining our map, just like poses, placing every feature into our state vector is unnecessary. To associate features through continuous poses (images), we will track their ID's. If the feature is matched in at least **3** images it is assigned an ID and initialized to the state vector. Selectively picking features based on their information gain can greatly reduce the total complexity. Our simple heuristic has

the capability of selecting features with information gain in mind but more comprehensive methods have been investigated in literature [89].

3D features are initialized by concatenating the first observation on the pose estimate that the feature was observed from. The harder problem is to initialize features without depth information. A common approach is to apply linear triangulation [26] on the observed points whilst transformed over pose estimates obtained from visual odometry. The resultant estimate of features locations are scaled over the decomposed essential matrix. However, there may exist some ambiguities in this approach, causing features to appear behind the camera or very far away due to parallax. Our current solution for these ambiguities is to remove these feature entirely; however, better initializations method have been proposed in literature, refer to Section 6.6.

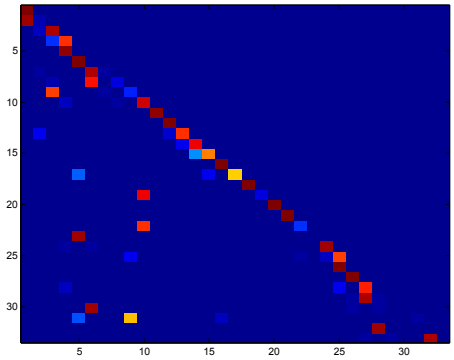
6.3.7 Loop Closing

Loop closure is handled via the *key frame* selection approach [72], during which frames for each camera poses are checked for scene difference. The first frame at pose 1 is selected as the first key frame, then every new frame is compared with the current to check for feature matches (RE-RANSAC). If the feature matches fall below a set threshold a new *key frame* is created and set to the current.

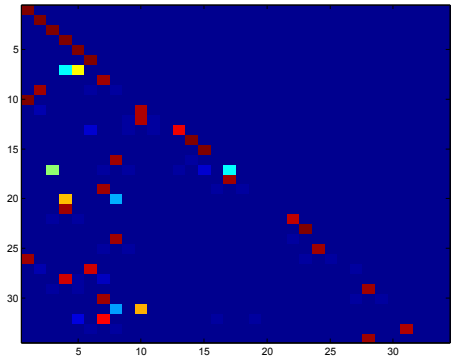
Once all the *key frames* are found, they are then fed into the Fast Appearance Based Mapping (FABMAP) algorithm [90] for place recognition [3]. FABMAP has become an extremely popular and efficient way of performing image based loop closing in SLAM. The power lies in the *bag of words* approach, where features are quantized into *vocabulary*. The vocabulary in our case is representitive of the SIFT features. The result is a probability matrix that determines if two observations have come from a similar location. From the most probable list of matches, we perform an additional RE-RANSAC to re-associate the feature, forming the loop closure in our feature based SLAM graph. Figure 6.3(a), conveys the probability matrix for two datasets and an example of the matches at one of the matched key frames.

[3] website: http://www.robots.ox.ac.uk/ mjc/Software.htm

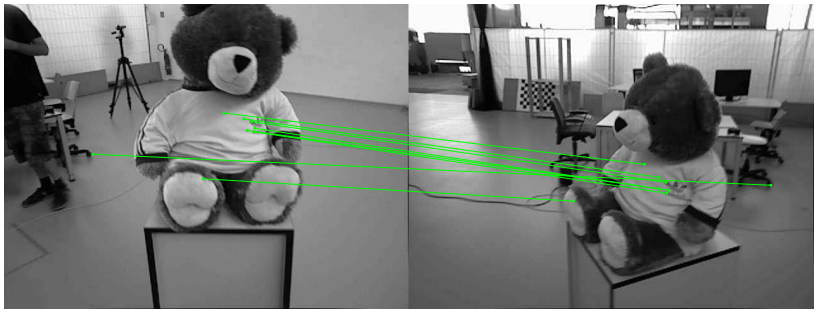
The advantage of using SIFT features as the vocabulary and the matching, is that scale and rotation invariance allows for association on scenes with very different viewing orientations. Figure 6.3(c) highlights this.



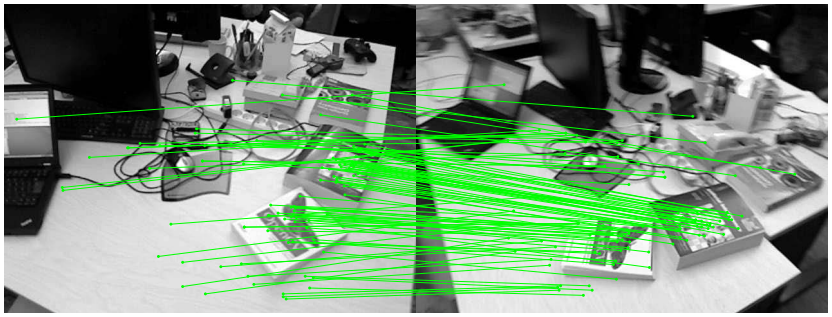
(a) `FABMAP probability matrix Teddy`



(b) `FABMAP probability matrix Room`



(c) `Loop closure match Teddy`



(d) `Loop closure match Room`

Figure 6.3: An example of loop closing using Key Frame

Using FABMAP avoids the need to perform a search with quadratic complexity across all the poses and greatly increases the efficiency of loop closing.

6.4 RGB-D SLAM

Often in the case of RGB-D SLAM, researchers have chosen to represent the complete problem in terms of pose graph [73, 74], disregarding the features entirely and instead opting for relative poses created by ICP as loop closures. Since scan matching is applied to evaluate the constraints, the information loss is justified by the strength or low uncertainties given to relative poses.

We identify three major downsides in these approaches. First, ICP has many defective properties when inferring feature observations into relative pose constraints. This is mainly caused by the limited view angle of the sensor resulting in insufficient observations of geometric structures.

Second, without the addition of features in the optimization problem it is less likely to find good consistency in the poses estimates. The reason being that sensor data uncertainties are not accurately represented in the optimization problem. Consequently, in these pose graph approaches there is no way of handling information reuse which may result in overconfident covariances.

Lastly, by not explicitly optimizing the features there is no existing representation of the map at the end of optimization. The maps are therefore created from sensor data projected over the estimated poses, which may be visually attractive but impractical to exploit in techniques such as localization.

Our RGB-D SLAM algorithm starts by formulating the complete feature based SLAM problem first by tracking and associating the features in the Front-End. From this we can start to make improvements on Back-End with Reliable Optimization, Sparse Map Joining (SMJ) and Pose graph representation (PGR). The end result is a consistent graph with near optimal estimates on the given datasets.

6.4.1 Flow Chart of RGB-D SLAM

The following flow chart represents how the individual steps are linked together.

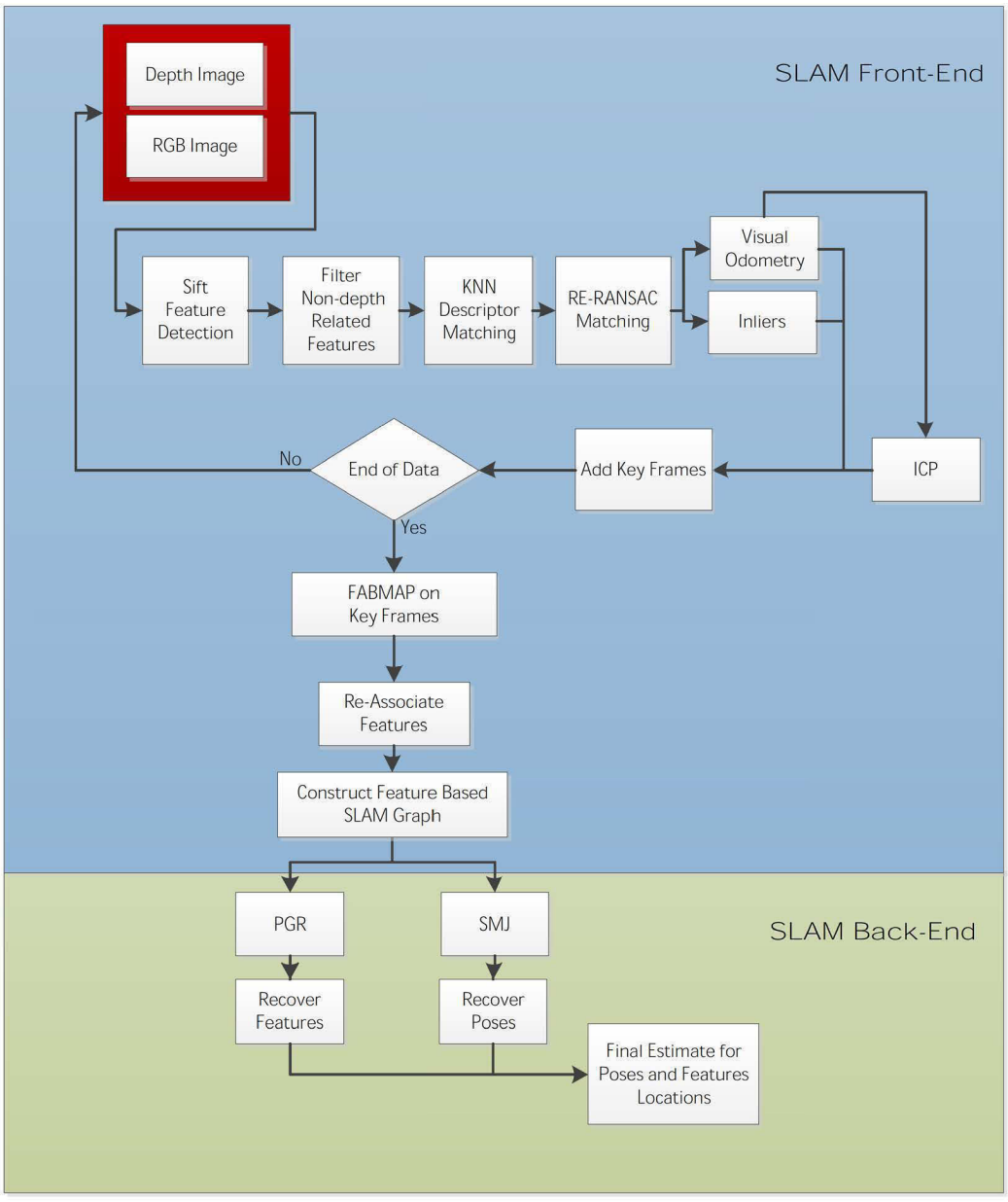


Figure 6.4: Flow chart for RGB-D SLAM

Front-End and Back-End

In Figure 6.4, we denote the clear division between the Front-End and the Back-End components in a complete SLAM algorithm. The Front-End is a sequential process of Extraction, Filtering, Matching and the obtaining of virtual odometry. Once the dataset is fully analyzed, loop closure is performed to re-associate features in each key frame match. From all this information, a graph is constructed and then passed into the SLAM Back-End.

From previous chapters, we asserted that the Back-End can be solved in multiple ways. We will be testing our two proposed methods (SMJ and PGR) for both accuracy and efficiency.

6.4.2 Experiments and Results

The dataset we have chosen comes from the Technical University of Munich[4] [91]. They have provided 3 distinct trajectories each with differing properties. **Teddy** is a dataset which closes a single loop, reconstructing the shape of a single object. The **Desk** dataset is a curved trajectory that doubles back to its original position closing several loops. **Room** is the largest of the three datasets, mapping out an entire office space closing several loops and forming a complex trajectory.

In these RGB-D datasets, not only are they well documented, in addition the ground truth has been obtained through a Vicon system[5] with sub-millimeters accuracy. To avoid any failures due to sensor limitations, the datasets selected consist of well defined scene structures and good availability of depth information.

Our preferred evaluation methods are Absolute Trajectory Error (ATE) and Relative Pose Error (RPE), proposed by J. Sturm [91]. ATE is very straight forward as it first aligns the estimate with ground truth and then checks for the overall translation error, allowing for a good visual inspection on the overall result. RPE on the other hand, takes the difference between the estimated motion and the true motion. This is especially useful

[4] http://vision.in.tum.de/data/datasets/rgbd-dataset/download
[5] http://www.vicon.com/

for examining loop closures and visual odometry. The Δ parameter lets us set how much drift is incurred after a fixed motion (Δ is set to 0.5 meters in our case). RPE also evaluates over both rotation and translation which is not the case in ATE.

Table 6.2 provides a quick summary of the datasets acquired from the Front-End and Table 6.3 details the results after Back-End optimization. The ML solution has been omitted from the results due to the large computational complexities associated with RGB-D SLAM.

Table 6.2: Summary of Datasets

Dataset	no. Poses	no. Features	no. Feature Observations
Teddy	325	60150	183480
Desk	286	32859	156583
Room	676	59223	274393

Table 6.3: Result of RGB-D SLAM

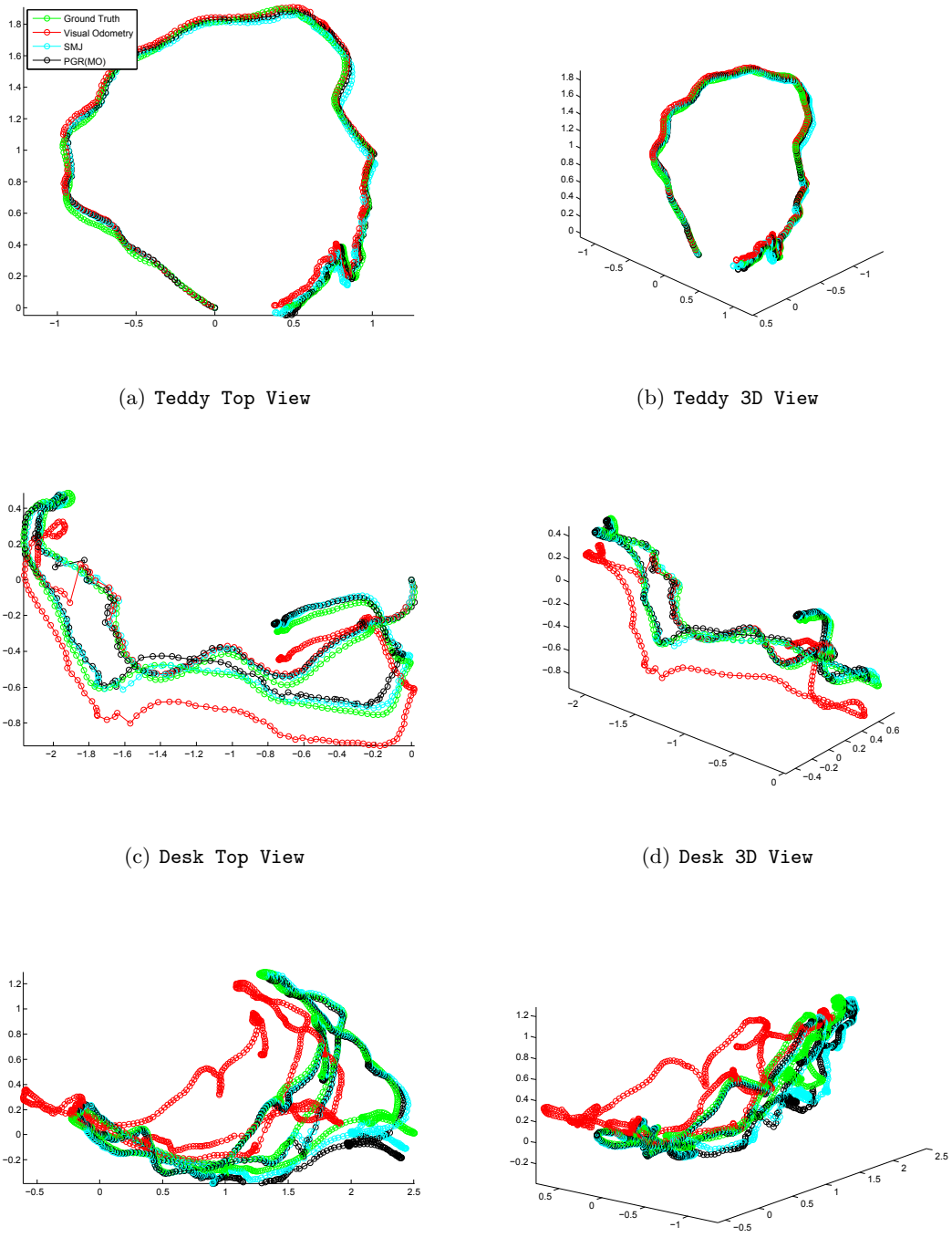
MAP		ATE	RPE	Total Time for Optimization (s)
Teddy	Visual Odometry	0.024	0.020	-
	SMJ	0.016	0.022	2818.8
	PGR(MO)	0.025	0.020	748.7
Desk	Visual Odometry	0.115	0.044	-
	SMJ	0.025	0.032	1521.1
	PGR(MO)	0.038	0.044	356.4
Room	Visual Odometry	0.207	0.065	-
	SMJ	0.069	0.056	5393.4
	PGR(MO)	0.077	0.058	3170.6

From the results, there is an indication that the visual odometry for a single loop closure trajectory, **Teddy** has only small drift. Therefore, optimization does not provide much improvement on the poses through optimization. Also there is very little rotation in the trajectory itself which may also explain the small errors.

For the datasets, **Desk** and **Room**, there are significant improvements on the poses. Both SMJ and PGR have much lower ATE's and RPE's compared to visual odometry. Due to the multiple loop closures and complex trajectories visual odometry quickly deviates from the ground truth. In terms of computation time, PGR is the better algorithm due to the dense population of features in the problem, although the approximation makes it slightly less accurate compared to SMJ.

Upon visual inspection on Figure 6.5, we can see how the optimization result differs from ground truth. We have also performed a point cloud overlay on the SMJ final pose estimate to reconstruct the scene in Figure 6.6.

Visual comparison to ground truth



(a) `Teddy Top View` (b) `Teddy 3D View`

(c) `Desk Top View` (d) `Desk 3D View`

(e) `Teddy Top View` (f) `Room 3D View`

Figure 6.5: Pose trajectories after optimization compared against the ground truth measurement.

Point Cloud Reconstruction

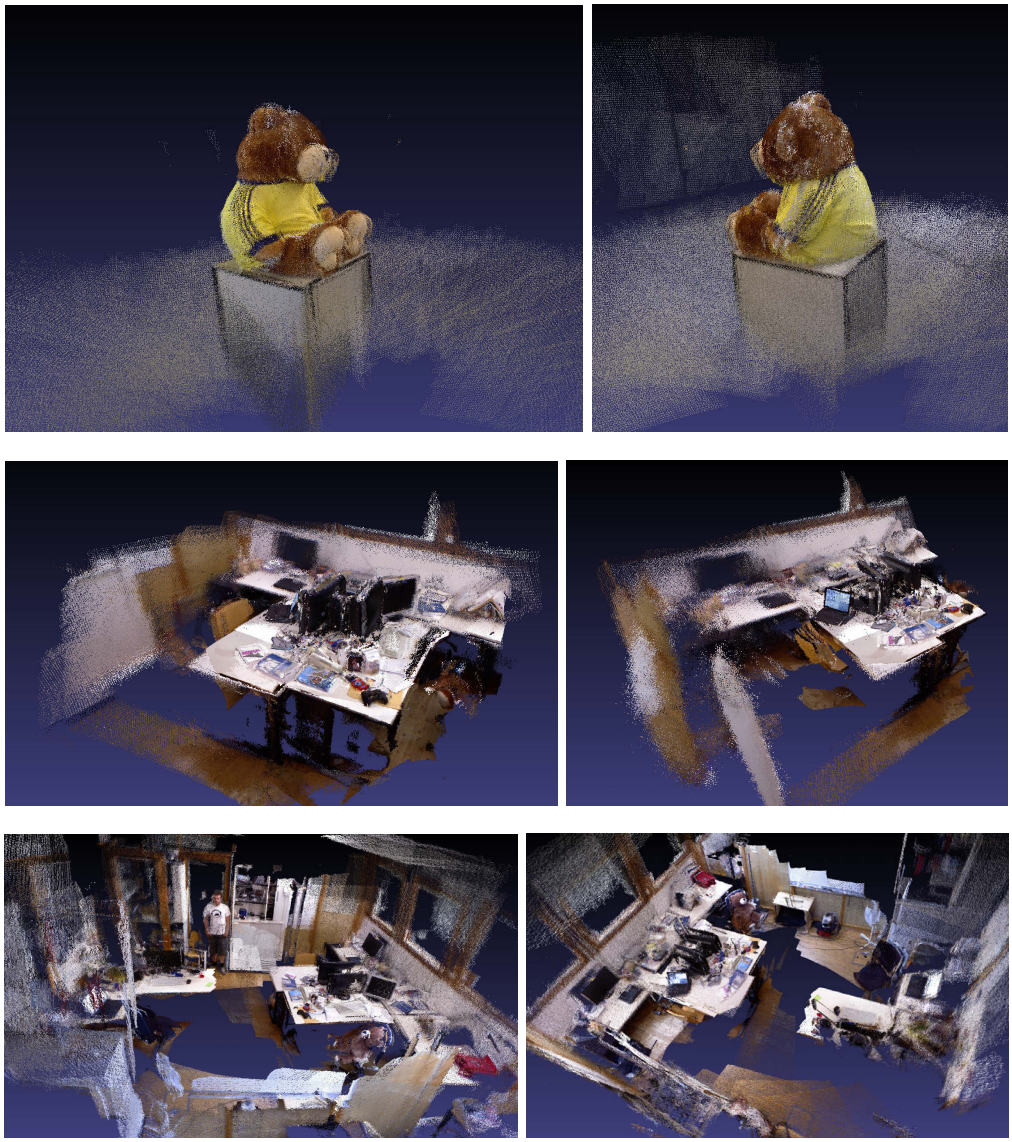


Figure 6.6: 3D point cloud overlay, Top: Teddy, Middle: Desk, Bottom: Room

Currently, our visualization strategy is to overlay the point clouds. More advance techniques such as Surfel representation [72], use surface normals to cluster the points, reducing the overall point density for faster rendering. Generating meshes is another possible alternative for reconstruction. Given sufficient memory, a dense mesh of the environment can be recovered with high fidelity [92]. We would like to look into these visual representations further, but they are not considered a major priority.

6.5 Robust RGB-D SLAM

In our previous RGB-D SLAM algorithm, there exist several cases where RE-RANSAC will fail. The most detrimental is the encountering of poor or no depth observability. This is common when features are too far away from the sensor or lie on a medium which the sensor cannot detect (e.g., Low/High reflectivity, transparency). Another major flaw of the sensor is the saturation of ambient infrared light, often occurring when the sensor is taken outdoors in daylight.

Our second proposed algorithm tries to rectify these issues by applying Sparse Map Joining combined with a Dual-Observation Model approach, refer to Section 4.3.3. From this idea, we know that it is theoretically possible to join two local maps of differing observation models. More specifically, a 3D feature based map and a bundle adjustment map.

In this section we will focus on the associated steps in developing this robust RGB-D SLAM algorithm and demonstrate experimentally on a dataset that has been exposed to these failure conditions.

6.5.1 Local Map Building and Joining

Local maps that have depth information are built following the standard range and bearing technique detailed in Section 4.3. For local maps that are bearing only, there exist several proposed algorithms for solving the bundle adjustment problem. Our implementation applies the popular Sparse Bundle Adjustment idea (SBA) [60], where features are parameterization in xyz space and Levenberg-Marquardt acts as the solver. Depending on how reliable the Front-End is (good initial estimates, correct data association, etc.), SBA converges reasonably.

6.5.2 Local Map Switching

To determine which local map is most appropriate for a given sensor reading, we must derive a scoring mechanism.

It is well-known, that the quality of visual odometry can be assessed either through algebraic means (quality of fitting a model, etc.) and/or through the geometric properties of a scene (plane structures, scene clutter, etc.) [84].

To plan how to switch between the two local maps, an analysis is made on the failure modes. A critical assumption is that we will always have enough observed features for at least one mapping approach to continue functioning. In reality, due to various configurations, environmental factors and sensor failures, poor quality features can violate this assumption. We have wisely chosen a dataset that avoids this problem.

Switching between RE-RANSAC and EM-RANSAC

Given that both RANSAC methods continue to operate, precedence of visual odometry is always given to RE-RANSAC. The reasons being that the RGB camera on the Kinect and Xtion are of poor quality (resolutions of 640 by 480 pixels) and therefore any measurement of features far from the camera location are subjected to larger noise errors. In addition, RE-RANSAC gives us extra geometric information through the depth sensor which provides us with better constraints in our models.

Our monitoring system needs to find the possibilities scenarios where RE-RANSAC is likely to fail, so that EM-RANSAC can take over. The following is a table listing what we think are the critical failure conditions.

Table 6.4: RE-RANSAC Failure Modes

Failure Cases	Description
Depth Drop Off	Little to no depth values associated causing RANSAC to completely fail.
Feature Clustering	A RANSAC estimate that is centered around a clustered region will return a poor estimates of visual odometry.
Planar Structure	If the geometric structure in the features are poor, ICP is likely to behave poorly due to weakness in constraints.

From the three failure cases, we have imposed three simple heuristics to monitor the feature quality in RE-RANSAC. The first problem is that of **Depth Drop Off** which is solved by applying a simple threshold on the minimum number of inlier from RANSAC. γ_1 is the threshold value and n is the number of inliers.

$$n < \gamma_1 \tag{6.12}$$

For **Clustering**, a second threshold called the cluster score α is applied. First the centroid C is calculated from image inliers, then α is found using a distributed average over the distance to C,

$$\alpha = \frac{1}{n} \sum_m d_1(C, \hat{Z}_m) \tag{6.13}$$

where d_1 is the distance function between two points and our new threshold becomes $\alpha < \gamma_2$. Finally for **Planar Structure**, first we assume that the scene only consists of a single plane and calculate the normal. β is found from the average inlier residuals to the plane normal P^N and points on the plane P^X.

$$\beta = \frac{1}{n} \sum_m d_2(P^N, P^X, \hat{Z}_m) \tag{6.14}$$

where d_2 function calculates the perpendicular distance from point to plane. The last threshold is then $\beta < \gamma_3$. If any thresholds are violated, a switch is made to start using EM-RANSAC. Only when all three criteria are satisfied over a set number of frames does switch back occur. This is to prevent noisy images from corrupting the switching process.

6.5.3 Flow Chart of Robust RGB-D SLAM

The major difference between this new algorithm and the one mentioned in Section 6.4 is the introduction of the switching system. This system monitors the visual odometry and appropriately selects the correct sensor model to use.

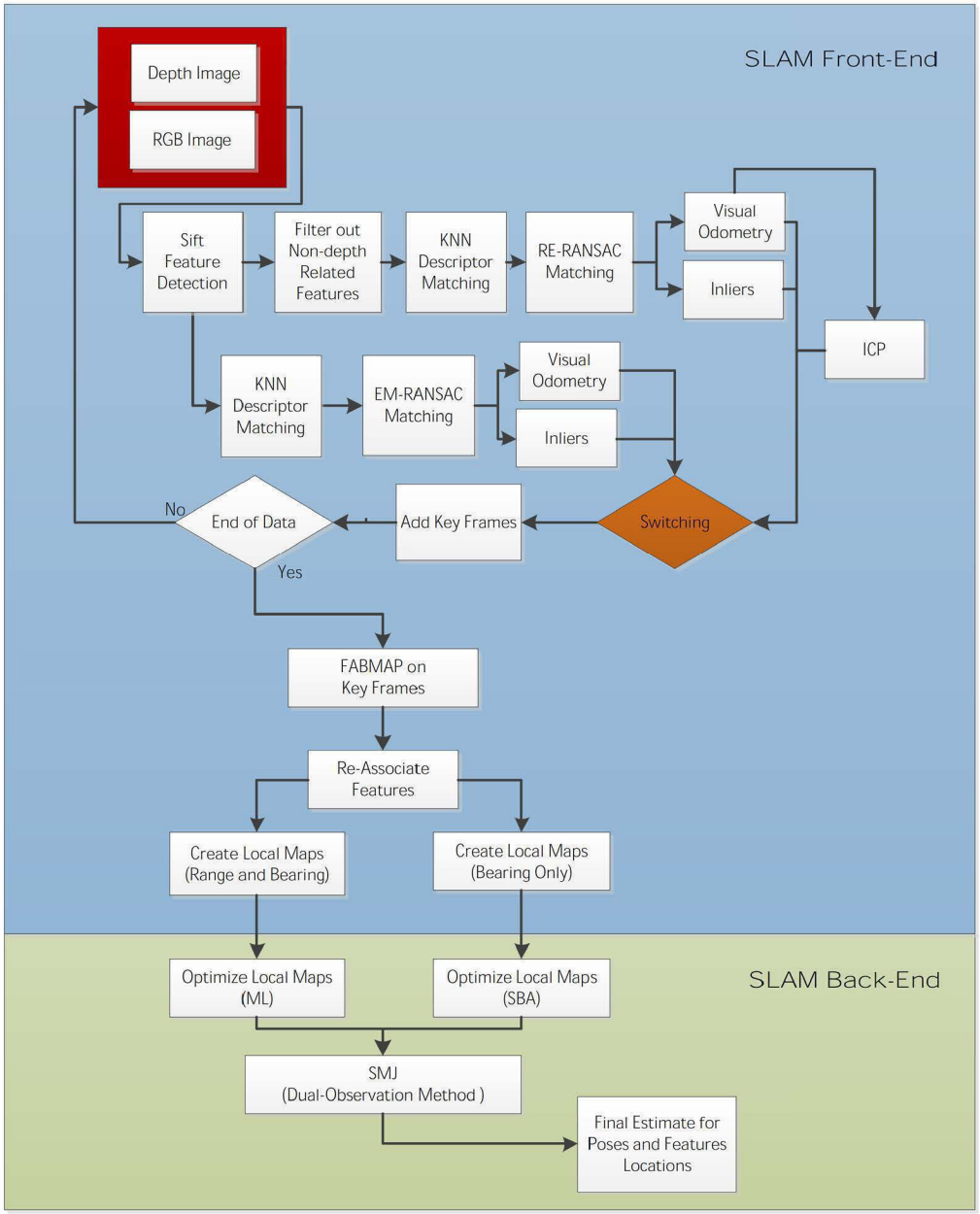


Figure 6.7: Flow chart for RGB-D SLAM

After all the features have been associated, the graph is separated into local maps based on the current sensor models along the trajectory. The maps are then solved independently by either Maximum Likelihood or SBA accordingly.

Joining can be performed through any SMJ optimization approach, although we prefer to utilize Batch Optimization. The dual observation model lets us rescale the entire map to the appropriate global scale given significant enough features overlap between the local maps. In SMJ we have the flexibility of keeping all the poses, or marginalizing them out according to Section 4.4.2. Keeping all poses will naturally compromise the efficiency, but can offer better reconstructions of the environment and avoid the pose recovery phase. In the following experiment, we have opted to keep all poses during SMJ.

6.5.4 Experiment

UTS Level 6 Dataset

The UTS level 6 dataset was collected on a narrow balcony, forming a loop around a multi-story building. During this trajectory, the openness of specific sections results in many features being observed far from the sensor. In addition, there are glass windows that cannot be directly detected by the sensor, contributing to the depth observability issues.

The trajectory itself is 25m across and 15m wide. A total of 1670 Kinect images were collected along the loop. In the visual odometry only 746 frames are used with 37 key frames. The total number of features is 11218 with 62319 features observations.

Visual Odometry

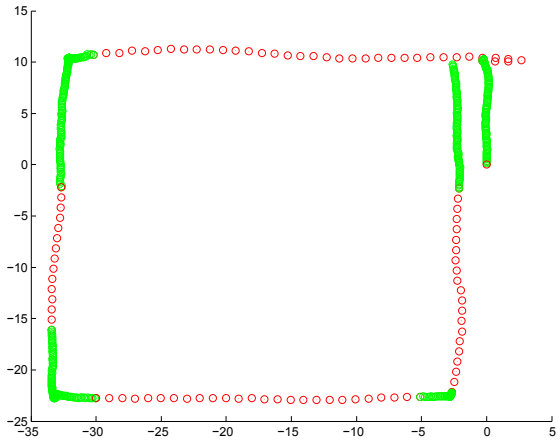


Figure 6.8: Initial estimate obtained from visual odometry. Anchor pose is set to $(0, 0, 0)$. Green: RE-RANSAC, Red: EM-RANSAC.

From Figure 6.8, we see how the inaccurate scale from EM visual odometry causes the initial value to be greatly different to the actual trajectory. We can also see the areas where the algorithm has decided to switch, indicated by the color change. The images in Figure 6.9, illustrate the situations where a switch was deemed necessary, namely the areas where there are glass walls or the environment is far from the camera.

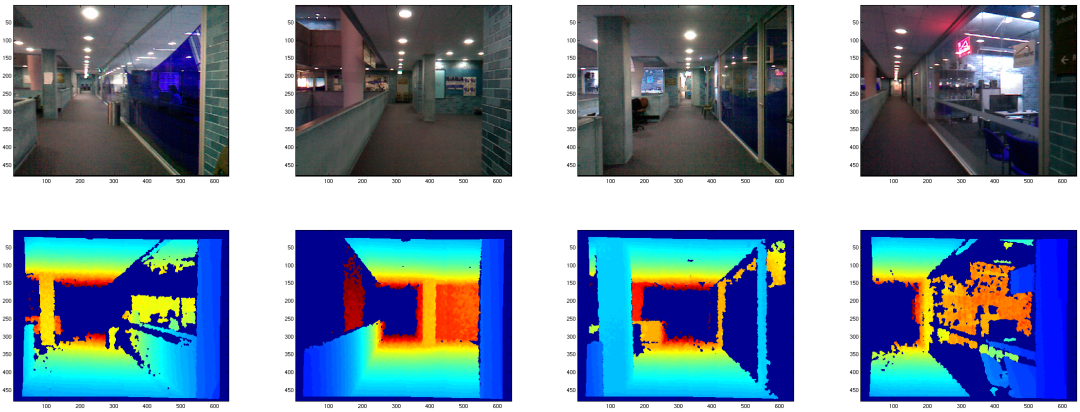
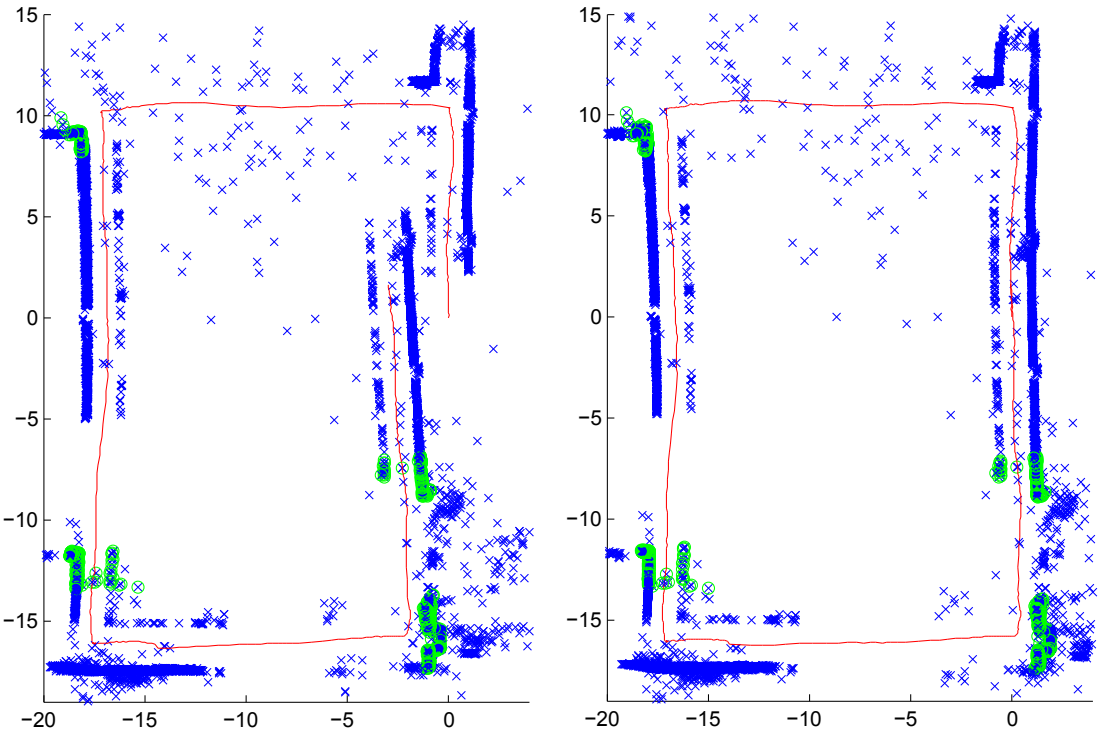


Figure 6.9: RGB and Depth images at the switching point, RGB-D → RGB

Optimized Graph (Scale & Loop Closure)



(a) `Before Loop closure` (b) `After adding Loop closure`

Figure 6.10: Map joining results from SMJ's dual observation method. Red: Camera Poses, Blue: Feature Locations, Green: Common Features between local maps.

After joining, the scale in all the RGB Local maps are corrected (Figure 6.10(a)). If we incorporate loop closuring (Figure 6.10(b)), the global map quality is improved. The major influence for the scale accuracy is in the overlapping features marked in green circles. We have found the scale to be most accurate when features are well distributed for good geometry. The most inaccurate local map scale exists in the 2nd RGB map (indicated by the second turn) where the overlapping features only exist on the left side of the camera. This is because the entire right side wall is glass. To estimate a good scale value, we must try to maintain good geometry and low feature uncertainty at the areas where the algorithm has decided to switch, or else SMJ joining is likely to fail.

Point Cloud Reconstruction

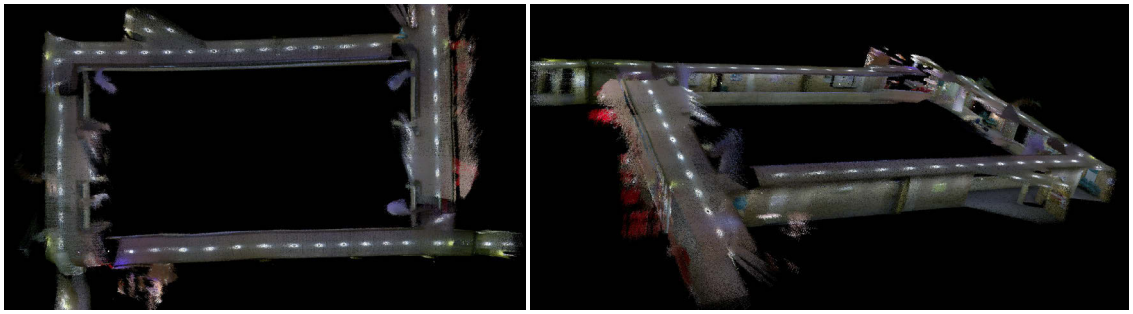


Figure 6.11: 3D point cloud overlay on pose estimate. Left: Top view, Right: Isometric view

Unfortunately no ground truth is available for the UTS dataset. The best evaluation we can achieve is to overlay the point clouds and compare against a 2D architectural floor plan. From Figure 6.12, we can see that the metric scale is correctly estimated and the boundaries of the floor plan are *close* to the boundaries of the point cloud.

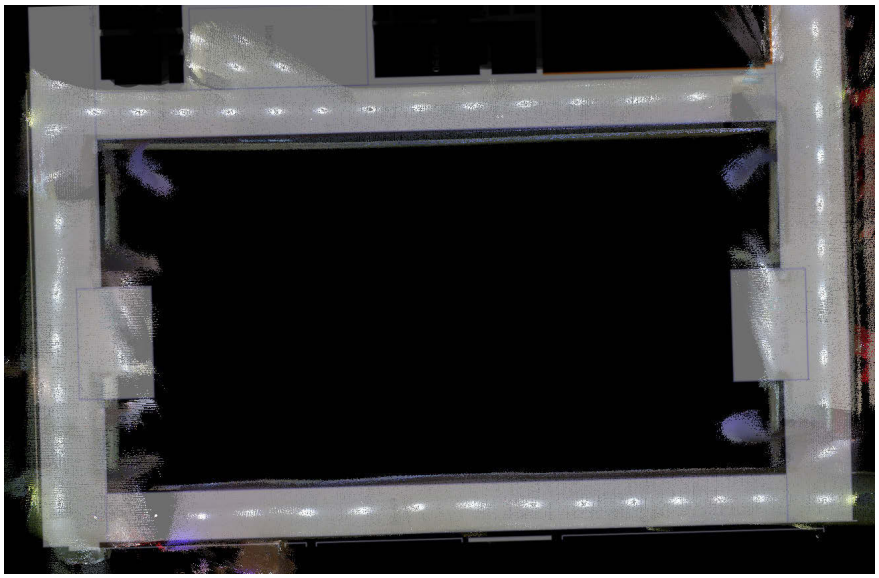


Figure 6.12: Point cloud overlay and architectural floor plan

6.6 Discussion

In this section we will discuss limitations that exist in our two RGB-D algorithms.

6.6.1 RGB-D SLAM

From what we have seen in the results, visual odometry in data with few loop closures is usually an adequate estimate of the given data. Applying additional optimization does not necessarily improve the incurred drift, even after associating many features together.

A major unsolved problem in both our algorithms, is the handling of outliers. An outlier handling approach was proposed in Section 5.5.3; however, reliance on a confident odometry is critical, such that the influence of outliers is devalued. We can only partially remove outliers with this approach in our RGB-D framework. This is because ICP is often unreliable in the Front-End process resulting in no odometry information at many steps. In future, we would like to explore methods to reduce the number of outliers in the Front-End as well as a robustified Back-End that does not need to rely on odometry.

6.6.2 Robust RGB-D SLAM

In Robust RGB-D SLAM, failures may be caused by a violation of a few underlying assumptions. These assumptions are discussed in turn. In our scheme, it is assumed that the model is a perfect pinhole camera. In reality, factors such as lens distortion and poor camera calibration can violate this assumption. In the Kinect Camera, the RGB sensor has a very limited sensor resolution and we cannot confidently say that our models are accurate. Thankfully, the lens distortion is relatively low, which means this can be disregarded as a cause of major error. As long as the environment can be kept indoors, very distant features with small parallax between frames can be restricted.

The uncertainties in the feature estimates associated between local maps should not be overlooked. A major failure case is when the camera starts to turn and loses depth information during or at the end of a turn. At this instance the monocular mapping scheme cannot gain enough observations about the overlapping features, resulting in high

uncertainty for those feature locations. As a result, the process of optimization will become unstable, either diverging or becoming singular. This is a common problem in monocular SLAM due to depth being inferred from bearing estimates, making it highly sensitive to noise and ambiguities.

From our experiment, we have found that common features found during the joining phase between RGB-D → RGB Map have the highest uncertainty. Typically when an RGB-D map starts to fail the common features are within sensor range. As the algorithm switches, the distant features in EM-RANSAC have larger influence on inliers causing the closer features (common local map features) to have less associations. Therefore, the failure case mentioned above is exhibited when joining occurs. In the results seen in Section 6.5.4, the scale is only estimated when joining RGB → RGB-D Map. During this transition, more common features are observed by the RGB Map to achieve a good estimation on scale.

As previously mentioned in the experiments, structure of common features also plays an important role in improving the estimate of scale. Typically clustering is a negative attribute and we must try to avoid this as much as possible. However, because we cannot always control the environment itself, there is no easy solution to this problem.

In the future, we propose to apply more robust monocular SLAM methods to counteract these issues. Due to the projective nature of the monocular cameras, a point on the image plane can represent an infinite depth value. Therefore, alternate parameterizations of Bundle adjustment are useful in these cases, e.g., Inverse Depth [93] or Parallax Angle [94]. In addition, data association and visual odometry may be improved by only selecting high quality features. For example, the quad tree method selects features based on their spatial separation [82]. Finally, we want to look into better triangulation methods for feature initialization, such as the one suggested by Klein and Murray [95].

6.7 Summary

In this chapter we have proposed two algorithms capable of utilizing only a single RGB-D camera for SLAM. One to handle the typical RGB-D style environments, and the other to handle environments where depth information is no longer observable due to sensor limitations.

We have designed a software architecture capable of handling both the Front-End and Back-End aspects of SLAM. The result is an estimate in which point clouds can be overlaid on poses for reconstruction or feature maps can be used in robot localization.

From a series of experiments conducted on real world datasets, the performance of our algorithms is evaluated. Given the large number of features in the problem, PGR is proven to be especially efficient.

The dual observation model lets us build scale consistent maps where there is poor depth observability along the length of a SLAM trajectory. Although, we have obtained *good* results, there is still much room for improvement.

Chapter 7

Conclusion and Future Work

In this thesis, we started by giving a brief introduction to SLAM and its applications. Then we formulated the problem as being feature based graph optimization, solved using non-linear least squares. SLAM itself can be further divided up into two individual parts, a Front-End and a Back-End. This thesis primarily focused on the latter, developing reliable and efficient solutions to the optimization problem. The main contributions have been detailed in Chapters 3, 4 and 5.

There are two main motivations being addressed in this thesis. The first is non-linearity, which causes convergence onto a local minimum for non-linear least squares based approaches. Since we always desire the exact Maximum Likelihood estimate, an unreliable optimization approach cannot be fairly evaluated or theoretically analyzed. The second motivation is scalability. As the number of robot poses and features increase, the computation complexity must still stay within reasonable bounds for a solution to be found.

7.1 Summary of Contributions

The following is a summary of the contributions and findings presented in this thesis.

7.1.1 Reliable Optimization

In Chapter 3 we proposed an algorithm that is able to achieve the global minimum of a non-linear least squares problem with improved *reliability*. The main insight is the importance of the initial estimate within Gauss-Newton. We started by presenting a general framework towards reliability (Section 3.2), then introduced the concept of Iterative Reweighted Least Squares bootstrapping to smoothly step through the conditions outlined in the framework (**C1,C2,C3**). One major advantage for our method is generalization, meaning that the same formulation can be kept across various SLAM variants. Tests have also confirmed that the IRLS bootstrapping algorithm can outperform other popular bootstrappers (TORO, LAGO and Spanning Tree), even under very noisy measurement conditions.

7.1.2 Sparse Map Joining

Our next contribution is the implementation of Sparse Map Joining (SMJ) for both 2D and 3D feature based SLAM. The sub-mapping process itself approximates odometry and features measurements as discrete local maps. The proposed method also takes advantage of pose marginalization to reduce the dimensionality, making the overall algorithm very efficient when compared to solving for the full Maximum Likelihood. In our experiments we test three sub-map joining procedures, Batch, Sequential, and Divide & Conquer. Each has their own way of solving the joining process, with Batch proving to be the most consistent and accurate. Depending on the time needed to solve each individual local map, increasing the number of local maps can also improve efficiency to a certain extent. Finally, by reformulating SMJ, we are able to solve problems in SLAM with differing observation models.

7.1.3 Pose Graph Representation

The third contribution describes a way to convert the feature based SLAM problem into a Pose Graph Representation. In doing so the dimension is reduced by not explicitly estimating the individual feature locations. However, converting the problem into a pose only graph may lead to information reuse. By dividing up observation information heuristically through **PGR(MO)**, we are still able to obtain the optimal number of relative poses while keeping the estimate consistent. In addition, we have also introduced a concept called *Key Poses* to handle the efficiency condition mention in Equation (5.6). If this condition is not handled properly, then the efficiency of PGR will be greatly compromised.

It is evident that PGR is not always recommended over SMJ given the greater loss of information (larger χ^2 ratios). However, for environments where the features are densely populated there is a significant boost in efficiency, making PGR the more feasible approximation in case.

In our discussion we detail other benefits of PGR, namely feature based SLAM problems in the presence of outliers. We demonstrate that a modified PGR algorithm is able to withstand even a 70% outlier rate.

7.1.4 Case Study

In Chapter 6, we have conducted a case study to examine the effectiveness of our techniques on a real world problem. By using a single RGB-D camera, we were able to construct two SLAM algorithms that take in sensor data and output the estimate for both feature and camera positions.

The first algorithm assumes a perfect environment where depth information is always available to the sensor. The Front-End then builds a pose and feature graph, optimized using either SMJ or PGR. We concluded from our findings that PGR is most effective of the two method. This is evident from the comparison of the efficiency and accuracy of the results against a known ground truth.

The second algorithm exploits the dual observation model, proposed in Section 4.3.3, to

overcome poor depth observability. This involves joining a Range and Bearing (RGB-D) map with a Bearing only (RGB) map. The idea is to jointly optimize scale alongside the other state variables. Given that the uncertainty in the overlapping features between local maps is low, we can accurately estimate a global scale value that can achieve metric accuracy in both map and poses. The evidence is supported through a real world experiment with unreliable depth observability.

7.2 Future Work

In this final section, we will provide insight into possible future directions as well as unforeseen challenges.

7.2.1 Improving the Reliability

From our experiments, after applying IRLS bootstrapping we can obtain a *high success rate* for convergence towards the global minimum. However, this can be further improved, especially for datasets with larger noise. In future work, we plan to investigate alternative forms of the influence function (e.g., dynamic influence), which may contribute to the reliability of our framework demonstrated in Section 3.2.

Currently we do not have a good theoretical basis on why the Cauchy M-Estimator ($\alpha = 1$) offers the *closest* estimate to the actual Maximum Likelihood solution. Therefore, there is no way of knowing when our algorithm can achieve the final condition **C3** (3.2). Nevertheless, Cauchy is confirmed to be the best static influence function when compared to other M-Estimators.

7.2.2 Optimal Splitting Strategy

As mentioned in Section 4.2.1, splitting of the original feature based graph into local maps is very important to secure efficiency and accuracy of the SMJ algorithm. Our current strategy is very naive, since it only considers the length of the trajectory and divides the graph accordingly. In the future, we wish to find better splitting strategies that may be

coupled with the reliability of local map convergence and/or the connectivity of a local map graph.

7.2.3 Finding the Optimal Subset of Key Poses

In PGR, our main idea to gain efficiency is to only connect relative poses that are within the *key poses* subset. These key poses are being selected based on their current location along the pose trajectory defined by the variable k. A better way would be to select the optimum subset of key poses base on metric accuracy or uncertainty.

7.2.4 Issues in RGB-D SLAM

Our current implementation of RGB-D SLAM is nowhere near perfect and the Front-End is susceptible to many vulnerabilities.

First, the randomness of RANSAC does not give us a deterministic solution every time the Front-End is run, which means our SLAM solution is also not repeatable. There are many variants of RANSAC in literature which we would like to further investigate (e.g., LMeds [96], MLESAC [97]), all with differing properties to improve on the standard approach.

As mentioned in Section 6.6.1, even with the best data association technique we cannot guarantee that no outliers will exist in the data. Since there is no direct way to obtain odometry in RGB-D SLAM, the outliers become difficult to remove with our suggested ideas. In the future, we wish to look into robust Back-Ends that can handle problems where there is no odometry. As far as we know, this idea has yet to be investigated.

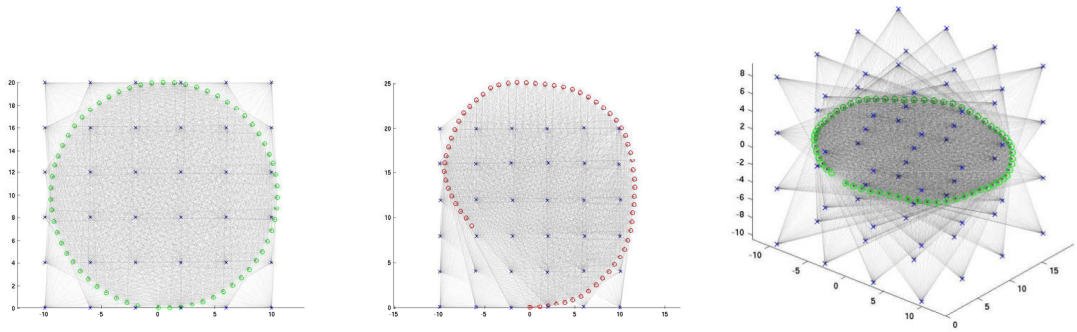
In the second algorithm (Robust RGB-D), the monocular SLAM part is still susceptible to degeneracy, ambiguity and depth uncertainty. When coupled with a poor quality RGB camera, our algorithm still has many failure cases to overcome. In the future we would like to look more into combining a high definition camera with the Depth sensor and parameterizing the bundle adjustment problem better through inverse depth [98] or parallax angle [94].

Appendix

A Simulated Datasets

In simulation, we have the liberty of using perfect Gaussian models, data association and controlled static environments. Here is a list of trajectories we have used in this thesis.

CIRCLE (2D,3D)

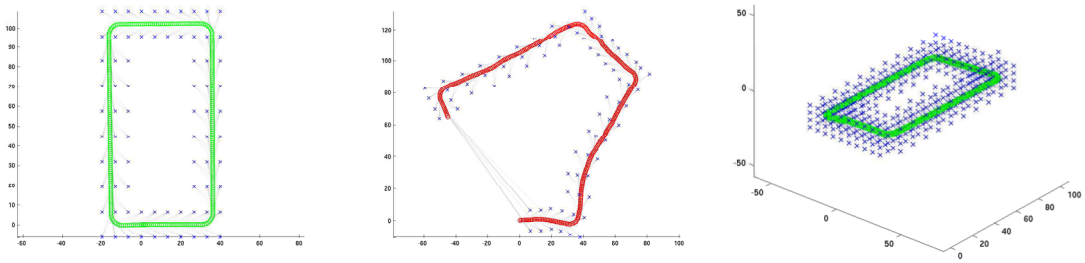


(a) 2D Ground truth (b) 2D Odometry (c) 3D Ground truth

Figure 1: Circle trajectory

This is a circular trajectory where the sensor range is large enough to observe every feature from every pose. Therefore, an edge will also exist connecting every feature to every pose. In SLAM graphs, we can consider this dataset the most optimal in terms of connectivity. For our 3D simulation the trajectories are kept the same except oscillation is applied on the pitch angle.

LOOP (2D,3D)

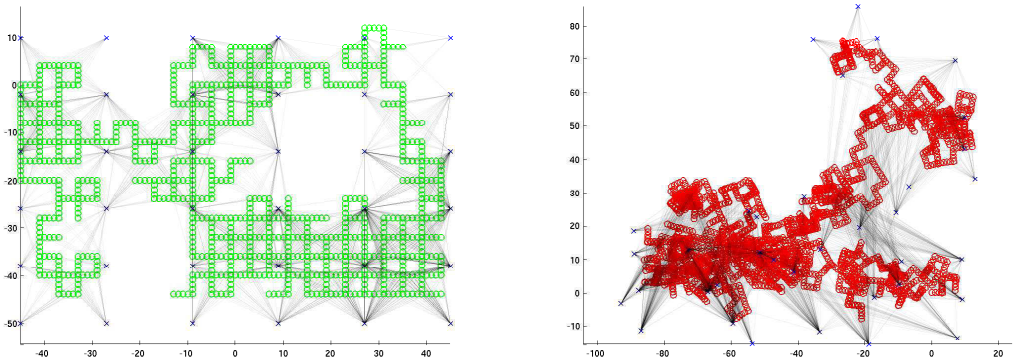


(a) 2D Ground truth (b) 2D Odometry (c) 3D Ground Truth

Figure 2: Loop trajectory

Using a standard sensor model for range and bearing, the Loop trajectory has a large but single loop, observing several features along the way. The objective is to keep overall node degree and feature observations low.

MANHATTAN FEATURES (2D)

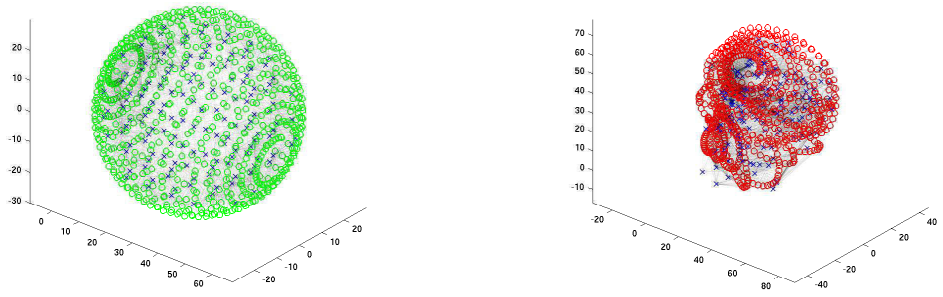


(a) Ground truth (b) Odometry

Figure 3: Manhattan features trajectory

This is a simulation of Manhattan [33] where features are observed instead of relative pose constraints. This trajectory is very complex in nature with many loops. The observations can be sparse or dense at different sections depending on the path taken.

SPHERE FEATURES (3D)



(a) `Ground truth` (b) `Odometry`

Figure 4: Sphere features trajectory

Sphere is a common trajectory used to test 3D SLAM algorithms for their reliability towards complex rotations. In this instance 3D features are observed throughout the trajectory using a typical 3D sensor model.

B Reliable Optimization

The following summarizes how to use IRLS as a bootstrapping algorithm

Algorithm 1: IRLS+GN

Input: $(X^{(0)}, Z, \Omega)$

Output: Gauss-Newton results $(X^{\star})$

$k \leftarrow 1$

INITIAL INFLUENCE

$\alpha \leftarrow 2$ `// Geman-McClure`

while $\alpha > 1$ **do**

  foreach *Edge* i, j **do**

    Compute $r_{ij}^{(k-1)}$ using $X_{(k-1)}^{\star}$ `// calculate residual`

    $w_{ij}^{(k)} = w(r_{ij}^{(k-1)}, \alpha)$ `// calculate M-Estimator weight`

    $\Omega_{ij}^{w} = w_{ij}^{(k)} \Omega_{ij}$

  end

  $X_{(k)}^{\star} = \texttt{GN}(\text{Equ } (3.1), X_{(k-1)}^{\star}, Z, \Omega^{w})$[1]

  $\alpha = \alpha - 0.5$

  $k \leftarrow k + 1$

end

ITERATIVE RE-WEIGHTED LEAST SQUARES

$\alpha = 1$ `// cauchy`

while *Not Converged* (Equ (3.8) $< \epsilon$) **do**

  foreach *Edge* i, j **do**

    Compute $r_{ij}^{(k-1)}$ using $X_{(k-1)}^{\star}$ `// calculate residual`

    $w_{ij}^{(k)} = w(r_{ij}^{(k-1)}, \alpha)$ `// calculate M-Estimator weight`

    $\Omega_{ij}^{w} = w_{ij}^{(k)} \Omega_{ij}$

  end

  $X_{(k)}^{\star} = \texttt{GN}(\text{Equ } (3.1), X_{(k-1)}^{\star}, Z, \Omega^{w})$

  $k \leftarrow k + 1$

end

BOOTSTRAPPING

$X^{\star} = \texttt{GN}(\text{Equ } (2.29), X_{(k)}^{\star}, Z, \Omega)$

[1] $\texttt{GN}(\cdot, \cdot)$ refers to the Gauss-Newton function: the first argument refers to the function itself, the other three respectively are the state vectors, measurement vectors and information matrix.

C Sparse Map Joining

C.1 Batch Optimization (BO)

Algorithm 2: Batch Joining

Input: Solved Local Maps: (Z^L, Ω^L, n)

Output: Solved Global Map: (X^G, Λ^G)

$X^G \leftarrow Z_1^L$

$i \leftarrow 2$

while $i < n+1$ **do**

$X^G \leftarrow$ `Concatenate`(X^G, Z_i^L)`// initialization`

$i \leftarrow i+1$

end

$Z^G = \left[Z_1^L, Z_2^L \cdots Z_n^L\right]^T$

$\Omega^G = \texttt{diag}(\Omega_1^L, \Omega_2^L \cdots \Omega_n^L)$

$(X^G, \Lambda^G) = \texttt{GN}(\text{Equ } (2.27)\ , X^G, Z^G, \Omega^G)$

return X^G

C.2 Sequential Optimization (SO)

Algorithm 3: Sequential Joining

Input: Solved Local Maps: (Z^L, Ω^L, n)

Output: Solved Global Map: (X^G, Λ^G)

$X^G, Z^G \leftarrow Z_1^L$

$\Omega^G \leftarrow \Omega_1^L$

$i \leftarrow 2$

while $i < n + 1$ **do**

$X^G = \texttt{Concatenate}(X^G, Z_i^L)\texttt{// initialization}$

$Z^G = \begin{bmatrix} Z^G, Z_i^L \end{bmatrix}^T$

$\Omega^G = \texttt{diag}(\Omega^G, \Omega_i^L)$

$(X^{*G}, \Lambda^G) = \texttt{GN}(\text{Equ } (2.27)\ , X^G, Z^G, \Omega^G)$

$X^G \leftarrow X^{*G}$

$Z^G \leftarrow X^{*G}$

$\Omega^G \leftarrow \Lambda^G$

$i \leftarrow i + 1$

end

return X^G

C.3 Divide & Conquer Optimization (DCO)

Algorithm 4: Divide and Conquer Joining

Input: Solved Local Maps: (Z^L, Ω^L)

Output: Solved Global Map: (X^G, Λ^G)

$k \leftarrow 1$ `// tree depth`

$(Z^L)^k \leftarrow Z^L$

while $dim((Z^L)^k) > 1$ **do**

  if $dim((Z^L)^k) = Odd$ **then**

    $Z^L_{\texttt{buff}} = (Z^L_{\texttt{end}})^k$ `// create buffer map`

    $\Omega^L_{\texttt{buff}} = (\Omega^L_{\texttt{end}})^k$

  end

  $i \leftarrow 2$

  $j \leftarrow 1$

  while $i < dim((Z^L)^k)$ **do**

    $(X^L_j)^{k+1} \leftarrow$ `Concatenate`$((X^L_{i-1})^k, (X^L_i)^k)$ `// initialization`

    $(Z^L_j)^{k+1} \leftarrow \left[(Z^L_{i-1})^k, (Z^L_i)^k\right]^T$

    $(\Omega^L_j)^{k+1} =$ `diag`$((\Lambda^L_{i-1})^k, (\Lambda^L_i)^k)$

    $((X^{*L}_j)^{k+1}, (\Lambda^L_j)^{k+1}) =$ `GN`(Equ (2.27) $, (X^L_j)^{k+1}, (Z^L_j)^{k+1}, (\Omega^L_j)^{k+1})$

    $(Z^L_j)^{k+1} \leftarrow (X^{*L}_j)^{k+1}$

    $(\Omega^L_j)^{k+1} \leftarrow (\Lambda^L_j)^{k+1}$

    $i \leftarrow i + 2$

    $j \leftarrow j + 1$

  end

  $(Z^L)^{k+1} \leftarrow \left[(Z^L)^{k+1}, Z^L_{\texttt{buff}}\right]$ `// add buffer map`

  $(\Omega^L)^{k+1} \leftarrow \left[(\Omega^L)^{k+1}, \Omega^L_{\texttt{buff}}\right]$

  $k \leftarrow k + 1$

end

return $X^G = (X^{*L})^{k+1}$

C.4 Sparse Map Joining Algorithm

Algorithm 5: Sparse Map Joining (SMJ) Algorithm

Input: Measurement, information and local maps size $(\hat{Z}^O, \hat{Z}^F, \Omega^O, \Omega^F, n)$

Output: SMJ state vector and information: (X^G, Λ^G)

OPTIMIZE AND MARGINALIZE LOCAL MAPS

$(X_1^L \cdots X_n^L, Z_1^L \cdots Z_n^L) \leftarrow$ `SplitMaps`$(\hat{Z}^O, \hat{Z}^F, \Omega^O, \Omega^F, n)$

$i \leftarrow 1$

while $i < n + 1$ **do**

$(X_i^L, \Lambda_i^L) \leftarrow$`IRLS+GN`$(X_i^L, Z_i^L, \Omega_i^L)$[2]

$\Omega_i^L \leftarrow \Lambda_i^{L(\mathcal{M})} \leftarrow \Lambda_i^L$

$Z_i^L \leftarrow X_i^{L(\mathcal{M})} \leftarrow X_i^L$ `// marginalization`

$i \leftarrow i + 1$

end

JOIN LOCAL MAPS

switch *Joining Algorithm* **do**

case *BO*

Use Algorithm 2

case *SO*

Use Algorithm 3

case *DCO*

Use Algorithm 4

end

return $(X^G)^\star$

[2] `IRLS+GN(`$\cdot, \cdot, \cdot$`)` refers to Iterative Re-weighted Least Squares bootstrapping in Algorithm 1.

D Pose Graph Representation

Algorithm 6: Relative Pose Searching

Input: Observations: (Z^F, Ω^F)

Output: Found Relative Poses: $(Count, RelativePoseList)$

$X^K \leftarrow$ `SelectSubset`(X^P, k) `// select pose subset`

$Count_{(\hat{Z}^F)} \leftarrow 0$ `// stores the frequency a observation is used`

foreach *pose* X_i^K **do**
  $n \leftarrow 1$
  foreach *pose* X_j^K, $j = i + n$ **do**
    $m \in$ all common features
    if $dim(m) > Obs_{min}$ **then**
      $RelativePoseList_{Z_{ij}^P} \leftarrow (\hat{Z}_{im}^F, \hat{Z}_{jm}^F, \Omega_{im}^F, \Omega_{jm}^F)$`// create link`
      $Count_{\hat{Z}_{im}^F} = Count_{\hat{Z}_{im}^F} + 1$`// increment frequency`
      $Count_{\hat{Z}_{jm}^F} = Count_{\hat{Z}_{jm}^F} + 1$
      $n \leftarrow n + 1$
    end
  end
end

return $Count_{\hat{Z}^F}, RelativePoseList_{Z_{ij}^P}$

Algorithm 7: Calculating Relative Pose

Input: Found Relative Poses: $(Count, RelativePoseList)$

Output: Pose Graph Edges and Information: Z^P, Ω^P

foreach *RelativePoseList* **do**

$\hat{Z}^F_{im}, \hat{Z}^F_{jm}, \Omega^F_{im}, \Omega^F_{jm} \leftarrow RelativePoseList_{Z^P_{ij}}$ `// extract observations`

$X^P_{ij} \leftarrow$ `Horn`$(\hat{Z}^F_{im}, \hat{Z}^F_{jm})$ `// calculate initial`

$\Omega^F{}_{im} \leftarrow \Omega^F_{im}/Count_{\hat{Z}^F_{im}}$ `// divide information, MO algorithm`

$\Omega^F{}_{jm} \leftarrow \Omega^F_{jm}/Count_{\hat{Z}^F_{jm}}$

TWO POSE GAUSS-NEWTON

if $j = i + 1$ **then**

$(X^\star, \Lambda) \leftarrow$ `GN`(Equ (2.27),$X^P_{ij}, (\hat{Z}^F_{im}, \hat{Z}^F_{jm}, \hat{Z}^O_{ij}), (\Omega^F_{im}, \Omega^F_{jm}, \Omega^O_{ij}))$ `// with odometry`

else

$(X^\star, \Lambda) \leftarrow$ `GN`(Equ (2.27),$X^P_{ij}, (\hat{Z}^F_{im}, \hat{Z}^F_{jm}), (\Omega^F_{im}, \Omega^F_{jm}))$ `// without odometry`

end

$Z^P_{ij} \leftarrow X^{P(\mathcal{M})} \leftarrow (X)^\star$`// Marginalize features`

$\Omega^P_{ij} \leftarrow \Lambda^{P(\mathcal{M})} \leftarrow \Lambda$

end

return Z^P, Ω^P

Algorithm 8: Pose Graph Representation (PGR) Algorithm

Input: Measurement and Information $(\hat{Z}^O, \hat{Z}^F, \Omega^O, \Omega^F)$

Output: Pose Graph state vector and information: (X, Λ)

RELATIVE POSE SEARCH

Algorithm 6

RELATIVE POSE CALCULATION

Algorithm 7

OPTIMIZATION

$X^P \leftarrow$ `Concatenate`$(\hat{Z}^O)$`// initialization`

$(X, \Lambda) \leftarrow$`IRLS+GN`(X^P, Z^P, Ω^P)

return (X, Λ)

E Schur Complement

Suppose

$$S = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \tag{1}$$

Given that C is a positive definite matrix and S is a positive semi-definite matrix then the complement of C is Ω is

$$A - BC^{-1}B^T \tag{2}$$

An example of feature marginalization on the information matrix is given below

$$\Omega = \begin{bmatrix} \Omega^{PP} & \Omega^{PF} \\ (\Omega^{PF})^T & \Omega^{FF} \end{bmatrix} \tag{3}$$

$$\Omega^{P(\mathcal{M})} = \Omega^{PP} - \Omega^{PF}(\Omega^{FF})^{-1}(\Omega^{PF})^T \tag{4}$$

F Transforming between Rotation Matrix and Euler Angles

The standard sequence for rotations in this thesis is, Yaw, Pitch, Roll or $R_z(\phi,)R_y(\theta), R_x(\psi)$.

Euler Angles to Rotation matrix

$$\begin{aligned} R_x(\psi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix}, \; R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \\ R_z(\phi) &= \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{5}$$

$$R = R_z R_y R_x \tag{6}$$

Rotation matrix to Euler Angles

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{7}$$

$$\theta = \begin{cases} R_{31} \neq \pm 1 & -\arcsin(R_{31})) \\ R_{31} = -1 & \pi/2 \\ R_{31} = 1 & -\pi/2 \end{cases}, \; \psi = \begin{cases} R_{31} \neq \pm 1 & \arctan(\frac{R_{21}}{cos(\theta)}, \frac{R_{11}}{cos(\theta)}) \\ R_{31} = \pm 1 & 0 \end{cases}$$

$$\phi = \begin{cases} R_{31} \neq \pm 1 & \arctan(\frac{R_{32}}{cos(\theta)}, \frac{R_{33}}{cos(\theta)}) \\ R_{31} = -1 & \arctan(\frac{R_{12}}{cos(\theta)}, \frac{R_{13}}{cos(\theta)}) \\ R_{31} = 1 & \arctan(\frac{-R_{12}}{cos(\theta)}, \frac{-R_{13}}{cos(\theta)}) \end{cases} \tag{8}$$

Bibliography

[1] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 376–382, 1991.

[2] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," in *Intelligent Robots and Systems (IROS), Proceedings of IEEE/RSJ International Conference on*. IEEE, 2006, pp. 3562–3568.

[3] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem." AAAI Press, 2002, pp. 593–598.

[4] T. Bailey, J. Nieto, and E. Nebot, "Consistency of the fastslam algorithm," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*. IEEE, 2006, pp. 424–429.

[5] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," in *Computer Vision and Pattern Recognition (CVPR), Proceedings of IEEE Computer Society Conference on*, 1994, pp. 935–938.

[6] S. Huang, Z. Wang, G. Dissanayake, and U. Frese, "Iterated d-slam map joining: evaluating its performance in terms of consistency, accuracy and efficiency," *Autonomous Robots*, vol. 27, no. 4, pp. 409–429, 2009.

[7] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research (IJRR)*, vol. 25, no. 12, pp. 1181–1204, 2006.

[8] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2d mapping," in *Intelligent Robots and Systems (IROS), Proceedings of IEEE/RSJ International Conference on.* IEEE, 2010, pp. 22–29.

[9] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research (IJRR)*, vol. 25, no. 5-6, pp. 403–429, 2006.

[10] T. Duckett, S. Marsland, and J. Shapiro, "Learning globally consistent maps by relaxation," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*, vol. 4. IEEE, 2000, pp. 3841–3846.

[11] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping," *Robotics, IEEE Transactions on*, vol. 21, no. 2, pp. 196–207, 2005.

[12] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research (IJRR)*, vol. 25, no. 12, pp. 1181–1203, 2006.

[13] S. Huang, H. Wang, U. Frese, and G. Dissanayake, "On the number of local minima to the point feature based slam problem," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on.* IEEE, 2012, pp. 2074–2079.

[14] L. Carlone, "Convergence analysis of pose graph optimization via gauss-newton methods," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on.* IEEE, 2013.

[15] 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*, vol. 17, no. 3, pp. 229–241, 2001.

[16] N. Sunderhauf and P. Protzel, "Towards a robust back-end for pose graph slam," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*. IEEE, 2012, pp. 1254–1261.

[17] F. Dellaert, A. Kipp, and P. Krauthausen, "A multifrontal qr factorization approach to distributed inference applied to multirobot localization and mapping," in *the 20th national conference on Artificial intelligence, Proceedings of*. AAAI Press, 2005, pp. 1261–1266.

[18] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press Cambridge, 2005, vol. 1.

[19] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," *Advances in neural information processing systems*, pp. 689–695, 2001.

[20] T. Dean and K. Kanazawa, "Probabilistic temporal reasoning." AAAI Press, 1988.

[21] K. P. Murphy, "Dynamic bayesian networks: representation, inference and learning," Ph.D. dissertation, University of California, 2002.

[22] 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*, vol. 2. IEEE, 1998, pp. 905–911.

[23] M. Avriel, *Nonlinear programming: analysis and methods*. Courier Dover Publications, 2003.

[24] J. E. Dennis and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*. Society for Industrial and Applied Mathematics, 1987,

vol. 16.

[25] M. J. Powell, *A new algorithm for unconstrained optimization.* UKAEA, 1970.

[26] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* Cambridge Univ Press, 2000, vol. 2.

[27] D. Rosen, M. Kaess, and J. Leonard, "An incremental trust-region method for robust online sparse least-squares estimation," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on.* IEEE, 2012, pp. 1262–1269.

[28] G. Tipaldi, G. Grisetti, and W. Burgard, "Approximate covariance estimation in graphical approaches to slam," in *Intelligent Robots and Systems (IROS), Proceedings of IEEE/RSJ International Conference on.* IEEE, 2007, pp. 3460–3465.

[29] A. Ranganathan, M. Kaess, and F. Dellaert, "Loopy sam," in *The 20th International Joint Conference on Artificial Intelligence (IJCAI), Proceedings of*, 2007, pp. 6–12.

[30] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *Robotics, IEEE Transactions on*, vol. 24, no. 6, pp. 1365–1378, 2008.

[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge university press, 2007.

[32] E. Olson and M. Kaess, "Evaluating the performance of map optimization algorithms," in *RSS Workshop on Good Experimental Methodology in Robotics*, 2009, p. 40.

[33] E. B. Olson, S. Teller, and J. Leonard, "Robust and efficient robotic mapping," Ph.D. dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2008.

[34] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking*

and navigation: theory algorithms and software. John Wiley & Sons, 2004.

[35] U. Frese, "A discussion of simultaneous localization and mapping," *Autonomous Robots*, vol. 20, no. 1, pp. 25–42, 2006.

[36] K. Madsen, H. B. Nielsen, and O. Tingleff, *Methods for non-linear least squares problems.* Informatics and Mathematical Modelling, Technical University of Denmark, 2004.

[37] T. Duckett, S. Marsland, and J. Shapiro, "Fast, on-line learning of globally consistent maps," *Autonomous Robots*, vol. 12, no. 3, pp. 287–300, 2002.

[38] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on.* IEEE, 2006, pp. 2262–2269.

[39] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[40] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Robotics: Science and Systems (RSS), Proceedings of*, 2007.

[41] L. Carlone, R. Aragues, J. Castellanos, and B. Bona, "A linear approximation for graph-based simultaneous localization and mapping," in *Robotics: Science and Systems (RSS), Proceedings of*, 2011.

[42] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[43] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, "View-based maps," *The International Journal of Robotics Research (IJRR)*, vol. 29, no. 8, pp. 941–957, 2010.

[44] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," in *Robotics: Science and Systems (RSS), Proceedings of*, 2007.

[45] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroeter, and L. Murphy, "Navigating, recognizing and describing urban spaces with vision and lasers," *The International Journal of Robotics Research (IJRR)*, vol. 28, no. 11-12, pp. 1406–1433, 2009.

[46] J. Folkesson and H. Christensen, "Graphical slam-a self-correcting map," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*, vol. 1. IEEE, 2004, pp. 383–390.

[47] K. Ni, D. Steedly, and F. Dellaert, "Tectonic sam: Exact, out-of-core, submap-based slam," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*. IEEE, 2007, pp. 1678–1685.

[48] M. A. Paskin, "Thin junction tree filters for simultaneous localization and mapping," in *International Joint Conference on Artificial Intelligence (IJCAI), Proceedings of*, G. Gottlob and T. Walsh, Eds. Morgan Kaufmann Publishers, 2003, pp. 1157–1164.

[49] U. Frese, "Treemap: An o (log n) algorithm for indoor simultaneous localization and mapping," *Autonomous Robots*, vol. 21, no. 2, pp. 103–122, 2006.

[50] A. Blake and A. Zisserman, *Visual reconstruction*. MIT press Cambridge, 1987, vol. 2.

[51] C. L. Zitnick, "Seeing through the blur," in *Computer Vision and Pattern Recognition (CVPR), Proceedings of the IEEE Conference on*. IEEE Computer Society, 2012, pp. 1736–1743.

[52] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[53] Z. Zhang, "Parameter estimation techniques: A tutorial with application to conic fitting," *Image and Vision Computing*, vol. 15, no. 1, pp. 59–76, 1997.

[54] S. Huang, Z. Wang, and G. Dissanayake, "Sparse local submap joining filter for building large-scale maps," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1121–1130, 2008.

[55] G. Grisetti, R. Kummerle, and K. Ni, "Robust optimization of factor graphs by using condensed measurements," in *Intelligent Robots and Systems (IROS), Proceedings of IEEE/RSJ International Conference on*. IEEE, 2012, pp. 581–588.

[56] L. M. Paz, P. Jensfelt, J. D. Tardos, and J. Neira, "Ekf slam updates in o (n) with divide and conquer slam," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1657–1663.

[57] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2d and 3d mapping," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*. IEEE, 2010, pp. 273–278.

[58] C. Hertzberg, "A framework for sparse, non-linear least squares problems on manifolds," Ph.D. dissertation, University of Bremen, 2008.

[59] L. Zhao, S. Huang, L. Yan, J. Wang, G. Hu, and G. Dissanayake, "Large-scale monocular slam by local bundle adjustment and map joining," in *Control Automation Robotics Vision (ICARCV), Proceedings of*, 2010, pp. 431–436.

[60] M. Lourakis and A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm," *Institute of Computer Science-FORTH, Heraklion, Crete, Greece, Tech. Rep*, vol. 340, 2004.

[61] J. Kurlbaum and U. Frese, "A benchmark data set for data association," *Univ. Bremen, Bremen, Germany, SFB/TR*, vol. 8, pp. 017–02, 2009.

[62] J. Guivant and E. Nebot, "Simultaneous localization and map building: Test case for outdoor applications," *Australian Centre for Field Robotics*, 2002.

[63] G. Hu, S. Huang, and G. Dissanayake, "3d i-slsjf: A consistent sparse local submap joining algorithm for building large-scale 3d map," in *Conference on Decision and Control held jointly with 28th Chinese Control Conference, Proceedings of*, 2009, pp. 6040–6045.

[64] A. Nüchter, *3D robotic mapping: the simultaneous localization and mapping problem with six degrees of freedom.* Springer, 2009, vol. 52.

[65] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping." in *Robotics: Science and Systems (RSS), Proceedings of*, 2012.

[66] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *Intelligent Transportation Systems Magazine, IEEE*, vol. 2, no. 4, pp. 31–43, 2010.

[67] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[68] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact pose slam," *Robotics, IEEE Transactions on*, vol. 26, no. 1, pp. 78–93, 2010.

[69] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on.* IEEE, 2013.

[70] D. Nistér, 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*, vol. 1. IEEE, 2004, pp. I–652.

[71] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton,

D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), Proceedings of 10th IEEE international symposium on*. IEEE, 2011, pp. 127–136.

[72] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgbd mapping: Using depth cameras for dense 3d modeling of indoor environments," in *Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*, 2010.

[73] H. Strasdat, A. J. Davison, J. Montiel, and K. Konolige, "Double window optimisation for constant time visual slam," in *Computer Vision (ICCV), Proceedings of 2011 IEEE International Conference on*. IEEE, 2011, pp. 2352–2359.

[74] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3d visual slam with a hand-held rgb-d camera," in *RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Proceedings of*, 2011.

[75] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns," 2008, united states of ameriaca patent app. 12/522,171.

[76] C. Audras, A. Comport, M. Meilland, and P. Rives, "Real-time dense RGB-D localisation and mapping," in *Australian Conference on Robotics and Automation (ACRA), Proceedings of*, 2011.

[77] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference, Proceedings of*, vol. 15. Manchester, UK, 1988, p. 50.

[78] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[79] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[80] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best

matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.

[81] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[82] C. Mei, G. Sibley, M. Cummins, P. M. Newman, and I. D. Reid, "A constant-time efficient stereo slam system." in *BMVC*, 2009, pp. 1–11.

[83] D. Nistér, "An efficient solution to the five-point relative pose problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 6, pp. 756–770, 2004.

[84] P. H. Torr, A. W. Fitzgibbon, and A. Zisserman, "The problem of degeneracy in structure and motion recovery from uncalibrated image sequences," *International Journal of Computer Vision*, vol. 32, no. 1, pp. 27–44, 1999.

[85] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Robotics: Science and Systems (RSS), Proceedings of*, 2009.

[86] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments," in *Experimental Robotics*. Springer, 2014, pp. 477–491.

[87] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically stable sampling for the icp algorithm," in *3-D Digital Imaging and Modeling. Proceedings of*. IEEE, 2003, pp. 260–267.

[88] C. Liu, "Beyond pixels: exploring new representations and applications for motion analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[89] W. Zhou, J. Miro, and G. Dissanayake, "Information-driven 6d slam based on ranging

vision," in *Intelligent Robots and Systems (IROS), Proceedings of IEEE/RSJ International Conference on.* IEEE, 2008, pp. 2072–2077.

[90] M. Cummins and P. Newman, "Accelerated appearance-only slam," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on.* IEEE, 2008, pp. 1828–1833.

[91] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Intelligent Robots and Systems (IROS), Proceedings of IEEE/RSJ International Conference on.* IEEE, 2012, pp. 573–580.

[92] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[93] J. Civera, A. J. Davison, and J. Montiel, "Inverse depth parametrization for monocular slam," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 932–945, 2008.

[94] L. Zhao, S. Huang, L. Yan, and G. Dissanayake, "Parallax angle parametrization for monocular slam," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on.* IEEE, 2011, pp. 3117–3124.

[95] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Mixed and Augmented Reality (ISMAR), Proceedings of 8th IEEE International Symposium on.* IEEE, 2009, pp. 83–86.

[96] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American statistical association*, vol. 79, no. 388, pp. 871–880, 1984.

[97] B. J. Tordoff and D. W. Murray, "Guided-mlesac: Faster image transform estimation by using matching priors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 10, pp. 1523–1535, 2005.

[98] J. Civera, A. J. Davison, and J. Montiel, "Inverse depth parametrization for monocular slam," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 932–945, 2008.