TECHNISCHE UNIVERSITÄT CHEMNITZ

Robust Optimization for Simultaneous Localization and Mapping

von der Fakultät für Elektrotechnik und Informationstechnik
der Technischen Universität Chemnitz

genehmigte

Dissertation

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von

Dipl.-Inf. Niko Sünderhauf

geboren am 10. April 1981 in Zwickau

eingereicht am 14. Februar 2012

Gutachter: Prof. Dr.-Ing. Peter Protzel
Prof. Dr.-Ing. Jozef Suchý

Tag der Verleihung: 19. April 2012

Sünderhauf, Niko
Robust Optimization for Simultaneous Localization and Mapping
Dissertation, Fakultät für Elektrotechnik und Informationstechnik
Technische Universität Chemnitz, April 2012

Keywords:
Simultaneous Localization and Mapping, Pose Graph SLAM, Appearance-Based Place Recognition, Nonlinear Least Squares Optimization, Factor Graph, Robust Optimization, Outlier Rejection, GNSS-based Localization, Multipath Mitigation

This document is available online at:
`http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-86443`

Abstract

SLAM (Simultaneous Localization And Mapping) has been a very active and almost ubiquitous problem in the field of mobile and autonomous robotics for over two decades. For many years, filter-based methods have dominated the SLAM literature, but a change of paradigms could be observed recently. Current state of the art solutions of the SLAM problem are based on efficient sparse least squares optimization techniques. However, it is commonly known that least squares methods are by default not robust against *outliers*. In SLAM, such outliers arise mostly from data association errors like false positive loop closures. Since the optimizers in current SLAM systems are not robust against outliers, they have to rely heavily on certain preprocessing steps to prevent or reject all data association errors. Especially false positive loop closures will lead to catastrophically wrong solutions with current solvers. The problem is commonly accepted in the literature, but no concise solution has been proposed so far.

The main focus of this work is to develop a novel formulation of the optimization-based SLAM problem that is robust against such outliers. The developed approach allows the back-end part of the SLAM system to change parts of the topological structure of the problem's factor graph representation *during* the optimization process. The back-end can thereby discard individual constraints and converge towards correct solutions even in the presence of many false positive loop closures. This largely increases the overall robustness of the SLAM system and closes a gap between the sensor-driven front-end and the back-end optimizers. The approach is evaluated on both large scale synthetic and real-world datasets.

This work furthermore shows that the developed approach is versatile and can be applied beyond SLAM, in other domains where least squares optimization problems are solved and outliers have to be expected. This is successfully demonstrated in the domain of GPS-based vehicle localization in urban areas where multipath satellite observations often impede high-precision position estimates.

Contents

List of Symbols and Notation **9**

List of Acronyms **11**

1 Introduction **13**
Objectives and Contributions 18
Thesis Outline 18

2 Simultaneous Localization And Mapping **21**
2.1 S, L, and M – The Parts of SLAM 21
2.1.1 M for Mapping 23
2.1.2 L for Localization 32
2.1.3 S for Simultaneous 35
2.2 Graph Representations for SLAM 41
2.2.1 Dynamic Bayesian Networks 41
2.2.2 Factor Graphs 42
2.2.3 Markov Random Fields 43
2.3 SLAM as a Nonlinear Least Squares Optimization Problem 44
2.3.1 The Pose Graph SLAM Problem 44
2.3.2 Deriving a Nonlinear Least Squares Formulation 45
2.3.3 An Intuitive Analogy for the Least Squares Optimization 48
2.3.4 Optimization-based SLAM – A Literature Review 49
2.4 Summary 51

3 Least Squares Optimization **53**
3.1 Introduction 53
3.1.1 A Taxonomy of Optimization Problems 54
3.1.2 Least Squares Optimization Problems 55
3.2 Linear Least Squares Problems 55
3.2.1 Solving Linear Least Squares Problems 56
3.2.2 Examples for Linear Least Squares Problems 57
3.3 Nonlinear Least Squares Problems 59
3.3.1 Gradient Descent 60
3.3.2 Newton's Method 61

3.3.3 Gauss-Newton . . . 63
3.3.4 Levenberg-Marquardt . . . 64
3.3.5 Summary . . . 65
3.4 Weighted Nonlinear Least Squares Problems . . . 66
3.5 Least Squares Optimization for SLAM . . . 68
3.5.1 A Sandbox Example . . . 69
3.5.2 Why Optimization-based SLAM is Efficient . . . 72
3.6 Least Squares Optimization in the Presence of Outliers . . . 74
3.6.1 Sample Consensus Methods for Outlier Rejection . . . 75
3.6.2 Robust Cost Functions . . . 77
3.7 Summary . . . 79

4 Motivation – When Optimization Fails 81
4.1 Data Association Errors and their Effects on SLAM . . . 82
4.2 Current Approaches for Outlier Mitigation and Avoidance . . . 83
4.2.1 Outlier Avoidance on the Front-End Side . . . 84
4.2.2 Outlier Mitigation on the Back-End Side . . . 86
4.3 Summary . . . 87

5 A Robust Back-End for SLAM 89
5.1 The Robustified Formulation for Pose Graph SLAM . . . 90
5.1.1 First Steps Towards a Mathematical Formulation . . . 90
5.1.2 Introducing the Switch Variables and Finding a Suitable Switch Function . . . 92
5.1.3 Introducing the Switch Priors . . . 95
5.1.4 Putting it all Together . . . 96
5.2 Discussion . . . 97
5.2.1 The Influence of s_{ij} on the Information Matrix Λ_{ij} . . . 98
5.2.2 Establishing the Connection to the Maximum a Posteriori Solution 100
5.2.3 The Influence of the Additional Variables and Constraints on the Problem Size . . . 102
5.2.4 The Influence of the Additional Variables and Constraints on the Sparse Structure of the Problem . . . 102
5.2.5 The Influence of the Additional Variables and Constraints on the Problem's Convergence Properties . . . 105
5.3 Summary and a First Working Example . . . 105

6 Evaluation 109
6.1 Error Metrics for SLAM . . . 110
6.1.1 The Root-Mean-Square Error (RMSE) . . . 110
6.1.2 The Relative Pose Error Metric . . . 111
6.1.3 Precision-Recall Statistics . . . 112
6.2 Datasets for the Evaluation . . . 113
6.3 General Methodology . . . 114

6.3.1 Policies for Adding Outlier Loop Closure Constraints 115
6.3.2 Loop Closure Displacement 117
6.3.3 The Switch Function 119
6.3.4 Used Framework and Implementation 119
6.4 The Influence of Ξ_{ij} on the Estimation Results 119
6.4.1 Methodology 120
6.4.2 Results and Interpretation 120
6.5 The Robustness in the Presence of Outliers 121
6.5.1 Methodology 121
6.5.2 Results and Interpretation 121
6.5.3 Discussion of the Failure Cases 125
6.6 Runtime and Convergence Behaviour 129
6.6.1 Methodology 129
6.6.2 Results and Interpretation 129
6.7 Performance in the Outlier-Free Case 130
6.7.1 Methodology 130
6.7.2 Results and Interpretation 132
6.8 The Influence of the Switch Function Ψ 133
6.9 Summary of the Evaluation and First Conclusions 134

7 Applying the Robust Back-End in a Complete SLAM System on a Real-World Dataset 137
7.1 The Front-End 138
7.1.1 Visual Odometry by Image Profile Matching 138
7.1.2 Place Recognition Using BRIEF-Gist 139
7.2 Results of the Complete SLAM System on the St. Lucia Dataset 142
7.3 Summary 145

8 Applications Beyond SLAM – Multipath Mitigation in GNSS-based Localization Problems using the Robust Back-End 147
8.1 GNSS-based Localization – A Gentle Introduction 148
8.1.1 Systematic Errors 149
8.1.2 Multipath Errors 150
8.2 Multipath Identification and Mitigation – Related Work 151
8.3 Modelling the GNSS-based Localization Problem as a Factor Graph . . . 152
8.3.1 The Vehicle State Vertices 152
8.3.2 The Pseudorange Factor 153
8.3.3 Additional Factors 154
8.3.4 Solving for the Maximum a Posteriori Solution 155
8.4 Towards a Problem Formulation Robust to Multipath Errors 156
8.4.1 The Switched Pseudorange Factor 156
8.4.2 The Switch Transition Factor 157
8.5 Multipath Mitigation in a Real-World Urban Scenario 158
8.5.1 The Chemnitz City Dataset 159

8.5.2 Methodology . . . 160
8.5.3 Results . . . 161
8.6 Interpretation and Summary . . . 165
8.7 Outlook . . . 167

9 An Outlook on Robust Optimization for Sensor Fusion and Calibration 169
9.1 Sensor Fusion by Robust Optimization . . . 169
9.2 Sensor Calibration by Robust Optimization . . . 171

10 Conclusions 173
10.1 What has been Achieved – Contributions of this Thesis . . . 174
10.2 Open Questions – An Outlook on Further Work . . . 175
10.2.1 Parameters of the Proposed Robust Formulation . . . 175
10.2.2 Convergence Behaviour and the Dangers of Local Minima . . . 176
10.2.3 Further Applications of the Robust Approach . . . 176

A Filter Algorithms for SLAM 179
A.1 The General Bayes Filter . . . 179
A.2 Gaussian Filters . . . 183
A.2.1 The Kalman Filter . . . 183
A.2.2 The Extended Kalman Filter . . . 186
A.2.3 The Unscented Kalman Filter . . . 189
A.3 The Particle Filter . . . 191
A.4 Summary . . . 193

Translations 195

Bibliography 205

List of Figures 221

List of Tables 225

Versicherung 227

Theses 229

Curriculum Vitae 231

List of Symbols and Notation

General Mathematical Notation

x	A scalar value.
$\mathbf{x}$	A vector.
$\mathbf{X}$	A matrix.
X	A set of individual values.
$\mathbf{x}^\mathsf{T}$	The transpose of $\mathbf{x}$.
$\mathbf{x}^{-1}$	The inverse of $\mathbf{x}$.
$\mathbf{x}^*$	The optimal / best $\mathbf{x}$.
$\mathbf{x}_{0:T}$	The set $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\}$.
X_T	The same set as above, $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\}$.
$P(\mathbf{x})$	The probability distribution over $\mathbf{x}$.
$P(\mathbf{x}\|\mathbf{z})$	The conditional probability distribution over $\mathbf{x}$, conditioned on $\mathbf{z}$.
$\|\cdot\|^2_{\mathbf{\Sigma}}$	The squared Mahalanobis distance with covariance $\mathbf{\Sigma}$.
$\propto$	The proportionality relation.
$(a, b, c)^\mathsf{T}$	A column vector of individual elements.
$\{a, b, c\}$	A set of individual elements.
$\operatorname{argmin}_x f(x)$	The minimizer of a function f.
$\operatorname{argmax}_x f(x)$	The maximizer of a function f.
$\exp(x)$	Meaning e^x with e expressing Euler's number.
$f'(x)$	The derivative of a function f.
$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$	The partial derivative of $\mathbf{f}$ with respect to $\mathbf{x}$.
$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$	$\mathbf{x}$ is normally distributed with mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$.

Roman Letters

$\mathbf{e}$	A general error term. E.g $\mathbf{e}^{\text{lc}}_{ij}$ is the error term associated with the loop closure constraint between $\mathbf{x}_i$ and $\mathbf{x}_j$.
s_{ij}	The switch variable for the loop closure constraint between $\mathbf{x}_i$ and $\mathbf{x}_j$.
S	The set of all switch variables.
$\mathbf{H}$	A Hessian matrix.
$\mathbf{J}$	A Jacobian matrix.
$\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$	A normal distribution with mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$.
$\mathcal{U}(a, b)$	A uniform distribution with lower and upper limits a and b.
$\mathbf{u}_i$	A control input of any kind, associated with $\mathbf{x}_i$.
$\mathbf{u}_{ij}$	The displacement associated with a loop closure between $\mathbf{x}_i$ and $\mathbf{x}_j$.

$\mathbf{x}_i$	A general system state vector.
$\mathbf{x}_t$	The system state at specific time t.
$\mathbf{x}^{x,y}$	The x and y components of the state vector $\mathbf{x}$.
$\mathbf{x}^{\theta}$	The orientation component θ of the state vector $\mathbf{x}$.
$\mathbf{z}$	A general measurement or observation.

Greek Letters

δ	Receiver clock error / bias terms.
$\boldsymbol{\delta}_{ij}$	The difference between two poses.
γ_{ij}	The prior value for the switch variable s_{ij}.
Γ	The set of all switch prior values.
ω_{ij}	A variable weight factor that is used inside a switched loop closure constraint.
Ψ	A switch function that maps a switch variable into a weight factor.
ρ_{ti}	The pseudorange observed to satellite i at time t.
Ξ_{ij}	The covariance matrix for the switch prior constraint of s_{ij}.
$\boldsymbol{\Sigma}, \boldsymbol{\Lambda}$	General covariance matrices.
$\boldsymbol{\Omega}$	A general information matrix.

Special Abbreviations in Mathematical Expressions

lc	loop closure
mm	motion model
odo	odometry
pr	pseudorange
slc	switched loop closure
sp	switch prior
spr	switched pseudorange
st	state transition
swt	switch transition

These abbreviations are used as upper index, e.g. $\mathbf{e}^{\text{odo}}$ to indicate the type of an error term, Jacobian etc.

List of Acronyms

BRIEF	Binary Robust Independent Elementary Features
CRF	Conditional Random Field
CTRV	Constant Turn Rate and Velocity Motion Model
DBN	Dynamic Bayesian Network
EGNOS	European Geostationary Navigation Overlay Service
EKF	Extended Kalman Filter
FAST	Features from Accelerated Segment Test
g^2o	Generalized Graph Optimization
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GTSAM	The Georgia Tech Smoothing and Mapping Library
IMU	Inertial Measurement Unit
iSAM	Incremental Smoothing And Mapping
KF	Kalman Filter
MAP	Maximum a Posteriori
ML	Maximum Likelihood
MSER	Maximally Stable Extrema Regions
RANSAC	Random Sample Consensus
RMSE	Root Mean Squared Error
RPE	Relative Pose Error
SBAS	Satellite-Based Augmentation System
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded-Up Robust Features
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
WAAS	Wide Area Augmentation System

1 Introduction

In the beginning, robots were in the factories, assembling our cars. Now they are in our homes, vacuuming our floors. There are robots for search and rescue, for demining and bomb disposal. Robots explore the forbidding craters and dunes on Mars, and the dark abysses of Earth's oceans. Robots help with surgery and lawn mowing and robotic technology helps to increase the efficiency of agriculture and forestry. Robots play soccer with each other, mimic human behaviour and biological systems, guide visitors through museums and entertain our kids and ourselves. Robots have been a fascinating constant in literature, television and movies and an enthralling and intriguing field of science.

Not all of these robots are autonomous. Sometimes autonomy is not necessary. Sometimes we do not dare to let them operate autonomously, because we do not fully trust the robustness of our algorithms and designs. *Robustness* in this sense describes the ability of a system to cope with whatever shortcomings, inadequacies and errors might occur, without breaking. Robustness means to be able to properly react to occurrences that were not foreseen during the design phase, that were not modelled and not pre-programmed into the system. Maybe this robustness is what today's autonomous robots lack the most and what prevents them from being truly universal and applied outside controlled environments or specialized domains. Helping to increase the robustness in a certain area of robotics is the scope of this thesis.

While simple reactive behaviours are sufficient to let an autonomous robot *appear* intelligent[1], a robot has to reason about its environment in order to fulfill a truly useful task. This reasoning has to go beyond what *is* perceived at one very moment in time and has to incorporate knowledge of what *was* perceived and learned in the past. This however requires the robot to collect information about the environment and fuse it with what it already learned or knows a priori. For a robot, *perception* is the beginning of all

[1] [Braitenberg, 1986]

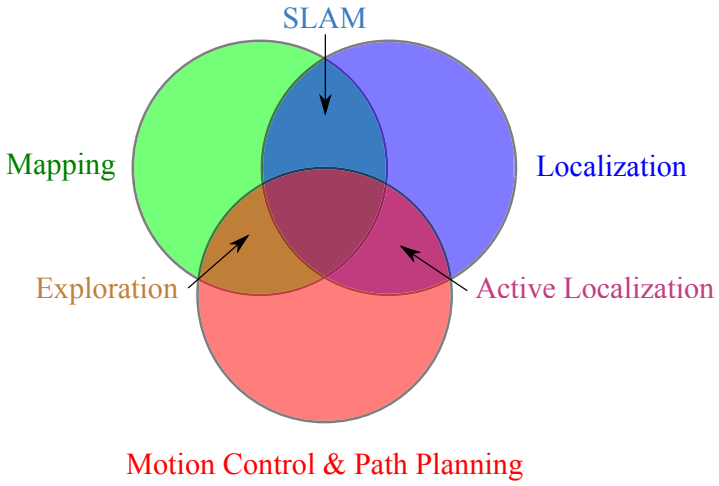


Figure 1.1: *SLAM at the intersection of localization and mapping, compared to other areas of mobile robotics research. Integrated approaches in the center of the figure that combine SLAM and exploration are sometimes referred to as* active SLAM. *Figure adapted after [Makarenko et al., 2002].*

reasoning. This thesis is about a special combination of perception and reasoning that is, at least to some extend, part of most autonomous and mobile robot systems.

Among the many tasks such a robot has to perform, Fig. 1.1 illustrates motion control and path planning, mapping and localization and their intersections. The combination of mapping and localization is called SLAM, which is an acronym for *simultaneous localization and mapping*. SLAM has been one of the major fields of research in mobile robotics during the past two decades. To perform SLAM means to perceive the unknown environment while moving through it and at the same time build a map of what was explored. SLAM is a hard problem for a robot and it used to be a hard problem for human explorers in the past. These explorers set out on their ships into the unknown and mapped the foreign coasts and islands they discovered. Just like the human explorers in the past, robots performing SLAM have to know *where* they are in order to draw *what* they see at the proper position onto the map. Sailors in the past used so called chip logs to estimate their velocity and extrapolated this velocity over time to calculate their position. This *dead reckoning* process inevitably leads to gross positioning errors over time and has therefore always been supported by using *landmarks* for navigation. Polaris[2], the North Star has served countless sailors as an important landmark. It enables one to determine the direction of geographic north but also the latitude of the ship's position. If that landmark was unobservable for several days due to overcast skies, the navigator had to rely solely on dead reckoning. This way, the uncertainty of the navigator's position

[2] α Ursae Minoris, the brightest star in the constellation Ursa Minor is only visible from the northern hemisphere. On the southern parts of the oceans, the Southern Cross roughly indicates the south direction.

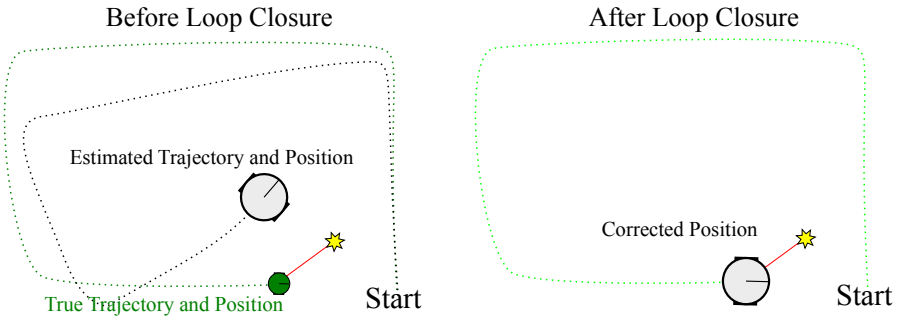


Figure 1.2: *Closing the loop allows SLAM to correct errors in the trajectory and position estimate that accumulated over time. In the left figure, a robot is about to re-observe a landmark with known position. In the right figure, the position estimate has been corrected. Depending on the SLAM algorithm used, the estimation error can be propagated back along the trajectory to correct past position estimates as well.*

estimate grew from day to day, and gross errors were very likely. When the sky was clear again, or a known coast was reached, the ship's assumed position could suddenly be corrected and determined with high certainty again. The same process happens in SLAM for robots. When the robot re-enters an area with known landmarks, it can correct its position estimate. This is called a *loop closure* and illustrated in Fig. 1.2.

However, the mere sighting of a landmark is not sufficient. The observed landmark has to be associated with one of the landmarks in the map. The question therefore is *which* among the presumably many landmarks in the map has just been observed. This problem is called *data association* and was well known to the navigators on the ancient sail ships. The worst data association error in the history of mankind can probably be credited to Christopher Columbus when in the morning hours of October 12^{th} 1492 he erroneously associated the cost of the Bahamas with Asia. This is what happens when the position of the landmarks in your map and your own position are only vaguely known. On the other side, the largest possible loop closure was successfully performed by the expedition of Ferdinand Magellan when his ships re-entered the realm of the Philippine and Maluku islands from the east in the spring of 1521 when these islands had only been reached from the west by Europeans before.

This thesis is about increasing the robustness of SLAM against data association errors, and in particular against false positive loop closure detections. Such false positive loop closure detections can occur when the robot erroneously assumes it re-observed a landmark it has already mapped or when the general appearance of the observed surroundings are very similar to the appearance of other places in the map. Ambiguous observations and appearances are very common in man-made environments (just think about office floors or suburban steets) and the problem is referred to as *perceptual aliasing*.

State of the art SLAM systems formulate the SLAM problem as a least squares optimization problem and use nonlinear optimization approaches to solve it. Such SLAM

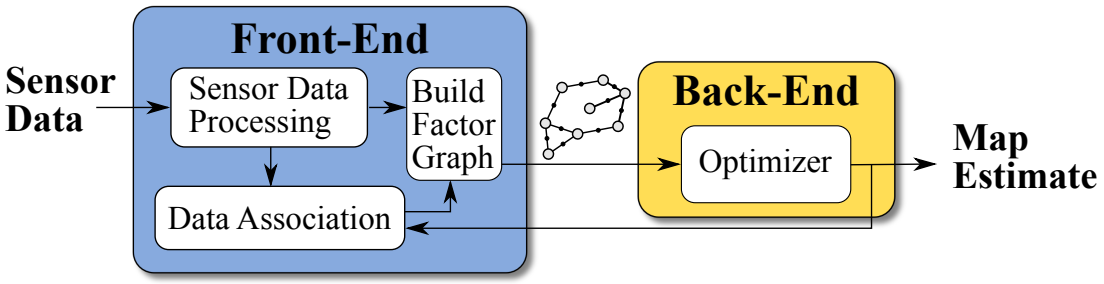


Figure 1.3: *State of the art SLAM systems consist of a front-end and a back-end part. The front-end processes the available sensor data, performs data association and builds and maintains the (factor) graph representation of the SLAM problem. The back-end solves this problem by applying nonlinear least squares optimization techniques.*

systems usually share a common general structure[3]: The system is divided into two parts, a so called *front-end* and a *back-end*. Fig. 1.3 illustrates the concept and the flow of information. The *front-end* has access to the sensor data and its task is to process these sensor data and perform data association. Thus the front-end detects loop closures and is also responsible for the false positive loop closures we want to avoid. The second part of a SLAM system is called the *back-end*. It contains the actual least squares optimizer. A suitable way of representing the least squares optimization problem of SLAM is using a so called *factor graph*. These graphs can be understood as being the interface between the front-end and the back-end. The factor graph is constructed by the front-end, using all available sensor data and data association results, and then handed to the back-end. The back-end in turn extracts the optimization problem encoded in the graph and solves it to gain a map that best represents all the available information.

The gap that can be seen between the front-end and the back-end does not only exist in the illustration. The division between front-end and back-end is very real in today's SLAM implementations (and sometimes even in the SLAM community) and has a number of severe ramifications in practical applications. If one regards the problem of ambiguity and perceptual aliasing and further acknowledges that any sensor measurement and processing is prone to error, it is easy to imagine situations where the front-end erroneously detects a loop closure between two places in the map that in reality do *not* correspond. As we are going to see, such false positive loop closures form misplaced edges in the graph representation that is exchanged between the front-end and the back-end. In this situation, the optimization in the back-end would fail and converge to a solution that corresponds to a corrupted and often useless map. The reason for this behaviour is that the back-end absolutely relies on the front-end to construct a topologically correct graph representation. Any misplaced edges in the graph almost inevitably lead to bad SLAM solutions, since the least squares optimization methods used in the back-ends are

[3] [Konolige et al., 2010b]

not robust against such so called *outliers*. Although the problem in general is largely acknowledged in the literature, no concise solution has been proposed so far. The back-end literature refers to the front-end's responsibility of graph construction. The front-end literature in turn accepts this responsibility and proposes sophisticated algorithms that try to actively prevent the occurrence of data association or place recognition errors. However, especially in large-scale scenarios or when considering long-term operation, current approaches are not sufficient since they cannot guarantee to be free of outliers. Instead of proposing yet another more sophisticated and involved place recognition or data association system, we should therefore accept a reasonable rate of errors on the front-end side and rather concentrate on making the back-end cope with these outlier constraints. This is exactly what I did in the research that led to this thesis. My findings increase the robustness of the overall SLAM system.

Joan Solà wrote in his thesis[4], "SLAM is just a black hole: once you fall inside, you cannot escape.". I am not sure if I fell into that hole, but apparently I got close to it – since I first learned about SLAM at the 2$^{\text{nd}}$ SLAM Summer School in 2004 at LAAS[5] in Toulouse, it never let me go. Maybe this is because SLAM is a more versatile problem than it first appears. The researcher "doing SLAM" can work with many different kinds of sensors, where cameras have been the most fascinating to me. I started to work with stereo cameras, later we used monocular cameras and omni-directional cameras and arrived at RGB-D cameras recently. You can do SLAM with all of these, but each opens a new world with its own set of ideas, problems and intuitions one can learn about. At the heart of SLAM all these sensor information have to be fused, and the SLAM researcher uses all kinds of probabilistic estimation techniques, filtering, smoothing, optimization, to extract information hidden in the data streams. Once the goal of SLAM is reached, and a map is created and constantly updated, new questions arise. What to do with the map? Where to go next? How to get there? What is in the map? What is the semantic meaning of the mapped structure? So for me, SLAM is not so much a black hole, it has been more like a harbour that sent me out to explore into different directions.

Despite my fascination for SLAM, I also wanted to search for ways my findings could be transferred into other areas and applied there. It is curious that although different fields of science try to solve very similar problems, this is often obfuscated by using a different language, giving different names to things, citing different papers, reading different books and attending different conferences. Unintentionally, the flow of ideas is hindered and advances are delayed. So from time to time, it may be worth to step back and try to see the greater picture, or at least try to glance into neighboring fields. Therefore, two chapters at the end of my thesis are dedicated to the application of my results beyond SLAM.

[4] [Solà, 2007]

[5] Laboratoire d'analyse et d'architecture des systèmes, Toulouse, France

Objectives and Contributions

This thesis proposes a novel[6] back-end formulation for SLAM that is robust against data association errors in general and false positive loop closures in particular. The key insight developed during this work is that the gap between front-end and back-end can be closed by allowing the back-end to influence and change the factor graph topology *during* the optimization. As I am going to show, the proposed robust back-end can cope with a large number of outliers in a variety of situations and scenarios.

With the proposed novel robust back-end, optimization-based SLAM systems are not longer dependent on the front-end to provide an absolutely correct and precise problem formulation. This implies that the front-ends can become simpler, more efficient and lightweight compared to current state of the art approaches, since the need for absolute correctness and precision (that cannot be guaranteed anyway) vanishes. While the core of this dissertation takes place on the back-end side of the system layout in Fig. 1.3, the results influence the developments on the front-end side as well. Sticking to the illustration, with the approach proposed here, the gap between both parts vanishes. In the end, the graph construction will merge into the optimization, leading to an integrated overall system.

Although the proposed approach has been developed and first implemented specifically with the SLAM problem in mind, the general concept behind it is rather universal and can be applied to a variety of least squares optimization problems. Two chapters at the end of the thesis are therefore specifically dedicated to applications beyond SLAM that can benefit from the approach.

Thesis Outline

This thesis can be divided into three parts, providing an introduction, the major contributions and an outlook on further applications.

Chapters 2 and 3 form the first part and provide introductory material on SLAM and least squares optimization. The most important developments in the field of SLAM during the past years are reviewed and references to the respective literature are provided. A special focus is put on the formulation of SLAM as a least squares optimization problem. Chapter 3 therefore repeats the essentials of least squares optimization techniques, with a focus on nonlinear least squares and iterative methods. Readers familiar with SLAM and least squares optimization techniques can skip these chapters or quickly browse through them.

The second part motivates and develops the main ideas and contributions of this thesis. In Chapter 4, I provide the main motivation for my work by examples and citations from the recent literature. The main contribution of this work is Chapter 5 which proposes a novel algorithmic and mathematical approach to increase the robustness of optimization-based SLAM back-ends. The evaluation of these ideas in Chapter 6 and the

[6] The main ideas of this work have been published by the author in [Sünderhauf and Protzel, 2011] and [Sünderhauf and Protzel, 2012].

application to a large-scale real-world dataset in Chapter 7 show the feasibility of the approach and affirm the claims and theses of this dissertation.

The last part sketches possible applications of the proposed robust back-end for problems different than SLAM: Chapter 8 shows how the techniques developed in Chapter 5 can be applied for multipath mitigation in GNSS-based localization problems. A short discussion on applications for general sensor data fusion and calibration is given in Chapter 9, before Chapter 10 concludes this thesis.

2

Simultaneous Localization And Mapping

The problem of simultaneous localization and mapping (SLAM) is an old problem in mobile robotics. Not only does it continue to be a very active field of research[7], it is a very versatile problem as well. SLAM adapts approaches, draws inspiration and uses ideas and techniques from very different research areas, ranging from Bayesian probability theory to computer vision, and even neuroscience.

Depending on different criteria, we can divide the research on SLAM into several classes. For instance we can distinguish different SLAM approaches by the kind of map they create, or by the types of sensors that are used. Another important criterion is the type of mathematical algorithm that is used to solve the problem. Fig. 2.1 illustrates the resulting taxonomy of SLAM, without claiming to be thorough and complete.

Since SLAM is in the focus of this thesis, I use this chapter to shortly explain the SLAM problem and review recent developments that are most relevant to the rest of this thesis.

2.1 S, L, and M – The Parts of SLAM

Summarized in a single sentence, the SLAM problem describes the process of a robot building a map of its unknown environment while it is exploring it. The robot in this scenario is equipped with sensors that measure its own movements (e.g. odometry) and other sensors that perceive its surroundings (e.g. a laser range finder, sonar sensors, or a camera). In order to map the collected information about the environment, the robot has to know its position and orientation relative to the map. It therefore has to constantly localize itself in the environment with the help of the still incomplete map.

[7] As these lines are written, Google Scholar lists over 11.000 scientific papers for the search term "SLAM + robot" that were published since 2005.

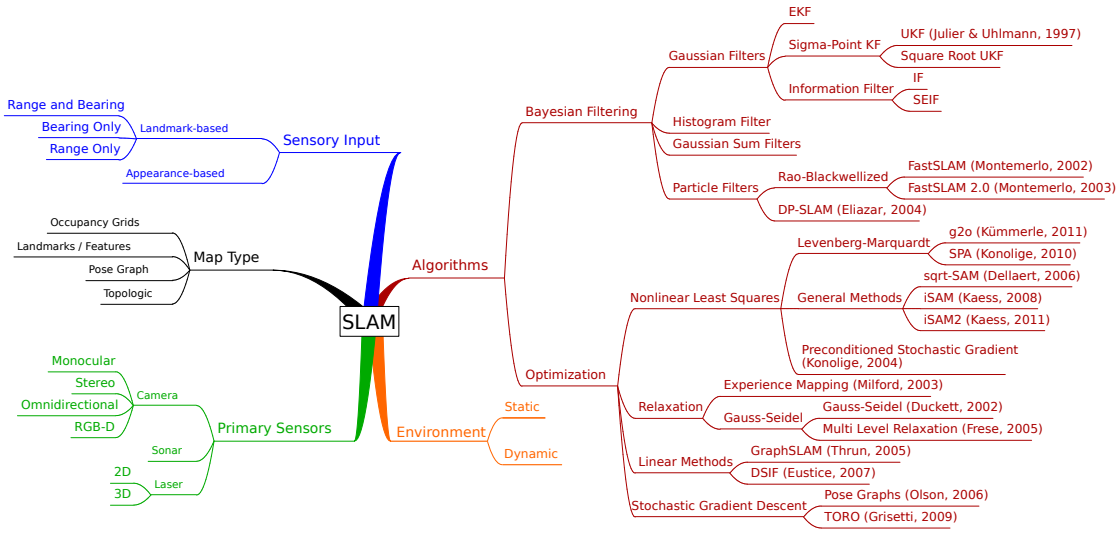


Figure 2.1: *A taxonomy of the SLAM problem.*

Accurate localization is crucial to the process since an inaccurate localization will also cause an inaccurate map, which makes future localization even harder and so on. In SLAM, localization and mapping depend on each other and are executed simultaneously in an intertwined way.

SLAM is a hard problem because due to sensor noise none of the measurements are perfect. This means that neither the robot's movement nor the structure of the environment is known absolutely precisely, but only up to some degree of uncertainty. In order to cope with these ubiquitous uncertainties, SLAM is usually understood and approached using probabilistic models and techniques.

Although [Siciliano and Khatib, 2008, ch. 37] consider the first SLAM problem in the scientific literature to date back to the days of Gauss who developed a least-squares method for calculating the orbits of the planets around the sun [Gauss, 1809], the birth of the "modern" SLAM problem can be traced back to the ICRA conference in 1986 where the first ideas and thoughts on the subject were discussed. Shortly after that, a number of seminal publications established this new field of robotics research. The acronym *SLAM* was later coined in [Durrant-Whyte et al., 1995]. For further references I want to refer the reader to [Durrant-Whyte and Bailey, 2006] where the authors[8] also provide more interesting details on the history of the SLAM problem.

The next three sections are dedicated to the three parts of SLAM: The mapping, the localization and the simultaneous estimation.

[8]The main author Hugh Durrant-Whyte has been participating in the SLAM research from the very beginning.

2.1.1 M for Mapping

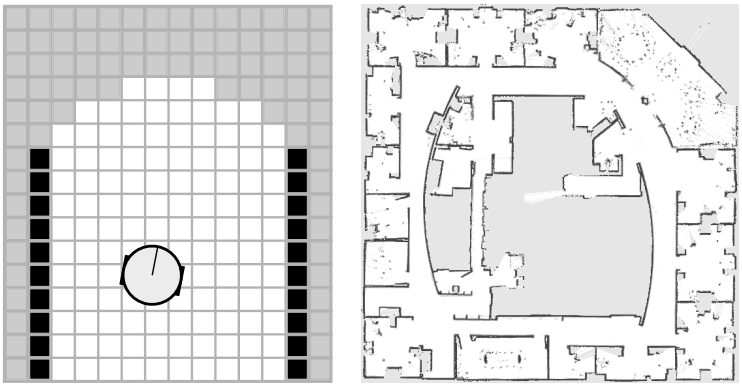
The maps that are created in SLAM can be quite different, depending mainly on the sensor equipment on the robot but also on the environment the robot operates in and its specific task. This section reviews a number of important map types that are known from the literature and have been used in past and present robotics research. Further references on robot mapping can be found in the survey paper [Thrun, 2002], in [Siciliano and Khatib, 2008, ch. 36] and also in [Thrun et al., 2005, ch. 9].

Occupancy Grid Maps

In the early days of SLAM, sonar sensors have been very popular (probably because they were available, cheap, and easy to process without requiring a lot of computational power). So called *occupany grid maps* [Moravec and Elfes, 1985] were therefore a natural representation of the environment at that time. These occupancy grids subsample the environment into discrete regions or cells m_i that cover a certain area of the environment, e.g. of size 10×10 cm. Each of these cells m_i is associated a value that expresses the probability $P(m_i|\mathbf{z}_{0:t}, \mathbf{x}_{0:t})$ that this region m_i is occupied by an obstacle, given all sensor information up to now ($\mathbf{z}_{0:t}$) and all (presumably known) robot poses $\mathbf{x}_{0:t}$. Yet unobserved cells are usually assigned the value 0.5. Fig. 2.2(a) illustrates these ideas.

Occupancy maps are well-suited for environments with a dense observable structure such as indoor scenarios and can be built conveniently from sonar or laser range finder data. Fig. 2.2(b) shows an example occupancy map that was built using the measurements of a laser range finder. In that map of a building, black regions contain obstacles and white areas correspond to regions that can be traversed freely. Grey parts of the map have not yet been observed and therefore are unknown. Due to their structure, occupancy grid maps have been widely used for obstacle avoidance algorithms such as the Vector Field Histogram [Borenstein et al., 1991] or the potential field methods [Khatib, 1986] and general motion planning like Rapidly Exploring Random Trees (RRTs) [Lavalle and Kuffner, 1999].

Occupancy grid maps can also be built from 3D data that is for instance collected by a 3D laser range finder, a stereo camera or a RGB-D camera. Fig. 2.3(a) shows the 3-dimensional point cloud acquired by a laser range finder in our lab. It consists of over 71,000 single points of measurement. Since such large point clouds are hard to handle, they can be compressed into an occupancy structure: Octrees are such efficient data structures that can handle large 3-dimensional datasets. Every node in these tree structures covers a certain volume in space (e.g. $(10\,\mathrm{cm})^3$) and can have up to eight children (hence the name octree). However, a node is expanded and divided further only if a scan point exists inside its volume. This way, only areas that contain structure are represented with high resolution in the tree. The processed and compressed octree map [Wurm et al., 2010; Nüchter et al., 2005] of the 3D laser scan of Fig. 2.3(a) is shown in Fig. 2.3(b). Notice that each of the visible cubes is a leaf in the octree. The tree contains 12,000 leafs which is a significant reduction compared to the original 71,000 scan points.



(a) (b)

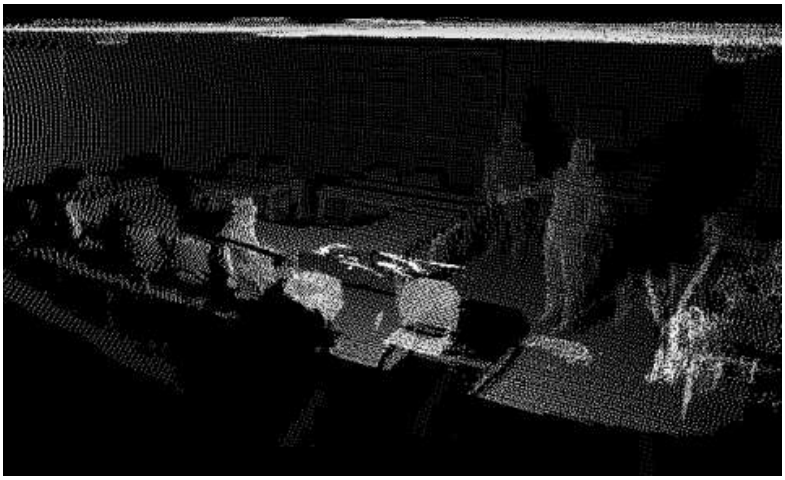
Figure 2.2: *Occupancy grid maps: (a) illustrates the general concept. The map and the space around the robot is partitioned into discrete regions and each region is associated a probability that it is occupied by an obstacle, according to the sensor measurements. In the illustration, black regions are considered to be occupied, white are free of obstacles. Grey regions are unobserved, either because they are blocked by obstacles or they are yet out of sensor range. (b) 2D occupancy grid map created using a Laser range finder in an indoor scenario, called the Intel dataset (Image courtesy of Cyrill Stachnis, raw data courtesy of Dirk Haehnel.)*

Feature Maps

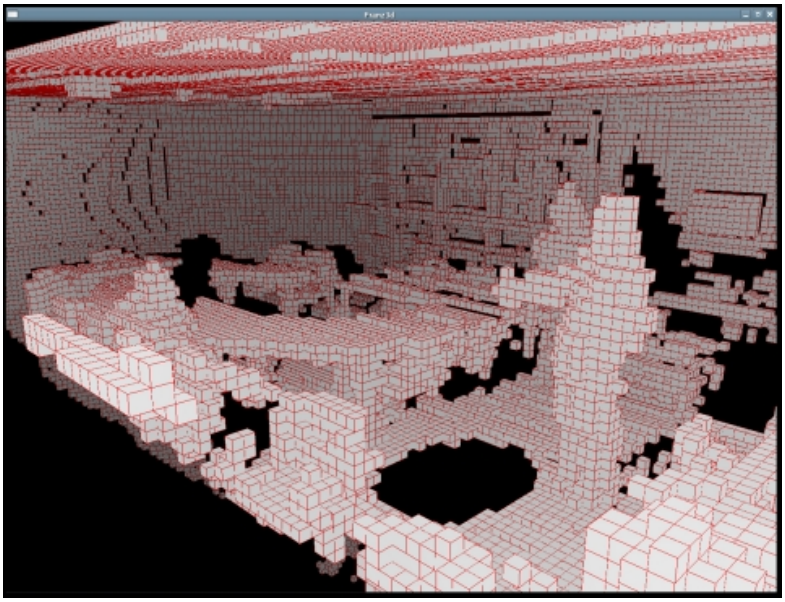
While occupancy grids provide a dense map of the environment, so called *feature maps* are sparse and contain only the position of distinct *features* or *landmarks* in the environment. Fig. 2.4(a) illustrates the general concept. In contrast to occupancy grids, feature maps can be maintained in a single map state vector $\mathbf{m} = (\mathbf{l}_0, \mathbf{l}_1, \ldots, \mathbf{l}_n)^\mathsf{T}$ that contains only the landmark coordinates. As the robot explores the environment and discovers more and more landmarks, this vector will of course grow larger because the newly discovered landmarks are appended to the state vector.

Feature maps are well suited for environments that contain distinctly recognizable landmarks. For instance in the Victoria Park dataset [Guivant et al., 2002] (see Fig. 2.4(b)), the landmarks are the trunks of trees that have been extracted from laser scan data. Artificial landmarks and markers can of course also serve as features for a map, as well es visual features. These special features are described after shortly explaining what types of landmarks can be distinguished in general.

Types of Landmarks Since the landmarks used for feature mapping are observed by the robot's sensors, we can distinguish three types or modes of landmark observations, depending on the sensor's measurement principles. These three modes are illustrated in



(a)



(b)

Figure 2.3: *(a) A 3D point cloud acquired with a 3D laser range finder in our lab. Visible are several chairs, tables, and two persons. The scan consists of over 71,000 single scan points and has been compressed into an octree structure in (b). This octree data structure contains only approximately 12,000 nodes, which are visible as cubes in the image.*

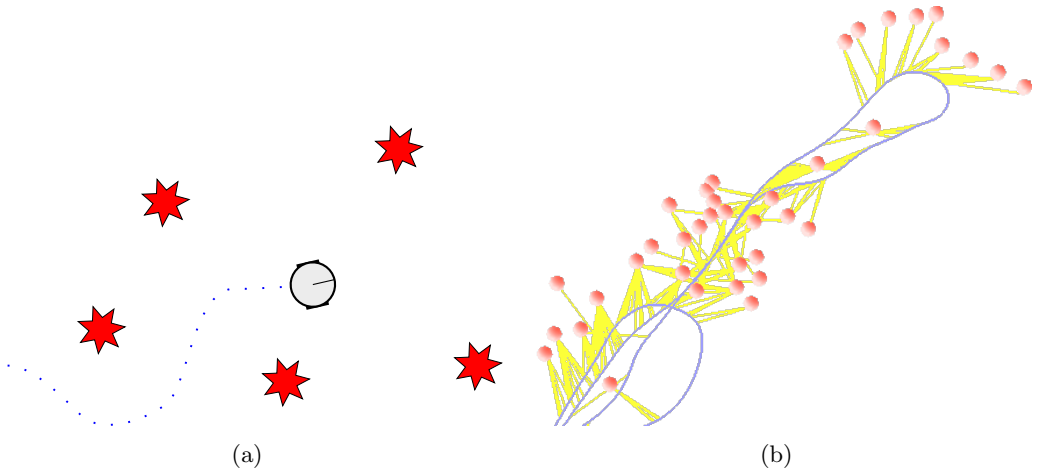


Figure 2.4: *(a) illustrates the general concept of feature maps that contain only the coordinates of distinguishable, discrete landmarks. In contrast to occupancy grid maps that provide dense information, such feature maps are therefore rather sparse. (b) shows a part of a feature map constructed from the Victoria Park dataset [Guivant et al., 2002]. The red points are the landmarks, which in this example are tree trunks extracted from laser scans. The estimated robot path is shown in blue. The yellow lines connect the landmarks with the places from where they have been observed.*

Fig. 2.5 and explained in the following:

- *Range and bearing* landmarks are the easiest to work with, since their position in space relative to the sensor (and hence to the robot) is well-defined by the observation. Examples for such landmarks are corner features in laser scans or visual landmarks retrieved from stereo cameras or RGB-D cameras.
- *Bearing only* observations are typically provided by monocular camera systems that can only measure the bearing of a visual landmark relative to the camera coordinate frame, but not the landmark's depth or distance.
- *Range only* measurements occur for instance when using the pseudoranges to GPS satellites or the signal strength of WiFi transmitters for localization.

Visual Landmarks for SLAM In recent years, visual landmarks have become popular for SLAM. They are extracted from the images provided by cameras of different types, such as stereo cameras, monocular cameras or RGB-D cameras. Depending on the type of camera, the feature observations will be bearing only (for monocular cameras) or provide both range and bearing for stereo and RGB-D cameras. The computer vision literature knows a vast number of feature detectors and descriptors that are suitable to extract

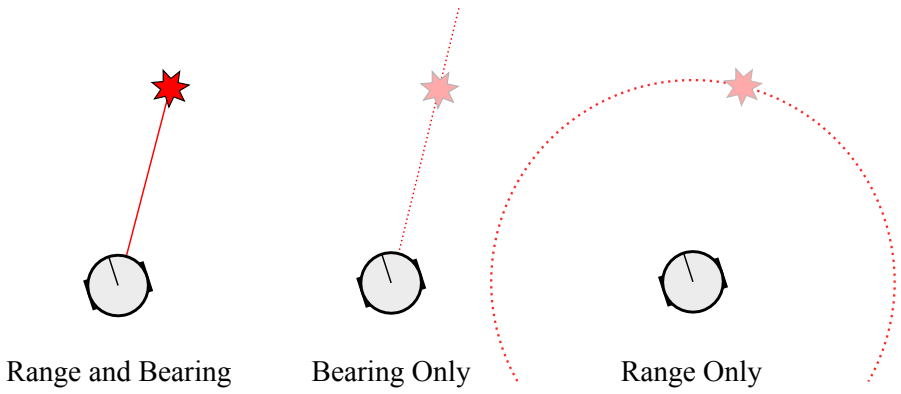


Figure 2.5: *Different modes of landmark observations: Often range and bearing cannot both be measured. For instance, in visual SLAM with a single (monocular) camera, only the bearing to a visual landmark is known. Its range has to be estimated over time, e.g. after re-observing that landmark a second time from a different location that introduces a so called motion parallax. In contrast, when using stereo or RGB-D cameras, the range to a landmark can be immediately measured. Range only measurements in contrast commonly occur when using GNSS or other radio sources like Wi-Fi for localization.*

visual features from images and describe their appearance. Well-known examples for such feature extractors are (among others) SIFT [Lowe, 2004], SURF [Bay et al., 2006], CenSurE [Agrawal and Konolige, 2008], Harris [Harris and Stephens, 1988], MSER [Matas et al., 2002] or FAST [Rosten and Drummond, 2006]. Fig. 2.6 shows two examples of visual features extracted from an urban scene.

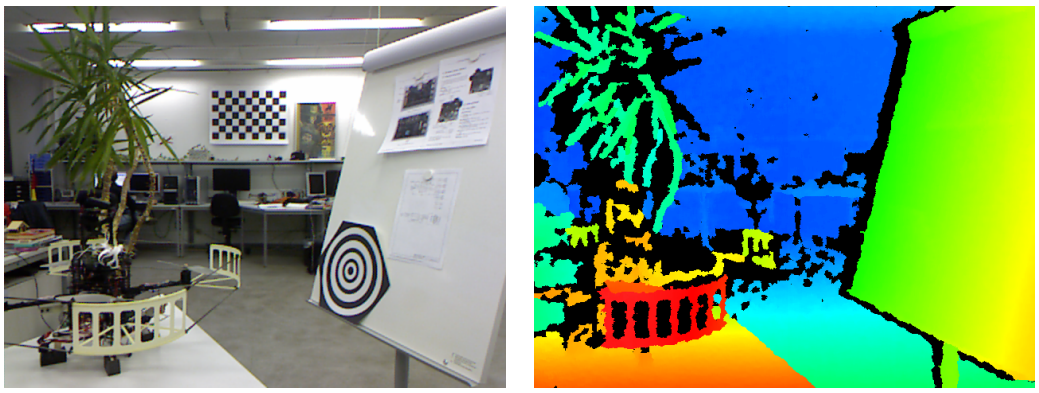
Examples for visual SLAM using stereo cameras (to measure the 3D coordinates of the visual landmarks) are [Se et al., 2002] or [Jung and Lacroix, 2005]. 3D point clouds [Rusu and Cousins, 2011; Rusu, 2009] that are acquired by RGB-D cameras like the Microsoft Kinect or the PrimeSense device (see Fig. 2.7) have recently become a very active field of research and have also been used for robotic mapping and SLAM [Henry et al., 2010; Engelhard et al., 2011].

[Davison, 2003] used visual landmarks retrieved from a monocular camera for SLAM. This *MonoSLAM*[9] approach was adapted by a number of other authors [Solà et al., 2005; Montiel et al., 2006; Solà, 2007; Klein and Murray, 2007]. We also contributed to this strand of SLAM in [Sünderhauf et al., 2007]. The main challenge in SLAM with monocular (bearing only) landmarks is that the position of the landmarks can only be determined by triangulation after they have been observed several times from different viewpoints as illustrated in Fig. 2.8. That means, a landmark cannot be automatically incorporated into the map when it is observed for the first time. More precisely, the conventional Cartesian parametrization of the landmark position, i.e. $\mathbf{x} = (x, y, z)^\mathsf{T}$,

[9] In the computer vision community, the term *structure from motion* refers to the same problem.

(a) (b)

Figure 2.6: *Examples for visual features: (a) SIFT features [Lowe, 2004] and (b) features based on perceptual saliency, following a biologically motivated model by [Itti et al., 1998].*



(a) (b)

Figure 2.7: *RGB-D cameras like the Microsoft Kinect or PrimeSense devices provide a conventional RGB image (a) and a depth image (b) of the observed scene. In the depth image in (b), small depth (distance) is depicted with reddish colors, while blue corresponds to larger distances. Notice that the depth could not be calculated for all pixels (black areas). Given the depth image and the internal camera parameters, the original position* $\mathbf{x}_i = (x_i, y_i, z_i)^\mathsf{T}$ *in 3D space of almost all pixels* $\mathbf{p}_i$ *can be determined.*

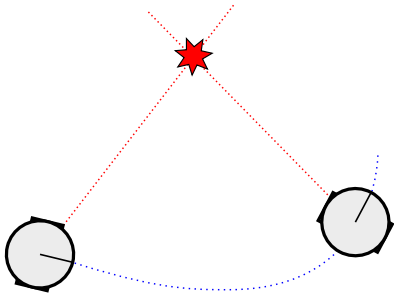


Figure 2.8: *If only the bearing (or the range) of a landmark can be observed, its position cannot be determined immediately. The landmark rather has to be observed multiple times from different viewpoints. If the parallax induced by the robot's motion is large enough, the landmark's position can be triangulated.*

Figure 2.9: *Landmarks in the direction of motion are hard to triangulate. Due to the small parallax between the two observations, the uncertainty in the landmark position is high in the longitudinal direction, but small in the lateral direction. While such landmarks can hardly help the robot to estimate its position, they still provide valuable information on the robot's orientation.*

prevents this *undelayed initialization* since the $(x, y, z)^\mathsf{T}$ position cannot be uniquely determined if only the bearing is observed. This is also the case when the robot moves towards the landmark as illustrated in Fig. 2.9. In this case the induced motion parallax is small, i.e. the viewing angle does hardly change and hence triangulation is not possible or only with small precision.

Alternative landmark parametrizations such as the inverse depth parametrization [Montiel et al., 2006] foster undelayed initialization. [Solà et al., 2011] compares different of such strategies for landmark parametrization in monocular SLAM. The most recent developments in the area of monocular SLAM however seem to point towards feature-less approaches that directly perform dense 3D reconstruction [Newcombe et al., 2011].

A good survey on monocular and stereo vision approaches to SLAM is provided by [Lemaire et al., 2007]. Landmarks of higher order, i.e. not only simple point features, but line segments have been proposed to be used as landmarks for SLAM in [Smith et al., 2006], [Eade and Drummond, 2009] and [Lemaire and Lacroix, 2007]. An overview on vision for SLAM, including high level features for mapping, is given in [Lacroix et al., 2008].

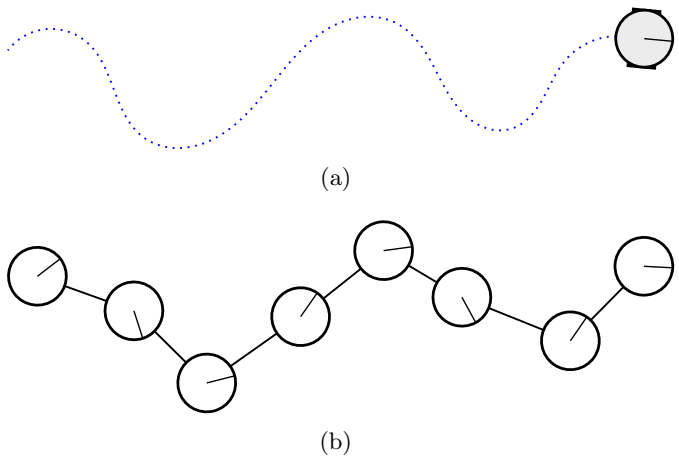


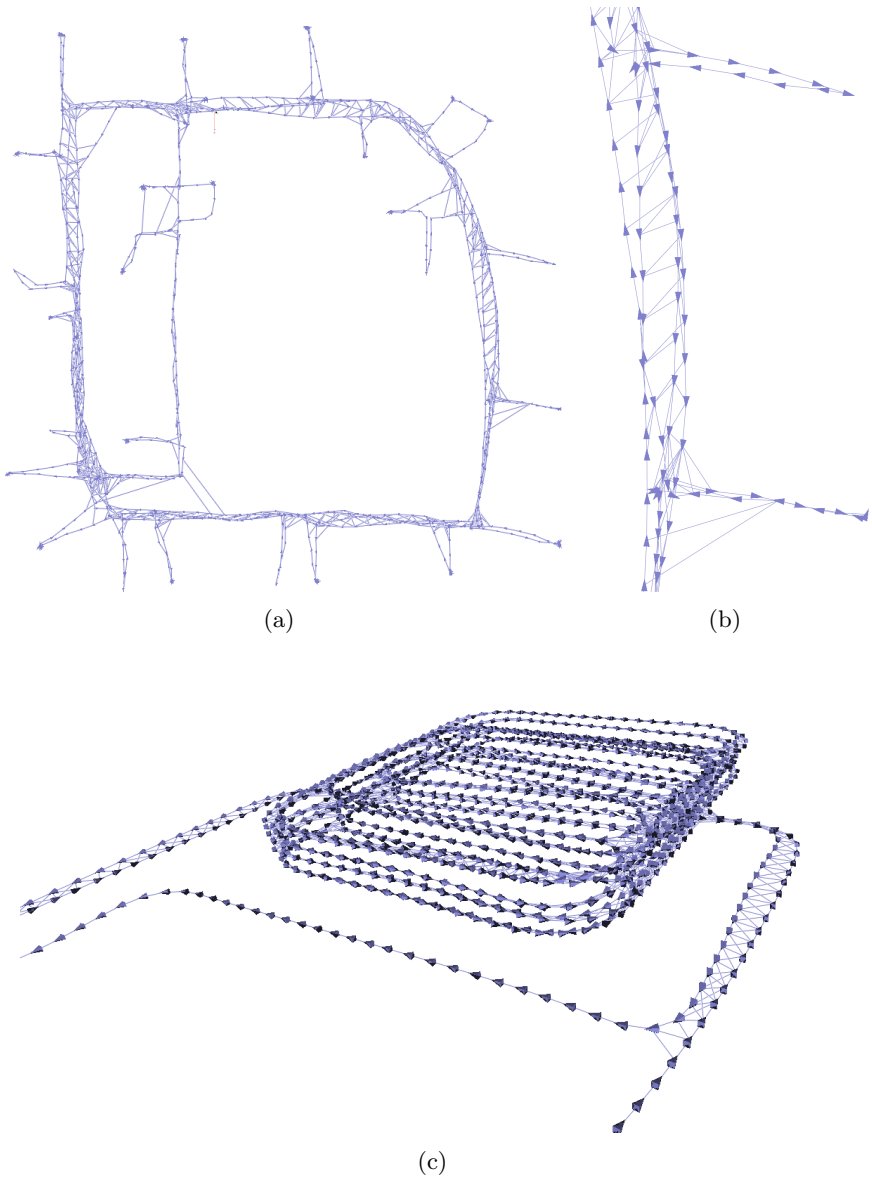
Figure 2.10: *The general concept of a pose graph: The robot's trajectory (a) is represented as a graph (b) where the nodes are the robot poses (expressing position and orientation) at a certain point in time and the edges contain the spatial relation between these poses. A new node is inserted into the graph whenever the robot's pose underwent significant changes, e.g. because the robot turned or moved.*

Pose Graphs

Besides occupancy grids or feature maps, the so called *pose graphs* are a more abstract and general type of map and have recently received a lot of attention in SLAM research [Olson et al., 2006; Kaess et al., 2008; Konolige et al., 2010a; Kümmerle et al., 2011b]. Due to their generality, pose graphs play an important role in this thesis. Especially when the contributions and novel ideas are formulated in Chapter 5 and evaluated later in Chapters 6 and 7, they will be the map type of choice.

Pose graphs represent the robot's trajectory as a graph structure where the nodes represent the robot poses[10] and the edges between the nodes represent the spatial connections between these poses. The edges contain for instance odometry information or express loop closures. Fig. 2.10 illustrates that concept, while further examples for 2D and 3D pose graphs can be seen in Fig. 2.11. Notice that explicit landmarks or other information about the environment are not part of the pose graph itself. However, such information can be *attached* to the nodes in the graph, hence allowing the pose graph to express occupancy grids in 2D or 3D or feature maps and any combination of those. For instance, if we would attach the laser scan for each pose in the map of Fig. 2.11(a), we would gain a similar structure as in Fig. 2.2(b), since both maps were calculated on the same raw data in the same environment. This way, despite their simplicity, pose graphs are a powerful tool in robotic mapping and SLAM today.

[10]The term *pose* typically means *position* together with *orientation*, i.e. in a 2D world a pose would be written as a vector $\mathbf{x}_i = (x_i, y_i, \theta_i)^\mathsf{T}$.



(a) (b)

(c)

Figure 2.11: *Pose graphs of the Intel dataset (a) and a close-up view of the right part of the map (b). The 3D graph in (c) shows the Parking Garage dataset. Notice how the graphical representation of the nodes illustrates both the position and orientation of the robot. The edges between the single nodes represent either odometry or loop closure information.*

2.1.2 L for Localization

While the M-part of SLAM is about how the world looks like and deals with questions of how the collected sensor information can be processed and saved into a map structure in a suitable way, the L-part answers the robot's question "Where am I?" [Borenstein et al., 1996].

Localization of a robot in a known map is a well-researched problem of its own. We can distinguish between *local* and *global* localization problems: In *global* localization, we do not have any prior knowledge about the position of the robot. It can be anywhere in the map and has to re-localize using the map and the available sensor information. This kind of problem is also known as *kidnapped robot problem*. Usually, the re-localization cannot be accomplished immediately, especially in man-made environments with a lot of repetitive and self-similar structure. Until a stable pose estimate is established, several hypotheses have to be maintained, which favours particle filtering approaches like [Fox et al., 1999].

In SLAM, the kidnapped robot situation usually does not occur and the localization is *local* since it can always use prior knowledge about the probable robot pose. Often, the robot pose is already known up to a small displacement or error. Scan matching techniques are commonly applied when the robot is equipped with a laser range finder. Examples for such localization techniques can be found in [Gutmann and Konolige, 1999] or more recently in [Diosi and Kleeman, 2007] or [Censi, 2008].

Place Recognition

A special sub-class of the localization problem very common when dealing with pose graph SLAM is *place recognition*. Place recognition means that the robot has to recognize known places in the environment, thus decide whether it has already been in that particular area of the environment it currently perceives at some time in the past or whether this area is completely new and unknown.

So called *appearance-based* methods approach the problem by evaluating the visual appearance of the environment to the robot. Fig. 2.12 illustrates that concept. Common systems apply the *visual vocabulary*[11] approach borrowed from the text retrieval community [Sivic and Zisserman, 2003]. Fig. 2.13 illustrates the training phase of such a system: From a large and preferably representative set of images, keypoints like SIFT [Lowe, 2004], SURF [Bay et al., 2006], CenSurE [Agrawal and Konolige, 2008] or others are extracted and descriptors for these keypoints and their surrounding image patches are calculated. These can be SIFT or SURF descriptors or others like BRIEF [Calonder et al., 2010]. The large collection of descriptor vectors are clustered e.g. by using k-means or hierarchical k-means. The centers of these clusters are kept as so called *visual words* and form a database which is, due to the roots of the approach, called the *dictionary* or visual vocabulary. In the query phase, keypoints and descriptors for the current scene are extracted and the descriptor vectors are classified using the database of learned visual words. That is each observed descriptor is associated with a visual word, e.g. by

[11] also known as *bag of words*

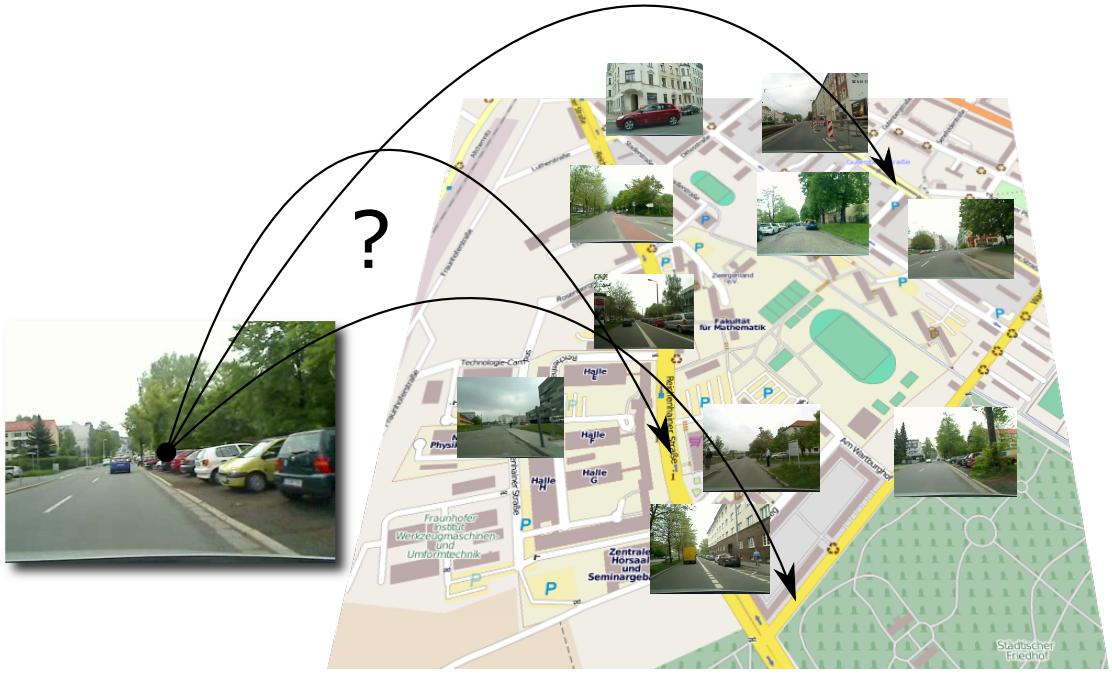


Figure 2.12: *The (visual) place recognition problem: Given a number of images from already visited places in the world and a query image, decide where in the world the query image has been recorded or if it originates from a previously unseen location.*

finding the most similar word in the database. The scene can then be described by a statistic over the occurrence of visual words, by counting how often individual visual words occur in that specific scene, weighted by the overall occurrence of that word etc. Place recognition can then be performed by finding the most similar scenes according to these statistical measures. [Sivic and Zisserman, 2003] e.g. proposed to use this simple *term frequency – inverse document frequency* scheme.

Further developments have been conducted by [Nister and Stewenius, 2006] who introduced tree structures to manage large databases of visual words or [Schindler et al., 2007] who demonstrated city-scale place recognition using these tree structures. [Angeli et al., 2008] proposed an incremental scheme to learn a visual dictionary online i.e. during the operation of the robot, in contrast to the previous approaches which used a static dictionary acquired beforehand.

A crucial issue in appearance-based place recognition systems are false-positives. That means if two places appear to be very similar but are actually at two distant positions in the environment, they can be erroneously recognized as a loop closure. This usually has devastating effects on the resulting map built by the SLAM system. Especially in man-made environments that suffer from a high self-similarity and repetitive structure, false place recognitions are very likely. In fact, this is exactly where this thesis tries to

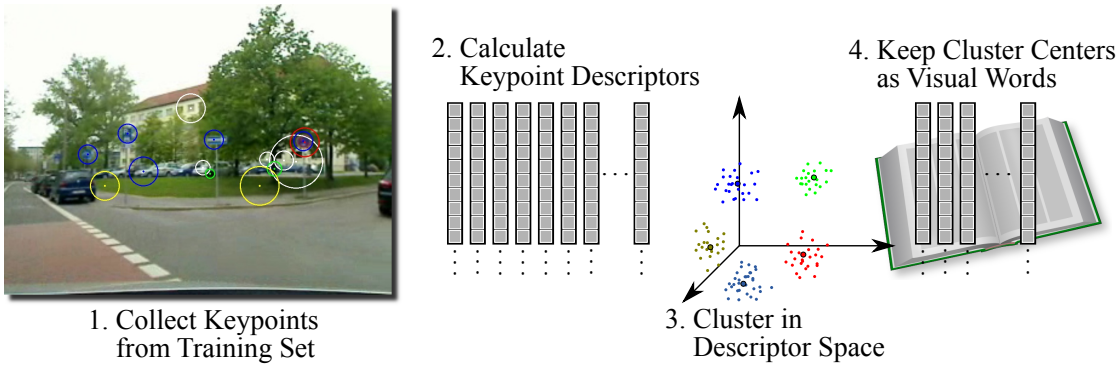


Figure 2.13: *Training phase of a simple appearance-based place recognition system: From a large collection of representative images, keypoints are extracted and their descriptors are calculated. By clustering these descriptors and keeping only the cluster centers, a "dictionary" or "vocabulary" of "visual words" (also called a "bag of words") is built.*

help, by making the SLAM systems robust against these place recognition errors.

To prevent these false-positives (or at least to decrease the false-positive rate), several advanced approaches have been proposed. For instance the FAB-MAP[12] system introduced in [Cummins and Newman, 2008] captures the mutual information and dependencies of the visual words and the correlations between them. The underlying idea is that some of the visual words tend to co-occur together with other words, since they are created from common objects in the environment. FAB-MAP applies an unsupervised learning stage to learn these dependencies between the visual words and approximates them in a special structure, a Chow-Liu tree [Chow and Liu, 1968]. FAB-MAP has been successfully applied to place recognition on very large datasets [Cummins and Newman, 2009] and extended to 3D environments by combination with a 3D laser range finder [Paul and Newman, 2010]. A further improvement of the robustness of FAB-MAP has been reported by [Maddern et al., 2011] after incorporating odometry information into the place recognition process. Although FAB-MAP has been shown to work well under different conditions, several authors reported that it still produces false positive place recognitions that will eventually lead to false positive loop closure constraints. We will come back to that problem in Chapter 4, where the motivation for this thesis is explained more elaborately. For now it suffices to keep in mind that although a number of approaches for appearance-based place recognition or loop closure detection exist, they are not guaranteed to work perfectly.

Another strand of appearance-based place recognition is the use of so called *holistic* descriptors, i.e. descriptors that describe the appearance of the complete scene and not of single keypoints in it. In own previous work [Sünderhauf and Protzel, 2011] we developed such a holistic descriptor which we coined BRIEF-Gist. However, the idea of a

[12]short for Fast Appearance Based Mapping

holistic scene descriptor is not new and was e.g. examined by [Oliva and Torralba, 2001] and [Oliva and Torralba, 2006] with the introduction of the Gist descriptor. This global image descriptor is built from the responses of steerable filters at different orientations and scales. More recently, [Murillo and Kosecka, 2009] demonstrated place recognition using the Gist descriptor on panoramic images in an urban environment.

Another interesting place recognition system based on biological principles was described by [Siagian and Itti, 2007]. They use models of human early vision to extract both salient features and the gist of the scene to form a feature vector that serves as a scene descriptor and demonstrated the recognition performance of their system on several smaller datasets in an urban environment[13].

2.1.3 S for Simultaneous

While both the mapping and the localization sub-problems can be considered well understood, the hardness of the SLAM problem originates from the S in SLAM: Localization and mapping have to be executed simultaneously.

Two main forms of SLAM can be distinguished: In the *full* SLAM problem, the map M and the complete *sequence* of robot poses $X_T = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\}$ have to be estimated at once. The *online* SLAM problem is interested only in the *most current* robot pose $\mathbf{x}_t$ and the map M. Notice that the map is not written with a temporal index, since we assume that the environment is static and therefore also the map remains constant over time[14]. Both M and X_T or $\mathbf{x}_t$ are estimated simultaneously, we will also speak of them as being estimated *jointly*.

Motion Models

Since the poses X_T and the map M are not known a priori, they have to be estimated using the given sensor information. Among these sensor measurements, we are usually given odometry information that describe the movement between the single poses. The odometry (that is sometimes referred to as *control input*) is denoted

$$U_T = \{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_T\} \tag{2.1}$$

[13] As a personal note, I find such biologically motivated systems very appealing since they offer a completely different viewpoint and motivation on certain problems in robotics. Although such approaches may lack the underlying huge mathematical framework and justification of conventional methods, they often work astonishingly well and produce competitive results. Maybe the most impressive example is RatSLAM, a complete working SLAM system that is built on a computational model of brain cells present in mammalian hippocampi [Milford and Wyeth, 2008; Milford, 2008]. In previous work, we reviewed this system and the involved brain cells [Sünderhauf and Protzel, 2010b] and proposed a more efficient implementation of the underlying neural models [Sünderhauf et al., 2010; Sünderhauf and Protzel, 2010a].

[14] The problem where the environment may change is referred to as dynamic SLAM. Most of the algorithms in the literature however assume a static world.

and the relation between two successive poses and the respective odometry measurement is given by a usually nonlinear motion model[15] f:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{w}_t \tag{2.2}$$

The noise of the odometry sensor system is usually modelled as zero-mean Gaussian and is (approximately) captured by the additional term $\mathbf{w}_t$ that follows a Gaussian distribution with covariance $\mathbf{\Sigma}_t$:

$$\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_t) \tag{2.3}$$

Therefore, $\mathbf{x}_{t+1}$ is modelled as a Gaussian distribution as well:

$$\mathbf{x}_{t+1} \sim \mathcal{N}(f(\mathbf{x}_t, \mathbf{u}_t), \mathbf{\Sigma}_t) \tag{2.4}$$

The term *odometry* here does not necessarily mean data collected from simple wheel encoders. It can also refer to more sophisticated approaches like visual odometry [Nister et al., 2004; Olson et al., 2000][16] or odometric information extracted from laser scan matching [Rusinkiewicz and Levoy, 2001; Diosi and Kleeman, 2007] etc.

Depending on the robot configuration and the actual sensor system used, the control inputs $\mathbf{u}_t$ can be very different: It can be a displacement vector $(\Delta x, \Delta y, \Delta\theta)^\mathsf{T}$ measured in the local robot coordinate system, which would for instance be acquired by a visual odometry system. An IMU[17] would provide accelerations and turn rate: $\mathbf{u}_t = (\ddot{x}, \ddot{y}, \dot{\theta})^\mathsf{T}$. A skid steered robot's simple odometry sensor may return the counted ticks of the wheel encoders $\mathbf{u}_t = (c_{\text{left}}, c_{\text{right}})^\mathsf{T}$ or forward velocity and turn rate, so $\mathbf{u}_t = (v, \omega)^\mathsf{T}$. For an Ackerman vehicle this may be forward velocity and steering angle, resulting in $\mathbf{u}_t = (v, \phi)^\mathsf{T}$. For aerial vehicles like quadrotors the control inputs can be given in terms of turn rates and thrust or lift force or in terms of propeller revolutions etc.

This is only a small collection of possible control inputs $\mathbf{u}_t$. It is obvious that the control input and motion model function f depend very much on the actual robot system and its configuration. A good starting point for further information on motion models for vehicles operating in 2D is [Thrun et al., 2005, ch. 5] that discusses different models and provides references for further information or [Schubert et al., 2011].

As an example, let us consider a skid steered robot with the three dimensional state vector $\mathbf{x} = (x, y, \theta)^\mathsf{T}$ and control inputs $\mathbf{u}_t = (v, \omega)^\mathsf{T}$. A suitable motion model could be

[15] Notice that other authors prefer to write $\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t)$, i.e. they change the involved indices.

[16] Previous own contributions to the field of visual odometry were [Sünderhauf et al., 2005; Sünderhauf and Protzel, 2006; Sünderhauf and Protzel, 2009] and the technical report [Sünderhauf and Protzel, 2007].

[17] Inertial Measurement Unit

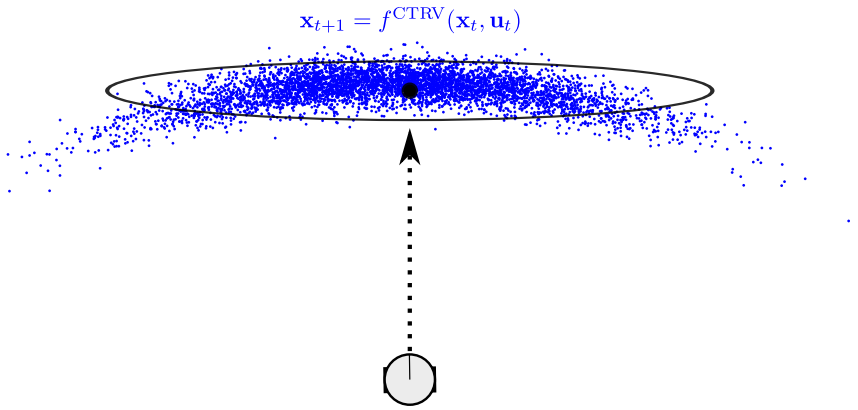


Figure 2.14: *A skid steered robot moving according to a constant turn rate and velocity motion model. If the control input* $\mathbf{u}_t = (v, \omega)^\mathsf{T}$ *is subject to noise (e.g.* v *and* ω *follow a Gaussian), the distribution over* $\mathbf{x}_{t+1}$*, illustrated by the blue points, resembles a banana-like shape. Each of the blue points is the result of an experiment where the robot moved from its initial position according to the motion model* f^{CTRV} *and the control input* $\mathbf{u}_t$ *was spoiled by Gaussian noise. Notice that due to the nonlinearities in the motion model, the resulting posterior distribution over* $\mathbf{x}_{t+1}$ *is clearly* not *Gaussian, although the control inputs* were *Gaussian. The Gaussian error model (black ellipse) can therefore only approximately capture the true distribution.*

the constant turn rate and velocity model f^{CTRV} which is given by

$$\mathbf{x}_{t+1} = f^{\text{CTRV}}(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t + \begin{cases} \begin{pmatrix} v_t \cos(\theta_t)\Delta t \\ v_t \sin(\theta_t)\Delta t \\ 0 \end{pmatrix} & , \omega_t = 0 \\ \begin{pmatrix} \frac{v_t}{\omega_t}\left(\sin(\theta_t + \omega_t \Delta t) - \sin(\theta_t)\right) \\ \frac{v_t}{\omega_t}\left(\cos(\theta_t) - \cos(\theta_t + \omega_t \Delta t)\right) \\ \omega_t \Delta t \end{pmatrix} & , \omega_t \neq 0 \end{cases} \tag{2.5}$$

In Fig. 2.14 we see how a robot's pose evolves under such a motion model if the control inputs are subject to Gaussian noise. The blue points form a very typical banana-like shape and have been acquired by Monte-Carlo simulation. They illustrate the correct distribution over $\mathbf{x}_{t+1}$ that is only approximately covered by the Gaussian model (2.4) represented by the black ellipse. The reason for the discrepancy between the real and the approximated Gaussian distribution lies in the nonlinearities of the motion model. It is important to keep in mind that although many processes in robotics and SLAM are modelled as Gaussians, they are often only a convenient approximation to the more complex real system behaviour.

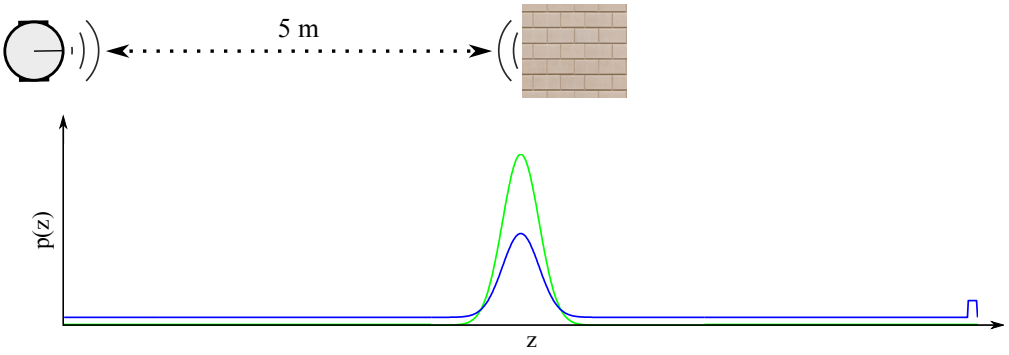


Figure 2.15: *Comparison of an ideal Gaussian (green) and more realistic (blue) model for a sonar sensor (adapted after [Thrun et al., 2005, ch. 6.3]). The ground truth measurement is* $\hat{z} = 5\,\mathrm{m}$.

Sensor Models and Measurement Functions

While $\mathbf{u}_t$ contained the control input or odometry readings, other sensor measurements are usually collected into a single variable vector and written as:

$$Z_T = \{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_T\} \tag{2.6}$$

These additional sensor readings could for instance be range sensor measurements, landmark observations, or global position information. Fig. 2.5 illustrated three different modes of landmark observations, where either the bearing or the distance to a landmark or both are directly observed.

A sensor model function h allows us to predict the measurements given the current estimates of the map and the robot pose:

$$\mathbf{z}_t = h(\mathbf{x}_t, M) + \boldsymbol{\lambda}_t \tag{2.7}$$

where $\boldsymbol{\lambda}_t$ captures the sensor noise and uncertainties and is usually modelled as zero-mean Gaussian:

$$\boldsymbol{\lambda}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_t) \tag{2.8}$$

and therefore

$$\mathbf{z}_t \sim \mathcal{N}(h(\mathbf{x}_t, M), \boldsymbol{\Lambda}_t) \tag{2.9}$$

Like the motion model f, the sensor model h usually is a nonlinear function in practical applications. It may for instance describe the measurement model of a sonar or laser range finder, or in more complicated cases encode a pinhole camera model for visual SLAM applications.

The Gaussian measurement models are often only a coarse approximation of the real sensor behaviour. Fig. 2.15 illustrates this for the example of a sonar sensor. Suppose the sonar sensor is measuring an obstacle in a ground truth distance of $\hat{z} = 5\,\mathrm{m}$. The

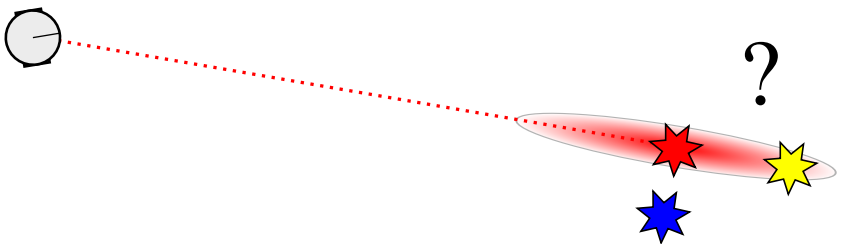


Figure 2.16: *The problem of data association: The robot already has mapped two landmarks (the yellow and blue star). It now observes a landmark at the position indicated by the red star. The resulting uncertainty in the position of the landmark (which is due to the sensor noise associated with the range-and-bearing measurement* $\mathbf{z}_i = (r, \varphi)^\mathsf{T}$*) is illustrated by the faint red covariance ellipse. Which of the two known landmarks was observed here? Or does the measurement originate from a third, previously unobserved landmark? Notice that when using the Euclidean distance, the nearest neighbor of the observed landmark is the blue one, but using the Mahalanobis distance (thus taking the measurement uncertainties into account), the yellow landmark is closer to the observation.*

green curve shows the ideal Gaussian sensor model with the mean μ at $\hat{z}$ and a standard deviation of 0.2 m. The real sensor behaviour may be more correctly captured by the blue curve. Here, the Gaussian is overlaid by a uniform distribution that captures the probability that the sensor may return *any* value because it captured an old sonar echo or because of sensor crosstalk. The maximum possible sensor range ($z_{\max} = 10\,\text{m}$ in the example) is usually also prone to an increased probability, since many sensors return their maximum readings in case of error or malfunction.

Data Association

The data association or correspondence problem describes the process of associating sensor data with real-world modalities. It is for instance apparent in landmark-based SLAM, where the observed landmarks have to be correctly identified in order to correctly process the sensor information and update the map. In other words, the measurement $\mathbf{z}_t$ has to be either associated to one of the n known landmarks $\mathbf{l}_1, \ldots, \mathbf{l}_n$, or a new landmark $\mathbf{l}_{n+1}$ has to be introduced. This is illustrated in Fig. 2.16. Different techniques can be used to decide which of the two known landmarks was observed in Fig. 2.16. A simple nearest neighbor approach using the Euclidean distance would associate the observation with the blue landmark, since it is closer to the observation than the yellow landmark. A maximum likelihood approach however would choose the yellow landmark, since its Mahalanobis distance (taking the measurement uncertainty into account) from the measurement is smaller than that of the blue landmark.

Other, more sophisticated data association approaches are *joint compatibility branch and*

bound [Neira and Tardos, 2001] which adheres to the mutual compatibility and correlation between landmarks. [Thrun et al., 2005] furthermore discusses several approaches of performing SLAM with unknown correspondences, when landmarks cannot be uniquely identified. Other approaches like CRF-Matching [Ramos et al., 2007] involve conditional random fields [Lafferty et al., 2001] or graph partitioning [Olson et al., 2005].

Notice that the place recognition problem (Fig. 2.12) is a special variant of data association, where the observed "landmarks" are complete scenes or places and the observations are the visual appearances of these places. Place recognition therefore is data association in appearance space.

Probabilistic Estimation

Now that motion models, sensor models and the data association problem have been introduced, we are ready to write down what SLAM does in a mathematical way. As I mentioned before, most SLAM approaches use probabilistic techniques. For the *full* SLAM problem, the quantity we want to estimate can therefore be written as:

$$P(X_T, M|U_T, Z_T) \tag{2.10}$$

This expresses that SLAM tries to estimate the probability distribution over the robot poses X_T and the map structure M, given (and using) the odometry measurements U_T and all other available sensor information Z_T. Notice that the large index T in X_T etc. refers to $X_T = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\}$, meaning all available points in time e.g. from the beginning to the end of the robot's mission. The *full* SLAM problem estimates the complete trajectory X_T all at once, using all the collected information.

In contrast, if we only estimate the single pose at the current time t, the estimated quantity is written as

$$P(\mathbf{x}_t, M|U_{t-1}, Z_t) \tag{2.11}$$

where $U_{t-1} = \{\mathbf{u}_0, \ldots \mathbf{u}_{t-1}\}$ and equally for Z_t. This *online SLAM* problem is generally referred to as *filtering* while the full SLAM problem is also known as *smoothing*. If $\mathbf{u}_t$ would be given as well, we could solve the *prediction* problem and estimate

$$P(\mathbf{x}_{t+1}, M|U_t, Z_t) \tag{2.12}$$

Due to recent developments in the SLAM community, the originally strict separation between filtering and smoothing, hence online and full SLAM, has been weakened. Incremental smoothing approaches (based on least squares optimization techniques which we are going to encounter in the following) can efficiently run *online* but at the same time estimate the *full* trajectory up to the current time t. We can write this as

$$P(X_t, M|U_{t-1}, Z_t) \tag{2.13}$$

with a small time index at X_t to indicate that the set of robot poses $X_t = \{\mathbf{x}_0, \ldots, \mathbf{x}_t\}$ is not fixed, but constantly augmented with additional robot poses $\mathbf{x}_{t+1}$ and so on. The same applies of course to the sets Z and U.

Table 2.1: *Different probabilistic estimation techniques.*

Estimated Quantity	Algorithm
$P(\mathbf{x}_t, M \mid U_{t-1}, Z_t)$	Filtering
$P(\mathbf{x}_{t+1}, M \mid U_t, Z_t)$	Prediction
$P(X_T, M \mid U_T, Z_T)$	Smoothing
$P(X_t, M \mid U_{t-1}, Z_t)$	Incremental Smoothing

The different probabilistic estimation techniques are summarized in Table 2.1. For many years, filter-based methods, such as extended Kalman filters or particle filters, have dominated the SLAM literature[18]. During the past few years however, a steady change of paradigms occurred in the SLAM community. This change led the focus of new developments away from filters, towards efficient smoothing approaches. In contrast to the conventional filters, these modern smoothers are optimization-based and build upon efficient algorithms for nonlinear least squares optimization that exploit the sparsity inherent in the SLAM problem. This way, even large-scale SLAM problems can be solved in a fraction of a second on standard hardware. A common feature of such optimization-based approaches is that they express the SLAM problem using a *graph* structure, which is a convenient way of encapsulating and modelling of the underlying optimization problem. The next section therefore reviews the most important graphical representations for SLAM, before section 2.3 explains how SLAM can be expressed as a least squares optimization problem.

2.2 Graph Representations for SLAM

The SLAM problem can be conveniently modelled and expressed using different kinds of graph representations. This section shortly reviews two of the commonly used representations for different kinds of SLAM problems. The literature [Dellaert and Kaess, 2006] gives further insight into the graph structures of SLAM, including new developments like the Bayes tree [Kaess et al., 2010].

2.2.1 Dynamic Bayesian Networks

Dynamic Bayesian Networks (DBNs) [Dean and Kanazawa, 1988] are directed acyclic graphs that represent time-dependent random variables and the conditional dependencies between them. Usually, two types of nodes are used to distinguish between observable and unobservable variables[19]. Fig. 2.17 for example illustrates the DBN-representation of a SLAM problem with distinct landmarks. The grey nodes represent the observations (i.e. the sensor measurements) while white nodes illustrate unobserved random variables, e.g.

[18] A gentle introduction into filtering approaches for SLAM, discussion and further references can be found in Appendix A. They are omitted here, since filters play no further role in this thesis.

[19] These unobservable variables are also called *hidden* or *latent* variables.

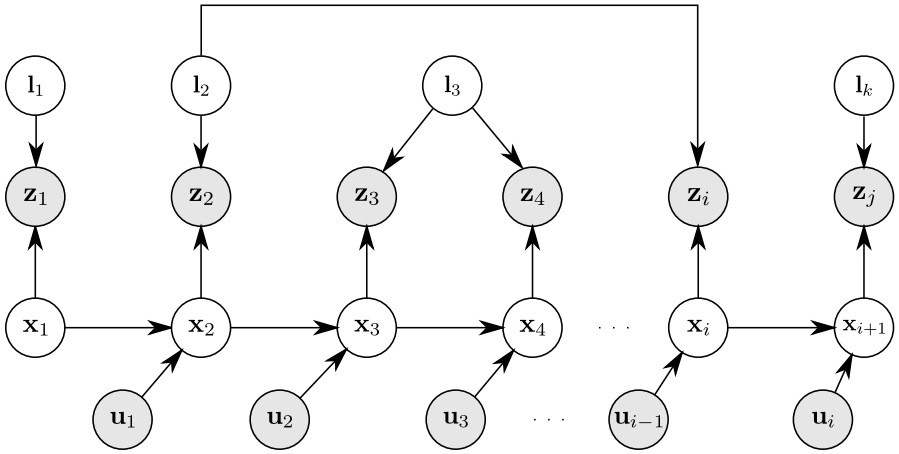


Figure 2.17: *The SLAM problem represented as a dynamic Bayesian network. White nodes are the unobservable hidden variables like the robot state $\mathbf{x}_t$ and the landmark positions $\mathbf{l}_i$. Dark nodes represent the observable random variables, e.g. the sensor measurements. In this example $\mathbf{u}_t$ are odometry measurements while $\mathbf{z}_i$ express landmark observations. SLAM tries to infer the states of the hidden variables, given the observations.*

the robot pose and the landmark positions. For each conditional dependency between two variables, there is a directed edge connecting the two corresponding nodes. [Russell and Norvig, 2010] provides introductory material for inference in DBNs, [Thrun et al., 2005] concentrates on DBNs for SLAM and state estimation problems.

2.2.2 Factor Graphs

Factor graphs are bipartite undirected graphs and have been proposed by [Kschischang et al., 2001] as a general tool to model factorizations of large functions with many variables into smaller local subsets. The idea can be applied to probabilistic problems like SLAM: The joint probability distribution that SLAM wants to estimate can be expressed as a product over several single *factors*, e.g.

$$P(X_T, M|U_T, Z_T) = \prod_i P(\mathbf{x}_i|\mathbf{u}_i, \mathbf{x}_{i-1}) \cdot \prod_{k,i,j} P(\mathbf{l}_k|\mathbf{x}_i, \mathbf{z}_j) \tag{2.14}$$

where $\mathbf{l}_k$ are positions of single landmarks that constitute the Map M.

Factor graphs contain two types of nodes: one for variables and the other for the probabilistic relations (the factors) between them. The edges in the graph therefore nicely capture the dependencies between the variables, since the edges always connect a variable node and a factor node (following the definition of a bipartite graph).

In the SLAM context, one type of nodes represents the unknown variables (robot states, landmark positions or map state in general) while the other type of node encodes the

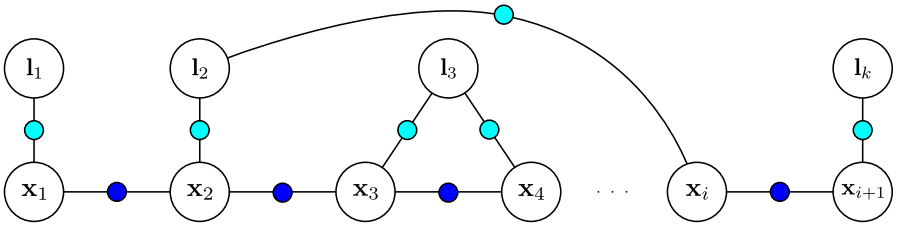


Figure 2.18: *Factor graph representation of the SLAM problem with landmarks. The large vertices represent the unknown robot poses, while probabilistic relationships between them are expressed by the small vertices. Odometry factors are represented by blue nodes, landmark observations are shown in cyan.*

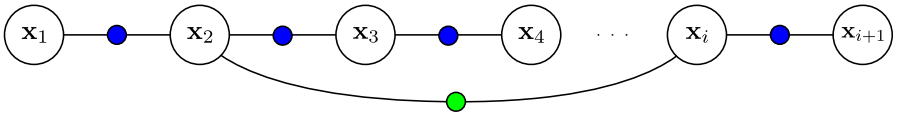


Figure 2.19: *Factor graph representation of the pose graph SLAM problem. The large vertices represent the unknown robot poses, while probabilistic constraints between them are expressed by the small vertices. Odometry constraints between successive robot states are shown in blue. The green vertex represents a loop closure factor between two non-successive poses.*

relations (conditional probabilities) between them. This is illustrated in Fig. 2.18 which is a direct translation of the DBN of Fig. 2.17. For illustrative purposes, the variable nodes are always depicted larger than the factor nodes between them.

While Fig. 2.18 represents a SLAM problem where the positions of landmarks are explicitly estimated, Fig. 2.19 illustrates the pose graph SLAM problem without distinct landmarks. Instead, loop closures are represented by a factor between non-successive robot state nodes.

2.2.3 Markov Random Fields

Appearing to be very similar to factor graphs, Markov random fields (MRF) are undirected graphs that contain only nodes for the variables. The structure of the edges in MRFs is almost the same as in factor graphs, except that the factor nodes have been eliminated and thus there are only edges between variable nodes. Markov random fields therefore capture the dependencies between single variables. In detail, two non-adjacent variables (i.e. those not connected by an edge) are conditionally independent, given all other variables. Furthermore, when the adjacent neighbors to a variable are given, that variable is conditionally independent of all other variables.

Fig. 2.20 illustrates the MRF that corresponds to the factor graph and DBN in Fig.

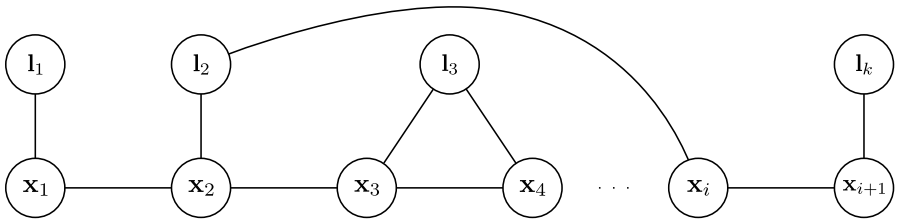


Figure 2.20: *Markov random field representation of the SLAM problem with landmarks. The nodes of MRFs represent the variables, i.e. the unknown robot poses and landmark positions. Two nodes are connected by an undirected edge if they are conditionally dependent.*

2.18 and Fig. 2.17 respectively. Notice that factor graphs can contain so called prior factors that only influence one variable and thus would become "open edges" in a MRF, which is of course not possible. Therefore, prior constraints cannot be modelled in MRFs.

We are going to re-encounter the adjacency matrices[20]of these MRF graph structures when talking about numerical methods for solving least squares problems. As we will see, the adjacency matrices are equivalent to the so called Hessian matrix of the optimization problem.

2.3 SLAM as a Nonlinear Least Squares Optimization Problem

After the previous sections laid out the foundations and explained the SLAM problem in general, I now want to concentrate on a specific kind of SLAM problem and show how it can be formulated (and solved) as a least squares optimization problem.

2.3.1 The Pose Graph SLAM Problem

The specific SLAM problem I am going to address here considers a robot operating in a 2D world. The robot has three degrees of freedom, i.e. its pose in the world can be expressed by the state vector $\mathbf{x} = (x, y, \theta)^\mathsf{T}$. The map we are interested in should only contain the robot's trajectory through the 2D world, while distinct landmarks are not part of the map. The map can therefore be expressed as a graph where the vertices represent robot poses $\mathbf{x}_i$ and edges represent the spatial constraints between these poses. Such a map is generally called a *pose graph*.

Two different kinds of constraints are necessary for pose graph SLAM. The first are odometric constraints that connect two successive states $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ via a motion model f, such that

$$\mathbf{x}_{i+1} \sim \mathcal{N}\left(f(\mathbf{x}_i, \mathbf{u}_i), \mathbf{\Sigma}_i\right) \tag{2.15}$$

[20] An introduction to graph theory can be found in standard computer science textbooks like [Sedgewick and Wayne, 2011] or [Cormen et al., 2009].

like we have seen before[21].

Depending on the type of robot the odometric sensor system may consist of simple wheel encoders, a laser scan matcher, visual odometry, etc. or a combination of those. The uncertainties of the sensor system are captured in the covariance $\mathbf{\Sigma}_i$.

Furthermore, in order to perform loop closing, the robot has to recognize places it already visited before. This place recognition is also a part of the front-end and provides the second type of constraint, the loop closure constraints. These Gaussian constraints connect two not necessarily successive poses $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$\mathbf{x}_j \sim \mathcal{N}\left(f(\mathbf{x}_i, \mathbf{u}_{ij}), \mathbf{\Lambda}_{ij}\right) \tag{2.16}$$

Notice that we can reuse the motion model function f since the constraint contains a pseudo-odometry measurement $\mathbf{u}_{ij}$. This allows the loop closure detection to express a displacement[22] between the two matched poses $\mathbf{x}_i$ and $\mathbf{x}_j$.

As the robot moves through its environment, it uses its odometry sensor to create an initial guess of its trajectory. Usually, due to the accumulated odometry errors, large discrepancies begin to show after even a small time. These map errors are most apparent when the robot closes loops, i.e. when it revisits areas it has traversed before.

The pose graph generated from the noisy odometry of a simulated robot can be seen in Fig. 2.21(b). The robot was steered on a square-shaped path (see Fig. 2.21(a) for the ground truth) and revisited the lower part of the trajectory. The figure clearly shows the effect of the noisy odometry sensors: The odometry-generated trajectory obviously diverges from the ground truth.

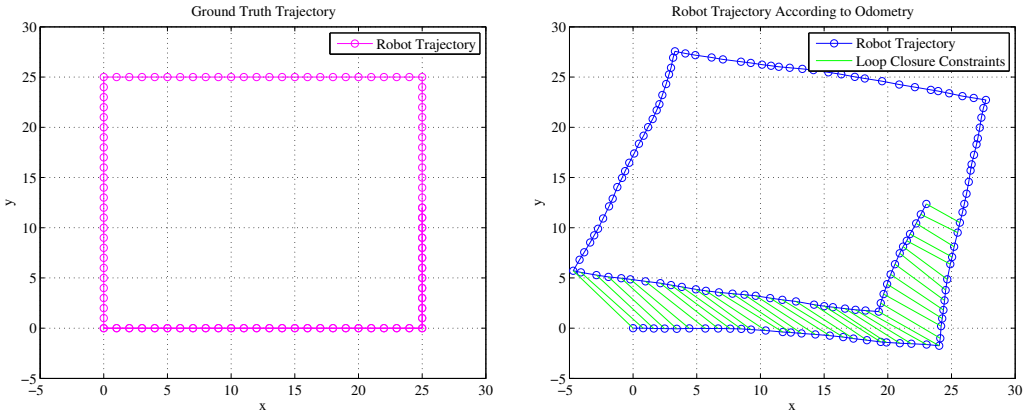
In addition, loop closure constraints are visible as green lines and connect two robot poses in the map. These constraints were generated by the place recognition algorithm in the front-end that considered the two connected robot poses to be one and the same in the real world.

2.3.2 Deriving a Nonlinear Least Squares Formulation

The key to the solution of the full SLAM problem for pose graph SLAM is the probability distribution $P(X|U)$. Given the set of odometry and loop closure constraints $\mathbf{u}_i, \mathbf{u}_{ij} \in U$, we seek the *optimal*, i.e. most likely configuration of robot poses, which we denote X^*. This most likely variable configuration is also called the *maximum a posteriori* or MAP

[21] Notice that i has been used for the indices instead of t as was the case before. Although this notation is a little bit inconsistent with regards to the previous sections, it stresses the fact that in optimization-based SLAM we often solve the *full* SLAM problem, *after* all the data has been gathered. Therefore, the usual notation does not stress the time-dependency as much as the notations generally used for online filtering approaches.

[22] Of course, we can always keep things simple if we set $\mathbf{u}_{ij} = \mathbf{0}$. Then any two matched poses would be required to align exactly. This can be used if the place recognition system is for instance based on the visual appearance of the scenes only and is not able to calculate an estimation of their displacement. If in contrast the front-end is able to do so, for instance by means of visual odometry between the two scenes, then $\mathbf{u}_{ij}$ can express that estimated displacement between the scenes, helping the overall solution to be more exact.



(a) (b)

Figure 2.21: *A small simulated dataset: (a) Ground truth trajectory. The robot started at coordinates* $(0, 0)$ *and was steered on a squared path, revisiting the first part of the trajectory. (b) Map generated from odometry measurements only. The robot positions are depicted in blue, omitting covariance information. Due to the noisy odometry measurements, the trajectory clearly diverges from the ground truth. Green links between two poses are loop closure constraints requested by the place recognition system in the front-end.*

estimate. It is equal to the *mode* of the joint probability distribution $P(X|U)$. In simpler words, X^* is the point where that distribution has its maximum:

$$X^* = \underset{X}{\operatorname{argmax}} P(X|U) \tag{2.17}$$

To solve this problem[23], we can factor the joint probability distribution as

$$P(X|U) \propto \underbrace{\prod_i P(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i)}_{\text{Odometry Constraints}} \cdot \underbrace{\prod_{ij} P(\mathbf{x}_j|\mathbf{x}_i, \mathbf{u}_{ij})}_{\text{Loop Closure Constraints}} \tag{2.18}$$

[23] Notice that in this problem formulation we ask for the *maximum* of a general posterior distribution $P(X|Z)$, which is a *point* estimate. This is in contrast to the classical Bayesian filters that ask for the full posterior *distribution* $P(X|Z)$. Furthermore, the difference between the maximum a posteriori (MAP) and the maximum likelihood (ML) solution is, that ML asks for $\operatorname{argmax}_X P(Z|X)$, which maximizes only the likelihood of the given data or measurements. MAP in contrast can incorporate prior information about X and estimates $\operatorname{argmax}_X P(X|Z)$, which is, according to the law of Bayes equal to $\operatorname{argmax}_X \eta P(Z|X)P(X)$. We therefore see that the solutions found by ML and MAP are equal if the prior $P(X)$ is a uniform distribution.

Under the assumption that (2.15) and (2.16) hold and thus the conditional probabilities above are all Gaussian, we can write for the odometry constraints:

$$\mathbf{x}_{i+1} \sim \mathcal{N}\left(f(\mathbf{x}_i, \mathbf{u}_i), \mathbf{\Sigma}_i\right) \tag{2.19}$$

$$P(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i) = \frac{1}{\sqrt{2\pi|\mathbf{\Sigma}_i|}} \exp\left(-\frac{1}{2}\ \left(f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\right)^\mathsf{T} \mathbf{\Sigma}_i^{-1} \left(f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\right)\right) \tag{2.20}$$

where we write $\exp(A)$ to express the exponential e^A with e the Euler number as usual. By collecting the first term $\frac{1}{\sqrt{2\pi|\mathbf{\Sigma}_i|}}$ into the normalizer η and applying the definition of the Mahalanobis distance[24] to the exponent, we can write more conveniently:

$$P(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i) = \eta \exp -\frac{1}{2}\|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i} \tag{2.21}$$

By following the same steps for the loop closure constraints we gain:

$$P(\mathbf{x}_j|\mathbf{x}_i, \mathbf{u}_{ij}) = \eta \exp -\frac{1}{2}\|f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j\|^2_{\mathbf{\Lambda}_{ij}} \tag{2.22}$$

This leads to the following factorization of the problem:

$$P(X|U) \propto \prod_i \exp -\frac{1}{2}\|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i} \cdot \prod_{ij} \exp -\frac{1}{2}\|f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j\|^2_{\mathbf{\Lambda}_{ij}} \tag{2.23}$$

We can now transform the products into more convenient sums by taking the negative logarithm:

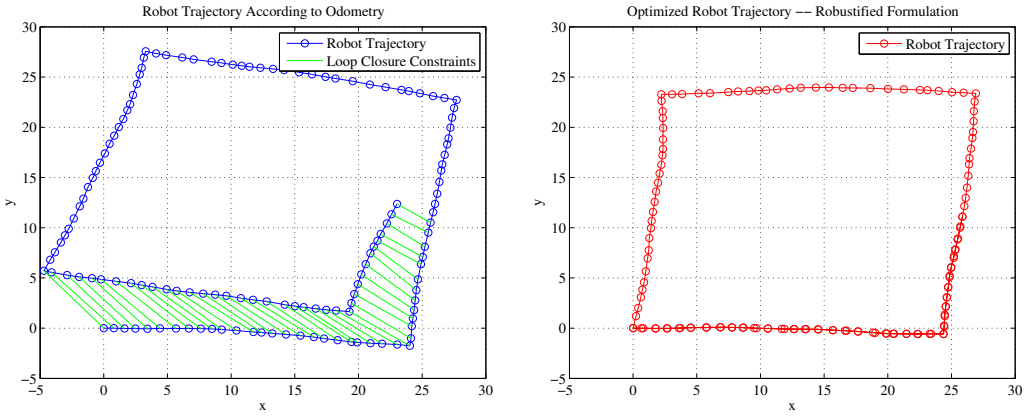
$$-\log P(X|U) \propto \sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i} + \sum_{ij} \|f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j\|^2_{\mathbf{\Lambda}_{ij}} \tag{2.24}$$

Notice that we dropped the normalizer η and the factors $\frac{1}{2}$ since for the following we are only interested in proportionality (hence the sign $\propto$) instead of exact equality. After these transformations, we can solve for the sought maximum a posteriori solution X^*:

$$\begin{aligned} X^* &= \operatorname*{argmax}_X P(X|U) \\ &= \operatorname*{argmin}_X -\log P(X|U) \\ &= \operatorname*{argmin}_X \underbrace{\sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i}}_{\text{Odometry Constraints}} + \underbrace{\sum_{ij} \|f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j\|^2_{\mathbf{\Lambda}_{ij}}}_{\text{Loop Closure Constraints}} \end{aligned} \tag{2.25}$$

This however is a least squares optimization problem, since the sought X^* is a minimizer over a sum of squared terms. One of the many available methods for nonlinear least squares optimization can be applied to solve this problem and find X^*.

[24] The squared Mahalanobis distance is defined as $\|a-b\|^2_{\mathbf{\Sigma}} = (a-b)^\mathsf{T}\mathbf{\Sigma}^{-1}(a-b)$



(a) (b)

Figure 2.22: *Pose graph map before (a) and after (b) the optimization. Notice how the loop at the bottom of the map has been correctly closed and the odometry errors have been distributed along the trajectory.*

We are going to review least squares optimization techniques in Chapter 3. Here, I want to denote that the reason why the least squares formulation of SLAM can be solved efficiently lies in the *sparse* structure of the constraints. This is nicely illustrated by the graph representations, which are sparsely connected. For example the degrees of the variable nodes in the MRF or the factor graph representation are low. Notice that especially in pose graph SLAM, the degree of most variable nodes will be 2. Only those nodes that are part of a loop closing will have a larger degree of $2 + k$ if they are involved in k loop closings (usually $k = 1$).

Coming back to the example from Fig. 2.21(b) and solving the optimization problem, the results can be seen in Fig. 2.22. The loop at the bottom of the plot has been correctly closed and the accumulated odometry errors have been distributed along the trajectory.

2.3.3 An Intuitive Analogy for the Least Squares Optimization

Very intuitively, we can interpret the optimization problem expressed by the graph representation as a spring-mass model (see Fig. 2.23). This analogy was first established by [Golfarelli et al., 1998]. In this understanding, the variable nodes would be small masses while the constraint edges would constitute the springs between the masses. The solution to the optimization problem would then correspond to the minimal energy configuration in that spring-mass system. This is the configuration that automatically sets in due to the forces of the springs working on the masses.

In this analogy, the power of the individual springs corresponds to the covariance matrix associated with the error function (the Mahalanobis distance). The smaller the covariance, the stronger the spring, and hence the stronger its influence on the overall

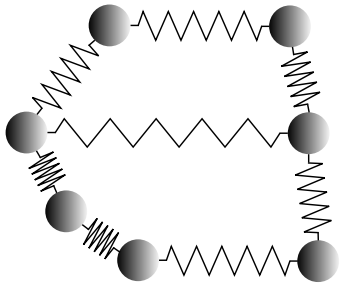


Figure 2.23: *Spring-mass model of the optimization for pose graph SLAM. The springs represent the constraint between the variables, which can be imagined as small masses. The solution X^* to the optimization problem (2.25) corresponds to the minimum energy configuration of the spring-mass system.*

Table 2.2: *Available frameworks and optimization-based SLAM back-ends.*

Name	Main Publication	Source
g^2o	[Kümmerle et al., 2011b]	`www.openslam.org`
MTK	[Wagner et al., 2011]	`www.openslam.org`
GTSAM	[Kaess et al., 2011]	`collab.cc.gatech.edu/borg/gtsam`
HOG-Man	[Grisetti et al., 2010]	`www.openslam.org`
TORO	[Grisetti et al., 2009]	`www.openslam.org`
iSAM 1	[Kaess et al., 2008]	`www.openslam.org`

network of masses and other springs. At the same time, the energy stored in the network corresponds to the residual error of the optimization: In the ideal case with no residual error, all springs in the network would be totally relaxed.

Notice that following the spring-mass analogy, a false positive loop closure constraint would correspond to a spring that erroneously connects two masses. This would result in a twisted and erroneous configuration of masses.

2.3.4 Optimization-based SLAM – A Literature Review

During the past few years a change of paradigms could be observed in the SLAM literature: First approaches to SLAM were based on filtering techniques, starting with the EKF and continuing with particle filters, UKF, and information filters. Lately, the focus of research has shifted to optimization-based approaches that have been found to be more efficient, versatile, scalable, and stable than solutions based on filtering algorithms. A number of optimization based SLAM back-ends are readily available to the SLAM researcher as open source libraries: Table 2.2 lists the frameworks and the associated main publications that are available at the time of writing.

The first paper that proposed an efficient solution to the full SLAM problem was [Lu and

Milios, 1997]. The authors explained a technique they called *consistent pose estimation* and applied it to indoor SLAM using laser range finders. The seminal paper represents the SLAM problem using a graph structure.

The idea has been developed further, e.g. by [Gutmann and Konolige, 1999] and [Konolige, 2004]. The latter publication pointed out the sparse structure inherent to the full SLAM problem and proposed a preconditioned gradient technique to solve it. Previous publications relied on simpler gradient descent techniques. GraphSLAM [Thrun and Montemerlo, 2005] proposed a scheme to reduce the number of variables involved in the SLAM problem by collapsing the constraints between robot poses and landmarks into pose–pose relations. A similar approach has been followed by [Folkesson and Christensen, 2004].

Relaxation techniques, such as Gauss-Seidel have been used for a while, e.g. by [Duckett et al., 2002; Frese et al., 2005; Milford et al., 2005]. [Olson et al., 2006] introduced stochastic gradient descent to further increase efficiency and solve pose graphs despite large initial errors. Later, [Grisetti et al., 2009] extended the approach towards non-flat environments with their system coined TORO.

Recent Nonlinear Optimization-Approaches

Nonlinear least squares optimization was used in an approach called $\sqrt{\text{SAM}}$ [Dellaert and Kaess, 2006] and its recent incremental enhancements iSAM [Kaess et al., 2008] and iSAM2 [Kaess et al., 2011]. Another strand of nonlinear approaches that explicitly exploits the sparse structure inherent in the SLAM problem was opened by SPA[25] [Konolige et al., 2010b]. The insight that due to the involved rotations, SLAM cannot be entirely correctly modeled using flat, Euclidean spaces, HOG-MAN [Grisetti et al., 2010] proposed a manifold approach that proved to outperform the simpler methods operating in Euclidean space. [Hertzberg et al., 2011] and [Wagner et al., 2011] developed that manifold approach further and extended it to general sensor fusion and calibration problems.

Combining the insights and learned lessons from HOG-MAN and SPA, the publicly available system g^{2}o [Kümmerle et al., 2011b] can be seen as the state of the art approach to solve large-scale SLAM problems containing several 10k variables (poses, landmarks) and constraints (observations, loop closings) in a matter of seconds on standard hardware.

[Kümmerle et al., 2011a] demonstrated the versatility of the g^{2}o framework by extending the state space and adding system parameters that might change over time. In their first experiments, the wheel diameters of a robot were estimated together with the trajectory and the map, leading to simultaneous calibration, localization, and mapping[26].

[25] Sparse Pose Adjustment, following the term Sparse Bundle Adjustment [Lourakis and Argyros, 2004] from the computer vision literature.

[26] Even more important would be a combination of SLAM and online-calibration for sensor parameters that are prune to drift like the biases of accelerometers or inertial sensors.

Incremental vs. Batch Processing

Most of the above graph- or optimization-based SLAM approaches were designed with the *full* SLAM system in mind. That is, they operate in *batch* mode, solving the whole problem all at once, *after* sensor data etc. has been collected.[27]

In contrast to these batch or smoothing methods, incremental approaches work more naturally *during* the operation of a robot and can be understood as combining the incremental nature of filtering with the advantages of smoothing. The whole strand of the approach coined *incremental smoothing and mapping* (iSAM) [Kaess et al., 2008; Kaess et al., 2010; Kaess et al., 2011; Kaess et al., 2012] explicitly concentrates on how the data structures required to express the SLAM problem for optimization-based approaches can be expanded (e.g. adding another observation or robot pose variable) efficiently, without rebuilding the data structures and recalculating everything from scratch.

While g^{2}o [Kümmerle et al., 2011b] also supports incremental processing, they did not explicitly concentrate their work on optimizing this mode of operation.

Linear Approximations and Closed-Form Solutions

The most recent development in optimization-based SLAM are linear approximations of the SLAM problem that lead to closed-form solutions [Carlone et al., 2011a; Carlone et al., 2011b]. Such techniques do not require an initial guess and can be solved in a single step instead of iteratively. The general idea is to separate the estimation of orientation and location. The reason for this approach is that an iterative solution is necessary mainly due to the nonlinearities introduced by the orientations. By estimating both quantities separately, the problem can be divided into two linear problems. However, this is only true when certain preconditions are met, e.g. spherical covariances. So far, the approach has only been shown to work in 2D scenarios. However, further advances may be expected in this area in the future.

Other recent work by [Huang et al., 2012; Huang et al., 2010] points in a similar direction and addresses questions regarding the convexity properties of SLAM, suggesting that SLAM has a special close-to-convex structure with only few local minima, especially in the case of spherical covariance matrices.

2.4 Summary

This chapter provided a brief introduction to SLAM and its various aspects. The introduction concluded with a review of the latest developments in SLAM research and outlined how SLAM can be formulated as a least squares problem. While I indicated that this problem formulation can be solved using algorithms and approaches from the

[27] This understanding bears some resemblance to *bundle adjustment* from the computer vision literature [Triggs et al., 2000]. In fact, at the mathematical core, the problems that have to be solved in both communities are very much the same. It is only natural that the used algorithms share the same principles, e.g. exploiting the sparse structure inherent in the nonlinear least squares optimization problems representing the problems at hand [Lourakis and Argyros, 2004].

field of optimization research, no further details were given. Therefore, the next chapter will provide a survey on general least squares optimization methods and least squares for SLAM in particular.

3
Least Squares Optimization

The last chapter concluded with a nonlinear least squares formulation of the (full) SLAM problem and provided an intuitive analogy that helps to understand the general idea behind this formulation. In this chapter I want to discuss some important properties of optimization problems and review methods that can be applied to solve least squares problems. The chapter provides a numerical example for a small, one-dimensional SLAM problem and ends with a discussion of optimization in the presence of outliers. Readers familiar with least squares optimization can safely skip this chapter and continue reading with Chapter 4 which discusses the main motivation of this work.

Readers interesting in more details and more elaborate discussions of the various aspects of optimization can rely on a set of excellent literature. The most valuable sources of information for this thesis were [Madsen et al., 2004] who provides a comprehensive and understandable introduction to nonlinear least squares methods, the textbooks by [Nocedal and Wright, 2006] and [Boyd and Vandenberghe, 2004] and (as so often) [Hartley and Zisserman, 2004].

3.1 Introduction

An *optimization problem* is generally defined as finding a minimum of an *objective function* or *cost function* $F(\mathbf{x})$. One is not so much interested in the function value of $F(\mathbf{x})$ at the minimum, but rather seeks the value of the *variable* $\mathbf{x}^*$ *where* that minimum occurs.

Formally, we can define an optimization problem as

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} F(\mathbf{x}) \tag{3.1}$$

From basic mathematical analysis, we know that an extrema point (i.e. a minimum or maximum) of a function F occurs, when its first derivative F' equals zero. And indeed,

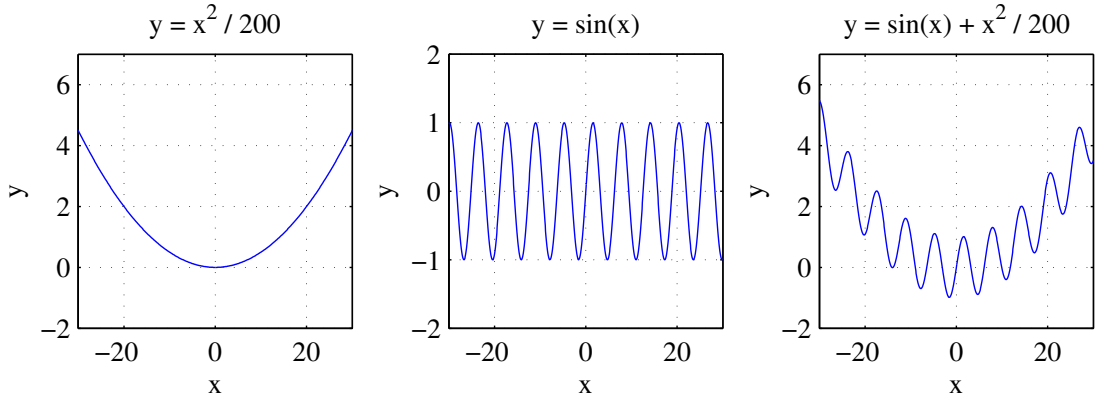


Figure 3.1: *Functions can have one global minimum (left), multiple local minima (middle), and a global minimum with multiple local minima (right).*

as we are going to see, derivatives of the cost function play an important role when it comes to solving the above problem. Notice that although I used the vector notation $\mathbf{x}$ in the above definition (3.1), F may also be defined on scalar variables. In general, the objective function is a function $F : \mathbb{R}^n \to \mathbb{R}$, where $n \geq 1$.

3.1.1 A Taxonomy of Optimization Problems

Optimization problems can be distinguished by a number of properties. The following taxonomy lists the most important types.

Global vs. local In the most desirable case, the objective function $F(\mathbf{x})$ may have only one unique minimum, which would be called the *global* minimum. More formally, $\mathbf{x}^*$ is a global minimizer of F if:

$$F(\mathbf{x}^*) < F(\mathbf{y}) \; \forall \mathbf{y} \tag{3.2}$$

Functions may also have multiple *local* minimizers. $\mathbf{x}^*$ is a local minimizer of F, if it minimizes the objective function within a certain region ϵ:

$$F(\mathbf{x}^*) \leq F(\mathbf{y}) \; \forall \mathbf{y} : ||\mathbf{y} - \mathbf{x}^*|| < \epsilon \tag{3.3}$$

Fig. 3.1 illustrates three functions with either a single, unique global minimum, multiple local minima without a unique global minimum and the combination of both cases.

Convex vs. non-convex The convexity of a cost function is an important criteria on how easy the associated optimization problem can be solved. In simple words, a convex function has only one minimum, which is a very preferable property. In Fig. 3.1, only the leftmost function is convex.

Constrained vs. unconstrained The optimization problem defined in (3.1) is called an *unconstrained* problem, as the sought solution $\mathbf{x}^*$ is not subject to any further constraints or preconditions. In contrast, we can require $\mathbf{x}^*$ to adhere to equality and / or inequality *constraints* of the form $c(\mathbf{x}^*) = 0$ or $c(\mathbf{x}^*) \geq 0$ respectively. In the following we shall only consider unconstrained optimization problems.

Continuous vs. discrete If the objective function F and the variables $\mathbf{x}$ are not defined over a continuous body like $\mathbb{R}^n$, but over a discrete set, like $\mathbb{Z}^n$, the optimization problem is called *discrete*. Many important problems from computer science and graph theory are discrete optimization problems, e.g. shortest path problems, maximum flow problems etc. The problems considered in this thesis however, are problems of continuous optimization.

Linear vs. nonlinear Depending on whether the objective function $F(\mathbf{x})$ depends linear or nonlinear on $\mathbf{x}$, we call the associated optimization problem *linear* or *nonlinear*. Most of the problems considered in the following are going to be nonlinear.

3.1.2 Least Squares Optimization Problems

The term *least squares optimization* describes a very important sub-class of general optimization problems. In least squares optimization, the objective function is a sum over squared terms, i.e

$$F(\mathbf{x}) = \frac{1}{2} \sum_{i}^{n} f_i(\mathbf{x})^2 \tag{3.4}$$

and the functions f_i are defined as scalar-valued functions $f_i : \mathbb{R}^n \to \mathbb{R}$. This formulation of the objective function is very common and the problems we will encounter later on will as well be formulated as (nonlinear) least squares problems.

Notice that we can reformulate (3.4) and, using vector formulation, write more conveniently

$$F(\mathbf{x}) = \frac{1}{2} \mathbf{f}(\mathbf{x})^\mathsf{T} \mathbf{f}(\mathbf{x}) \tag{3.5}$$

if we let $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x}))^\mathsf{T}$. Then the least squares optimization problem is

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} F(\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2} \mathbf{f}(\mathbf{x})^\mathsf{T} \mathbf{f}(\mathbf{x}) \tag{3.6}$$

3.2 Linear Least Squares Problems

In case of a linear least squares problem, all f_i depend linearly on $\mathbf{x}$, i.e. the function value is a linear combination of the variables in $\mathbf{x}$. Then we can write:

$$f_i(\mathbf{x}) = a_{i,1}x_1 + a_{i,2}x_2 + \ldots + a_{i,n}x_n - b_i \tag{3.7}$$

Notice that we wrote $-b_i$ instead of $+b_i$ without loss of generality because the following derivations can be done more conveniently. Given (3.7) we can obviously stack the single

f_i onto each other to gain the column vector $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x}))^\mathsf{T}$. Then by collecting all coefficients $a_{i,j}$ into the matrix $\mathbf{A}$ and all b_i into the vector $\mathbf{b}$ we can write very simply

$$\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} \tag{3.8}$$

3.2.1 Solving Linear Least Squares Problems

In order to determine the location of the extrema points for $F(\mathbf{x})$, we need its derivative $F'(\mathbf{x})$:

$$\begin{aligned} F'(\mathbf{x}) &= \frac{\partial F}{\partial \mathbf{x}} \\ &= \frac{\partial(\frac{1}{2}\mathbf{f}^\mathsf{T}\mathbf{f})}{\partial \mathbf{x}} \\ &= \mathbf{f}^\mathsf{T}\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \end{aligned} \tag{3.9}$$

The last line was derived applying the chain rule. Now, because

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \frac{\partial(\mathbf{A}\mathbf{x} - \mathbf{b})}{\partial \mathbf{x}} = \mathbf{A} \tag{3.10}$$

we finally obtain:

$$F'(\mathbf{x}) = (\mathbf{A}\mathbf{x} - \mathbf{b})^\mathsf{T} \cdot \mathbf{A} \tag{3.11}$$

The position of the extrema, $\mathbf{x}^*$ is found by setting the first derivative to zero:

$$\begin{aligned} 0 &= F'(\mathbf{x}^*) \\ &= (\mathbf{A}\mathbf{x}^* - \mathbf{b})^\mathsf{T} \cdot \mathbf{A} \\ &= \mathbf{A}^\mathsf{T}(\mathbf{A}\mathbf{x}^* - \mathbf{b}) \\ &= \mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x}^* - \mathbf{A}^\mathsf{T}\mathbf{b} \\ \mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x}^* &= \mathbf{A}^\mathsf{T}\mathbf{b} \end{aligned} \tag{3.12}$$

The equation in (3.12) is called a *normal equation*. Such normal equations can be solved using different methods:

1. If $\mathbf{A}^\mathsf{T}\mathbf{A}$ is invertible, the solution is found directly by $\mathbf{x}^* = (\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}\mathbf{A}^\mathsf{T}\mathbf{b}$.

2. A better alternative uses the *QR decomposition* of $\mathbf{A}$ into an upper triangular matrix $\mathbf{R}$ and an orthogonal matrix $\mathbf{Q}$, so that $\mathbf{A} = \mathbf{Q}\mathbf{R}$. The solution $\mathbf{x}^*$ is then given by $\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{Q}^\mathsf{T}b$. This method prevents the calculation of $\mathbf{A}^\mathsf{T}\mathbf{A}$ and its inversion, which can become numerically unstable.

3. Instead of the QR decomposition, the *Cholesky decomposition* of $\mathbf{A}^\mathsf{T}\mathbf{A}$ can be used. This method decomposes $\mathbf{A}^\mathsf{T}\mathbf{A}$ into $\mathbf{L}\mathbf{L}^\mathsf{T}$, then solves $\mathbf{L}\mathbf{y} = \mathbf{A}^\mathsf{T}\mathbf{b}$ for $\mathbf{y}$ and finds $\mathbf{x}^*$ by solving $\mathbf{L}^\mathsf{T}\mathbf{x} = y$. Notice that $\mathbf{L}$ is a lower triangular matrix with positive diagonal elements. Compared to the QR decomposition, Cholesky is faster, but numerically more unstable because $\mathbf{A}^\mathsf{T}\mathbf{A}$ has to be calculated.

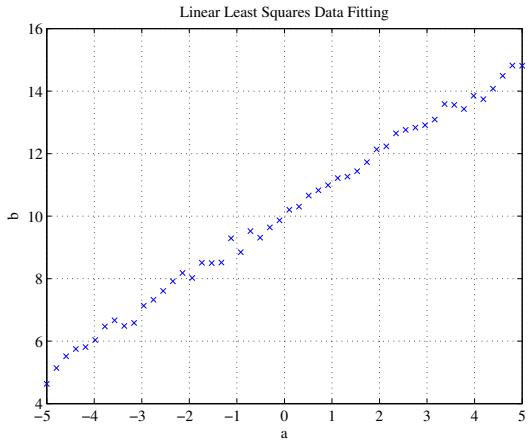


Figure 3.2: *A simple example to curve fitting with noisy input points.*

4. The solution to the linear least squares problem with $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$ can as well be determined using the *singular value decomposition*, SVD. Here, $\mathbf{A}$ would be decomposed into $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T}$. After calculating a new vector $\mathbf{b}' = \mathbf{U}^\mathsf{T}\mathbf{b}$ and defining a vector $\mathbf{y}$ with $y_i = b'_i/d_i$ where d_i is the i-th diagonal entry of $\mathbf{D}$, the solution $\mathbf{x}^*$ is found by $\mathbf{x}^* = \mathbf{V}\mathbf{y}$. Further details can be found in [Hartley and Zisserman, 2004, ch. A5].

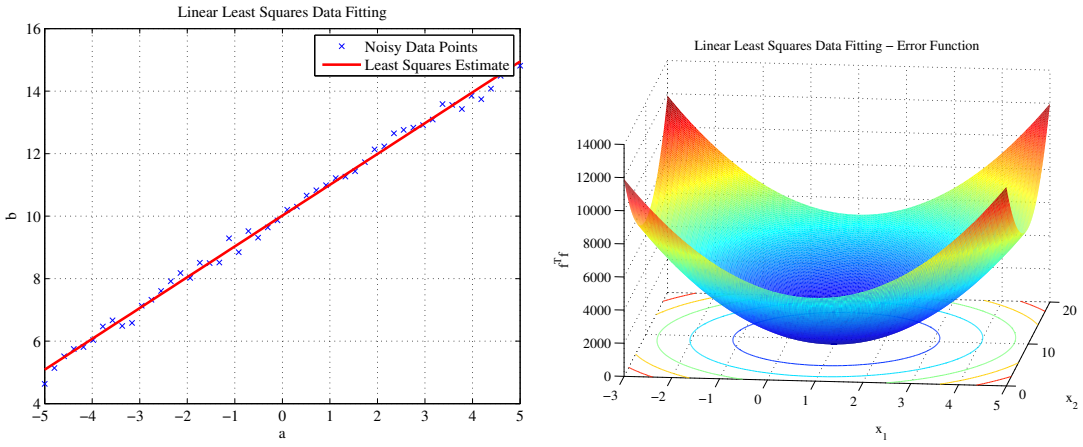
3.2.2 Examples for Linear Least Squares Problems

After the previous pages laid out the theory on linear least squares problems, we shall now consider a typical application of least squares optimization: *regression*, *data fitting* or *curve fitting*.

Example 3.1. *Consider we are given n noisy data points (a_i, b_i), as shown in Fig. 3.2. Suppose further we know that the points should follow a linear model of the form $g_\mathbf{x}(a) = b = x_1 a + x_2$. The parameters x_1 and x_2 however, are unknown to us. How can we find values for these two parameters, so that the resulting linear function fits the data best?*

We can use optimization to determine the sought parameter vector $\mathbf{x}^*$ if we set up an objective function appropriate to the problem formulation. Intuitively, for the optimal line model $g_{\mathbf{x}^*}$, all of the data points should be very close to the line. A natural criteria on how well a function $g_\mathbf{x}$ (with particular values for the parameters $\mathbf{x}$) fits the given data set, is the distance $||g_\mathbf{x}(a_i) - b_i||$ of the data point b_i from its predicted value $g_\mathbf{x}(a_i)$. Notice that instead of the distance $||\cdot||$, we can as well use the squared *distance $||\cdot||^2$ and therefore formulate a squared objective function for every data point as*

$$f_i^2(\mathbf{x}) = ||g_\mathbf{x}(a_i) - b_i||^2 = ||x_1 a_i + x_2 - b_i||^2 \tag{3.13}$$



(a) (b)

Figure 3.3: *(a) Result of the least squares solution to the simple curve fitting problem with a linear function model. The red line is the graph of the found optimal solution that minimizes the squared distances of the noisy data points from the line. (b) The error function associated with this optimization problem is convex with a single global minimum at the found solution.*

By summing over all f_i^2, we obtain the overall objective function:

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} f_i^2 = \frac{1}{2}\sum_{i=1}^{n}(x_1 a_i + x_2 - b_i)^2 \tag{3.14}$$

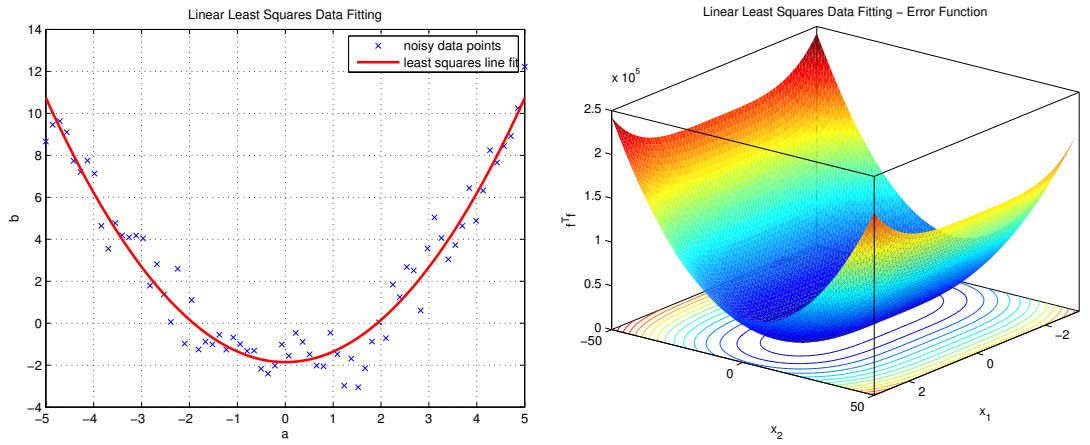
This however, corresponds exactly to the linear least squares problem defined above in (3.8) where

$$\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} \tag{3.15}$$

In our particular example we have

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= \mathbf{A}\mathbf{x} - \mathbf{b} \\ &= \begin{pmatrix} a_1 & 1 \\ a_2 & 1 \\ \vdots & \vdots \\ a_n & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \end{aligned} \tag{3.16}$$

With this insight, we can immediately apply one of the possible strategies and solve for the optimal least squares solution $\mathbf{x}^*$, e.g. by solving the normal equation $\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x}^* = \mathbf{A}^\mathsf{T}\mathbf{b}$. In this example, using the values shown in Fig. 3.2, the result is $\mathbf{x}^* = (0.9997, 10.0351)^\mathsf{T}$. This least squares estimate *of the parameters $(x_1, x_2)^\mathsf{T}$ is very close to the real values that I used to generate the data, which were $(1, 10)^\mathsf{T}$. The data points are overlaid with*



(a) (b)

Figure 3.4: *(a) Noisy data points following a polynomial of 2^{nd} degree. The graph of the optimal fit is shown in red. (b) The error function associated with this optimization problem is convex with a single global minimum at the found solution $\mathbf{x}^*$.*

the graph of the function $g_{\mathbf{x}^*}$ in Fig. 3.3(a). The error function $\mathbf{f}^\mathsf{T}\mathbf{f}$ is illustrated in Fig. 3.3(b). Notice that it is a convex function with a single unique minimum.

Example 3.2. *In this second example, we consider fitting noisy data points to a polynomial of 2nd degree, specifically $g_{\mathbf{x}}(a) = b = x_1 a^2 + x_2$. It is important to understand that even if the polynomial is of course nonlinear in its variable a, it is still linear in its parameters x_1 and x_2. Therefore, the resulting fitting problem for all polynomials is a linear least squares problem and can be solved using the methods presented before.*

Fig. 3.4(a) and 3.4(b) illustrate the noisy data points along with the least squares fit and the associated, 2-dimensional convex error function.

3.3 Nonlinear Least Squares Problems

In contrast to linear least squares problems, the objective functions $f_i(\mathbf{x})$ of the *nonlinear* least squares problem

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, F(\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmin}}\, \frac{1}{2}\mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{f}(\mathbf{x}) \tag{3.17}$$

depend on $\mathbf{x}$ in a nonlinear way. That means that the functions f_i cannot be expressed as a linear combination of the elements of $\mathbf{x}$, as we saw in (3.7).

Therefore, nonlinear least squares problems cannot be solved directly via a normal equation (3.12) but rather require the application of iterative approaches: Starting from

an initial guess of the solution, $\mathbf{x_0}$, these methods converge towards $\mathbf{x}^*$, which is hopefully the globally optimal solution. This however, can only be guaranteed if the objective function is convex or quasi-convex. In the general case, the found solution $\mathbf{x}^*$ may be only a local minimizer of the objective function, and, starting from different initial guesses $\mathbf{x_0}$, different local minima $\mathbf{x}^*$ may be found.

The literature knows a large number of iterative approaches for solving nonlinear least squares problems. Before we proceed to explore the most important strategies, we want to establish the following result: The first derivative (or *Jacobian*) of the objective function F is

$$F(\mathbf{x})' = \mathbf{J}_F = \mathbf{J}_\mathbf{f}^\mathsf{T}\mathbf{f} \tag{3.18}$$

This is because

$$\begin{aligned} F(\mathbf{x}) &= \frac{1}{2}\mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{f}(\mathbf{x}) && (3.19)\\ &= \frac{1}{2}\sum_i f_i(\mathbf{x})^2 && (3.20)\\ & && (3.21) \end{aligned}$$

and further because the partial derivatives of F are given by

$$\begin{aligned} \frac{\partial F}{\partial x_j} &= \frac{1}{2}\sum_i \frac{\partial f_i^2}{\partial x_j} && (3.22)\\ &= \frac{1}{2}\sum_i \frac{\partial f_i}{\partial x_j}\cdot f_i + f_i \cdot \frac{\partial f_i}{\partial x_j} && (3.23)\\ &= \sum_i \frac{\partial f_i}{\partial x_j}\cdot f_i && (3.24)\\ & && (3.25) \end{aligned}$$

The second derivative of F, called the *Hessian* of F, is

$$\begin{aligned} F(\mathbf{x})'' = \mathbf{H}_F &= \mathbf{J}'_F \\ &= (\mathbf{J}_\mathbf{f}^\mathsf{T}\mathbf{f})' \\ &= \mathbf{J}'^\mathsf{T}_\mathbf{f}\mathbf{f} + \mathbf{J}_\mathbf{f}^\mathsf{T}\mathbf{f}' \\ &= \mathbf{H}_\mathbf{f}^\mathsf{T}\mathbf{f} + \mathbf{J}_\mathbf{f}^\mathsf{T}\mathbf{J}_\mathbf{f} && (3.26) \end{aligned}$$

3.3.1 Gradient Descent

Gradient descent is maybe the most straight-forward iterative strategy for finding a local minimum of an arbitrary differentiable cost function $F(\mathbf{x})$. From the definition of the gradient

$$\nabla F(\mathbf{x}) = F'(\mathbf{x}) = \frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{J}_F \tag{3.27}$$

we instantly know that the function F decreases most rapidly in the direction of the negative gradient. Thus, we call

$$\Delta\mathbf{x} = -\mathbf{J}_F \tag{3.28}$$

a *descent direction* on F at $\mathbf{x}$ (where $\mathbf{J}_F$ is, as defined, the gradient or first derivative of F, evaluated at $\mathbf{x}$). The idea behind the gradient descent strategy is to repeatedly move into the direction of the negative gradient (thus it is also called *steepest descent*) until convergence.

However, directly using the gradient $-\mathbf{J}$ as the step in the iteration setting $\mathbf{x_{n+1}} = \mathbf{x_n} - \mathbf{J}$ would not guarantee $F(\mathbf{x_{n+1}}) < F(\mathbf{x_n})$. Therefore, the step *length* α has to be determined so that $F(\mathbf{x_n} + \alpha\Delta\mathbf{x}) < F(\mathbf{x_n})$. This can be done by searching along the line determined by the descent direction until a suitable value of α is found. Details on how to perform this line search strategy can be found in [Nocedal and Wright, 2006, ch. 3].

Finally, the step of gradient descent is

$$\Delta\mathbf{x} = -\alpha\mathbf{J}_F \tag{3.29}$$

or, in the case of least squares problems, due to (3.18),

$$\Delta\mathbf{x} = -\alpha\mathbf{J}_\mathbf{f}^\mathsf{T}\mathbf{f}(\mathbf{x}) \tag{3.30}$$

This allows us to summarize the gradient descent algorithm:

1. find the descent direction $-\mathbf{J}_F$ at the current $\mathbf{x_n}$
2. determine a suitable step length α through line search
3. repeat from $\mathbf{x_{n+1}} = \mathbf{x_n} - \alpha\mathbf{J}_F$ until convergence

Although the gradient descent strategy is easy to implement and guaranteed to converge to a minimum of the objective function, using more sophisticated methods can significantly speed up convergence. Such methods are reviewed next.

3.3.2 Newton's Method

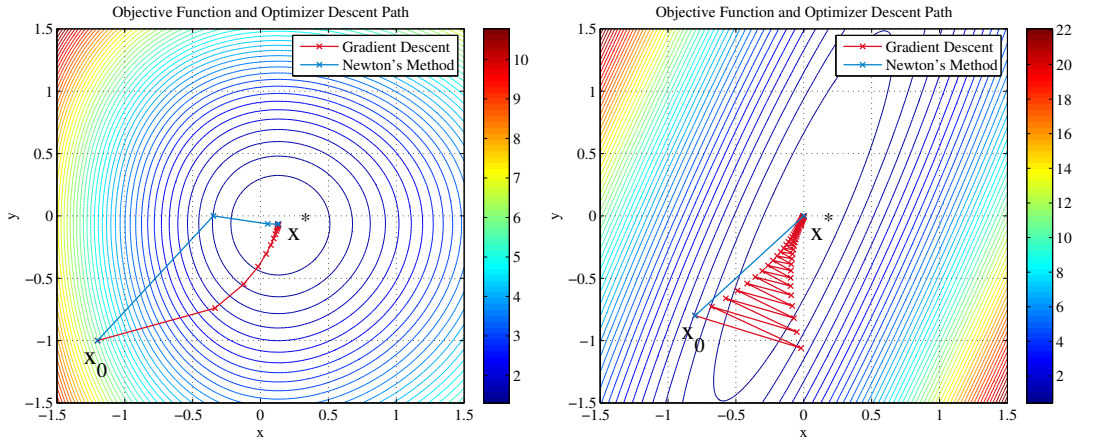
Newton's Method is a well-known and simple approach to finding the *root* of a function. For the one-dimensional case, the root of a differentiable function $g(x)$ is found by starting with an initial guess x_0 and then repeatedly calculate

$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)} \tag{3.31}$$

The geometrical meaning of the above equation is that x_{n+1} is the intersection of the tangent $g'(x_n)$ with the abscissa, because

$$g'(x_n) = \frac{\Delta y}{\Delta x} = \frac{g(x_n) - 0}{x_n - x_{n+1}} \tag{3.32}$$

Reformulating directly leads to (3.31).



(a) (b)

Figure 3.5: *Comparison of gradient descent and Newton's Method on two different objective functions. (a) The gradient descent strategy (red) choses steps along the negative gradient. Newton's Method (blue) approaches the minimum faster by exploiting the curvature information encoded in the second derivative. (b) illustrates the typical zigzag-behaviour of the gradient descent strategy that leads to suboptimal convergence. Notice that Newton's Method is able to solve the problem instantly as it finds the optimum in one step. This is possible, because the objective function is quadratic in its arguments.*

Of course, in optimization we are not so much interested in finding the *root* of the objective function F, but its *minimum*. However, from basic analysis we know that the extrema of F occur at the roots of the derivative F'.

Therefore, Newton's Method finds an extrema of a twice-differentiable function F by searching a root of its derivative F' through

$$x_{n+1} = x_n + \Delta x = x_n - \frac{F'(x_n)}{F''(x_n)} \tag{3.33}$$

The subtrahend in the equation above, $\Delta x = -\frac{F'(x_n)}{F''(x_n)}$ is usually called the *Newton step*.

For the multi-dimensional case, where we are interested in optimizing a vector-valued objective function $F(\mathbf{x})$, we likewise find the step as the solution to the equation

$$\mathbf{H}_F \Delta \mathbf{x} = -\mathbf{J}_F \tag{3.34}$$

where $\mathbf{J}_F$ and $\mathbf{H}_F$ are the Jacobian (i.e. the gradient or first derivative) and the Hessian (i.e. the second derivative) of the objective function F both evaluated at the current solution $\mathbf{x_n}$.

It is interesting to note, that Newton's Method can be applied to arbitrary nonlinear optimization problems, as long as the objective function F is differentiable twice. In

particular, F is not required to be the sum of squared terms as in (3.4). However, if Newton's Method is applied to an objective function of the form $F(\mathbf{x}) = \frac{1}{2}\mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{f}(\mathbf{x})$ as is the case in least squares problems, because of (3.18) and (3.26) we have

$$\begin{aligned} \mathbf{H}_F\Delta\mathbf{x} &= -\mathbf{J}_F \\ (\mathbf{H}_\mathbf{f}^\mathsf{T}\mathbf{f}(\mathbf{x}) + \mathbf{J}_\mathbf{f}^\mathsf{T}\mathbf{J}_\mathbf{f})\Delta\mathbf{x} &= -\mathbf{J}_\mathbf{f}^\mathsf{T}\mathbf{f}(\mathbf{x}) \end{aligned} \tag{3.35}$$

with $\mathbf{J}_\mathbf{f}$ and $\mathbf{H}_\mathbf{f}$ are the Jacobian and Hessian of $\mathbf{f}$. Notice that both (3.34) and (3.35) are linear problems of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$, and can be solved using one of the standard approaches, e.g. Cholesky decomposition.

Fig. 3.5 illustrates the superior behaviour of Newton's method compared to the simple gradient descent strategy described before. While the gradient descent approach shows a typical zigzag behaviour during the descent along the gradient, Newton's method converges quicker on a more direct descent path, since it exploits the curvature information encoded in the Hessian $\mathbf{H}_\mathbf{f}$. Although Newton's Method converges quickly towards a solution, a drawback is that it requires the second derivative $\mathbf{H}$, which is often hard to specify or calculate.

3.3.3 Gauss-Newton

In contrast to Newton's Method presented above, Gauss-Newton can only be applied to quadratic objective functions, i.e. least squares problems of the form

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, F(\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{f}(\mathbf{x}) \tag{3.36}$$

The algorithm is derived by applying a first order Taylor linearization of $\mathbf{f}(\mathbf{x})$ around $\mathbf{x}$:

$$\mathbf{f}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x} \tag{3.37}$$

As $F(\mathbf{x}) = \frac{1}{2}\mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{f}(\mathbf{x})$, we have

$$\begin{aligned} F(\mathbf{x} + \Delta\mathbf{x}) \approx L_\mathbf{x}(\Delta\mathbf{x}) &= \frac{1}{2}(\mathbf{f}(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x})^\mathsf{T} \cdot (\mathbf{f}(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x}) \\ &= \frac{1}{2}\mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{J}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{J}\Delta\mathbf{x} \\ &= F(\mathbf{x}) + \mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{J}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{J}\Delta\mathbf{x} \end{aligned} \tag{3.38}$$

Notice that $\mathbf{J}$ is the Jacobian of $\mathbf{f}$, evaluated at $\mathbf{x}$.

Given this approximation to the real objective function in the vicinity of $\mathbf{x}$, we seek the $\Delta\mathbf{x}$ that minimizes it. $L_\mathbf{x}(\Delta\mathbf{x})$ obviously is quadratic in $\Delta\mathbf{x}$ and therefore has only one global minimizer which can be found directly by setting the first derivative to zero:

$$\begin{aligned} 0 = L'(\Delta\mathbf{x}) &= \mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{J} + \mathbf{J}^\mathsf{T}\mathbf{J}\Delta\mathbf{x} \\ \mathbf{J}^\mathsf{T}\mathbf{J}\Delta\mathbf{x} &= -\mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{J} \\ \mathbf{J}^\mathsf{T}\mathbf{J}\Delta\mathbf{x} &= -\mathbf{J}^\mathsf{T}\mathbf{f}(\mathbf{x}) \end{aligned} \tag{3.39}$$

This is a *normal equation* as in (3.12) and can be solved using any of the methods presented in section 3.2.1.

It is interesting to notice that the right hand side of the equation above, $\mathbf{J}^\mathsf{T}\mathbf{f}(\mathbf{x})$, is equal to the derivative of the objective function F, because of (3.18). Furthermore, by comparing the left hand side of above result with the step of Newton's Method (3.35), we understand that the essential difference between Gauss-Newton and Newton's Method is that Gauss-Newton approximates the Hessian of the objective function by $\mathbf{J}^\mathsf{T}\mathbf{J}$. This way, Gauss-Newton avoids having to evaluate second derivatives which can become tedious for many non-trivial problems.

3.3.4 Levenberg-Marquardt

So far, we encountered three iterative solvers for nonlinear least squares optimization problems: Gradient descent, Newton's Method and Gauss-Newton.

Of those three, only the step $\Delta\mathbf{x} = \alpha\mathbf{J}$ of gradient descent is guaranteed to decrease the value of the objective function $F(\mathbf{x})$. Although Newton's Method converges very fast (quadratically) when already close to the solution, it can fail to give a valid descent step when the optimizer is still far from the sought minimum. The same applies for Gauss-Newton, whose only difference is the approximation of the Hessian through the square of the Jacobian.

The method generally known as Levenberg-Marquardt addresses this problem by gradually switching between gradient descent and the Gauss-Newton strategy. This is achieved by introducing a parameter λ, called the *damping factor* and solving the following problem to find the descent step:

$$(\mathbf{J}^\mathsf{T}\mathbf{J} + \lambda\mathbf{I})\Delta\mathbf{x} = -\mathbf{J}^\mathsf{T}\mathbf{f}(\mathbf{x}) \tag{3.40}$$

If the solution $\Delta\mathbf{x}$ to the above problem leads to a decrease of the objective function, the solution is accepted, λ is decreased (e.g. divided by 10) and the algorithm continues from $\mathbf{x_{n+1}} = \mathbf{x_n} + \Delta\mathbf{x}$. If the solution did not lead to a decrease of the objective function, the damping parameter λ is increased (e.g. by multiplying it by 10) and (3.40) is solved again until a λ that leads to a decrease is found.

Large values of λ will make the algorithm behave like the gradient descent strategy, as in this case the left term in (3.40) is dominated by $\lambda\mathbf{I}$ and thus

$$\Delta\mathbf{x} \approx -\frac{1}{\lambda}\mathbf{J}^\mathsf{T}\mathbf{f}(\mathbf{x}) = -\frac{1}{\lambda}\mathbf{J}_F \tag{3.41}$$

which results in a small step along the negative gradient of F, i.e. we are applying the steepest descent strategy. On the other hand, if λ is small, we arrive at

$$\mathbf{J}^\mathsf{T}\mathbf{J}\Delta\mathbf{x} = -\mathbf{J}^\mathsf{T}\mathbf{f}(\mathbf{x}) \tag{3.42}$$

which equals the Gauss-Newton step as in (3.39).

This way, by adapting the damping parameter λ, Levenberg-Marquardt can gradually switch between gradient descent and Gauss-Newton behaviour, which makes it a very favourable optimization approach. For the initial value of λ, [Hartley and Zisserman, 2004] proposes to use 10^{-3} times the average of $\mathrm{diag}(\mathbf{J}^\mathsf{T}\mathbf{J})$.

3.3.5 Summary

The previous sections reviewed four typical strategies for least squares optimization. All of them are incremental, but each of them applies a slightly different update step to find the next best variable configuration on the descent down the objective function. Table 3.1 summarizes the steps applied by the different approaches.

Table 3.1: *Steps for different iterative strategies for nonlinear least squares problems.*

Method	Step $\Delta\mathbf{x}$
Gradient Descent	$\Delta\mathbf{x} = -\alpha\mathbf{J}^\mathsf{T}\mathbf{f}(\mathbf{x})$
Newton's Method	$(\mathbf{H}^\mathsf{T}\mathbf{f} + \mathbf{J}^\mathsf{T}\mathbf{J})\Delta\mathbf{x} = -\mathbf{J}^\mathsf{T}\mathbf{f}(\mathbf{x})$
Gauss-Newton	$\mathbf{J}^\mathsf{T}\mathbf{J}\Delta\mathbf{x} = -\mathbf{J}^\mathsf{T}\mathbf{f}(\mathbf{x})$
Levenberg-Marquardt	$(\mathbf{J}^\mathsf{T}\mathbf{J} + \lambda\mathbf{I})\Delta\mathbf{x} = -\mathbf{J}^\mathsf{T}\mathbf{f}(\mathbf{x})$

While linear least squares problems like the ones presented in 3.2.2 are convex and therefore have a unique minimum, nonlinear least squares problems can have several minima. This makes nonlinear least squares problems more difficult to solve. Even more, depending on the initial guess $\mathbf{x_0}$ (i.e. the starting point of the iterative descent), the optimizer can converge to different solutions. Since it cannot be guaranteed, to which of the local minima the optimizer converges, one can never be sure whether the found solution is the global minimum or just one of potentially many local minima. Different strategies can be applied to mitigate the effects of this problem, like starting the optimization from different initial guesses, hoping that the global optimum will be among the different solutions. More on strategies for global optimization can be found in textbooks like [Pintér, 2006].

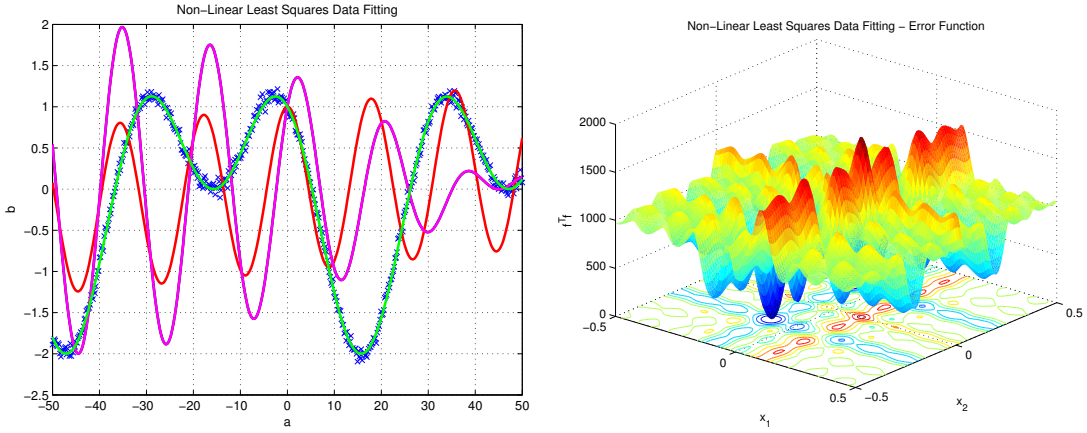
Depending on the specific error function, there may even be an infinite number of local minima. Such an extreme case is presented in the following example.

Example 3.3. *Consider we are given the measurements $(a_i, b_i)^\mathsf{T}$ illustrated in blue in Fig. 3.6(a). Like before, we know these points follow a function with two unknown parameters x_1 and x_2 and we shall estimate these parameters by applying least squares optimization. This time however, the functional model is nonlinear in its parameters:*

$$b_i = \sin(x_1 \cdot a_i) + \cos(x_2 \cdot a_i) \tag{3.43}$$

The error function depicted in 3.6(b) as an infinite number of local minima. Depending on the initial guess, the optimizer converges to very different solutions $\mathbf{x}^* = (x_1, x_2)^\mathsf{T}$ that, when inserted into the function definition above lead to very different behaviours. Three of these solutions are illustrated in Fig. 3.6(a). The green function corresponds to the parameter settings at the global optimum, while the red and magenta functions originate from the parameters found in one of the local minima.

Furthermore, how the optimizer descends the error function from different initial guesses is illustrated in Fig. 3.7.



(a) (b)

Figure 3.6: *Fitting the parameters x_1 and x_2 for the function $b = \sin(x_1 \cdot a) + \cos(x_2 \cdot a)$ to the given noisy input data is a highly nonlinear problem. (a) shows the data points (blue crosses) along with three solutions (green, red and pink lines) found by the optimizer. Each of these solutions corresponds to one of the many local minima of the error function shown in (b). See also Fig. 3.7.*

While the Gauss-Newton and Levenberg-Marquardt methods summarized above are very regularly used and the algorithm of choice in current SLAM back-ends, there might be better choices. Very recently, [Rosen et al., 2012] discussed the application of an approach coined *Powell's Dog Leg method* [Powell, 1970; Madsen et al., 2004] for optimization-based SLAM, after [Lourakis and Argyros, 2005] pointed out its advantages over Levenberg-Marquardt. According to [Rosen et al., 2012], Powell's Dog Leg method was implemented in a new version of iSAM (version 1.6) and proved to be considerably faster than both Gauss-Newton and Levenberg-Marquardt, while sharing the high accuracy of the Levenberg-Marquardt approach. Further developments in this area can be expected.

3.4 Weighted Nonlinear Least Squares Problems

In the methods and examples we encountered so far, all data points and partial objective functions f_i had the same influence on the optimization result. In many applications however, additional information is provided that allows to *weight* individual data points differently, thus some points will have more influence on the result of the optimization than others.

The mechanism to incorporate such weights into a least squares optimization process is easy. Consider that in the previous examples the objective function F was the sum over point-wise distances, using the squared euclidean norm $\|\cdot\|^2$, which lead to the squared

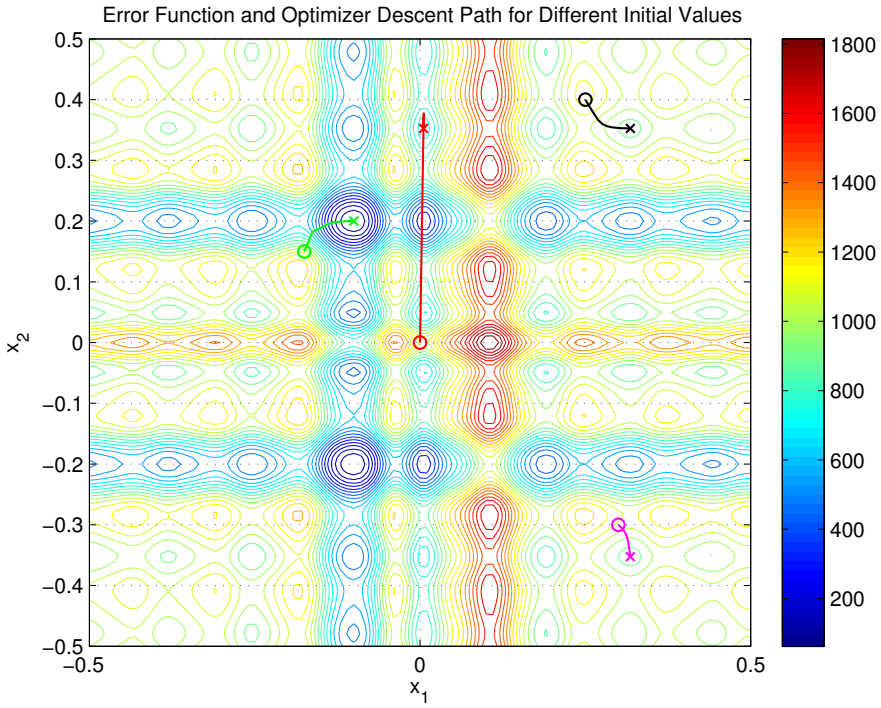


Figure 3.7: *The error function for the function fitting problem* $b = \sin(x_1 \cdot a) + \cos(x_2 \cdot a)$ *has infinitely many local minima. The figure shows different descent paths through the 2-dimensional state space* $(x_1, x_2)^\mathsf{T}$*. Depending on the starting point (circles), the optimizer converges towards different solutions (crosses). Only the descent path depicted in green converges to the global optimum. Notice that the colors of the descent paths correspond to the colors of the function plots in Fig. 3.6(a) and that the pink and black solution lead to identical functions. Also notice how the red descent path "jumps over" a deeper minimum than the one it finally converges to.*

objective functions:

$$\begin{aligned} F &= \frac{1}{2}\sum_i \|f_i\|^2 = \frac{1}{2}\sum_i \|\mathbf{x}_i - \mathbf{y}_i\|^2 \\ &= \frac{1}{2}\mathbf{f}^\mathsf{T}\mathbf{f} \end{aligned} \tag{3.44}$$

If covariance matrices $\mathbf{\Sigma}_i$ are given for the measurements $\mathbf{y}_i$, we can use the squared Mahalanobis distance instead of the squared Euclidean distance to form the objective function. The squared Mahalanobis distance $\|\mathbf{x}_i - \mathbf{y}_i\|^2_{\mathbf{\Sigma}_i}$ with covariance matrix $\mathbf{\Sigma}_i$ is defined as

$$\|\mathbf{x}_i - \mathbf{y}_i\|^2_{\mathbf{\Sigma}_i} = (\mathbf{x}_i - \mathbf{y}_i)^\mathsf{T}\mathbf{\Sigma}_i^{-1}(\mathbf{x}_i - \mathbf{y}_i) \tag{3.45}$$

By applying the definition of the Mahalanobis distance we define the weighted objective function:

$$\begin{aligned} F &= \frac{1}{2}\sum_i \|f_i\|^2_{\mathbf{\Sigma}_i} = \frac{1}{2}\sum_i \|\mathbf{x}_i - \mathbf{y}_i\|^2_{\mathbf{\Sigma}_i} \\ &= \frac{1}{2}\mathbf{f}^\mathsf{T}\mathbf{\Sigma}^{-1}\mathbf{f} \\ &= \frac{1}{2}\mathbf{f}^\mathsf{T}\mathbf{\Omega}\mathbf{f} \end{aligned} \tag{3.46}$$

$\mathbf{\Omega}$, the inverse covariance matrix $\mathbf{\Sigma}$, ($\mathbf{\Omega}$ is also called the *information matrix*), serves as a weight matrix that puts a different weight on each of the terms f_i that constitute the overall objective function F.

The *weighted least squares optimization problem* then has the form

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} F(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{x}} \frac{1}{2}\mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{\Omega}\mathbf{f}(\mathbf{x}) \tag{3.47}$$

and can in general be solved by the exact same methods explained above. The only difference is the incorporation of the weight matrix into the equations. For the sake of completeness, I want to shortly repeat the derivation of the Gauss-Newton update step equation with the information matrix:

As above, from $F(\mathbf{x}) = \frac{1}{2}\mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{\Omega}\mathbf{f}(\mathbf{x})$ and a first order Taylor expansion follows that

$$\begin{aligned} F(\mathbf{x}+\Delta\mathbf{x}) \approx L_{\mathbf{x}}(\Delta\mathbf{x}) &= \frac{1}{2}(\mathbf{f}(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x})^\mathsf{T}\mathbf{\Omega}(\mathbf{f}(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x}) \\ &= \frac{1}{2}\mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{\Omega}\mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{\Omega}\mathbf{J}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{\Omega}\mathbf{J}\Delta\mathbf{x} \\ &= F(\mathbf{x}) + \mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{\Omega}\mathbf{J}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{\Omega}\mathbf{J}\Delta\mathbf{x} \end{aligned} \tag{3.48}$$

Now, setting the first derivative to zero in order to find the minimizing step, we gain:

$$\begin{aligned} 0 = L'(\Delta\mathbf{x}) &= \mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{\Omega}\mathbf{J} + \mathbf{J}^\mathsf{T}\mathbf{\Omega}\mathbf{J}\Delta\mathbf{x} \\ \mathbf{J}^\mathsf{T}\mathbf{\Omega}\mathbf{J}\Delta\mathbf{x} &= -\mathbf{f}(\mathbf{x})^\mathsf{T}\mathbf{\Omega}\mathbf{J} \\ \mathbf{J}^\mathsf{T}\mathbf{\Omega}\mathbf{J}\Delta\mathbf{x} &= -\mathbf{J}^\mathsf{T}\mathbf{\Omega}^\mathsf{T}\mathbf{f}(\mathbf{x}) \end{aligned} \tag{3.49}$$

Or, if we apply Levenberg-Marquard's strategy:

$$(\mathbf{J}^\mathsf{T}\mathbf{\Omega}\mathbf{J} + \lambda\mathbf{I})\Delta\mathbf{x} = -\mathbf{J}^\mathsf{T}\mathbf{\Omega}^\mathsf{T}\mathbf{f}(\mathbf{x}) \tag{3.50}$$

Obviously the information matrix $\mathbf{\Omega}$ is incorporated very easily in the standard algorithms.

3.5 Least Squares Optimization for SLAM

After the previous sections reviewed a number of algorithms that solve least squares optimization problems, it is time to wrap things up and explore how these methods can be used to solve the SLAM problem in its least squares formulation.

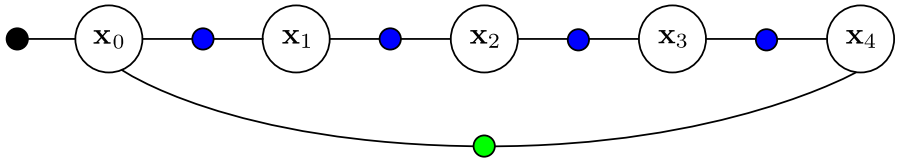


Figure 3.8: *Factor graph representation for the one-dimensional sandbox example described in the text. The black factor node represents the prior constraint that anchors the first robot position $\mathbf{x}_0$ at the origin. The blue factors represent odometry constraints while the green node shows the loop closure constraint between $\mathbf{x}_4$ and $\mathbf{x}_0$.*

Table 3.2: *A sandbox example for 1D SLAM: Observed values.*

Quantity	Ground Truth	Observed / Measured
$\mathbf{u}_0$	1.0	1.1
$\mathbf{u}_1$	1.0	1.0
$\mathbf{u}_2$	1.0	1.1
$\mathbf{u}_3$	-3.0	-2.7
$\mathbf{u}_{0,4}$	0.0	0.0

3.5.1 A Sandbox Example

A very simple one dimensional SLAM example will help to understand how optimization based SLAM approaches work and why they are very efficient. In our example, a robot will drive through its 1D environment and close a loop by returning to its starting point. Suppose we are given the measurements in table 3.2. As we see, there are four odometry measurements $\mathbf{u}_i$ that give rise to four odometry constraints of the form $\|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i}$ and one loop closure between $\mathbf{x}_0$ and $\mathbf{x}_4$ that creates the constraint $\|f(\mathbf{x}_0, \mathbf{u}_{0,4}) - \mathbf{x}_4\|^2_{\mathbf{\Lambda}_{0,4}}$. The standard deviations $\boldsymbol{\sigma}_i$ for the odometry readings and $\boldsymbol{\lambda}_{0,4}$ for the loop closure displacement are initialized as $\boldsymbol{\sigma}_i = 0.1$ and $\boldsymbol{\lambda}_{0,4} = 0.1$.

Table 3.3: *A sandbox example for 1D SLAM: Robot position according to odometry measurements and ground truth.*

Quantity	Ground Truth	According to Odometry
$\mathbf{x}_0$	0.0	0.0
$\mathbf{x}_1$	1.0	1.1
$\mathbf{x}_2$	2.0	2.1
$\mathbf{x}_3$	3.0	3.2
$\mathbf{x}_4$	0.0	0.5

Obviously there is a discrepancy between the odometry measurements and the ground truth which leads to a wrong initialization of the robot positions (see table 3.3). However, there is also a discrepancy induced by the loop closure constraint: According to odometry, $\mathbf{x}_4 = 0.5$, but according to the loop closure constraint $\mathbf{x}_4 = \mathbf{x}_0 = 0.0$.

We will now see how we can solve the problem using the least squares method derived before. The factor graph that represents our problem is depicted in Fig. 3.8. According to (2.25) we can find the optimal configuration of robot positions by minimizing over the sum of the constraints. This is a (weighted) least squares problem if the objective function $\mathbf{F}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^\top \mathbf{\Omega} \mathbf{f}(\mathbf{x})$ is built from the single constraints by setting

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f(\mathbf{x}_0, \mathbf{u}_0) - \mathbf{x}_1 \\ f(\mathbf{x}_1, \mathbf{u}_1) - \mathbf{x}_2 \\ f(\mathbf{x}_2, \mathbf{u}_2) - \mathbf{x}_3 \\ f(\mathbf{x}_3, \mathbf{u}_3) - \mathbf{x}_4 \\ f(\mathbf{x}_0, \mathbf{u}_{0,4}) - \mathbf{x}_4 \\ \mathbf{x}_0 - \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_0 + \mathbf{u}_0 - \mathbf{x}_1 \\ \mathbf{x}_1 + \mathbf{u}_1 - \mathbf{x}_2 \\ \mathbf{x}_2 + \mathbf{u}_2 - \mathbf{x}_3 \\ \mathbf{x}_3 + \mathbf{u}_3 - \mathbf{x}_4 \\ \mathbf{x}_0 + \mathbf{u}_{0,4} - \mathbf{x}_4 \\ \mathbf{x}_0 - 0 \end{pmatrix} \tag{3.51}$$

Notice that the last line is a so called *prior* constraint that anchors the first pose $\mathbf{x}_0$ at the origin.

The information matrix $\mathbf{\Omega}$ is formed by

$$\begin{aligned} \mathbf{\Omega} = \begin{pmatrix} \mathbf{\Sigma}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Lambda}_{0,4} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Pi} \end{pmatrix}^{-1} &= \begin{pmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 \end{pmatrix} \end{aligned} \tag{3.52}$$

where $\mathbf{\Pi}$ is the covariance associated with the aforementioned prior constraint. Notice how the information matrix (that serves as a weight matrix) has a block-diagonal structure and is built from the single covariance matrices of all constraints.

From the previous sections we know that we can solve such nonlinear[28] least squares problems by applying different iterative strategies, like Gauss-Newton or Levenberg-Marquardt. To solve our sandbox problem, we will pick Gauss-Newton. According to (3.49) we have to evaluate the Jacobian of $\mathbf{f}$ at the current estimate of $\mathbf{x}^*$ to calculate $\Delta\mathbf{x}$.

[28] To be correct, this sandbox example is a *linear* least squares problem. It could be solved with direct linear methods of course, but in order to demonstrate the general case of a nonlinear problem, I am going to apply a nonlinear method.

This $\Delta\mathbf{x}$ will then be added to the current estimate of $\mathbf{x}^*$ and the process is repeated until convergence. The Jacobian of $\mathbf{f}$ is

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{x}_0} \frac{\partial \mathbf{f}}{\partial \mathbf{x}_1} \cdots \frac{\partial \mathbf{f}}{\partial \mathbf{x}_4} \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.53}$$

We see immediately that $\mathbf{J}$ has a block diagonal structure for the odometry constraints (top four rows) and is sparse, since large areas of the matrix are zero. This will become important in a moment.

Recall from (3.49) that we have to solve

$$\mathbf{J}^\mathsf{T}\mathbf{\Omega}\mathbf{J}\Delta\mathbf{x} = -\mathbf{J}^\mathsf{T}\mathbf{\Omega}^\mathsf{T}\mathbf{f}(\mathbf{x}) \tag{3.54}$$

for $\Delta\mathbf{x}$. We can write this shorter as

$$\mathbf{H}\Delta\mathbf{x} = -\mathbf{b} \tag{3.55}$$

by setting

$$\mathbf{H} = \mathbf{J}^\mathsf{T}\mathbf{\Omega}\mathbf{J} = \begin{pmatrix} 1200 & -100 & 0 & 0 & -100 \\ -100 & 200 & -100 & 0 & 0 \\ 0 & -100 & 200 & -100 & 0 \\ 0 & 0 & -100 & 300 & -100 \\ -100 & 0 & 0 & -100 & 200 \end{pmatrix} \tag{3.56}$$

and

$$\mathbf{b} = \mathbf{J}^\mathsf{T}\mathbf{\Omega}^\mathsf{T}\mathbf{f}(\mathbf{x_0}) = \begin{pmatrix} -50 & 0 & 0 & 0 & 50 \end{pmatrix}^\mathsf{T} \tag{3.57}$$

with $\mathbf{x_0} = \begin{pmatrix} 0 & 1.1 & 2.1 & 3.2 & 0.5 \end{pmatrix}^\mathsf{T}$ being the initial guess of the robot poses according to odometry. Therefore the initial error is $\mathbf{f}(\mathbf{x_0}) = \begin{pmatrix} 0 & 0.1 & 0.1 & 0.2 & -0.5 \end{pmatrix}^\mathsf{T}$.

The linear system of equations (3.55) can be solved in different ways, for instance using a Cholesky decomposition of $\mathbf{H}$. Notice however, that due to $\mathbf{J}$ being sparse, $\mathbf{H}$ is sparse as well and has a special structure:

Result 3.1. $\mathbf{H}_{i,j}$ *is non-zero only if the variables* $\mathbf{x}_i$ *and* $\mathbf{x}_j$ *are connected by a constraint.*

Result 3.2. $\mathbf{H}$ *is an adjacency matrix of the Markov random field graphical representation of the optimization problem. The sparsity of* $\mathbf{H}$ *corresponds to the sparse structure of the graphical representation of the underlying problem. That is, the (variable) nodes in those graphs have a low degree*[29].

[29] The *degree* of a node in a graph corresponds to the number of edges at this node.

The fill-in factor of $\mathbf{H}$ (i.e. the proportion of non-zero entries) in our simple example is 60% but will be much smaller for larger problems. This sparsity of $\mathbf{H}$ can be exploited. It allows us to apply very efficient decomposition algorithms to solve the linear system $\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$ [Davis, 2006].

Here we will apply the Cholesky decomposition of $\mathbf{H}$ to solve (3.55). Cholesky decomposes $\mathbf{H}$ into $\mathbf{H} = \mathbf{L}\mathbf{L}^\mathsf{T}$ with

$$\mathbf{L} = \begin{pmatrix} 34.6410 & 0 & 0 & 0 & 0 \\ -2.8868 & 13.8444 & 0 & 0 & 0 \\ 0 & -7.2232 & 12.1584 & 0 & 0 \\ 0 & 0 & -8.2248 & 11.5045 & 0 \\ -2.8868 & -0.6019 & -0.3576 & -8.9479 & 10.5409 \end{pmatrix} \tag{3.58}$$

We can now solve $\mathbf{L}\mathbf{y} = -\mathbf{b}$ by simple variable insertion and gain

$$\mathbf{y} = \begin{pmatrix} 1.4434 & 0.3010 & 0.1788 & 0.1278 & -4.2164 \end{pmatrix}^\mathsf{T} \tag{3.59}$$

With equal ease we solve $\mathbf{L}^\mathsf{T}\Delta\mathbf{x} = \mathbf{y}$ for our sought $\Delta\mathbf{x}$ which is

$$\Delta\mathbf{x} = \begin{pmatrix} 0 & -0.1 & -0.2 & -0.3 & -0.4 \end{pmatrix}^\mathsf{T} \tag{3.60}$$

Our first improved solution on the sought robot positions is therefore

$$\mathbf{x_1} = \mathbf{x_0} + \Delta\mathbf{x} = \begin{pmatrix} 0 & 1.0 & 1.9 & 2.9 & 0.1 \end{pmatrix} \tag{3.61}$$

This was the first iteration of the algorithm, for the second iteration we calculate the new error to be $\mathbf{f}(\mathbf{x_1}) = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & -0.1 & 0 \end{pmatrix}^\mathsf{T}$ and $\mathbf{b} = \mathbf{J}^\mathsf{T}\mathbf{\Omega}^\mathsf{T}\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$. This however means that the new $\Delta\mathbf{x}$ is $\mathbf{0}$, and thus we converged to the solution of $\mathbf{x}^*$ as given above. Notice how the remaining error was equally distributed among the robot states, so that the residual error is equal for each state.

Although our example converged immediately, real applications will usually not converge in only one iteration. Our sandbox example is strictly linear, therefore the resulting least squares problem is linear as well and thus can be solved directly in a single step. If nonlinearities are involved, the Jacobian $\mathbf{J}$ has to be re-evaluated and will change after each iteration, leading to a convergence behaviour that requires several iterations.

3.5.2 Why Optimization-based SLAM is Efficient

Matrix decompositions like Cholesky or QR play a crucial role when it comes to numerically solving nonlinear least squares optimization problems. Especially for large problems, these decompositions have a major influence on the overall algorithm runtime and complexity [Krauthausen et al., 2006; Dellaert and Kaess, 2006].

The small sandbox example above demonstrated how the system of linear equations $\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$ is formed for a SLAM problem from the Jacobian $\mathbf{J}$ of the system's error function. Fig. 3.9 shows both $\mathbf{J}$ and $\mathbf{H}$ for the somewhat larger example presented in

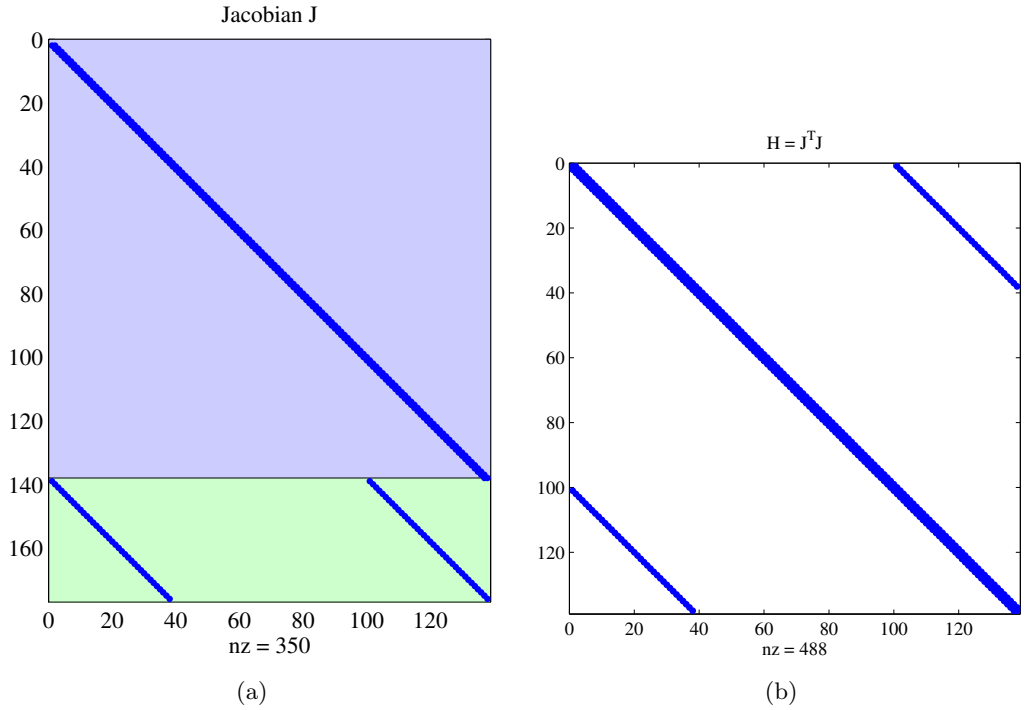


Figure 3.9: *Jacobian* $\mathbf{J}$ *and Hessian* $\mathbf{H} = \mathbf{J}^\mathsf{T}\mathbf{J}$ *for the example SLAM problem from Fig. 3.9. The non-zero values ("nz") are marked by a blue point. All other entries in the matrices are zero, thus both matrices are very sparse. The upper part of the Jacobian in (a) contains the derivatives of the 138 odometry constraints (blue background), while the derivatives of the loop closure constraints are visible in the lower part (green background).*

Fig. 2.21(b). As can be seen, the matrices are very sparse. Furthermore, the two blocks in $\mathbf{J}$ that contain the derivatives of the odometry and loop closure constraints can be clearly identified.

We have seen how the matrix $\mathbf{H}$ can be decomposed (using Cholesky or QR) to solve the system of equations $\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$. The matrix $\mathbf{H}$ will be of size $n \times n$ for n being the number of variables (e.g. pose variables). For large problems, $\mathbf{H}$ will therefore grow quickly. For example the Manhattan dataset, a 2D dataset commonly used to benchmark optimization-based SLAM algorithms, contains 3500 poses. Therefore, $n = 3500 \cdot 3 = 10500$ since every pose consists of 3 entries $\mathbf{x} = (x, y, \theta)^\mathsf{T}$. Performing Cholesky or QR decomposition of a matrix of size 10500×10500 would be intractable, if $\mathbf{H}$ would not have that special sparse structure. For the Manhattan dataset, only 189219 entries in the 10500×10500 matrix $\mathbf{H}$ are non-zero, which corresponds to a fill-in factor of only 0.17%. In general $\mathbf{H}$ will have a block-diagonal structure with off-diagonal elements originating from loop closure constraints or landmark observations.

A further interesting fact about **H** is, that it corresponds to the information matrix of the estimated system. That means, the covariances of the estimated variables can be retrieved from **H** by inverting the complete matrix or by marginalization of individual entries [Kaess et al., 2008].

To summarize this section, the important point to understand why modern SLAM backends like g^{2}o [Kümmerle et al., 2011b], iSAM2 [Kaess et al., 2011] and its predecessors can solve large-scale SLAM problems with many thousands of variables and constraints very efficiently is that the numerical algorithms for Cholesky or QR decomposition excessively exploit the sparse structure of the problem and the resulting sparsity of **H**. While tight complexity bounds are hard to give, [Krauthausen et al., 2006] established a general $\mathcal{O}(n^{1.5})$ bound for SLAM exploiting the sparsity for matrix factorization instead of the $\mathcal{O}(n^3)$ complexity for dense matrices.

3.6 Least Squares Optimization in the Presence of Outliers

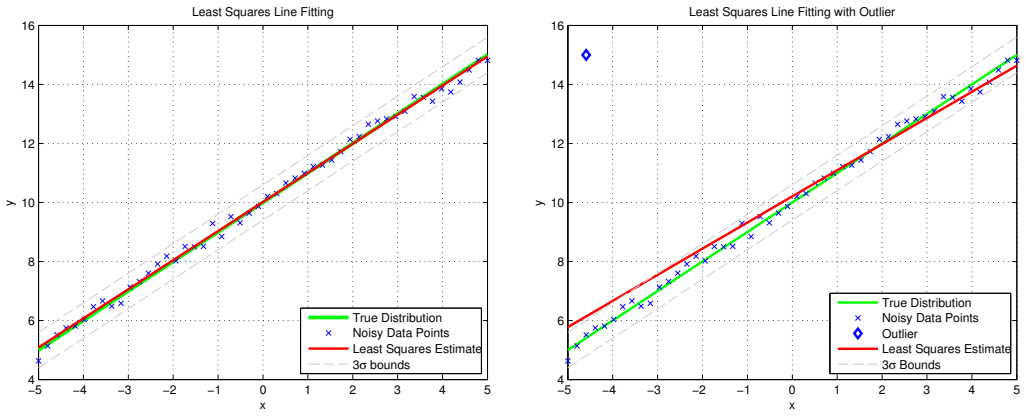
As we have seen, least squares methods can be used to efficiently solve a large number of linear and nonlinear optimization problems.

However, when dealing with real-world applications, certain serious problems can arise. Normally, the data points that constitute the optimization problem have been acquired by non-ideal sensor systems or pre-processed by algorithms that made certain assumptions on the nature of the measured quantities or the behaviour of the sensor system. All these ideal assumptions may be violated from time to time, the sensor may produce measurement errors that the sensor model does not account for, the pre-processing algorithms may misinterpret the data etc. The result of these undesired effects are so called *outliers*. The Encylopedia of Mathematics [Balakrishnan and Childs, 2001] defines an *outlier* more generally to be "any observation in a set of data that is inconsistent with the remainder of the observations in that data set." Outliers pose serious problems to any least squares optimization as they commonly lead to wrong or defective solutions. What effects outliers can have on the result of least squares optimization is illustrated by the following example:

Example 3.4. *To illustrate the effects of outliers on least squares optimization, we reuse the data from example 3.1: Fig. 3.10(a) shows a number of observed noisy data points depicted as blue crosses. They follow a certain, but unknown model, whose parameters have to be determined. The true underlying linear model* $y = x + 10$ *is illustrated by the green line.*

The data points deviate from the true model by a Gaussian error term with small variance $\sigma^2 = 0.04^2$*. As we have seen before, we can determine the unknown model parameters by a least squares estimation. Despite the Gaussian noise in the observations, the resulting estimate (shown in red) follows the true line model almost exactly, with a root-mean-squared-error (RMSE) of* 0.04*.*

In Fig. 3.10(b), however, a single outlier (illustrated by the blue diamond) has been added to the observations. Is lies far outside the 3σ *bounds of the assumed Gaussian*



(a) (b)

Figure 3.10: *Fitting a polynomial of 1st and 3rd degree in the presence of outliers. Obviously, even a single outlier is enough to lead to a defective solution.*

error model and is therefore inconsistent with the rest of the observations. Although there is only one single outlier present in the data set, the least squares estimate of the model parameters now significantly deviates from the true values, leading to a much higher RMSE of 0.36.

In simple words, the outlier drags the least squares solution towards itself. The reason for this behaviour and the seemingly over-proportional influence of that single data point lies in the formulation of the error function: Due to its quadratic characteristic, the influence of a data point on the overall estimation result raises with the square of its error. Outliers, that by definition have large errors, therefore have a large influence on the result of least squares optimizations.

3.6.1 Sample Consensus Methods for Outlier Rejection

The well known vulnerability of least squares methods to outliers has of course lead to a number of approaches that are commonly found in the literature. In the domain of model-fitting, regression and the like, sample consensus methods like RANSAC [Fischler and Bolles, 1981] are state of the art. In the following, I want to shortly summarize the main ideas.

RANSAC

RANSAC (RANdom SAmple Consensus) [Fischler and Bolles, 1981] is probably the best known robust estimation scheme. Its main idea is relatively simple:

1. Randomly choose n points from the given data set. n is the number of points that are sufficient to uniquely determine the parameters of the sought model. For

instance, if we want to fit points to a line in $\mathbb{R}^2$, two points are enough to uniquely determine that line. Therefore $n = 2$. To fit points to a plane in $\mathbb{R}^3$, we have to choose $n = 3$ points, and so on. The n chosen points form the so called *minimal set*.

2. Calculate the uniquely determined model parameters from that minimal set. These model parameters form the current *hypothesis*.

3. Determine the *inlier set* for the current hypothesis. The inlier set consists of those data points that have a distance below a certain threshold τ from the model determined in the last step.

4. Iterate steps 1 to 3 k times.

5. Return the parameters of the hypothesis with the largest inlier set.

Two steps of the algorithm are illustrated in Fig. 3.11 using the line fitting example in the presence of a single outlier. An open question is, how large k, the number of iterations, should be chosen. Its calculation follows the insight that the algorithm should with probability p choose at least one minimal set that contains no outliers. In this case, the resulting inlier set can be expected to be larger than all others.

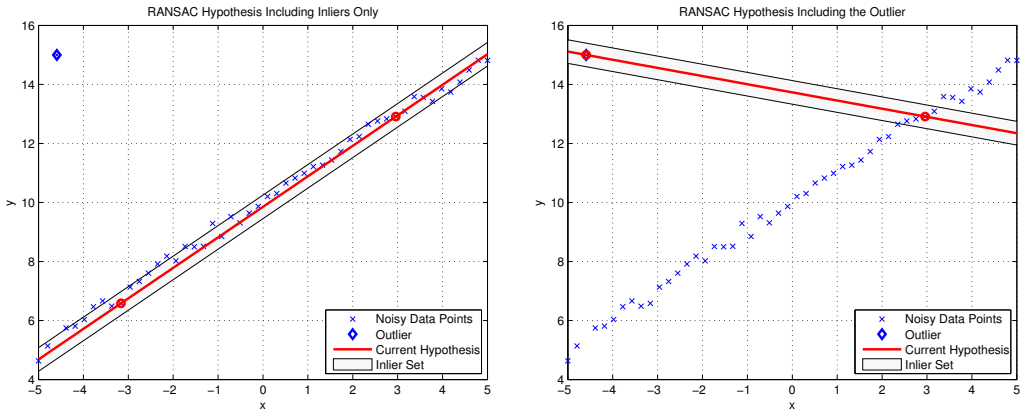
If we knew ε, the ratio of outliers to inliers in the data set, a randomly chosen point would be an inlier with probability $1 - \varepsilon$. Then the probability that all n points in the current minimal set are inliers is $(1 - \varepsilon)^n$. Therefore the probability that at least one point is an outlier is $1 - (1 - \varepsilon)^n$ and finally the probability that each of the k minimal sets contains at least one outlier is $(1 - (1 - \varepsilon)^n)^k$. As we want to ensure this probability to be below $1 - p$, we arrive at:

$$k = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^n)} \tag{3.62}$$

Further Developments

More recently, a number of extensions have been proposed that increase the performance of RANSAC. While RANSAC simply counts the number of inliers and chooses the hypothesis that scored the highest inlier count, M-SAC chooses the hypothesis for which the sum of all residual errors (the distance from a point to the current model) was minimal.

Another approach is MLESAC [Torr and Zisserman, 2000] that chooses the hypothesis with the maximum likelihood instead the one with the minimal sum of residual errors. [Nister, 2003] proposed preemptive RANSAC to reduce the time spent evaluating probably bad hypothesis. PROSAC [Chum and Matas, 2005] in turn proposes to use a guided (only semi-random) sampling process. More recently, [Raguram et al., 2009] explicitly incorporated uncertainty information into the RANSAC scheme and proposed an algorithm called Cov-RANSAC.



(a) (b)

Figure 3.11: *Two steps of the RANSAC algorithm: In (a), the minimal set consists of the two data points marked with red circles. The hypothesis arising from these two points is illustrated in red. The inlier set (grey area) includes most of the data points, resulting in a high consensus measure for this hypothesis. In contrast, hypotheses including an outlier (b) will include less data points, thus gaining low consensus.*

3.6.2 Robust Cost Functions

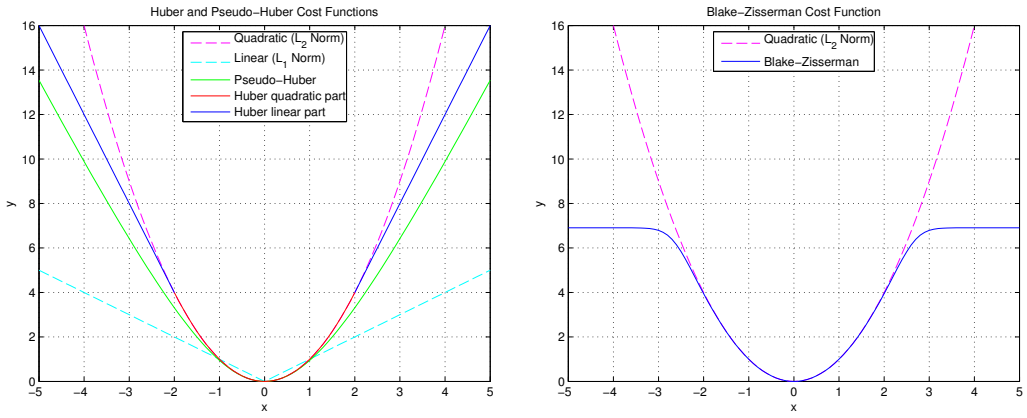
Under the assumption of normally distributed data with Gaussian noise, the squared cost function leads the optimizer to the Maximum Likelihood solution. However, due to the quadratic characteristics of the cost function, outliers can have a very large negative influence on the solution. The general idea of robust cost functions therefore is to modify the cost function in a way that it is quadratic for small errors, but raises slower (e.g. linearly) or stays even constant for larger errors induced by potential outliers.

[Blake and Zisserman, 1987] for instance proposed a function of the form

$$f_{BZ}(x) = -\log(e^{-x^2} + \epsilon) \tag{3.63}$$

which approximates the squared cost function x^2 for small x, but results in an asymptotically constant error value of $-\log\epsilon$ for large x (see Fig. 3.12(b)). Although the influence of outliers is bounded this way, the Blake-Zisserman function has the disadvantage of being non-convex and introducing local minima where the optimizer can get stuck.

Better alternatives are the convex Huber and Pseudo-Huber functions which are illustrated in Fig. 3.12(a). I will shortly discuss them below, a more elaborate overview and analysis of different robust cost functions can be found in [Hartley and Zisserman, 2004, ch. A6.8]. Further comments appear in [Boyd and Vandenberghe, 2004, ch. 6].



(a) (b)

Figure 3.12: *(a) The Huber function that can be used as a robust cost function. It consists of a quadratic part (red) for small $|x|b$ and raises linearly for $|x| \geq b$ (blue). The pseudo-Huber function (green) is a smooth approximation to the Huber function. For comparison, the quadratic and linear cost functions are shown in magenta and cyan. (b) The Blake-Zisserman cost function approximates the quadratic L_2 norm for small values of $|x|$ and remains asymptotically constant for larger values. Although it is robust against outliers, it has the disadvantage of being non-convex.*

The Huber cost function combines the characteristics of the squared and linear cost function [Huber, 1973]. It is defined as

$$f_H(x) = \begin{cases} x^2 & : |x| < b \\ 2b|x| - b^2 & : \text{otherwise} \end{cases} \tag{3.64}$$

Below a threshold b it is identical to the squared cost function and raises linearly above that. As the parameter b controls the change from quadratic to linear behaviour, it should be chosen as a threshold between inliers and outliers.

The Pseudo Huber cost function is a smoother approximation of the Huber function. It has continuous derivatives of all orders, in contrast to the Huber function which is only differentiable once. The Pseudo Huber function is defined as

$$f_{pH} = 2b^2 \left(\sqrt{1 + (x/b)^2} - 1 \right) \tag{3.65}$$

Applying a Robust Cost Function

As we already know, least squares methods such as Levenberg-Marquardt work by minimizing a squared error term $\mathbf{f}(\mathbf{x})^\mathsf{T} \boldsymbol{\Omega} \mathbf{f}(\mathbf{x})$. When using toolboxes or libraries for

optimization, the user often only has to provide the initial $\mathbf{x_0}$ and an implementation of the function $\mathbf{f}(\mathbf{x})$, without being able to change the quadratic characteristic of the objective function that is minimized.

If a robust cost function like the Huber function should be used instead of the normal squared cost function, we can work around this limitation and replace the vector $\mathbf{f} = \mathbf{f}(\mathbf{x})$ by $(w_1 f_1, w_2 f_2, \ldots, w_n f_n)^\mathsf{T}$. The single weight factors w_i are determined by

$$w_i = \sqrt{f_H(f_i)}/f_i \tag{3.66}$$

with f_H being the Huber function, for example. This results in the robust optimization problem

$$\underset{\mathbf{x}}{\operatorname{argmin}} \, \mathbf{\Omega f_H}(\mathbf{x}) \tag{3.67}$$

with $\mathbf{f_H} = (f_H(f_1), \ldots, f_H(f_n))^\mathsf{T}$ as desired.

3.7 Summary

This chapter reviewed the basics of least squares optimization problems that come in linear and nonlinear form. The chapter provided some insights into the algorithms that are applicable to solve such problems and demonstrated how the least squares formulation of pose graph SLAM can be solved. As we have seen, the reason why SLAM in its least squares formulation can be solved efficiently lies in the sparsity of the problem that is apparent in the structure of the graph representation (e.g. Markov random field or factor graph) and in the matrix $\mathbf{H}$ that approximates the Hessian of the weighted objective function and at the same time is the adjacency matrix of the SLAM problem's Markov random field.

The chapter concluded with the well-known problem of outliers and demonstrated the severe effects such outliers have on the least squares solution to a given problem. Since least squares optimization methods are not robust against outliers, a number of approaches have been developed to mitigate their influence, some of which have been shortly reviewed or mentioned.

The introductory part of my thesis ends at this point. Up to now, the material has in large parts been a review of what is already known in the community and what has been developed and published by others before. My own contributions will begin to unfold with the next chapter that motivates the following developments. I will show that outliers are, as we might already expect, a severe problem in optimization-based SLAM and that no sufficient suitable method for outlier mitigation for optimization-based SLAM has been proposed so far. The approach developed in Chapter 5 will address this open problem and the evaluations in Chapters 6 and 7 will prove its feasibility.

4

Motivation – When Optimization Fails

The previous chapters reviewed the SLAM problem in general and recent advances in optimization-based SLAM approaches in particular. As we have seen, efficient algorithms exploiting the sparsity of SLAM have been proposed and implementations in the form of open-source libraries are available to the robotics researcher, ready to be applied. These well-documented frameworks support batch and incremental processing, thus can be used to solve both the full and the online SLAM problem. It seems that 25 years after the first thoughts on the problem, SLAM has finally been solved[30].

However, appearances can be deceiving. At the utmost, the *back-end* part of SLAM might seem to be solved for now. That is, the known algorithms and corresponding numeric implementations for sparse optimization seem to be sufficient for all current practical and most academic purposes. The sensor-driven front-end part, however, has not so clearly converged towards a commonly accepted approach that has proven successful in a broad number of problems. Data association, place recognition, and loop closure detection are still very active research areas.

Even more, a major problem of current approaches to optimization-based SLAM arises from the strict division into a front-end and a back-end part. Current state of the art SLAM back-ends are least squares optimizers and as such, they are naturally not robust against outliers. For the general case of least squares optimization, this is known and understood, and I shortly discussed possible workarounds in section 3.6. For the special case of SLAM back-ends, the problem is widely acknowledged but usually ignored: While the front-end is responsible for sensor data processing, data association and graph construction, the back-end optimizer considers the data association problem solved. That means the back-end relies heavily on the front-end and expects it to produce

[30] A very interesting interview with Sebastian Thrun and José Neira on the topic "Is SLAM solved?" can be found in [Frese, 2010].

(a) (b)

Figure 4.1: *Failed place recognition: Due to the self-similarity of man-made environments, the place recognition system erroneously declared both images to originate from one and the same place. This is an example from [Sünderhauf and Protzel, 2011], showing a false-positive place recognition of our system based on BRIEF-Gist.*

a topologically correct factor graph. Although any failure in the data association can have catastrophic implications on the resultant map and robot state estimates, state of the art back-ends do little or nothing at all to mitigate these potentially severe effects.

This chapter constitutes the motivation of my dissertation and lays the foundations for the next chapter that proposes a solution to the problems stated here.

4.1 Data Association Errors and their Effects on SLAM

Typical data association errors in SLAM with potentially severe effects are false-positive loop closure constraints: Due to the high self-similarity of many indoor and non-natural outdoor environments, two distinct places can actually appear to be very similar to the sensor (e.g. a camera or laser range finder). Due to this so called *perceptual aliasing*, appearance-based SLAM front-ends can be lured into inserting a loop closure constraint between two similar looking, but not corresponding scenes that in reality were recorded at two very distinct (and distant) places in the environment. An example can be seen in Fig. 4.1. The two scenes from an urban dataset[31] have been erroneously assigned to the same place in the environment by our visual place recognition system based on

[31] The dataset that was used here was recorded from a car driving through St. Lucia, a suburb of Brisbane, Australia and covers roughly 66 km of urban roads [Milford and Wyeth, 2008]. I very much thank David Milford for providing the video footage. The dataset is very challenging, since the self-similarity of the environment is high and even for a human observer it is hard to tell non-corresponding places apart and to recognize the many large and small loop closures that occur. The dataset will be used again in Chapter 7 as an example for robust SLAM in a large-scale real-world environment.

BRIEF-Gist [Sünderhauf and Protzel, 2011] although they have been recorded in different places.

Spoken more technically, if the data association step in the front-end fails and erroneously detects a loop closure between two poses $\mathbf{x}_i$ and $\mathbf{x}_j$, a loop closure factor $\mathbf{u}_{ij}$ is introduced between the two corresponding nodes in the factor graph. This factor forces the optimizer to map the two poses onto each other according to $\mathbf{u}_{ij}$, which will very likely lead to divergence and a defective solution.

Notice that data association errors can also occur in landmark-based SLAM systems. In this case, observations would be erroneously associated with the wrong landmark, leading to "ghost" observations of landmarks in the wrong places.

Result 4.1. *False-positive loop closure constraints or false-positive landmark observations correspond to additional, erroneous constraint edges in the graph representation of the SLAM problem. Thus the* topology *of the graph becomes incorrect with respect to the ground truth representation.*

Following the terminology of general least squares optimization, we can call these additional, erroneous constraint edges *outliers*. The following short example will demonstrate the effect a few of these outliers can have in pose graph SLAM.

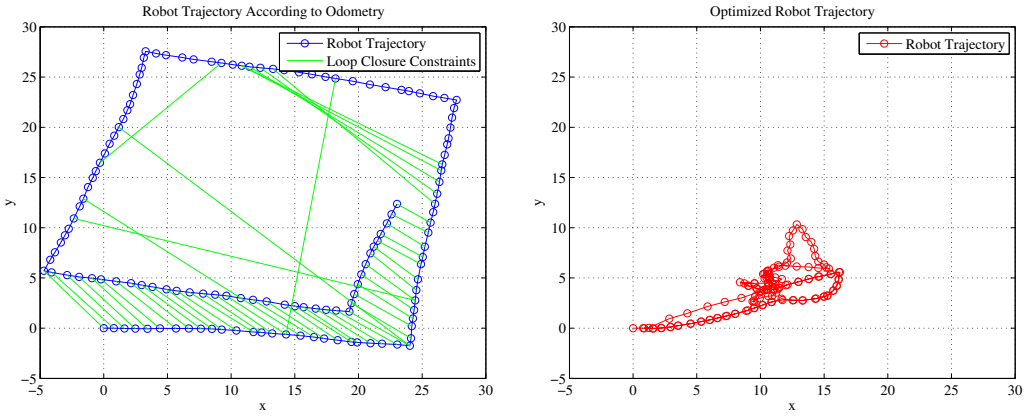
Example 4.1. *We can now consider an example that illustrates the effects of outliers in optimization-based SLAM systems. Again we use a simple example of a robot driving in a squared trajectory and revisiting some parts of the environment (as shown previously in Fig. 2.21(b)). As we remember, the place recognition module recognizes the revisited parts of the trajectory and correctly inserts a number of true positive loop closure constraints. This time however, due to perceptual aliasing and self similarity, the front-end also detects ten wrong loop closures and inserts them into the graph representation of the SLAM problem.*

Fig. 4.2(a) shows the trajectory estimated by the noisy odometry sensor along with the true positive and false positive loop closure constraints. Given this corrupted problem formulation, the back-end is not able to converge towards a meaningful solution, as can be seen in Fig. 4.2(b). This solution was determined with the state of the art back-end g^{2}o *. The Huber robust cost function which was used to mitigate the effects of the outlier constraints, proved to be insufficient to handle the situation*[32]*. Obviously, the additional constraints forced the optimizer to converge towards a collapsed and twisted trajectory. This result however, is not usable for any further application since it does not at all represent the true trajectory the robot was driven.*

4.2 Current Approaches for Outlier Mitigation and Avoidance

To avoid or mitigate the potentially catastrophic effects of false positive loop closure constraints, different strategies have been proposed in the literature. I am going to review the most important and influential ideas separately for the front-end and the back-end side.

[32] Several values for the kernel width were tried here, none of which provided satisfactory results.



(a) (b)

Figure 4.2: *A small simulated dataset with data association errors. Compare (a) to Fig. 2.21(b). In addition to the correct loop closure constraints, ten additional false-positive loop closures have been inserted by the front-end after place recognition failed as illustrated in Fig. 4.1. The results are catastrophic: (b) shows the optimized (maximum a posteriori) estimate of the robot trajectory as calculated by* g^{2}o *. Despite the Huber cost function was used to mitigate the effects of outlier constraints, the optimization converged towards a meaningless result.*

4.2.1 Outlier Avoidance on the Front-End Side

The best outlier mitigation strategy is to avoid outliers in the first place. Many proposed place recognition approaches explicitly try to detect and filter potential false positive loop closures. Although many of these strategies are appealing and reach high precision measures in several benchmarks, the bottom line is that *none* of the current approaches can *guarantee* to always work perfectly and never let a false positive pass.

The recently proposed FAB-MAP system [Cummins and Newman, 2008] for instance addresses the problem by combining the visual vocabulary tree approach [Nister and Stewenius, 2006] with a Bayesian probabilistic framework, thereby formulating place recognition as a recursive Bayesian estimation problem. Although FAB-MAP has been reported to perform well on extremely large datasets of up to 1000 km [Cummins and Newman, 2009], several other authors confirmed that FAB-MAP still produces false positive loop closure constraints, depending on the environment and the camera configuration used [Mar et al., 2010; Glover et al., 2010; Maddern et al., 2009]. Using FAB-MAP alone, without an outlier mitigation is therefore not sufficient.

Other authors apply a post-processing step to filter false positives from the loop closure candidates. [Konolige et al., 2010a] and [Konolige and Bowman, 2009] use a stereo camera setup to detect potential loop closures and propose a geometric consistency check based

on perspective view geometry to filter false positives. Place recognition with stereo cameras has been as well demonstrated by [Cadena et al., 2010; Cadena et al., 2011], building on work by [Ramos et al., 2008]. A technique coined CRF-Matching [Ramos et al., 2007] that builds upon Conditional Random Fields [Lafferty et al., 2001] has been used to identify false positive loop closure candidates. This technique has recently been re-used in multi-session SLAM [McDonald et al., 2011].

Other sophisticated data association strategies include the joint compatibility branch and bound algorithm (JCBB) [Neira and Tardos, 2001], a common approach based on maximum likelihood and mutual compatibility. Olson et al. proposed a compatibility check based on graph partitioning (SCGP) [Olson et al., 2005].

Discussion

Given the references above, it seems that a sufficient number of algorithms and approaches for outlier avoidance exist on the front-end side of SLAM. However, a closer examination of the recent literature on SLAM reveals that the problem cannot be considered solved. As it turns out, the proposed approaches are not sufficient and outliers still pose serious threats to working SLAM systems, especially – but not exclusively – in long-term or large-scale operations.

This is acknowledged by a number of authors. For instance [Olson, 2009] writes

> This process of data association [...] is highly susceptible to errors, and data-association errors can cause catastrophic divergence of the map.

[Cummins and Newman, 2009] write about "false positives that would cause filter divergence.", talking about loop closure detections in the context of SLAM. [Cadena et al., 2011] argue that false positives have catastrophic effects, while false negatives are more tolerable:

> A limitation of this system is that perceptual aliasing, or false positives, occurs in scenes where near information is very similar [...] These errors are catastrophic for the environment model being built, since it falsely connects unrelated areas. False negatives, or not being able to identify images from the same scene as correspondent, are not as catastrophic, although the precision of the resulting model is negatively affected when common areas are not identified. Ideally, all false positives as well as all false negatives should be avoided.

Other authors like [Maddern et al., 2011] stress the 100% precision requirement for loop closure detection systems. That means that no false positive loop closures should be detected:

> For use in loop closure detection for metric SLAM, the desired performance is high recall at 100% precision.

[Galvez-Lopez and Tardos, 2011] also denote the "desired working point" of a place recognition system to be at 100% precision.

Interestingly, a remarkable system that explicitly deals with false positive loop closure constraints is RatSLAM [Milford and Wyeth, 2008], a biologically inspired approach to SLAM. Combining RatSLAM with FAB-MAP, [Glover et al., 2010] point out that false positives are inevitable and have to be explicitly handled by the SLAM system:

> However, the false positive data association has not caused catastrophic failure [...] because the RatSLAM pose filtering addresses the false positives produced by FAB-MAP. This is a necessary addition to any data association system, as false positives are inevitable when dealing with large long-term real-world datasets.

The same paper confirms that even a state of the art front-end data association is not enough for a stable and robust long term SLAM system:

> FAB-MAP fails as any false positive causes an incorrect loop closure [...] Mapping using only FAB-MAP data association, results in catastrophic failure over full day datasets.

More references can be found in the literature, but the examples above are sufficient to conclude that none of the current data association techniques is considered to work perfectly, i.e. none is guaranteed to reach a precision of 100%.

Result 4.2. *Despite a number of approaches for outlier avoidance on the front-end side of modern SLAM systems exist, outliers arising from data association errors such as false positive loop closures are still considered a serious problem by the current SLAM literature. None of the proposed approaches for loop closure detection can guarantee to prevent that false positive loop closure constraints are created.*

As even a single wrong loop closure constraint can cause the whole SLAM system to fail, the back-end should not have to rely solely on the front-end data association. It should rather be able to mitigate the existing outliers or even change the data association decisions made by the front-end, if they appear to be false at a later time during the optimization.

4.2.2 Outlier Mitigation on the Back-End Side

On the back-end side, outlier mitigation is a widely unregarded topic. Sample consensus methods (see section 3.6.1) such as RANSAC [Fischler and Bolles, 1981], MSAC, or MLESAC [Torr and Zisserman, 2000] are state of the art in many applications such as model-fitting and regression. However due to their main idea (repeatedly pick a minimal set of data points at random, determine a hypothesis for the sought model and search for the hypothesis with the largest inlier set), these approaches can hardly be applied to SLAM and familiar problems, since there is no such thing as a minimal set of constraints that could be picked to solve a SLAM problem.

So called *robust cost functions* (see section 3.6.2), like the Huber function [Huber, 1973] are better suited for SLAM back-ends. They are typically applied to mitigate the effects of outliers in least squares problems. If outliers are expected, g^{2}o [Kümmerle et al., 2011b] and recently also iSAM 1.6 [Rosen et al., 2012] allow to use the Huber or pseudo-Huber functions instead of the usual squared error function. The idea of Huber is that the error function for data points whose error is above a certain threshold should raise linearly instead of quadratically as is normally the case in least squares. However, as we have seen in the example in Fig. 4.2(b) robust cost functions are not sufficient to deal with outlier constraints like false-positive loop closures since the influence of outliers is merely reduced, but not removed. Furthermore, a suitable threshold has to be chosen manually beforehand and is fixed thereafter.

Result 4.3. *The known mechanisms for outlier mitigation on the back-end side are not sufficient to cope with outlier loop closure constraints.*

4.3 Summary

To summarize this chapter, optimization-based SLAM is fragile when outliers such as false positive loop closure constraints are present. The problem is acknowledged in the literature, but the measures to avoid or mitigate it are insufficient at the moment.

This constitutes the motivation for my work. Since neither the front-end nor the back-end approaches provide the desired and necessary robustness, a novel approach had to be developed. However, I did not want to come up with yet another method for appearance-based place recognition that might have been even more complex or mathematically involved than the existing ones, or a scheme for outlier rejection on the front-end side. Instead, my goal was to solve the outlier problem on the back-end side. I therefore accepted that every front-end will inevitably produce data association errors from time to time and thus for instance erroneously insert additional loop closure constraints edges into the formulation of the pose graph SLAM problem. The approach I wanted to develop should be able to cope with these false constraints edges. How this goal was achieved will become clear in the next chapter. It contains the main contribution of my work and proposes a novel approach for robust optimization-based SLAM that can cope with outliers like false positive loop closure constraints.

5
A Robust Back-End for SLAM

In the previous chapter we saw that false positive loop closure constraints are a severe problem. They corrupt the pose graph formulation of the SLAM problem with erroneous edges, leading to a topologically incorrect graph. If we accept that data association errors, false positive place recognitions or false positive landmark observations will inevitably occur, the back-ends and thus the optimization itself has to become robust against these outliers. Current back-ends however are *not* robust and this is widely accepted in the literature as we saw in the previous chapter. Although robust cost functions like the Huber function can help to reduce the influence of these outlier edges, they do not resolve problem completely. Thus current solvers that rely on the front-end to produce a correct graph are doomed to converge towards a defective solution in the presence of outliers.

Since false positive loop closures are expressed as additional constraint edges in the factor graph representation, my main idea to increase the robustness of SLAM back-ends is that **the topology of the graph should be subject to the optimization instead of keeping it fixed.** This idea is illustrated in Fig. 5.1. If constraint edges representing outliers and data association errors could be identified and removed during the optimization process, the graph topology would be corrected and the optimization could converge towards a correct solution.

Now we have an augmented optimization problem: We do not only seek the optimal configuration X^* of robot poses, but at the same time we also seek the optimal topology of the constraint graph. With *optimal* topology I mean that it is free of outlier edges.

Optimizing the topology of the graph means to alter the optimization problem during the optimization process, which appears to be a "chicken or the egg" type of problem. How can the optimizer change the formulation of the optimization problem it is solving? This seems too complex and too costly to be feasible at first. Is it even possible that the problem formulation can be changed by the optimizer?

It is clear that (apart from exactly *how* it can be done at all, mathematically) we

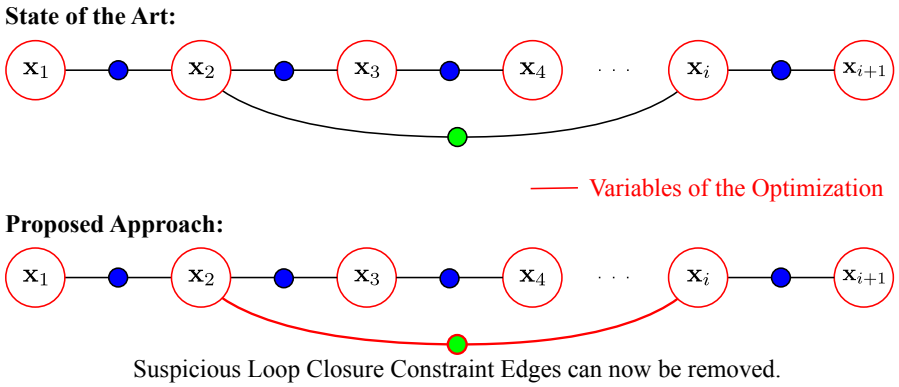


Figure 5.1: *In current state of the art approaches, only the robot poses $\mathbf{x}_i$ are variables in the optimization. I propose to augment the problem and also make the loop closure constraint edges subject to the optimization. In my proposed approach, the edges that are suspicious of being outliers, thus representing data association errors, can be removed during the optimization.*

can not allow the optimizer to change or reformulate the optimization problem totally randomly. What we want to achieve is that suspicious edges representing data association errors or outliers can be *removed* from the graph. So we can limit the operations the optimizer can conduct on the graph representation of its problem: We only allow to *remove* existing edges. No other operations are permitted, especially not to add new edges or to add or remove any of the vertices. With this limitation, the approach may now be in reach of possibility, as we are going to see.

5.1 The Robustified Formulation for Pose Graph SLAM

With my general idea phrased informally above, I am now going to transform it into a mathematical formulation. I will do so step-by-step, explaining my considerations on the way. We will finally arrive at a mathematically sound robustified formulation for the pose graph SLAM problem that is ready to be applied with the available optimization frameworks.

The main challenge is to enable the optimizer to *remove* constraint edges from the graph. As the constraint graph is merely a *representation* of the optimization problem, but not in any way a tool to solve it, we must concentrate on the mathematical relations and probabilistic constraints that constitute the optimization problem. I will, however, continue to use the graph representation to visualize certain steps and ideas.

5.1.1 First Steps Towards a Mathematical Formulation

Removing an edge from the factor graph corresponds to disabling the constraint associated with that edge. As we recall from (2.25), a loop closure constraint between poses $\mathbf{x}_i$ and

$\mathbf{x}_j$ is expressed by the squared Mahalanobis distance

$$\|\mathbf{e}_{ij}^{\text{lc}}\|^2_{\mathbf{\Lambda}_{ij}} = \|f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j\|^2_{\mathbf{\Lambda}_{ij}} \tag{5.1}$$

Together with the odometry constraints, it is part of the cost function that is to be minimized:

$$X^* = \underset{X}{\operatorname{argmin}} \underbrace{\sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i}}_{\text{Odometry Constraints}} + \underbrace{\sum_{ij} \|f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j\|^2_{\mathbf{\Lambda}_{ij}}}_{\text{Loop Closure Constraints}} \tag{5.2}$$

Given that formulation, how can we disable any of the loop closure constraints? To *disable* here means that the disabled constraint should not have any influence on the optimization process, it should be completely removed from the problem formulation. A binary weight factor ω_{ij} would allow us to do just that: It could disable or enable its associated constraint if $\omega_{ij} \in \{0, 1\}$, i.e. ω_{ij} is either 0 or 1:

$$X^* = \underset{X}{\operatorname{argmin}} \sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i} + \sum_{ij} \omega_{ij} \cdot \|f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j\|^2_{\mathbf{\Lambda}_{ij}} \tag{5.3}$$

Equally we can draw the weight into the squared Mahalanobis distance and write

$$X^* = \underset{X}{\operatorname{argmin}} \sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i} + \sum_{ij} \|\omega_{ij} \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)\|^2_{\mathbf{\Lambda}_{ij}} \tag{5.4}$$

Fig. 5.2 illustrates the idea. If the weight ω_{ij} is 1, the associated constraint is fully respected in the optimization process. In contrast, if the weight is 0, the constraint is completely ignored in the optimization and has no influence on the optimization result, as if it would not exist at all.

The current state of the art approaches correspond to the case where the weights are constant and fixed to $\omega_{ij} = 1$. However, if these weights were not fixed, but were themselves subject to the optimization and could be changed by the optimizer during the optimization process, we would in principle have achieved the desired behaviour: The topology of the constraint graph would be subject to the optimization process.

As the weights ω_{ij} shall not be fixed but subject to the optimization, they have to be *variables* of the optimization problem, just like the unknown robot poses $\mathbf{x}_i$ are variables. Since the weights shall either enable or completely disable their associated loop closure constraint, the domain of the weights should be the set $\{0, 1\}$. However, such discrete variables are not suited for our least squares optimization methods, which require continuous domains.

This thought will be developed further in the following. Meanwhile, we can summarize this section and declare:

Result 5.1. *The topology of the graph representation of the pose graph SLAM problem can be influenced by associating weight factors with each of the loop closure constraints. These weight factors can disable their associated constraint, which is equivalent to removing the corresponding edge from the graph.*

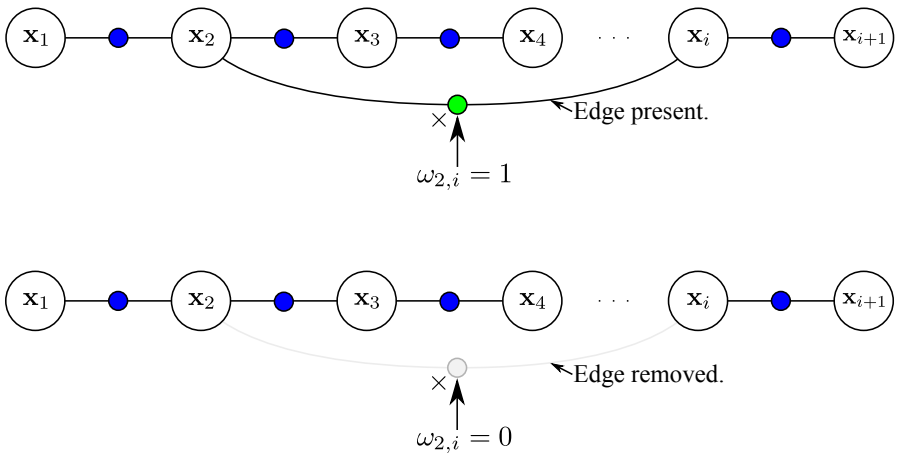


Figure 5.2: *A first idea how to disable certain constraint edges during the optimization: A binary weight $\omega_{ij} \in \{0, 1\}$ is associated with each loop closure constraint. Depending on the value of ω_{ij}, the constraint is either left untouched (top) or is disabled or removed (bottom). If those weights are modelled as variables in the optimization problem, the constraints could be disabled as part of the optimization.*

5.1.2 Introducing the Switch Variables and Finding a Suitable Switch Function

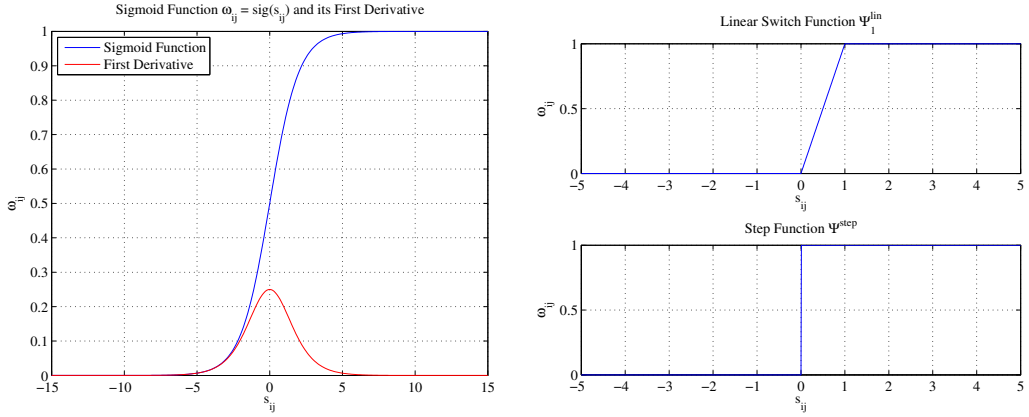
Instead of using the discrete weights themselves as variables in the optimization problem, we introduce a continuous variable $s_{ij} \in \mathbb{R}$ for each weight ω_{ij}. I call the set $S = \{s_{ij}\}$ the *switch variables*. We furthermore need a *switch function*

$$\omega_{ij} = \Psi(s_{ij}) : \mathbb{R} \to \{0, 1\} \tag{5.5}$$

that maps the continuous inputs s_{ij} to the desired weights $\omega_{ij} \in \{0, 1\}$. The s_{ij} would then be the variables in the optimization problem. A possible suitable function could be the step function

$$\omega_{ij} = \Psi^{\text{step}}(s_{ij}) : \mathbb{R} \to \{0, 1\} = \begin{cases} 0 & : s_{ij} < 0 \\ 1 & : s_{ij} \geq 0 \end{cases} \tag{5.6}$$

This function would allow to use real valued continuous variables s_{ij} that the optimizer could handle well. It would also implement the desired switching behaviour since the function's results ω_{ij} are either 0 or 1. However, the step function is not continuously differentiable since it has a discontinuity at $s_{ij} = 0$. Since iterative optimization algorithms like Levenberg-Marquardt use gradient information to converge towards the solution, continuously once-differentiable cost functions are required to ensure stable convergence behaviour.



(a) (b)

Figure 5.3: *Three possible choices for the switch function Ψ: (a) The sigmoid function and its first derivative. (b) The linear function Ψ_1^{lin} and the step function. See the text for further explanation and function definitions.*

A better solution is to use the sigmoid function for Ψ:

$$\omega_{ij} = \Psi^{\text{sigmoid}}(s_{ij}) = \text{sig}(s_{ij}) : \mathbb{R} \rightarrow (0,1) = \frac{1}{1+e^{-s_{ij}}} \tag{5.7}$$

which is continuously differentiable with derivation

$$\text{sig}'(s_{ij}) = \text{sig}(s_{ij}) \cdot (1 - \text{sig}(s_{ij})) \tag{5.8}$$

The sigmoid function and its derivative are illustrated in Fig. 5.3(a). Notice that the sigmoid function asymptotically converges towards 0 and 1 but never exactly reaches those values:

$$0 < sig(s_{ij}) < 1 \; \forall s_{ij} \in \mathbb{R} \tag{5.9}$$

During the first experiments with the proposed robust back-end, the sigmoid function was used. This was also reported in [Sünderhauf and Protzel, 2011]. A later evaluation however revealed, that a simple piecewise linear function of the form

$$\omega_{ij} = \Psi_a^{\text{lin}}(s_{ij}) : \mathbb{R} \rightarrow [0,1] = \begin{cases} 0 & : s_{ij} < 0 \\ \frac{1}{a} s_{ij} & : 0 \leq s_{ij} \leq a \\ 1 & : s_{ij} > a \end{cases} \tag{5.10}$$

with parameter $a = 1$ (see Fig. 5.3(b)) is superior to the sigmoid function. We will come back to that point later during the evaluation in section 6.8. For now the important point is that a suitable switch function Ψ has to be chosen to map the continuous real-valued switch variables s_{ij} to the appropriate weights ω_{ij}.

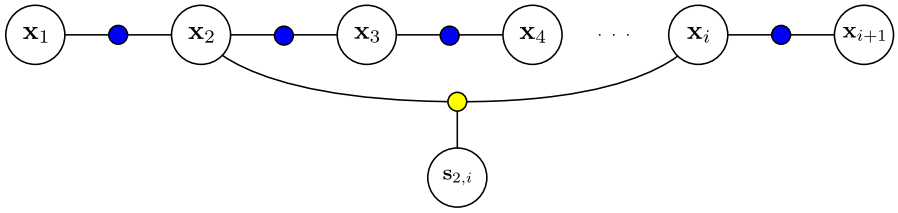


Figure 5.4: *Factor graph representation of the augmented pose graph SLAM problem after introducing the switch variables. Notice how the additional switch variable $s_{2,i}$ governs the loop closure factor (now shown in yellow). Depending on the value assigned to the switch variable s_{ij}, the loop closure factor is "switched" on or off, i.e. it is activated or deactivated as part of the optimization process.*

We are now ready to augment the optimization problem (5.2) and introduce the new *switch variables* $S = \{s_{ij}\}$ into the cost function:

$$\begin{aligned} X^*, S^* &= \underset{X,S}{\operatorname{argmin}} \sum_i \|\mathbf{e}_i^{\text{odo}}\|^2_{\mathbf{\Sigma}_i} + \sum_{ij} \|\mathbf{e}_{ij}^{\text{slc}}\|^2_{\mathbf{\Lambda}_{ij}} \\ &= \underset{X,S}{\operatorname{argmin}} \underbrace{\sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i}}_{\text{Odometry Constraints}} + \underbrace{\sum_{ij} \|\Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)\|^2_{\mathbf{\Lambda}_{ij}}}_{\text{Switched Loop Closure Constraints}} \end{aligned} \tag{5.11}$$

As laid out above, the function Ψ now acts as a switch on the loop closure constraints, which I therefore call *switched* loop closure constraints. Notice that we now optimize over two sets of hidden variables, the robot poses $X = \{\mathbf{x}_i\}$ and the switch variables $S = \{s_{ij}\}$.

Fig. 5.4 illustrates the augmented problem formulation of (5.11). Compared to the original factor graph representation in Fig. 2.19, we now have an additional type of vertices which represent the switch variables s_{ij}. The loop closure factors now connect three (instead of two) vertices.

To summarize the central idea developed in this section I want to state the following result:

Result 5.2. *The topology of the factor graph can be made subject to the optimization by introducing additional hidden variables into the problem formulation. Each of these switch variables controls the value of a weight via a switch function. The weights act on the loop closure constraints which are practically removed from the problem formulation when the associated weight factor is* 0*. This corresponds to removing the associated constraint edge from the factor graph.*

5.1.3 Introducing the Switch Priors

Like all other variables, the switch variables must be initialized before the optimization starts. What would be a reasonable initial value? Remember that the loop closure constraints were proposed or requested by the front-end. Some of them might be false-positive loop closures and we want to identify and disable those. It is therefore reasonable to initially accept all loop closure constraints, i.e. letting $\omega_{ij} = \Psi(s_{ij}) \approx 1$, or equal to 1, depending on the actual switch function chosen. For Ψ^{lin}, a proper and convenient initial value for all switch variables would therefore be $s_{ij} = 1$. For the following, I call these initial values γ_{ij} and collect them in the set $\Gamma = \{\gamma_{ij}\}$.

Notice that although it is intuitively right to initialize $s_{ij} = \gamma_{ij}$ with γ_{ij} so that $\Psi(\gamma_{ij}) \approx 1$, any other value could be used in principle. The exact value for γ_{ij} could be chosen problem-dependent, set to a fixed value for all γ_{ij}, or provided individually for each γ_{ij} by the front-end. This could be used to model a fuzzy characteristic of "loop-closureness", e.g. the place recognition module in the front-end could express a degree of belief or element of doubt on whether the loop closure actually exists or if it might be a false-positive.

Like any other variable or observation in our probabilistic framework[33], the switch variables s_{ij} are modelled as normally distributed Gaussian variables. The initial value is used as mean of the distribution:

$$s_{ij} \sim \mathcal{N}\left(\gamma_{ij}, \boldsymbol{\Xi}_{ij}\right) \tag{5.12}$$

The initial values γ_{ij} (that can be understood as observations of their switch variable) have to be incorporated into the optimization-based framework using *prior constraints* or *prior factors*. These prior factors express the (trivial) problem of maximizing $P(S|\Gamma)$:

$$\begin{aligned} S^* &= \underset{S}{\operatorname{argmax}}\, P(S|\Gamma) \\ &= \underset{S}{\operatorname{argmin}} \sum_{ij} \|\gamma_{ij} - s_{ij}\|^2_{\boldsymbol{\Xi}_{ij}} \\ &= \underset{S}{\operatorname{argmin}} \sum_{ij} \|\mathbf{e}^{\text{sp}}_{ij}\|^2_{\boldsymbol{\Xi}_{ij}} \end{aligned} \tag{5.13}$$

The solution to this sub-problem obviously is $s_{ij} = \gamma_{ij}\ \forall s_{ij} \in S$. Notice however, that the switch variables are also involved in the error term $\mathbf{e}^{\text{slc}}_{ij}$ in (5.11), so that on a global scale the optimal solution may be $s_{ij} \neq \gamma_{ij}$ for some s_{ij}, namely exactly those s_{ij} associated with a false loop closure constraint.

Like the initial value γ_{ij}, the covariance $\boldsymbol{\Xi}_{ij}$ can be provided by the front-end and would express its *confidence* in the correctness of the loop closure constraint associated with s_{ij}: Deactivating the constraint, thus driving s_{ij} away from its initial value γ_{ij}, would result in a higher penalty $\mathbf{e}^{\text{sp}}_{ij}$, the smaller the covariance in $\boldsymbol{\Xi}_{ij}$ is. Together with γ_{ij}, the

[33] remember from (2.15) that $\mathbf{x}_{i+1} \sim \mathcal{N}\left(f(\mathbf{x}_i, \mathbf{u}_i), \boldsymbol{\Sigma}_i\right)$ and according to (2.16) $\mathbf{x}_j \sim \mathcal{N}\left(f(\mathbf{x}_i, \mathbf{u}_{ij}), \boldsymbol{\Lambda}_{ij}\right)$

front-end now has two mechanisms to fine-tune the behaviour of the back-end system and to incorporate additional information like degree of belief, doubt, and confidence[34].

Apart from the probabilistic justification, the importance of the switch prior constraint can also be understood intuitively: The optimal solution to the problem (5.11) would be reached by deactivating all loop closure constraints (by driving their switch variables s_{ij} towards values so that $\Psi(s_{ij}) \approx 0$) and by arranging the robot poses according to the odometry measurements. This would even lead to an overall error of zero, since all loop closure constraints are ignored and cannot contribute to the overall cost function, hence cannot influence the solution of the sought robot poses X^*. This is of course not desirable and can be easily avoided if we would, in simple words, penalize the optimizer whenever it tries to disable a loop closure constraint. That penalty should therefore be proportional to the difference between s_{ij} and its initial value γ_{ij}, which is exactly what the switch prior constraint introduced above provides.

Result 5.3. *Switch prior factors anchor the switch variables s_{ij} at their initial values γ_{ij}. These additional factors are necessary to prevent the optimizer from deactivating all loop closure constraints. Like all other constraints, the switch priors are modelled as Gaussians with covariance Ξ_{ij}.*

5.1.4 Putting it all Together

We can now add the switch prior constraint to the problem formulation of (5.11) to arrive at the final robust optimization problem for pose graph SLAM:

$$\begin{aligned} X^*, S^* = \underset{X,S}{\operatorname{argmin}} & \underbrace{\sum_i \| f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1} \|^2_{\mathbf{\Sigma}_i}}_{\text{Odometry Constraints}} \\ & + \underbrace{\sum_{ij} \| \Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j) \|^2_{\mathbf{\Lambda}_{ij}}}_{\text{Switched Loop Closure Constraints}} \\ & + \underbrace{\sum_{i,j} \| \gamma_{ij} - s_{ij} \|^2_{\mathbf{\Xi}_{ij}}}_{\text{Switch Prior Constraints}} \end{aligned} \tag{5.14}$$

or shorter:

$$X^*, S^* = \underset{X,S}{\operatorname{argmin}} \sum_i \| \mathbf{e}_i^{\text{odo}} \|^2_{\mathbf{\Sigma}_i} + \sum_{ij} \| \mathbf{e}_{ij}^{\text{slc}} \|^2_{\mathbf{\Lambda}_{ij}} + \sum_{ij} \| \mathbf{e}_{ij}^{\text{sp}} \|^2_{\mathbf{\Xi}_{ij}} \tag{5.15}$$

Above formula (5.14) constitutes my proposed robust problem formulation for pose graph SLAM. It is an optimization problem over two sets of hidden variables, the robot poses $X = \{\mathbf{x}_i\}$ and the switch variables $S = \{s_{ij}\}$, and consists of three types of

[34] Developing front-ends that can create these measures and evaluating how they can influence the performance of the overall robust SLAM system is an interesting challenge for future research.

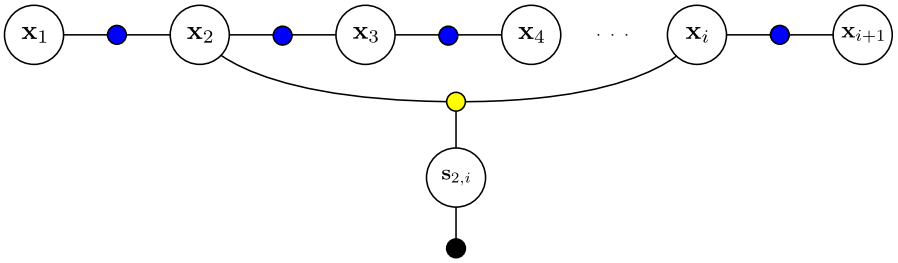


Figure 5.5: *Factor graph representation of the proposed robust back-end. The optimization problem now consists of two types of hidden variables (represented by the large nodes) and three types of constraints (represented by edges with small nodes). The hidden, unobserved variables are the robot poses $\mathbf{x}_i$ and the switch variables s_{ij}. The loop closure constraints (yellow) between two poses connect three variables (in contrast to only two variables in the state of the art approaches). Each switch variable s_{ij} is governed by a prior factor that penalizes the deactivation of loop closures.*

constraints that incorporate the observations into the system. The odometry constraints equal those in the standard problem formulation, and represent the pose-to-pose motion information. The loop closure constraints are now constraints between three variables ($\mathbf{x}_i, \mathbf{x}_j$, and s_{ij}) and are weighted by the variable weight factor $\omega_{ij} = \Psi(s_{ij})$. This way, by driving the switch variable to small values, the optimization procedure can deactivate the associated loop closure constraints. The third type of constraints, the switch prior constraints, penalize the deactivation of loop closure constraints.

Fig. 5.5 illustrates the robustified factor graph, showing the new switch variable along with its prior factor.

Result 5.4. *I proposed a problem formulation that allows the optimizer to change the topology of the factor graph representing the pose graph SLAM problem. Individual loop closure constraint edges can be removed from the problem by driving the associated weight towards 0. The behaviour of the system can be influenced by the actual choice of switch function, switch priors and switch prior covariance.*

5.2 Discussion

After an approach that allows the optimizer to change the problem topology during the optimization has been proposed in the previous section, the proposal has to be discussed more elaborately. This section is dedicated to certain details that help to understand the mechanisms behind the approach and prepares the evaluation in Chapter 6.

5.2.1 The Influence of s_{ij} on the Information Matrix $\mathbf{\Lambda}_{ij}$

Remember from (5.14) that each switched loop closure constraint contributes to the overall cost function with the term

$$\|\Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)\|^2_{\mathbf{\Lambda}_{ij}} = \|\mathbf{e}^{\text{slc}}_{ij}\|^2_{\mathbf{\Lambda}_{ij}} \tag{5.16}$$

We arrived at this formulation after considerations that the optimizer should be able to disable any loop closure constraint: By driving s_{ij} towards a value so that $\Psi(s_{ij}) \approx 0$, the optimizer can "switch off" the associated loop closure constraint, because in this case $\|\Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)\|^2_{\mathbf{\Lambda}_{ij}} = 0$ and the difference between $f(\mathbf{x}_i, \mathbf{u}_{ij})$ and $\mathbf{x}_j$ does not add to the global error terms.

Although this interpretation is easy to understand intuitively, the effect of the switch variable can also be understood as acting upon the entries of the information matrix $\mathbf{\Lambda}^{-1}_{ij}$ that is associated with the loop closure constraint via the squared Mahalanobis distance $\|\cdot\|^2_{\mathbf{\Lambda}_{ij}}$. This second possible interpretation will help us with the probabilistic analysis of the switched loop closure factors. As a first step, I will therefore show the influence of the switch variables s_{ij} on the information matrix used in the Mahalanobis distance.

Starting from (5.16) and using the definition of the Mahalanobis distance we can write

$$\|\mathbf{e}^{\text{slc}}_{ij}\|^2_{\mathbf{\Lambda}_{ij}} = [\Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)]^{\mathsf{T}} \, \mathbf{\Lambda}^{-1}_{ij} \, [\Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)] \tag{5.17}$$

or more simple, if we set $f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j = \boldsymbol{\delta}_{ij}$:

$$\|\mathbf{e}^{\text{slc}}_{ij}\|^2_{\mathbf{\Lambda}_{ij}} = [\Psi(s_{ij}) \cdot \boldsymbol{\delta}_{ij}]^{\mathsf{T}} \, \mathbf{\Lambda}^{-1}_{ij} \, [\Psi(s_{ij}) \cdot \boldsymbol{\delta}_{ij}] \tag{5.18}$$

Using the fact that $\Psi(s_{ij})$ is scalar, we can apply the transpose and get

$$\|\mathbf{e}^{\text{slc}}_{ij}\|^2_{\mathbf{\Lambda}_{ij}} = \left[\Psi(s_{ij}) \cdot \boldsymbol{\delta}^{\mathsf{T}}_{ij}\right] \mathbf{\Lambda}^{-1}_{ij} \, [\Psi(s_{ij}) \cdot \boldsymbol{\delta}_{ij}] \tag{5.19}$$

Now we can separate the scalar switch function terms Ψ and write

$$\|\mathbf{e}^{\text{slc}}_{ij}\|^2_{\mathbf{\Lambda}_{ij}} = \Psi(s_{ij})^2 \cdot \left[\boldsymbol{\delta}^{\mathsf{T}}_{ij} \mathbf{\Lambda}^{-1}_{ij} \boldsymbol{\delta}_{ij}\right] \tag{5.20}$$

or equally

$$\|\mathbf{e}^{\text{slc}}_{ij}\|^2_{\mathbf{\Lambda}_{ij}} = \boldsymbol{\delta}^{\mathsf{T}}_{ij} \left[\Psi(s_{ij})^2 \mathbf{\Lambda}^{-1}_{ij}\right] \boldsymbol{\delta}_{ij} \tag{5.21}$$

This last formulation is interesting, because we see that the switch variables s_{ij} directly influence the resulting information matrices

$$\Psi(s_{ij})^2 \cdot \mathbf{\Lambda}^{-1}_{ij} = \mathbf{\Phi}^{-1}_{ij} \tag{5.22}$$

In this interpretation, if the weight $\omega_{ij} = \Psi(s_{ij})$ is driven towards a small value close to zero during the optimization, the resulting information matrix $\mathbf{\Phi}^{-1}_{ij}$ will be close to zero too. In other words, the associated uncertainty expressed in the covariance matrix $\mathbf{\Phi}_{ij}$ approaches infinity. This however, informally expresses that the associated

constraint is to be ignored in the optimization process, because literally nothing is known about it. Both interpretations, driving the information measure or the resulting error towards zero, topologically correspond to removing the associated edge from the graph that represents the optimization problem. However, the interpretation where the switch variables influence the information matrix is to be preferred, as it allows us to still consider the augmented loop closure constraints as Gaussian. This will become clear in the following.

Remember that, according to the (state of the art) loop closure constraint, $\mathbf{x}_j$ follows a normal distribution with mean $f(\mathbf{x}_i, \mathbf{u}_{ij})$ and covariance $\mathbf{\Lambda}_{ij}$:

$$\mathbf{x}_j \sim \mathcal{N}\left(f(\mathbf{x}_i, \mathbf{u}_{ij}), \mathbf{\Lambda}_{ij}\right) \tag{5.23}$$

For the proposed novel switched constraint, the mean of the normal distribution for $\mathbf{x}_j$ is the same as in the state of the art formulation. The associated covariance matrix however is given by

$$\begin{aligned} \mathbf{\Phi}_{ij} &= \left(\mathbf{\Phi}_{ij}^{-1}\right)^{-1} \\ &= \left(\Psi(s_{ij})^2 \cdot \mathbf{\Lambda}_{ij}^{-1}\right)^{-1} \\ &= \frac{1}{\omega_{ij}^2}\mathbf{\Lambda}_{ij} \end{aligned} \tag{5.24}$$

So for the augmented constraint, $\mathbf{x}_j$ is normally distributed according to

$$\mathbf{x}_j \sim \mathcal{N}\left(f(\mathbf{x}_i, \mathbf{u}_{ij}), \frac{1}{\omega_{ij}^2}\mathbf{\Lambda}_{ij}\right) \tag{5.25}$$

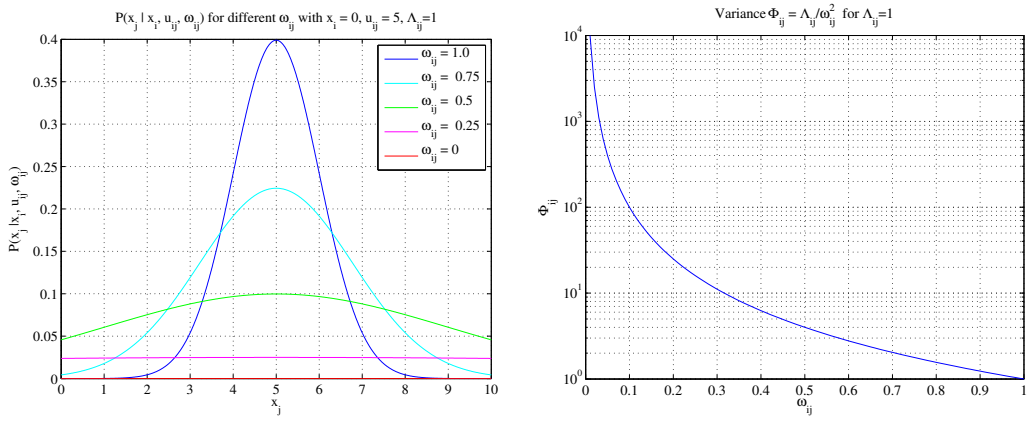
which can be expressed in the conditional probability $P(\mathbf{x}_j | \mathbf{x}_i, \mathbf{u}_{ij}, s_{ij})$.

Fig. 5.6 illustrates this distribution for different values of ω_{ij}. As we can see in this example, when the loop closure constraint is activated, i.e. $\omega_{ij} \approx 1$, $\mathbf{x}_j$ follows a Gaussian distribution with the mean at $f(\mathbf{x}_i, \mathbf{u}_{ij}) = 5$ and variance $\mathbf{\Lambda}_{ij} = 1$ (blue curve). When the constraint is deactivated because $\omega_{ij} \approx 0$, we see that $\mathbf{x}_j$ approaches a uniform distribution over its complete domain[35], as indicated by the red curve. In other words, the resulting covariance $\mathbf{\Phi}_{ij} = 1/\omega_{ij}^2 \mathbf{\Lambda}_{ij}$ quickly approaches infinity as ω_{ij} approaches 0.

Result 5.5. *The proposed switch variables influence the information matrices of the loop closure constraints via the switch function. This way, the loop closure information matrices can be driven away from their initial values and towards zero, which expresses that the associated constraint provides no information and thus does not influence the result of the overall problem optimization.*

This probabilistic interpretation is equivalent to the previous topological interpretation where the constraint edge was considered to be removed from the factor graph representation.

[35] The domain of $\mathbf{x}$ in this small example is $\mathbb{R}$. For real pose graph SLAM applications it will rather be $SE(2)$ for the 2D case or even $SE(3)$ for the 3D case.



(a) (b)

Figure 5.6: *The weight factor $\omega_{ij} = \Psi(s_{ij})$ influences the covariance $\mathbf{\Phi}_{ij}$ associated with $\mathbf{x}_j$ via the switched loop closure constraint. As the weight ω_{ij} approaches 0, the resulting covariance $\mathbf{\Phi}_{ij}$ quickly deviates from its initial value 1 towards ∞. Notice the log-scale in (b).*

5.2.2 Establishing the Connection to the Maximum a Posteriori Solution

Notice that the robust problem formulation in (5.14) was given in terms of a cost function that is to be minimized in order to find the optimal variable configuration of X^* and S^*. We now have to analyze the constraints or factors from a probabilistic perspective, as we have to make sure that the optimal configuration of X^* and S^* corresponds to the maximum a posteriori solution.

The factor graph in Fig. 5.5 represents a factorization of the joint probability over all hidden variables (robot poses X, and switch variables S) and observations (switch priors Γ and odometry measurements U):

$$P(X, S|\Gamma, U) \propto \underbrace{\prod_i P(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i)}_{\text{Odometry Factors}} \cdot \underbrace{\prod_{ij} P(\mathbf{x}_j|\mathbf{x}_i, \mathbf{u}_{ij}, s_{ij})}_{\text{Switched Loop Closure Factors}} \cdot \underbrace{\prod_{ij} P(s_{ij}|\gamma_{ij})}_{\text{Switch Prior Factors}} \tag{5.26}$$

The maximum a posteriori estimate of this quantity is by definition

$$X^*, S^* = \arg\max_{X,S} P(X, S|\Gamma, U) \tag{5.27}$$

$$= \operatorname*{argmin}_{X,S} -\log P(X, S|\Gamma, U) \tag{5.28}$$

$$= \operatorname*{argmin}_{X,S} -\log \left(\prod_i P(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i) \cdot \prod_{ij} P(\mathbf{x}_j|\mathbf{x}_i, \mathbf{u}_{ij}, s_{ij}) \cdot \prod_{ij} P(s_{ij}|\gamma_{ij}) \right) \tag{5.29}$$

We assumed all of these conditional probability distributions are Gaussians, therefore:

$$\begin{aligned} P(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i) &\sim \mathcal{N}\left(f(\mathbf{x}_i, \mathbf{u}_i), \mathbf{\Sigma}_i\right) \\ &\propto \exp -\frac{1}{2}\|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i} \end{aligned} \tag{5.30}$$

$$\begin{aligned} P(\mathbf{x}_j|\mathbf{x}_i, \mathbf{u}_{ij}, s_{ij}) &\sim \mathcal{N}\left(f(\mathbf{x}_i, \mathbf{u}_{ij}), \frac{1}{\Psi(s_{ij})^2}\mathbf{\Lambda}_{ij}\right) \\ &\propto \exp -\frac{1}{2}\|\Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)\|^2_{\mathbf{\Lambda}_{ij}} \end{aligned} \tag{5.31}$$

$$\begin{aligned} P(s_{ij}|\gamma_{ij}) &\sim \mathcal{N}\left(\gamma_{ij}, \mathbf{\Xi}_{ij}\right) \\ &\propto \exp -\frac{1}{2}\|s_{ij} - \gamma_{ij}\|^2_{\Xi_{ij}} \end{aligned} \tag{5.32}$$

We insert these quantities into the products and continue with

$$\begin{aligned} X^*, S^* = \underset{X,S}{\operatorname{argmin}} - \log\Bigg(&\prod_i \exp -\frac{1}{2}\|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i} \\ &\cdot \prod_{ij} \exp -\frac{1}{2}\|\Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)\|^2_{\mathbf{\Lambda}_{ij}} \\ &\cdot \prod_{ij} \exp -\frac{1}{2}\|s_{ij} - \gamma_{ij}\|^2_{\Xi_{ij}} \Bigg) \end{aligned} \tag{5.33}$$

After applying the log inside the brackets we gain

$$\begin{aligned} X^*, S^* = \underset{X,S}{\operatorname{argmin}} -\frac{1}{2}\Bigg(&\sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i} \\ &+ \sum_{ij} \|\Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)\|^2_{\mathbf{\Lambda}_{ij}} \\ &+ \sum_{ij} \|s_{ij} - \gamma_{ij}\|^2_{\Xi_{ij}} \Bigg) \end{aligned} \tag{5.34}$$

The constant factor $-\frac{1}{2}$ has no influence on the result (X^*, S^*) and can therefore be removed:

$$\begin{aligned} X^*, S^* = \underset{X,S}{\operatorname{argmin}} &\sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i} \\ &+ \sum_{ij} \|\Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)\|^2_{\mathbf{\Lambda}_{ij}} \\ &+ \sum_{ij} \|s_{ij} - \gamma_{ij}\|^2_{\Xi_{ij}} \end{aligned} \tag{5.35}$$

This however is equal to (5.14). We therefore conclude:

Result 5.6. *The solution (X^*, S^*) to the nonlinear optimization problem (5.14) maximizes $P(X, S|\Gamma, U)$ and thus (X^*, S^*) is the maximum a posteriori solution.*

5.2.3 The Influence of the Additional Variables and Constraints on the Problem Size

Obviously, the proposed robustified problem formulation significantly enlarges the original optimization problem.

Result 5.7. *For each of the m loop closure constraint present in the original problem, another variable and an associated prior factor are added in the robustified problem formulation. The problem size thus increases linearly in the size of m.*

However, given today's efficient solvers that exploit the sparseness of the optimization problem, the *size* of the problem (i.e. the number of variables and constraints) is not the most crucial factor that determines the runtime behaviour. By far more important are the sparse structure of the system's Jacobian and a beneficial convergence behaviour (e.g. convexity or close-convexity of the problem).

The evaluation in Chapter 6 will provide some details on how the additional variables and constraints change the required convergence time for a number of datasets. As expected, the convergence time is higher for the robust back-end when compared to state of the art formulations (but of course this is a small price to pay when it means to gain correct results even in the presence of outliers).

5.2.4 The Influence of the Additional Variables and Constraints on the Sparse Structure of the Problem

In section 3.5 we discussed that the sparseness of the Jacobian of the system's error function is crucial and determines the feasibility of the optimization problem. Only problems with a sparse Jacobian can be solved efficiently if the number of variables and constraints is large. Large but densely occupied problems are infeasible and cannot be solved in reasonable time with current solvers.

We have seen in Fig. 3.9, that the Jacobian and the Hessian of the state of the art formulation to pose graph SLAM are indeed sparse. The question now is whether the newly introduced switch variables and switch prior constraints or the alterations made to the loop closure constraints prevent a sparse structure of the Jacobian and the resulting Hessian. I will proceed analyzing the Jacobians of the switched loop closure constraint and the switch prior constraint. Since the odometry constraint remains unchanged, we do not have to revisit its Jacobian. It suffices to recall from section 3.5 that its structure is indeed sparse.

The Jacobian of the Switched Loop Closure Constraint

If a switched loop closure constraint contributes with

$$\|\Psi(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)\|^2_{\mathbf{\Lambda}_{ij}} = \|\mathbf{e}^{\text{slc}}_{ij}\|^2_{\mathbf{\Lambda}_{ij}} \tag{5.36}$$

to the global cost function, then the Jacobian for that constraint is formed by the partial derivatives with respect to the variables $\mathbf{x}$ and $\mathbf{s}$:

$$\mathbf{J}_{ij}^{\text{slc}} = \left(\frac{\partial \mathbf{e}_{ij}^{\text{slc}}}{\partial \mathbf{x}} \; \frac{\partial \mathbf{e}_{ij}^{\text{slc}}}{\partial \mathbf{s}} \right) \tag{5.37}$$

Notice that the only non-zero entries in that Jacobian are the partial derivatives with respect to $\mathbf{x}_i$, $\mathbf{x}_j$, and s_{ij}. All other entries are $\mathbf{0}$, thus the Jacobians are very sparse:

$$\mathbf{J}_{ij}^{\text{slc}} = \left(\mathbf{0} \dots \mathbf{0}, \frac{\partial \mathbf{e}_{ij}^{\text{slc}}}{\partial \mathbf{x}_i}, \mathbf{0} \dots \mathbf{0}, \frac{\partial \mathbf{e}_{ij}^{\text{slc}}}{\partial \mathbf{x}_j}, \mathbf{0} \dots \mathbf{0}, \frac{\partial \mathbf{e}_{ij}^{\text{slc}}}{\partial s_{ij}}, \mathbf{0} \dots \mathbf{0} \right) \tag{5.38}$$

with

$$\frac{\partial \mathbf{e}_{ij}^{\text{slc}}}{\partial \mathbf{x}_i} = \Psi(s_{ij}) \frac{\partial f}{\partial \mathbf{x}_i} \tag{5.39}$$

$$\frac{\partial \mathbf{e}_{ij}^{\text{slc}}}{\partial \mathbf{x}_j} = -\Psi(s_{ij}) \tag{5.40}$$

$$\frac{\partial \mathbf{e}_{ij}^{\text{slc}}}{\partial s_{ij}} = \frac{\partial \Psi(s_{ij})}{\partial s_{ij}} \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j) \tag{5.41}$$

The Jacobian of the Switch Prior Factor

Like above, the Jacobian for the switch prior constraint is given by the derivative of the error term $\mathbf{e}_{ij}^{\text{sp}}$ with respect to the pose and switch variables:

$$\mathbf{J}_{ij}^{\text{sp}} = \left(\frac{\partial \mathbf{e}_{ij}^{\text{sp}}}{\partial \mathbf{x}} \; \frac{\partial \mathbf{e}_{ij}^{\text{sp}}}{\partial \mathbf{s}} \right) \tag{5.42}$$

Since the error term is given by

$$\mathbf{e}_{ij}^{\text{sp}} = \gamma_{ij} - s_{ij} \tag{5.43}$$

only the partial derivative with respect to s_{ij} is non-zero:

$$\mathbf{J}_{ij}^{\text{sp}} = (0 \dots 0, -1, 0 \dots 0) \tag{5.44}$$

because

$$\frac{\partial \mathbf{e}_{ij}^{\text{sp}}}{\partial s_{ij}} = -1 \tag{5.45}$$

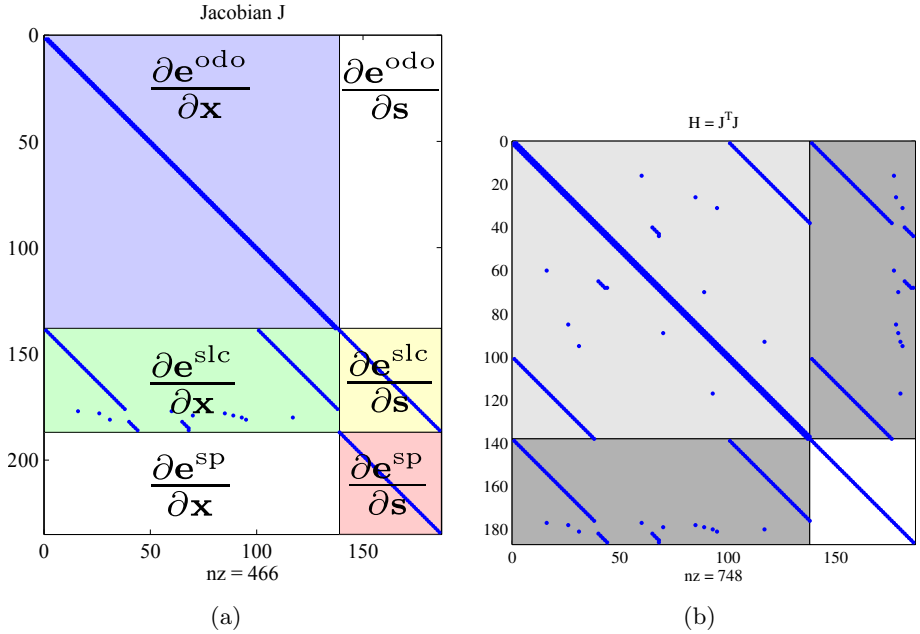


Figure 5.7: *Jacobian* $\mathbf{J}$ *and Hessian* $\mathbf{H} = \mathbf{J}^\top\mathbf{J}$ *for the example SLAM problem from Fig. 5.9, modelled using the proposed robust back-end. The non-zero values are marked by a blue point. All other entries in the matrices are zero, thus both matrices are very sparse. Notice the clear block structure of the sparse Jacobian in (a). The Hessian* $\mathbf{H}$ *in (b) is equivalent to the adjacency matrix of the Markov random field graph. The light grey block contains the connections between two pose nodes, while connections between a switch variable and a pose node are represented in the darker areas. Notice that* $\mathbf{H}$ *is symmetric.*

The overall Jacobian

The complete Jacobian of the system is formed by stacking the partial Jacobians of the three types of constraints:

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}^{\text{odo}} \\ \mathbf{J}^{\text{slc}} \\ \mathbf{J}^{\text{sp}} \end{pmatrix} \tag{5.46}$$

Since all partial Jacobians are sparse, the stacked system Jacobian is sparse as well. We can therefore conclude that the robust problem formulation's Jacobian has a sparse structure and the proposed extensions and changes to the problem formulation have no negative influence on the sparsity of the problem.

Result 5.8. *The overall Jacobian of the proposed robust problem formulation remains sparse. The resulting Hessian is sparse as well, which is expressed in the sparse intercon-*

nectivity of the problem's graph representation.

Fig. 5.7 illustrates this result on the example of the scenario provided in Fig. 5.9. Notice how several blocks are clearly visible in the Jacobian $\mathbf{J}$ in Fig. 5.7(a). These blocks contain the partial Jacobians of the odometry constraints, switched loop closure constraints, and switch prior constraints (from top to bottom) with respect to the robot poses and switch variables (left to right). The approximated Hessian $\mathbf{H} = \mathbf{J}^\mathsf{T}\mathbf{J}$ in Fig. 5.7(b) is equivalent to the adjacency matrix of the Markov random field and is naturally sparse as well. This is also intuitively clear because each of the switch variables governs only one loop closure edge. The same way, each of the switch variables is influenced by only one prior factor.

5.2.5 The Influence of the Additional Variables and Constraints on the Problem's Convergence Properties

While the influence of the newly proposed switch variables and constraints on the problem size and sparsity structure were determined in the previous two sections, their influence on the convergence behaviour during the optimization is much harder to determine.

First of all, a literature review shows that the convergence properties of the standard pose graph SLAM problem have not yet been exhaustively explored or even concluded: [Huang et al., 2010] raised the question of how far SLAM is from a linear (i.e. convex) least squares problem and suggested a *close-convexity* under certain circumstances, e.g. small initial angular deviations and spherical covariance matrices without off-diagonal entries. They also showed that a relative pose formulation helps to strengthen this close-to-convex property. Recently [Carlone et al., 2011a] presented a working closed-form, linear approximation to SLAM. In this formulation the SLAM problem, like all linear least squares problems, can be solved immediately without requiring an iterative solver or an initial guess. The prerequisite for this approach again are diagonal shaped covariances, with independent position and orientation measurements.

Result 5.9. *The very recent ongoing work shows that the convergence properties of the SLAM problem are still under research. It would be desirable to show that the robustified SLAM formulation proposed here does not negatively influence the convergence properties of the underlying standard SLAM problem. Although the results presented in the next chapter indicate a good convergence behaviour, the mathematical proof is still left for future work.*

5.3 Summary and a First Working Example

This chapter proposed to augment the least squares formulation of the pose graph SLAM problem by an additional kind of variable, the *switch variables*. Each of the variables is associated with a loop closure constraint and influences how that constraint is regarded in the optimization process. As we have seen, there are two equivalent interpretations of how the switch variables influence the optimization problem:

1. Topological Interpretation: The switch variables can *remove* their associated constraint edges from the factor graph representation of the SLAM problem.

2. Probabilistic Interpretation: The switch variables influence the information matrix of their associated constraint and can – in the limit – drive this information measure towards zero.

Either way, the augmented optimization problem formulation allows the optimizer to take back data associations conducted by the front end. It can thus remove the influence of some loop closure constraints and hence converge to a correct solution even in the presence of outliers. The *switch priors* penalize the removal or deactivation of a loop closure constraint and thus prevent the trivial solution of removing all loop closures.

The augmented problem formulation proposed here allows the SLAM back-end to become robust against false positive loop closure constraints and data association errors. A first implementation of the proposed system has been conducted for the GTSAM framework which was also used during the first experiments reported in [Sünderhauf and Protzel, 2011; Sünderhauf and Protzel, 2012]. However, the g^{2}o framework turned out to be more versatile later on and thus the robust approach was implemented for this framework as well.

Now that the general idea and the mathematical details of the robustified formulation for pose graph SLAM are laid out, its practical feasibility has to be proven and evaluated. I will begin with a small example dataset that demonstrates the general feasibility of the proposed approach. The next chapter will provide a more elaborate evaluation using different well-known datasets and Chapter 7 will demonstrate how the robust back-end performs on a large-scale real-world SLAM problem.

Example 5.1. *We already encountered the small synthetic dataset of this example in the previous chapter where it demonstrated the need for a robustified SLAM formulation. I am going to repeat the figures for the convenience of the reader: Fig. 5.9(a) shows the ground truth trajectory. The robot is driven in a squared trajectory, starting from* $(0, 0)$ *towards positive* x*, and revisiting the lower part of the trajectory. On its way, the front-end performs place recognition and correctly recognizes the loop closures in the revisited part of the environment. However, the place recognition fails a few times and introduces 10 false-positive loop closures. These are outliers because they connect positions in the world that do not correspond. The trajectory according to the noisy odometry sensor is visible in Fig. 5.9(b) along with the loop closures requested by the front-end. Feeding this problem definition into a state of the art SLAM back-end will result in a meaningless solution: Due to the erroneous loop closure constraints, state-of-the-art approaches to pose graph SLAM are not able to converge towards a correct solution. Fig. 5.9(c) illustrates the result of* g^{2}o *[Kümmerle et al., 2011b]. The false-positive loop closure requests force the optimization to converge towards a defective solution.*

My proposed robust back-end however is able to identify and disable the erroneous loop closure constraints: The result in Fig. 5.9(d) is the maximum a posteriori solution that maximizes the joint probability over all unknown robot poses and switch variables, given the observations. As one can clearly see, despite the wrong loop closures, the

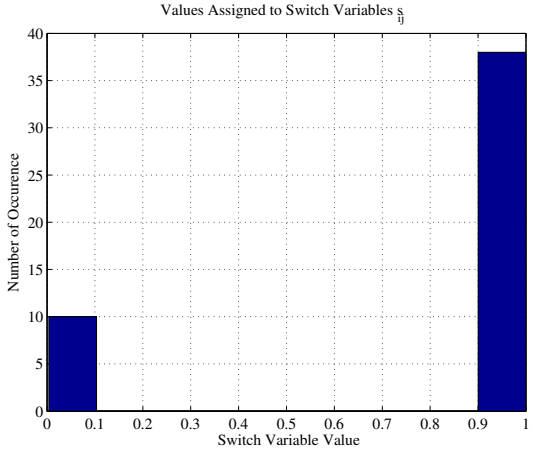


Figure 5.8: *Histogram over the final values of the switch variables s_{ij} for the simple example in Fig. 5.9. Those switch variables associated with correct loop closures maintain their initial value $s_{ij} = \gamma_{ij} = 1$, so that the resulting weight factor $\omega_{ij} = \Psi(s_{ij}) = 1$ and thus the associated constraint is maintained. In contrast, the variables associated with false-positive loop closures are assigned values $s_{ij} \approx 0$, so that $\omega_{ij} = \Psi(s_{ij}) \approx 0$. This way, the false-positive loop closure constraints are disabled and do not influence the solution for the robot poses X^*.*

robustified optimization converges towards a correct solution. The grey lines in the plot represent the loop closure requests that have been disabled during the optimization, i.e. their associated switch variables s_{ij} have been assigned values so that the resulting weight factor $\omega_{ij} = \Psi(s_{ij}) \approx 0$. This is illustrated in Fig. 5.8, which shows a histogram of the values assigned to the switch variables. On the left, one can see the switch variables that are associated with the 10 wrong loop closure constraints. They have been assigned small values approximately of 0. The correct loop closure constraints have been assigned values close to 1, thus do not diverge from their initial values.

It is important to notice that all of the correct loop closures at the bottom of the trajectory were adhered, none of the correct loop closures was disabled in error. Their associated switch values do not differ from the initial values and remain stable at 1. In terms of precision-recall statistics this example therefore yields an optimal result with both 100% precision and recall.

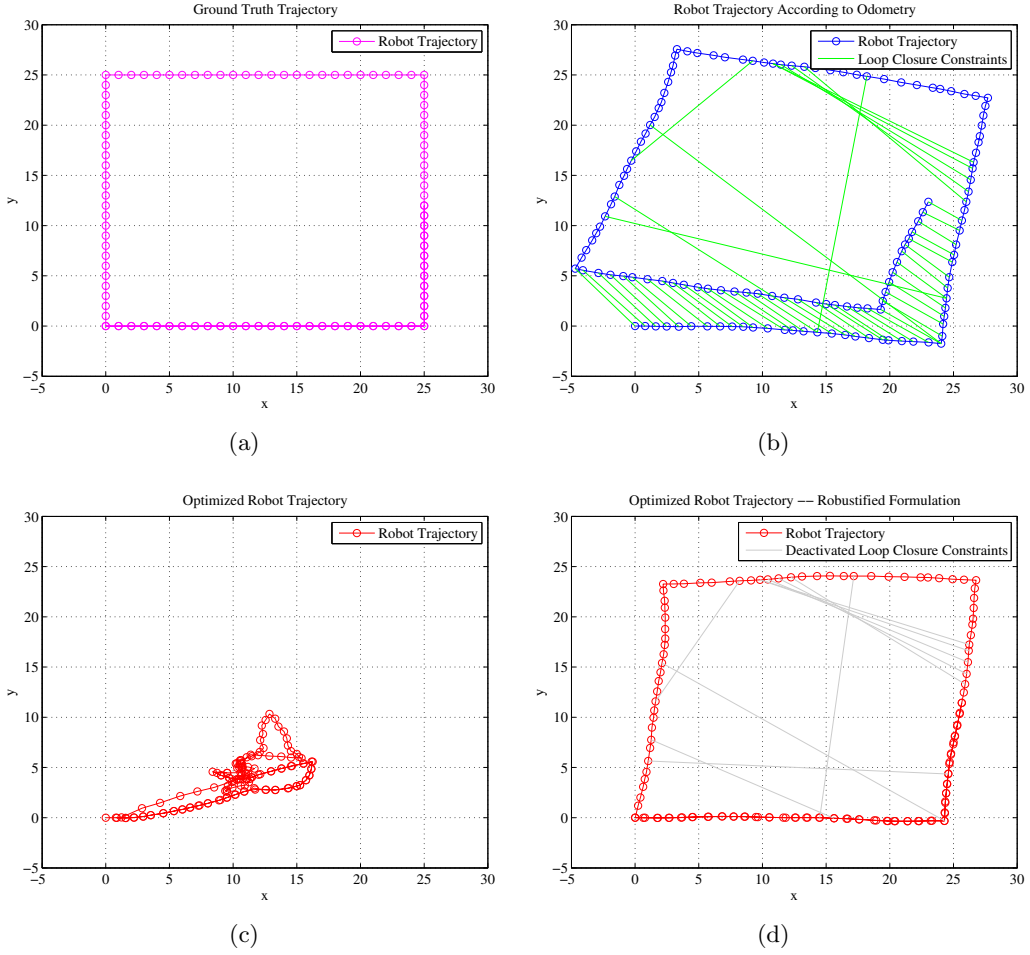


Figure 5.9: *A simple synthetic example for robust pose graph SLAM. (a) shows the ground truth path, (b) illustrates the trajectory according to the noisy odometry sensor along with the loop closures requested by the front-end. Notice the ten false positive loop closures. The solution provided by the state of the art back-end* g^{2}o *is depicted in (c). The outlier constraints forced the optimization to converge towards a defective solution. (d) shows the result of my proposed robust back-end. Notice that the trajectory is correctly estimated given the odometry readings. All erroneous loop closure constraints have been disabled.*

6 Evaluation

The previous chapter proposed a novel approach for a robust back-end for graph-based SLAM systems. While the general feasibility was shown using a simple example, this chapter will provide a more elaborate evaluation and analysis of the performance of the proposed system.

The common procedure in the literature is to examine the performance of the SLAM back-ends on several different datasets, measuring characteristics such as the deviation from the ground truth solution or convergence speed. I will follow these methods and use a number of commonly accepted datasets that have been used for evaluation in a number of publications. All of these datasets are pose graphs in 2D or 3D and contain many hundred or several thousand poses and loop closure constraints (see Table 6.1).

The main focus of the evaluation is of course on the robustness of the proposed robust back-end and not on how well optimization-based SLAM systems perform in general[36]. The evaluation that follows will therefore provide answers to the following questions:

- How robust is the robust back-end? That is, how many outliers can be present in the dataset while the system still produces a correct result?
- How large is the influence of outlier loop closure constraints on the final error of the optimization result, compared to the ground truth?
- What is the influence of the free parameter $\mathbf{\Xi}_{ij}$ (the switch prior covariance) on the performance of the system?
- What influence on the runtime behaviour can be expected when the robust back-end is implemented for a state of the art framework?

[36] This has already been done in a large number of publications and we know that they perform extremely well regarding correctness of the achieved results, runtime behaviour and scalability. That is, at least in outlier-free situations.

With the current understanding of SLAM as an optimization problem, it is unfortunately not possible to give a *guarantee* in a mathematical sense, or to *prove* that the iterative optimization will not get trapped in a local minimum. This is true for state of the art approaches as well as for the proposed robust extension. The reason is that SLAM in its default formulation is a non-convex problem, therefore the existence of local minima in the cost function has to be expected. The question where these local minima appear in the state space, relative to the initial guess and the ground truth solution, or how pronounced and accentuated they are, is of strong theoretical interest, but beyond the scope of this work[37].

6.1 Error Metrics for SLAM

While we can often judge the general quality of an optimization result by visual inspection of the resulting map or trajectory, a *quantitative* measure has to be preferred. In the following I will explain two meaningful error metrics for pose graphs, the *root-mean-square error* (RMSE) and the *relative pose error* (RPE). A third method, precision-recall statistics is introduced as well. This measure will allow to judge how well the robust back-end distinguishes between true and false positive loop closure constraints.

A more in-depth discussion on error metrics for pose graph SLAM algorithms can be found in [Kümmerle et al., 2009] or [Burgard et al., 2009].

6.1.1 The Root-Mean-Square Error (RMSE)

In general, the RMSE measures the mean deviation of an estimated value from its true value. It is reasonable to use two different RMSE measures for the position and for the orientation: For 2D and 3D problems, RMSE_{pos} measures the deviation of the robot positions from their ground truth values and RMSE_{ori} measures the deviation of the orientation angles.

Formally, for 2D SLAM, RMSE_{pos} is defined as

$$\text{RMSE}_{\text{pos}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^{x,y} - \hat{\mathbf{x}}_i^{x,y})^2} \tag{6.1}$$

where $\mathbf{x}_i^{x,y} = (x_i, y_i)^\mathsf{T}$ is the estimated robot position and $\hat{\mathbf{x}}_i^{x,y}$ is the ground truth position. The 3D case is defined respectively using $\mathbf{x}_i^{x,y,z}$ etc.

[37] As I shortly discussed before, recent work suggests that depending on the reference frame in which the constraints are formulated in, SLAM has close-convex properties [Huang et al., 2010]. Other authors proposed a linear approximation to 2D SLAM, which allows to apply a linear method to solve the problem, leading to a closed-form solution that requires no initial guess and has a unique solution [Carlone et al., 2011a; Carlone et al., 2011b]. These are all promising developments that are worth further research, also in combination with the approach for more robustness proposed in this thesis.

The root-mean-square error with respect to the orientation angle θ can be defined in the same way for the 2D case:

$$\mathrm{RMSE}_{\mathrm{ori}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^{\theta} - \hat{\mathbf{x}}_i^{\theta}\right)^2} \tag{6.2}$$

Notice that the minus operator above has to respect the special angular subtraction rules[38].

For the 3D case with full 6 degrees of freedom and 3 orientation angles, quaternion algebra is used to determine the difference in orientation between the estimated and ground truth poses:

$$\mathrm{RMSE}_{\mathrm{ori}} = \frac{1}{n}\sum_{i=1}^{n}\arccos\left(\mathbf{q}_i \cdot \hat{\mathbf{q}}_i^{-1}\right) \tag{6.3}$$

where $\mathbf{q}_i$ and $\hat{\mathbf{q}}_i$ are the quaternions expressing the 3D orientation of the respective estimated or ground truth pose, $\mathbf{q}^{-1}$ is the quaternion inverse and the operator $\cdot$ is the quaternion multiplication. The arccos operator here only takes the first element of the resulting quaternion difference into account.

The RMSE metric measures how well the estimated robot trajectory is aligned to the ground truth solution on a global level. This bears some commonly known drawbacks: Any absolute displacement or rotation of the estimated solution with respect to the ground truth will lead to large errors in the RMSE metric, even if the map structure itself is inherently correct. Imagine the solution of the optimization fits the ground truth trajectory exactly, but is rotated by a small angle. This will of course lead to large errors in $\mathrm{RMSE}_{\mathrm{pos}}$.

Second, single errors in the estimation are penalized multiple times and the time of their occurrence determines the amount of penalty that is given. These disadvantages of RMSE have been explicitly pointed out in [Burgard et al., 2009] and an alternative metric has been defined. This metric is explained next.

6.1.2 The Relative Pose Error Metric

To overcome the drawbacks of the RMSE metric, [Burgard et al., 2009] and [Kümmerle et al., 2009] proposed a metric based on relative pose transformations. I will abbreviate this error as RPE.

The general relative pose error is defined as:

$$\mathrm{RPE} = \frac{1}{n}\sum_{i,j}(\boldsymbol{\delta}_{ij} - \hat{\boldsymbol{\delta}}_{ij})^2 \tag{6.4}$$

[38] For reasons of clarity I avoid using special operator symbols like $\ominus$ or $\boxplus$ for cases where the normal euclidean subtraction or addition does not apply and manifold approaches should be used instead. I rely on the reader to recognize these situations which generally occur whenever rotations are involved. Recently [Hertzberg et al., 2011] elaborately discussed the differences between euclidean spaces and manifold spaces and the ramifications that arise for applications in sensor fusion or SLAM.

where $\boldsymbol{\delta}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ is the relative transformation between the estimated poses $\mathbf{x}_i$ and $\mathbf{x}_j$ and $\hat{\boldsymbol{\delta}}_{ij}$ is the relative transformation between the ground truth poses $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ respectively. The RPE is an average measure with n being the number of constraints that is used in the sum in (6.4) above. Therefore, the RPE can be understood as a measure for the energy that is needed to transform the estimated trajectory into the ground truth solution.

For better interpretation and comparability, I calculate the RPE metric separately for the translational and rotational part of the transformations between poses. Following the notation used for the RMSE metrics, the translational RPE will be called RPE_{pos}, while the measure for the orientation is written as RPE_{ori}.

For the 2D case, RPE_{pos} is defined as:

$$\begin{aligned}\text{RPE}_{\text{pos}} &= \frac{1}{n}\sum_{i,j}(\boldsymbol{\delta}_{ij}^{x,y} - \hat{\boldsymbol{\delta}}_{ij}^{x,y})^2 \\ &= \frac{1}{n}\sum_{i,j}\left(\mathbf{R}_{\mathbf{x}_i^\theta}\left(\mathbf{x}_i^{x,y} - \mathbf{x}_j^{x,y}\right) - \mathbf{R}_{\hat{\mathbf{x}}_i^\theta}\left(\hat{\mathbf{x}}_i^{x,y} - \hat{\mathbf{x}}_j^{x,y}\right)\right)^2\end{aligned} \tag{6.5}$$

where $\mathbf{R}_{\mathbf{x}_i^\theta}$ is the 2D rotation matrix with angle $\mathbf{x}_i^\theta$. It rotates the difference vector into the local coordinate system of $\mathbf{x}_i$, hence we speak of the *relative* pose error.

The angular component in the 2D case is easily defined as:

$$\begin{aligned}\text{RPE}_{\text{ori}} &= \frac{1}{n}\sum_{i,j}(\boldsymbol{\delta}_{ij}^{\theta} - \hat{\boldsymbol{\delta}}_{ij}^{\theta})^2 \\ &= \frac{1}{n}\sum_{i,j}\left(\left(\mathbf{x}_i^\theta - \mathbf{x}_j^\theta\right) - \left(\hat{\mathbf{x}}_i^\theta - \hat{\mathbf{x}}_j^\theta\right)\right)^2\end{aligned} \tag{6.6}$$

Notice again that the minus operator has to adhere angular subtraction rules.

For the 3D case, the translational RPE_{pos} is defined respectively, of course using all three dimensions of the translational part of the poses (i.e. $\mathbf{x}_i^{x,y,z}$ and so on). For RPE_{ori}, quaternion algebra is used again to calculate the difference between the rotational part of the given poses, similar to the RMSE_{ori} metric. In this case, we have

$$\text{RPE}_{\text{ori}} = \frac{2}{n}\sum_{i,j}\arccos\left(\left(\mathbf{q}_i \cdot \mathbf{q}_j^{-1}\right) \cdot \left(\hat{\mathbf{q}}_i \cdot \hat{\mathbf{q}}_j^{-1}\right)^{-1}\right) \tag{6.7}$$

where the different $\mathbf{q}$ are the quaternions expressing the 3D orientation of the respective estimated or ground truth pose, and like before $\mathbf{q}^{-1}$ is the quaternion inverse, the operator $\cdot$ is the quaternion multiplication and arccos uses only the first element of its quaternion argument.

6.1.3 Precision-Recall Statistics

While the error metrics RPE or RMSE measure how well the estimated trajectory corresponds to the ground truth trajectory, it is also meaningful to specify how many

of the wrong loop closure constraints have been correctly deactivated by the robust back-end, or how many correct constraints have been deactivated in error.

A widely used metric that can measure these quantities is *precision* and *recall*.

Recall is defined as the ratio of true positives to all real positives. In our case where we want to judge the quality of loop closure deactivation, a recall of 100% means that all false loop closure constraints have been deactivated:

$$\text{recall} = \frac{\#\text{ correctly deactivated constraints}}{\#\text{ false constraints in dataset}} \tag{6.8}$$

Precision in return measures how many of the deactivated loop closures were actually wrong loop closures, thus the ratio of correctly deactivated constraints to all deactivated constraints.

$$\text{precision} = \frac{\#\text{ correctly deactivated constraints}}{\#\text{ all deactivated constraints}} \tag{6.9}$$

Both measures are given in percent and a high value (close to 100%) is desirable for both quantities. This is clear, as we want to deactivate all false loop closures (thus reaching a high recall) while not deactivating the correct loop closures (thus reaching a high precision).

It is important to understand what the term *deactivation* means in the context of precision-recall metrics for this specific domain. Remember that in the robust back-end, there is no binary decision on whether a constraint is active or deactivated. The system rather assigns a continuous weight $\omega_{ij} \in (0, 1)$ to each loop closure constraint. It is only for the precision-recall analysis described above, that a threshold τ on ω_{ij} is used to divide the loop closure constraints into deactivated and active constraints: A constraint is declared deactivated if $\omega_{ij} < \tau$. Precision and recall are calculated, and the procedure is repeated for different values of τ, so that $0 \leq \tau \leq 1$. The result is a plot that shows precision vs. recall for all chosen values of τ. Notice that in the extreme, for $\tau = 0$, all constraints would be declared "active". For the analysis conducted in the evaluation, 100 equally distributed values between 0 and 1 were used for τ.

6.2 Datasets for the Evaluation

In order to show the versatility and general feasibility of the proposed approach, six very different datasets were used for the evaluation. Table 6.1 lists and summarizes their important properties. The synthetic datasets are created from simulation, while the two real-world datasets have been recorded in a 2D (Intel) or 3D (Parking Garage) environment respectively. These datasets are publicly available to the community and have been used as examples and benchmarks in a number of SLAM publications before.

The Manhattan dataset is available in two versions: The original dataset (Fig. 6.1(a)) was first published by [Olson et al., 2006] and the second version (Fig. 6.1(b)) was included in the open source implementation of g^{2}o [Kümmerle et al., 2011b]. The difference between the two versions is the quality of the initial estimate: It is much closer to the ground truth for the g^{2}o version than for Olson's original dataset. This can be

Table 6.1: *The datasets used during the evaluation.*

Dataset	synthetic / real	2D/3D	Poses	Loop Closures
Manhattan (original)	synthetic	2D	3500	2099
Manhattan (g^2o version)	synthetic	2D	3500	2099
City10000	synthetic	2D	10000	10688
Sphere2500	synthetic	3D	2500	2450
Intel	real	2D	943	894
Parking Garage	real	3D	1661	4615

Table 6.2: *Initial errors for the various datasets.*

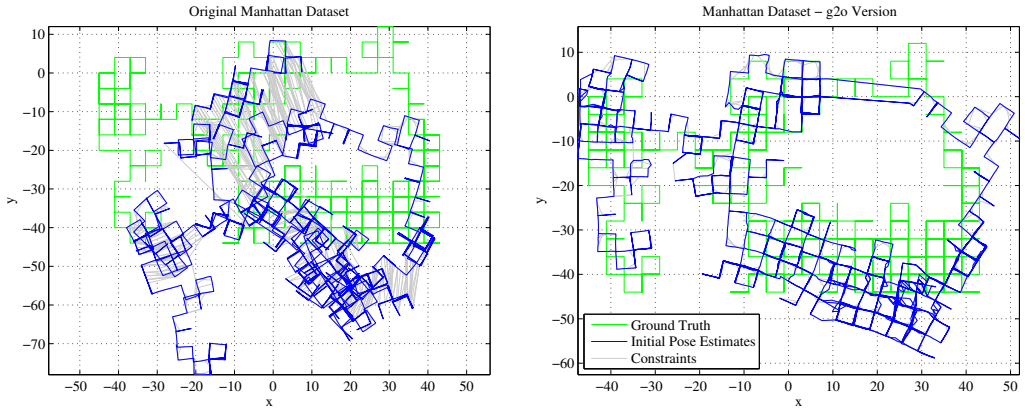
Dataset	RMSE_{pos} [m]	RMSE_{ori} [°]
Manhattan (Olson's Original)	22.44	36.8
Manhattan (g^2o Version)	9.97	18.9
City10000	37.2	29.6
Sphere2500	41.2	57.9
Intel	0.16	0.9
Parking	8.8	4.3

verified from Fig. 6.1 and from the error metrics in table 6.2. For evaluation purposes, having the same dataset with different initialisations allows us to see the influence of the initial estimates on the overall behaviour of the back-end system. I will therefore use both versions and indicate whether the original or the g^2o version has been used.

The datasets City and Sphere (see Fig. 6.2(a) and 6.2(b)) shipped with the open-source implementation of iSAM [Kaess et al., 2008]. The two real-world datasets Intel and Parking Garage are part of g^2o [Kümmerle et al., 2011b] and are depicted in Fig. 6.3(a) and 6.3(b). For the two latter real-world datasets, no ground truth information is available. Instead, the estimation results for the outlier-free dataset are used as a pseudo ground truth when necessary.

6.3 General Methodology

The available datasets that are frequently used for the evaluation of SLAM back-ends are pose graphs consisting of odometry measurements and loop closure constraints. These datasets are free of outliers, i.e. all loop closure constraints are correct. This is perfectly understandable, since state of the art back-ends cannot cope with false constraints and fail in their presence. To evaluate and benchmark the robust back-end I proposed in the previous chapter, the available datasets are spoiled by additional, wrong loop closure constraints. That means, loop closure constraints which do not connect corresponding poses and thus are outliers are added to the dataset. These additional loop closure



(a) (b)

Figure 6.1: *Two versions of the Manhattan dataset: (a) The original dataset as published in [Olson et al., 2006] and (b) the* g^{2}o *version. Notice that the initial error is much higher in (a) than it is in (b).*

constraints can be added totally random or following a certain policy, which will be explained later on. As discussed before, in real applications, these outliers might have been introduced by the front-end after a failed place recognition etc.

Given the spoiled datasets, the performance of the robust back-end can be evaluated using two different methods:

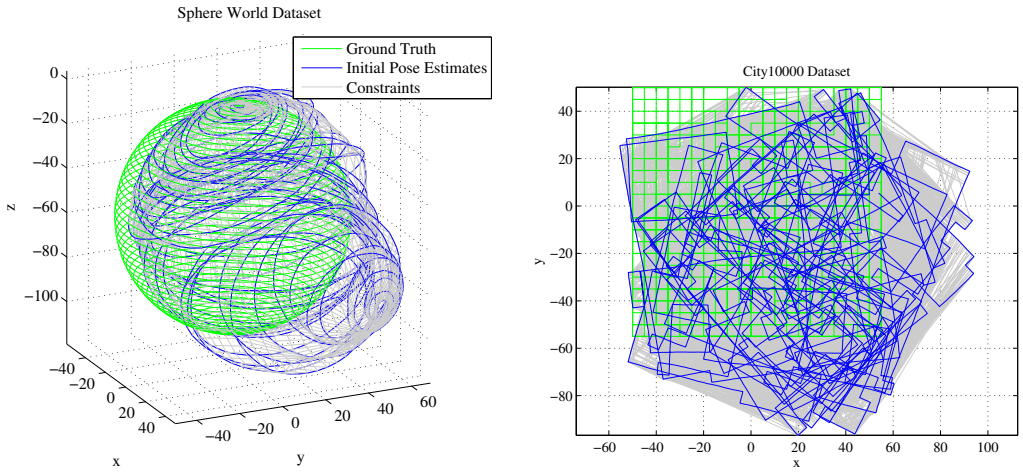
1. Use a suitable error metric (like RPE or RMSE) to compare the resulting *trajectory* against the ground truth solution and the solution reached by state of the art non-robust back-ends.

2. Use precision-recall statistics to identify how many of the added wrong outlier constraints could be identified and disabled, while maintaining the correct loop closure constraints.

6.3.1 Policies for Adding Outlier Loop Closure Constraints

For the evaluation, the datasets are spoiled by adding false positive loop closure constraints between two poses $\mathbf{x}_i$ and $\mathbf{x}_j$. The indices i and j are determined using four different policies, which are explained below and illustrated in Fig. 6.4.

Random Constraints This policy adds constraints between two randomly chosen pose vertices $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e. the indices i and j are drawn from a uniform distribution over all available indices. Most of the constraints that are created using this policy will span over large areas of the dataset since they connect two distant poses $\mathbf{x}_i$ and $\mathbf{x}_j$.



(a) (b)

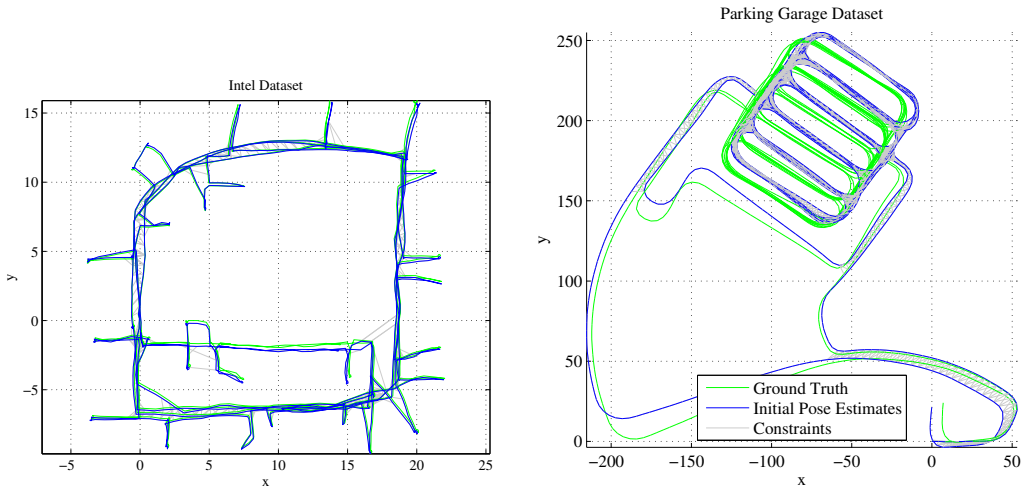
Figure 6.2: *Two more synthetic datasets used during the evaluation: For the sphere world (a) a simulated robot was driven on the surface of a sphere. (b) shows the planar City10000 dataset.*

Local Constraints Following this policy, constraints are added only locally. That means that the first pose vertex of the constraint is chosen randomly from all available vertices. The second vertex however is chosen so that it is in the spacial vicinity of the first vertex. This follows the intuition that in reality nearby places are more likely to appear similar than distant places and thus false place recognitions are more likely to be established between these nearby poses.

Randomly Grouped Constraints In real front-ends, false positive loop closure constraints can be expected to appear in groups. Imagine a robot driving through a corridor or street where the visual appearance is very similar to that of another corridor or street already mapped. The front-end may erroneously recognize loop closures for several successive frames while the robot traverses the ambiguous part of the environment.

This is simulated by the two grouped policies. The randomly grouped policy first picks i and j randomly from all available indices, but then adds 20 successive constraints between the poses with indices $i \dots i + 20$ and $j \dots j + 20$.

Locally Grouped Constraints This last policy is a combination of the local and grouped policies and creates groups of short constraints that connect nearby places. Following this policy, the first index i is chosen randomly. The second index j is chosen from the vicinity of i, like with the local constraint policy above. Then 20 successive constraints between the vertices with indices $i \dots i + 20$ and $j \dots j + 20$ are added.



(a) (b)

Figure 6.3: *Two real-word datasets: The Intel dataset (a) has been recorded in an indoor environment from a ground-based robot performing laser scan matching. The dataset shown in (b) is a 3D dataset recorded in a four-story parking garage. The image shows a bird's eye view on the trajectory. Both datasets shipped with* g^{2}o *[Kümmerle et al., 2011b]. Since no real ground truth measurements are available for both datasets, the results of the optimization without outliers serve as pseudo ground truth for the conducted experiments.*

6.3.2 Loop Closure Displacement

Remember from (5.14) that a loop closure constraint $\|\Psi(s_{ij})\cdot(f(\mathbf{x}_i,\mathbf{u}_{ij})-\mathbf{x}_j)\|^2_{\mathbf{\Lambda}_{ij}}$ contains a displacement $\mathbf{u}_{ij}$ between the two associated poses $\mathbf{x}_i$ and $\mathbf{x}_j$. The displacements $\mathbf{u}_{ij}$ associated with the added constraints are chosen randomly. The translational part is drawn from a uniform distribution, while the relative rotation is drawn from a Gaussian distribution. For the 2D case, this is:

$$\mathbf{u}_{ij} \sim \begin{pmatrix} \mathcal{U}(-1,1) \\ \mathcal{U}(-1,1) \\ \mathcal{N}(0,10\,^\circ) \end{pmatrix} \tag{6.10}$$

For the 3D datasets, the definition of $\mathbf{u}_{ij}$ is extended, using the same uniform distribution for the z-coordinate and the normal distribution for the two additional angles.

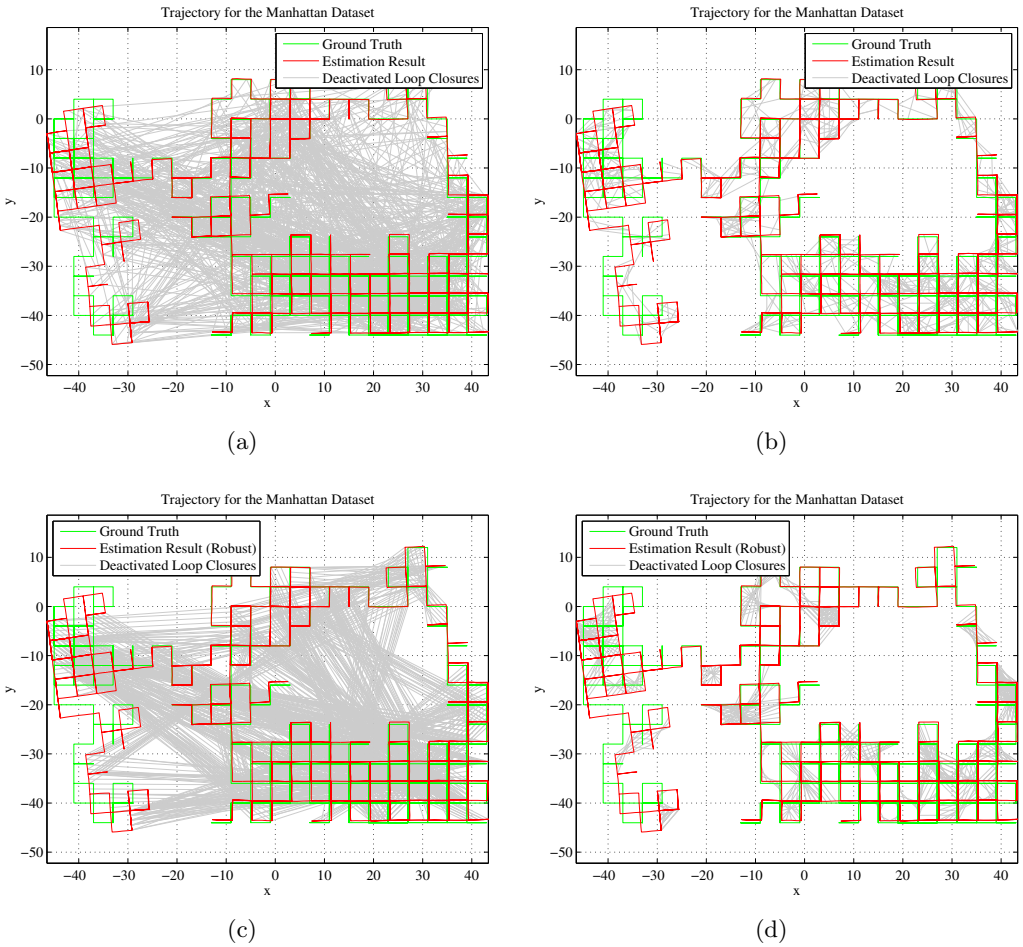


Figure 6.4: *Illustration of the different outlier policies. In all figures, 1000 outlier loop closure constraints were added to the Manhattan dataset and are visible as grey links between poses. The used outlier policies were: (a) random, (b) local, (c) randomly grouped, (d) locally grouped. The ground truth trajectory is shown in green, while the estimation result of the robust back-end is plotted in red. Notice that despite 1000 additional outliers, the proposed back-end was able to converge to a correct solution, disabling all of the false positive constraints, independent of the used outlier policy.*

6.3.3 The Switch Function

Unless otherwise noted, for all the experiments described below the linear switch function Ψ_a^{lin} with parameter $a = 1$ was used. It was previously defined as

$$\omega_{ij} = \Psi_a^{\text{lin}}(s_{ij}) : \mathbb{R} \rightarrow [0,1] = \begin{cases} 0 & : s_{ij} < 0 \\ \frac{1}{a} s_{ij} & : 0 \leq s_{ij} \leq a \\ 1 & : s_{ij} > a \end{cases} \tag{6.11}$$

However, other switch functions could be chosen as well, therefore the influence of the switch function on the estimation results is discussed in section 6.8.

6.3.4 Used Framework and Implementation

The robust back-end as proposed in the previous chapter was implemented for two available frameworks: g^{2}o [Kümmerle et al., 2011b] and GTSAM / iSAM [Kaess et al., 2011]. However, all experiments below were conducted using the robust back-end implementation for the g^{2}o framework which appeared to be more comfortable to work with and more versatile. The Gauss-Newton algorithm was used to iteratively solve the optimization problems.

6.4 The Influence of Ξ_{ij} on the Estimation Results

Now that all necessary preliminary information are given, the evaluation of the proposed robust-back end commences.

Remember, that the formulation of the proposed robust back-end in (5.14), involved the switch prior constraints

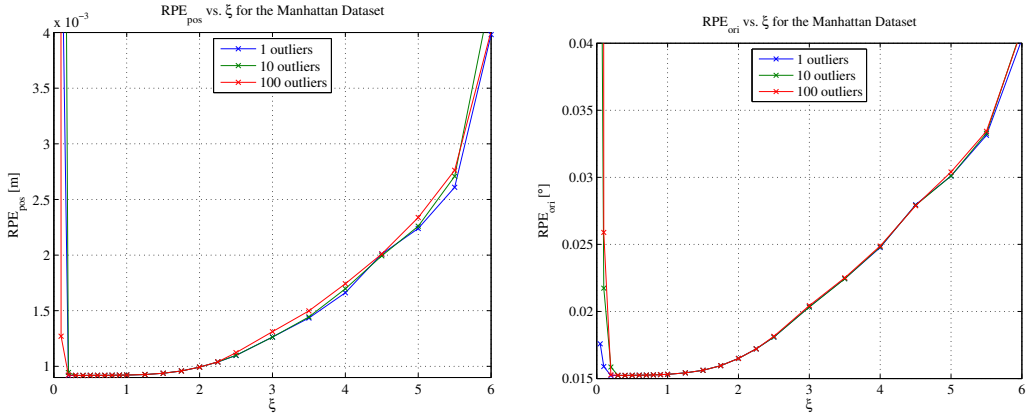
$$\|\gamma_{ij} - s_{ij}\|^2_{\boldsymbol{\Xi}_{ij}} \tag{6.12}$$

The exact value of the switch prior variances $\boldsymbol{\Xi}_{ij}$ could not be deduced in a mathematically sound way. It rather has to be set empirically. Therefore, the first question I want to explore in this evaluation is the influence of $\boldsymbol{\Xi}_{ij}$ on the estimation results and what a suitable value for $\boldsymbol{\Xi}_{ij}$ would be.

As I mentioned before in section 5.1.3, $\boldsymbol{\Xi}_{ij}$ controls the penalty the system gains for the deactivation of a loop closure constraint. By adapting this value individually for each constraint, the front-end could express a degree of certainty about that particular loop closure constraint[39]. If the front-end is not capable of determining an individual degree of certainty, all constraints could be assigned the same $\boldsymbol{\Xi}_{ij}$.

For the following, to show the general influence of $\boldsymbol{\Xi}_{ij}$, I assume the same value $\boldsymbol{\Xi}_{ij} = \xi$ was assigned to all constraints.

[39] A small value of $\boldsymbol{\Xi}_{ij}$ leads to a high penalty if it is deactivated. Thus the front-end would assign small $\boldsymbol{\Xi}_{ij}$ to loop closures it is very certain about and large $\boldsymbol{\Xi}_{ij}$ to those loop closures that appear more doubtful.



(a) (b)

Figure 6.5: *The influence of $\Xi_{ij} = \xi$ on the estimation quality. (a) shows the translational error RPE_{pos} while (b) plots the orientation error RPE_{ori}. The best choice for ξ is in the interval $0.3 \leq \xi \leq 1.5$ where the error measures are relatively constant and have their minimum.*

6.4.1 Methodology

To determine the influence of ξ, the relative pose errors $\mathrm{RPE}_{\mathrm{pos}}$ and $\mathrm{RPE}_{\mathrm{ori}}$ were determined for three different values of random outliers (1, 10, and 100) and varying ξ on the Manhattan dataset.

Fig. 6.5 shows the results. Every data point represents the mean $\mathrm{RPE}_{\mathrm{pos}}$ and $\mathrm{RPE}_{\mathrm{ori}}$ of 10 trials for that particular pairing of ξ and number of outliers. In total, 720 optimization runs were conducted to create the graph.

6.4.2 Results and Interpretation

The curves in Fig. 6.5 reveal a number of facts: The most obvious (and trivial) one is that the value of $\xi = \Xi_{ij}$ indeed influences the quality of the optimization result which is measured by the RPE metrics. The second insight is that the quality of the estimation drops drastically if ξ is too small or too large. This can be seen from the raising RPE measures for $\xi < 0.3$ and $\xi > 2.0$.

The most interesting result however is, that the RPE stays relatively constant for a broad range of values for ξ between 0.3 and 1.5, where the error is minimal. This behaviour is independent from the number of outliers.

Result 6.1. *If the front-end is not able to assign sound individual values for Ξ_{ij}, it is safe to set all $\Xi_{ij} = 1$, since this value is close to the individual optimal choice of Ξ_{ij} for a large range of outliers.*

Therefore, for all evaluations that follow, all Ξ_{ij} have been set to 1. Although that value was determined using the Manhattan dataset, it proved to be a suitable value for

all other evaluated datasets and even for another application beyond SLAM we are going to encounter in Chapter 8.

6.5 The Robustness in the Presence of Outliers

After a sound value for the free parameter of the system, Ξ_{ij} has been determined, we can now examine how well the proposed back-end performs in the presence of outliers. The evaluation will compare the results of the proposed robust solution to the solution of the state of the art problem formulation in the presence of outliers. We will furthermore see how much the number of outliers influences the estimation result.

6.5.1 Methodology

A large number of test cases were considered for the different datasets. The number of added outliers was varied between 0 and 1000, using all of the four policies described above. For each number of additional wrong outliers, 10 trials per policy were calculated, resulting in a total of 500 trials per dataset. For each trial, the error metrics RPE and RMSE were determined.

Notice from Table 6.1 that the number of correct loop closure constraints in the datasets varied between 10688 (City10000) and only 894 (Intel). Therefore, 1000 additional outlier loop closures is a huge number, leading to outlier ratios between 9.4% for the City10000 dataset and almost 112% for the Intel dataset. In real applications, we can expect much smaller outlier ratios in the range of a few percent or even below, depending on how sophisticated the front-end is built. The evaluation here uses much higher outlier ratios to demonstrate the remarkable robustness of the system.

6.5.2 Results and Interpretation

Table 6.3 summarizes the results for the different datasets. The minimum, maximum, and median RPE_{pos} measures are listed, as well as a success rate which measures the percentage of correct solutions. In the following, a small representative collection of plots will illustrate the most important results.

From table 6.3 we see that except for the parking garage dataset, the overall success rates are very high. In total, from all 2500 trials, only two failed, leading to success rates equal or close to 100%. The two failure cases and the special case of the Parking Garage dataset are discussed in detail in section 6.5.3. I want to remark that the two failure cases for the original Manhattan dataset and for the sphere world dataset could be successfully resolved by using the Huber cost function in combination with the proposed back-end. If we allow this further extension (which comes at the cost of slower convergence, due to the partly linear cost function), we can conclude that except for the Parking Garage dataset, a success rate of 100% was reached.

Result 6.2. *The proposed back-end was able to successfully solve 2498 out of 2500 trials on different datasets with up to 1000 outlier constraints. In combination with the Huber*

Table 6.3: *Overall* RPE_{pos} *metric for the different datasets, with 0 ... 1000 outliers and 500 trials per dataset.*

Dataset	max outl. ratio	min RPE_{pos}	max RPE_{pos}	median RPE_{pos}	incorrect solutions	success rate
Manhattan (g^2o)	47.6%	0.0009	0.0009	0.0009	0	100%
Manhattan (orig.)	47.6%	0.0009	5.9659	0.0009	1	99.8%
City10000	9.4%	0.0005	0.0005	0.0005	0	100%
Sphere2500	40.8%	0.0953	18.1674	0.0964	1	99.8%
Intel	111.9%	0.2122	0.2147	0.2132	0	100%

cost function, all 2500 trials were solved successfully, leading to a success rate of 100%.

Fig. 6.6 illustrates how the number of outliers affects the estimation error for different datasets. The plots reveal that the overall error, expressed in terms of the RPE metric, is surprisingly constant, regardless of the number of outliers in the dataset. Even for 1000 outlier constraints, the resultant map of the robust SLAM back-end does in general not diverge more from the ground truth solution than the solution for the original dataset with no outlier constraints does. Notice that the slight increase for the Intel dataset in 6.6(c) between 0 and 1000 outliers is approximately only 0.2% and therefore neglectable. Notice further that the failure case for the sphere dataset in 6.6(d) has been replaced by the solution from combination with the Huber function.

These are a quite remarkable findings, and we can conclude:

Result 6.3. *With the proposed robust back-end, the resulting estimation error is constant for a broad range of 0 to 1000 outlier constraints for a variety of datasets, both 2D and 3D, synthetic and real-world.*

Fig. 6.7 compares the results of the robust back-end with the non-robust state of the art implementation. Although for the non-robust trials the Huber error function was used (using a kernel width 1.0) as proposed by [Grisetti et al., 2011], g^2o was not able to converge to a correct solution in the presence of outliers. Table 6.4 furthermore reveals that the influence of the outlier policy on the result was in general neglectable.

Result 6.4. *For large numbers of outliers, the proposed robust implementation is up to two orders of magnitude more accurate than state of the art approaches using the Huber cost function. Despite the outliers, the robust back-end reaches the same low error measures as in the zero-outlier case.*

While the RPE metric compares the deviation of the estimated trajectory from the ground truth, precision-recall statistics allow us to determine how well the proposed robust technique is able to identify and disable the outlier constraints while leaving the true positive (i.e. correct) constraints intact. A system operating at a precision and recall rate both equal to 1 would be optimal, since all false positives are disabled, while

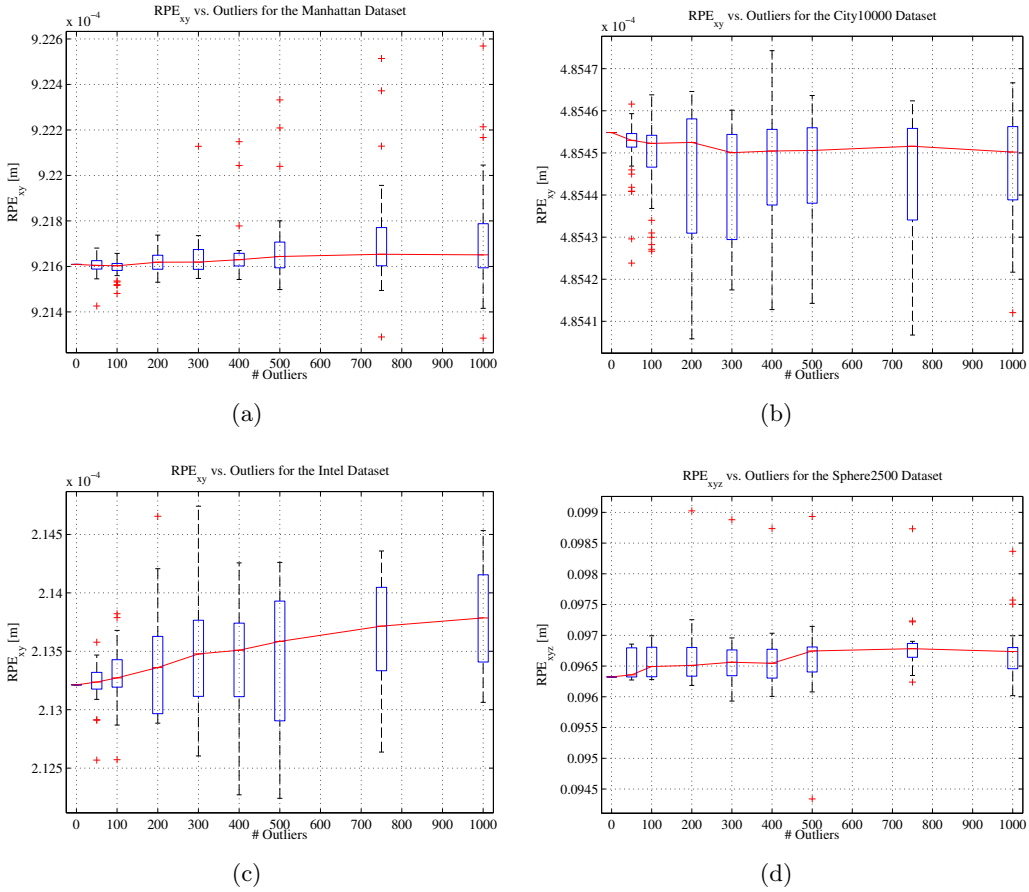


Figure 6.6: RPE *measures against outliers for various datasets. Notice how the error is approximately constant, regardless of the number of outlier constraints. In (d), the failure case that occurred at 300 outliers has been replaced by the solution gained by combination with the Huber cost function. In (a) the* g^{2}o *variant of the Manhattan dataset is shown.*

Table 6.4: RPE$_{pos}$ *metric for the* g^{2}o *version of the Manhattan dataset. All values* $\times 10^{-4}$.

Policy	min RPE$_{\text{pos}}$	max RPE$_{\text{pos}}$	avg RPE$_{\text{pos}}$	std. dev.	median
random	9.213	9.218	9.216	0.0006	9.216
local	9.214	9.226	9.217	0.0019	9.216
randomly grouped	9.216	9.217	9.216	0.0002	9.216
locally grouped	9.214	9.237	9.216	0.0029	9.216
overall	9.213	9.237	9.216	0.0017	9.216

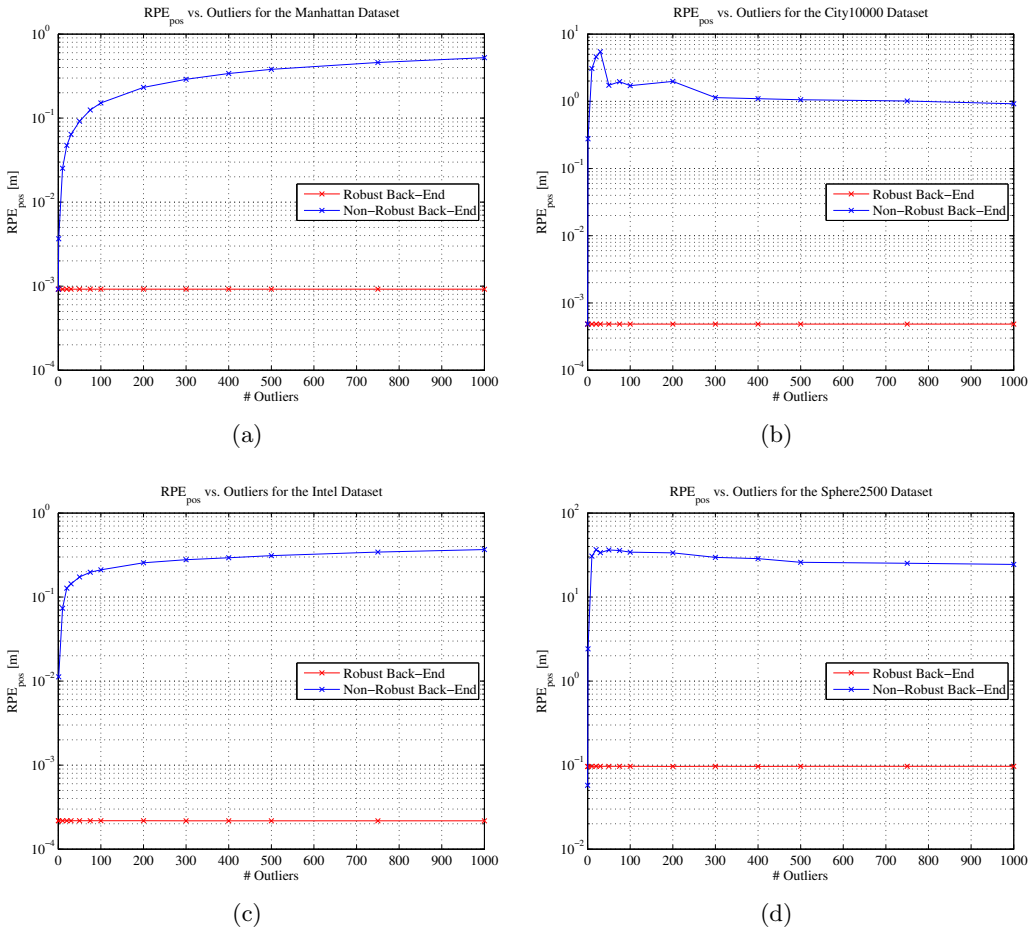


Figure 6.7: *Comparison of* RPE *measures between the proposed robust and the state of the art non-robust back-ends. Notice how the robust solution is up to two orders of magnitude more accurate for large numbers of outliers. The non-robust solution was supported by the Huber cost function, as proposed in [Grisetti et al., 2011].*

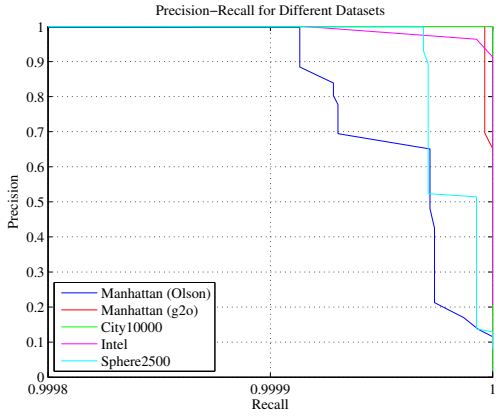


Figure 6.8: *Precision-recall statistics for the various datasets. Notice the scale of the X-axis, representing recall. The results indicate a close to optimal performance of the proposed system.*

all true positives are left untouched. However, we have to regard that the back-end never really performs a binary decision on whether a constraint is supposed to be active or deactivated. It is only the precision-recall benchmark, that emulates such a behaviour.

From Fig. 6.8 we can see that the results for the proposed back-end almost reach that point of optimal performance. For the datasets Intel, City10000, and Manhattan (g^2o), the recall is exactly 1 for a large span of precision. This means that there are values for ω_{ij} where all false positive constraints would be considered disabled, while a large amount of true positives are enabled. For the datasets sphere2500 and Olson's version of the Manhattan world, the recall is slightly smaller, since the statistics include the two failure cases that occurred with these two datasets and were mentioned before.

In general, the conclusion drawn from these results is:

Result 6.5. *In terms of precision-recall statistics, the proposed back-end reaches almost optimal results, indicating that almost all true positive loop closure constraints are left intact while all false positives are disabled. More precisely, at* 100% *precision, meaning no correct loop closure is erroneously deactivated, the system reached a recall of over* 99.99%*.*

A few exemplary robust solutions for several datasets containing large numbers of outliers are illustrated in Fig. 6.9.

6.5.3 Discussion of the Failure Cases

Manhattan and Sphere Datasets

As we saw in Table 6.3, from 2500 trials conducted using the datasets Manhattan, Sphere, City and Intel with up to 1000 added outlier constraints, only 2 trials failed to converge to a correct solution. One of these failures occurred with the Sphere2500 dataset and the

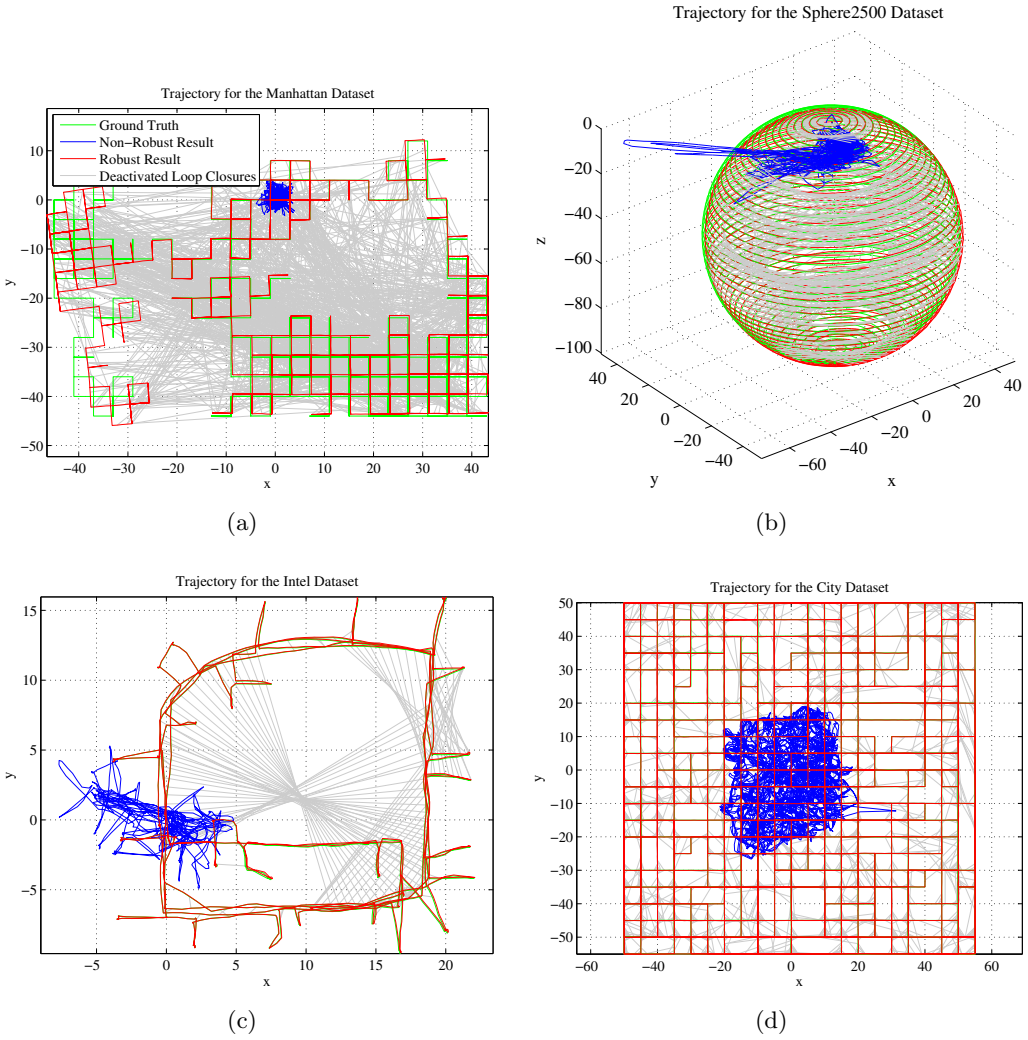
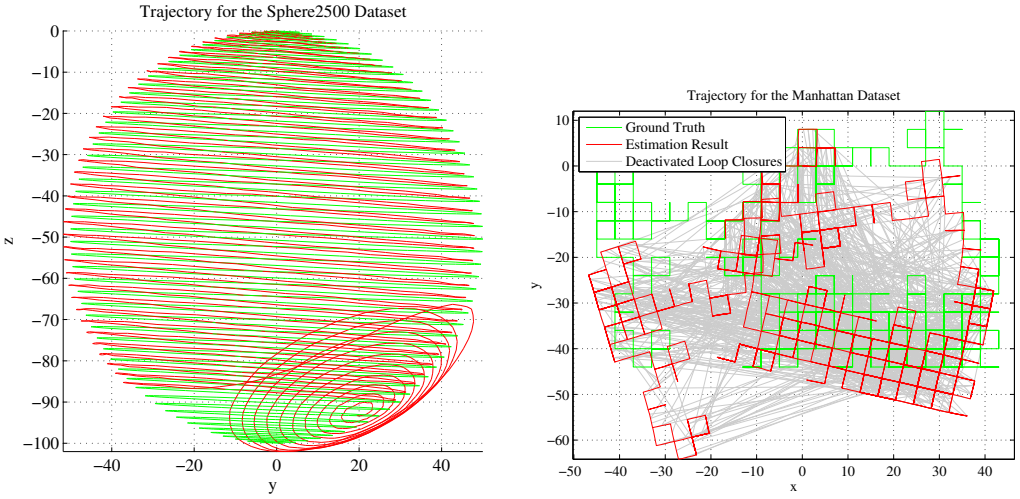


Figure 6.9: *The proposed robust back-end converges to correct results despite a high number of false positive loop closure constraints for different Datasets and outlier policies. In contrast, the conventional non-robust estimation (shown in blue) fails absolutely. (a) and (b) show the Manhattan and Sphere datasets with 1000 outliers. The Intel dataset in (c) has been spoiled by 5 groups of 20 outliers, while the City dataset in (d) shows 1000 local outliers.*



(a) (b)

Figure 6.10: *Two failure cases for the Sphere World dataset with 300 random outliers in (a) and the Manhattan dataset (Olson's original) containing 750 random outliers. Notice how the maps are still locally consistent. Both cases could be successfully resolved by combining the robust back-end with the Huber cost function.*

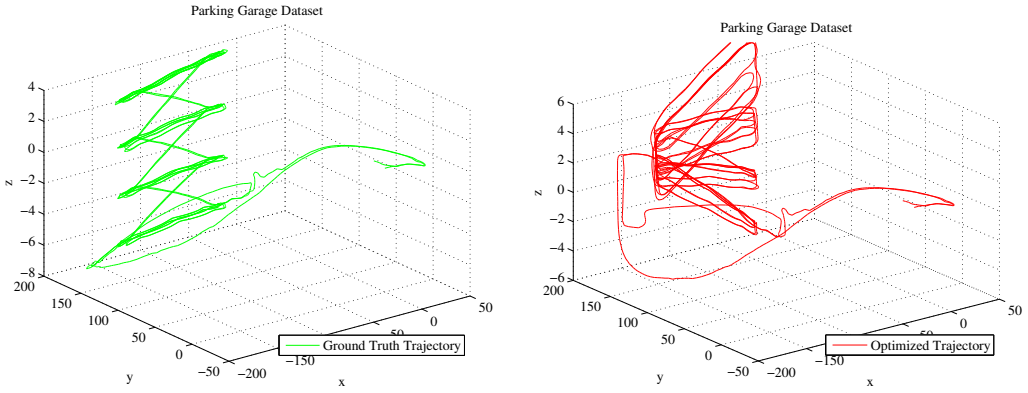
other one occurred with Olson's version of the Manhattan dataset. Both failure cases are illustrated in Fig. 6.10.

We can see from the figures that although the resulting maps are significantly distorted when compared to the ground truth on a global level, they are still locally intact. That is, for the sphere world the resulting map consists of two individually consistent and intact sub-maps that are however misaligned to each other. A similar situation occurs for the Manhattan dataset: Except for the region in the center of the map, where a false positive loop closure constraint was not deactivated, the map is locally consistent and would still be somewhat useful to a robot operating in that environment.

Result 6.6. *Even in the case of failure, the robust back-end degrades gracefully.*

The main reason for that beneficial behaviour is that apparently only *single* false positives are not deactivated correctly, leading to punctual errors that cause global distortion but retain local consistency. In the examples above, exactly *one* false positive was incorrectly *not* disabled. On the other hand, that means that still 299 out of 300 or 749 out of 750 false positive loop closure constraints were correctly disabled.

Both of the failure cases for the sphere and Manhattan datasets could be correctly resolved by combining the robust back-end with the Huber error function [Huber, 1973], using the default kernel width of 1.0. That means that instead of the default squared error function, the Huber function was used (see section 3.6.2), which can be easily done (e.g. when using the g^{2}o framework).



(a) (b)

Figure 6.11: *The parking garage dataset is in particular difficult: (a) shows the ground truth trajectory from the side. An exemplary failure case is illustrated in (b). See the text for further discussion and explanation.*

Result 6.7. *The proposed approach can be beneficially combined with robust cost functions such as the Huber function. This can help the robust back-end to converge to a correct solution in otherwise difficult situations.*

The Parking Garage Dataset

The parking garage dataset is a particular difficult dataset. Fig. 6.11(a) shows the pseudo ground truth trajectory (generated from the estimation result of the outlier-free data, since no real ground truth is available) from the side. Compare this view with the bird's eye perspective in Fig. 6.3(b). Also notice that the z-axis is scaled differently to better show the spatial structure of the data. The dataset was recorded in a parking garage with four parking decks, which are clearly visible in Fig. 6.11(a). The single decks are connected by only two strands of odometry constraints that originate from the driveways. The dataset thus consists of four main sub-maps (plus the trajectory outside the garage) that are only sparsely interconnected. The problems arising from this sparse connection structure can be seen in 6.11(b). Here the proposed robust back-end failed to deactivate a group of false positive loop closures between the parking levels, leading to a corrupted result. The reason for the failure is the insufficient amount of information (carried by odometry constraints) on the relative pose of the individual decks, due to the small number of constraints between these decks. In simple words, a group of 20 false positive loop closure constraints "voted" for a wrong connection between the decks on the side of the garage, and there are not enough odometry constraints that could "vote" against that false loop closure request. Since the SLAM system has no further knowledge about the structure of the environment, e.g. that certain regions of the map can never intersect or are required to be level, the error introduced by the false positive loop closure requests cannot be resolved by the proposed back-end.

Notice however, that as was the case for the sphere world and Manhattan datasets, at least the local consistency of the map is maintained.

Result 6.8. *Environments consisting of distinctive parts that are only sparsely interconnected are prone to errors if false positive loop closure constraints are established between these parts. Further high-level knowledge about the environment and its spatial structure may be needed to successfully resolve these situations.*

6.6 Runtime and Convergence Behaviour

In general, we expect convergence time to increase with an increasing number of constraints in the dataset. A short analysis in this section reveals the influence of the number of outliers on the required convergence time. Furthermore, the convergence behaviour i.e. how fast the initial error decreased during the optimization process allows some insights into the optimization process.

6.6.1 Methodology

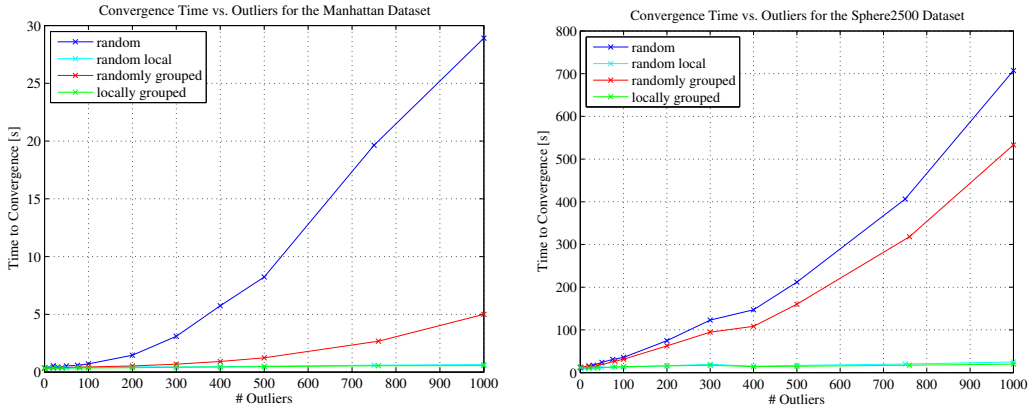
For this part of the evaluation, the same trials that were discussed in the previous section were reused: The datasets were spoiled by different numbers of outliers (from 0 to 1000) using all four policies. The required time until convergence was measured on a Intel Core2-Duo desktop machine running at 2.4 GHz. All measurements were conducted using the robust back-end implementation for the g^{2}o framework.

To examine the convergence behaviour, the 10 trials with 1000 outliers were selected from each dataset and each outlier policy. The χ^2 error (i.e. the error measure the optimizer tries to minimize) was recorded during the optimization and normalized so that the final χ^2 error is mapped to a value of 1. Equally, the required time to convergence was normalized to 1 second. This way, the data from different trials with different parameters can be compared. The normalized timing and χ^2 values were averaged over the 10 trials belonging to the same dataset and outlier policy.

6.6.2 Results and Interpretation

The influence of the outliers on the runtime behaviour can be seen from Fig. 6.12. Interestingly, the results are very different depending on the outlier policy that was used to add the false positives to the datasets. For the two non-local policies, the required time until convergence increases quickly with the number of outliers while for the local policies, the convergence time increases much slower. Obviously the non-local outlier constraints that often connect two very distant places in the dataset are more difficult to resolve.

Result 6.9. *The required time until convergence is largely dependent on the structure of the dataset, the number of outliers and the applied outlier policy. Outlier constraints that only locally connect parts of the trajectory result in faster convergence than constraints connecting distant places of the dataset.*



(a) (b)

Figure 6.12: *Convergence time for different outlier policies for the Manhattan and Sphere world datasets. Notice that the two local policies require much less time for convergence than the non-local policies.*

How the estimation error is minimized during the optimization is visible from Fig. 6.13. These plots reveal that the optimizer behaves very differently, depending on the structure of the dataset and the outlier policy used. For some datasets like the g^{2}o version of the Manhattan dataset or the Intel dataset, the χ^2 error drops quickly and monotonically. For others however, the Gauss-Newton optimizer does not decrease the χ^2 error monotonically, but rather even increases it before finally finding its minimum. This behaviour either indicates the low quality of the initial guess $\mathbf{x_0}$ from which the iterative optimization is started or the difficult non-convex structure of the error function that is to be minimized.

Result 6.10. *The convergence behaviour of the robust back-end depends very much on the structure of the dataset, the quality of the initial guess and the outlier policy. The non-monotonic behaviour of the descent on the error function supports the understanding of the error function's non-convex nature.*

6.7 Performance in the Outlier-Free Case

The proposed robust back-end was specifically designed to mitigate the effects of outliers in pose graph SLAM problems. While we have seen that it outperforms the state of the art systems when outliers are present, it is interesting to compare the performance of both approaches in the outlier-free case.

6.7.1 Methodology

To conduct this comparison, a single trial was calculated for each of the six datasets for the robust and the non-robust back-ends. RPE metrics were calculated and the time

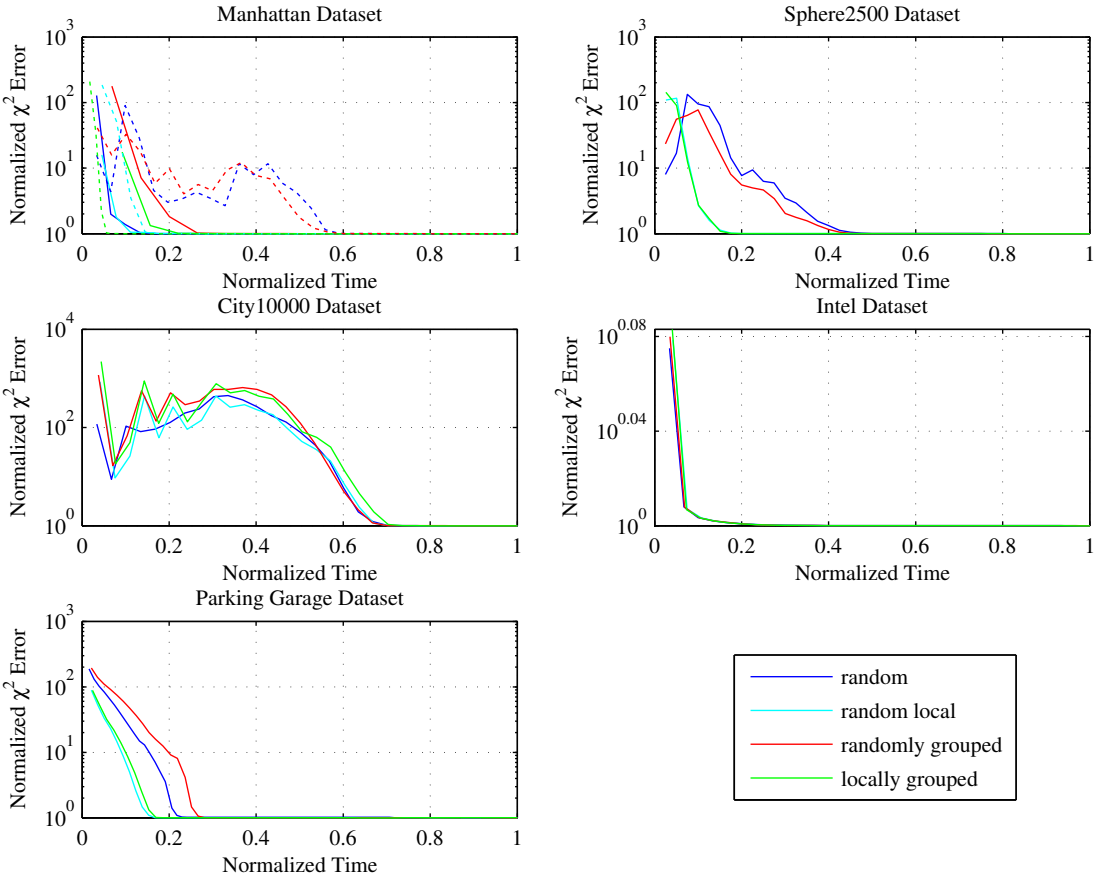


Figure 6.13: *Convergence behaviour for all datasets and all four policies with 1000 outliers. Both the time and the χ^2 errors have been normalized so that the time to convergence and the final residual error correspond to* 1*. The four outlier policies are color coded. For the Manhattan dataset the solid lines correspond to the* g^{2}o *version, while the dashed lines are for Olson's original dataset. Notice that the convergence behaviour is very different depending on the dataset and also the policy in some cases. Due to the Gauss-Newton algorithm used here, the optimizer first steps into regions of higher error before converging towards lower error measures.*

until convergence was measured. All tests were performed on an Intel Core2-Duo desktop machine running at 2.4 GHz.

6.7.2 Results and Interpretation

Table 6.5 summarizes the RPE and convergence time measurements. From these measurements we can immediately state the obvious and expected result:

Result 6.11. *Due to the increased number of variables and constraints that have to be adhered, the robust back-end requires more time for convergence compared to the non-robust formulation.*

Exactly how much more time is needed depends very much on the dataset. Table 6.6 compares the relative increase in convergence time to the relative increase in the number of constraints (Δn) for the different datasets. Remember that by switching from the non-robust to the robust problem formulation, a new variable and a new constraint is added for every loop closure constraint that exists in the dataset. The relative increase in the number of constraints therefore is different for each dataset. From the numbers in Table 6.6 no systematic relation can be established. The increase in convergence time does not only depend on the number of constraints but also on the quality of the initial guess and on the coherence of odometry and loop closure constraints.

The quality of the estimated trajectory can be seen from Table 6.5 as well. For both Manhattan datasets and the City dataset, the robust and non-robust back-end converged to the same low error values, indicating a result that closely fits the ground truth. For the Intel and Parking Garage datasets, the deviation in the RPE metric is marginal and negligible. Furthermore, there is no real ground truth available for these two datasets, since they have been collected in a real-world scenario, in contrast to the other datasets which are simulated. Therefore, the result of the non-robust state of the art back-end for the zero-outlier case was used as a pseudo ground truth in the previous evaluations. Comparing the non-robust results with itself of course leads to RPE values of exactly 0.0 and a slight deviation for the results of the robust approach that does not necessarily have to express a "worse" result. The Sphere dataset shows a slightly worse behaviour. Here the final RPE error measures differ between the two solutions. The reason is not known at this time, but the discrepancy might be caused because the ground truth covariances of the odometry and loop closure constraints are not properly captured by the given covariance matrices in the dataset. Regardless, although the error difference is approximately 4 cm, compared to the absolute size of the dataset, which is a sphere with diameter of 100 m, it is still almost neglectable.

Result 6.12. *In the outlier-free case, the robust-back end converged to the same solution as the state of the art non-robust back-end for five out of six datasets. Only one dataset converged towards slightly worse results that can, however, still be considered tolerable.*

Table 6.5: *Performance comparison for the outlier-free case (timings measured on a Intel Core2-Duo, 2.4GHz, using the* g^{2}o *framework).*

Dataset	Method	RPE_{pos} [m]	RPE_{ori} [°]	Time [s]
Manhattan (Olson's Original)	robust	9.2e-04	0.015	0.38
	non-robust	9.2e-04	0.015	0.21
Manhattan (g^{2}o Version)	robust	9.2e-04	0.015	0.34
	non-robust	9.2e-04	0.015	0.17
City10000	robust	4.8e-04	0.003	7.5
	non-robust	4.8e-04	0.003	1.7
Sphere2500	robust	0.096	0.013	11.9
	non-robust	0.057	0.013	4.7
Intel	robust	2.18e-04	0.002	0.66
	non-robust	0.0	0.0	0.03
Parking Garage	robust	4.81e-08	2e-06	0.63
	non-robust	0.0	0.0	0.36

Table 6.6: *Performance comparison for the outlier-free case.*

Dataset	Δn [%]	Δt [%]
Manhattan (Olson's Original)	37.5	81
Manhattan (g^{2}o Version)	37.5	100
City10000	51.6	341
Sphere2500	49.5	153
Intel	48.7	2100
Parking Garage	73.5	75

6.8 The Influence of the Switch Function Ψ

In the first publication of the robust back-end in [Sünderhauf and Protzel, 2011] the sigmoid function was proposed to be used as switch function, so $\Psi = \Psi^{\text{sigmoid}}$. However, during the evaluations conducted to collect the results presented in this chapter, the linear function Ψ_a^{lin} with $a = 1$ showed superior behaviour when solving the various datasets in batch mode. That means, when the whole dataset is optimized all at once, instead of incrementally adding smaller subsets. In batch mode, when using the sigmoid function as switch function Ψ, the optimization often did not converge to a correct solution in the presence of outliers. The robust back-end using Ψ^{sigmoid} did only converge reliably when it was combined with the Huber cost function using a very small kernel width. This practically led to a linear cost function instead of the conventional squared one.

Although correct results could be reached repeatedly and reliably this way, the linear switch function Ψ_a^{lin} with $a = 1$ (that was used throughout the evaluations in this chapter), is to be preferred. It does not require being used with Huber and thus converges faster.

An indication why the sigmoid function is inferior is that its gradient $\nabla\Psi^{\text{sigmoid}}$ is very small for values where the sigmoid approaches both 0 and 1. Since the switch variables are initialized to values so that $\Psi(s_{ij}) \approx 1$, the optimizer starts on a plateau of a very small gradient that is hard to overcome in the beginning of the optimization. For the linear switch function, the gradient $\nabla\Psi_a^{\text{lin}} = \frac{1}{a}$ is comparably steep and constant in its domain $(0, 1)$ which appears to be of advantage and ensures reliable convergence.

Result 6.13. *The linear switch function Ψ_a^{lin} with parameter $a = 1$ has proven to be superior to the sigmoid function $\Psi^{sigmoid}$. The parameter a shows little influence on the estimation results or the convergence behaviour.*

Despite these findings, the influence of the switch function on the quality of the estimation results, the robustness of the overall SLAM system and the convergence speed should be explored further. A mathematical proof and systematic description of its influences is desirable to better understand the behaviour of the proposed robust back-end.

6.9 Summary of the Evaluation and First Conclusions

The evaluation has shown that the proposed robust back-end is capable of solving large scale SLAM problems in 2D and 3D for a variety of datasets although they contain a large number of false positive loop closure constraints. The proposed extension to optimization-based SLAM correctly identifies and disables false positive loop closure constraints, while maintaining the correct (true positive) constraints. Even in the case of failure, the system degrades gracefully. If the proposed back-end fails to disable a false positive constraint, the resulting map will be globally distorted, but retains its local consistency. Furthermore, the presented system can be successfully combined with other measures of increased robustness such as the Huber cost function that decreases the influence of outliers. The free parameter $\mathbf{\Xi}_{ij}$ which constitutes the switch prior covariance can be chosen safely from a broad range of values.

As the evaluation revealed, the presented novel approach largely improves the robustness of optimization-based pose graph SLAM. When combined with a suitable front-end, the overall SLAM system becomes tolerant and robust against errors in the place recognition. This removes the requirement for the data association / place recognition algorithm in the front-end to work perfectly (with 100% precision). The data association or place recognition module can be kept simple and fast, since a reasonable rate of false positives among the loop closure constraints is acceptable for the robust back-end. Furthermore, no hard data association decisions are necessary in the front-end, the optimizer can take back decisions at any time.

The proposed robust problem formulation can be understood as transferring parts of the responsibility for correct data association from the front-end into the back-end. The robustified back-end optimizer can change the topological structure of the pose graph representation during the optimization process by adapting the covariances of individual

loop closure constraints. Therefore, it can account for possible data association errors and ignore erroneous loop closure constraints.

7

Applying the Robust Back-End in a Complete SLAM System on a Real-World Dataset

The datasets evaluated so far were either simulated or medium sized real-world datasets. Although these datasets already contained several thousand poses and constraints, this chapter shows results of the proposed robust back-end in an even larger real-world scenario.

To demonstrate this, the St. Lucia dataset that was first presented in [Milford and Wyeth, 2008] was chosen. It consists of video footage taken on a 66 km long course along the roads in a suburb of Brisbane, Australia. The camera was mounted on top of a car that drove through the street network for a little more than 1:40 hours, resulting in 57,858 image frames which correspond to distinct poses. No additional information is available, notably no GPS, or odometry information.

The front-end part of our SLAM system therefore has to extract inter-frame motion information and detect loop closures solely from the camera images. Coarse *visual odometry* information was extracted from the images using image profile matching [Sünderhauf and Protzel, 2010a]. Although this technique is rather simple, the extracted inter-frame motion estimates provide sufficient metric information for the SLAM back-end. Potential loop closures were detected by a light-weight place recognition system we call BRIEF-Gist [Sünderhauf and Protzel, 2011]. This rather simple place recognition system has a low false-negative rate, but a pretty high false-positive rate. However, due to the robust problem formulation, the optimizer is able to deactivate these wrong loop closures and converge to a correct solution.

In the following, the parts of the front-end are described in more detail. As we are going to see, the techniques used in the front-end are rather simple when compared to other

state of the art approaches for visual odometry or appearance-based place recognition and are highly prone to errors. For real applications one might choose more sophisticated approaches, but the intention of this chapter is to show that a working SLAM system with a very simple front-end is possible when using the proposed robust back-end.

7.1 The Front-End

The two main tasks of the SLAM system's front-end for the St. Lucia urban scenario is the extraction of odometry information and place recognition. This section reviews and describes the approaches that have been used to accomplish both tasks.

7.1.1 Visual Odometry by Image Profile Matching

Since no other sources of odometry information are available for the dataset, all information on the movement between two camera frames have to be extracted from the camera images themselves.

To demonstrate that even very simple approaches can be successful, I chose a visual odometry estimation technique based on image profile matching. This approach is able to rapidly extract coarse translation and rotation information from the video stream and is very similar to the approach described in [Milford and Wyeth, 2008] in the context of RatSLAM, a biologically motivated SLAM system. Here I want to shortly review the general idea behind this approach.

A horizontal image profile is a vector whose elements contain the column sums of the original image's grey scale pixel values. A vertical profile contains the row sums respectively. Thus, a horizontal profile of a $w \times h$ image i is defined as the w-vector

$$\pi_{\text{hor}}(x) = \sum_{y=0}^{h-1} i(x, y) \tag{7.1}$$

and likewise the vertical profile is a h-vector:

$$\pi_{\text{vert}}(y) = \sum_{x=0}^{w-1} i(x, y) \tag{7.2}$$

In order to extract rotation and velocity information from the image profiles, a simple correlation technique can be used. The horizontal profiles of two consecutive images are shifted against each other and an error measure is calculated by summing up the element-wise differences of the overlapping parts of the profiles. The error for two profiles at a given shift s as defined by [Milford and Wyeth, 2008] is:

$$e_{(\pi^{[1]},\pi^{[2]},s)} = \frac{1}{w - |s|} \sum_{n=1}^{w-|s|} |\pi^{[1]}_{n+\max(s,0)} - \pi^{[2]}_{n-\min(s,0)}| \tag{7.3}$$

Figure 7.1: *A typical scene from the St. Lucia dataset [Milford and Wyeth, 2008] and the image profiles we used to calculate coarse visual odometry. The horizontal profiles of the green boxes estimate the rotation and the vertical profile in the blue box is used to estimate the translation.*

By finding the shift that minimizes this error, the estimated horizontal rotation in pixels between the two images is determined:

$$s^* = \underset{s \in [-\delta, \delta]}{\operatorname{argmin}} \; e_{(\pi^{[1]}, \pi^{[2]}, s)} \tag{7.4}$$

Here δ bounds the search space for the optimal shift. A certain tradeoff between search space size and efficiency has to be found. If the search space is too small, the optimal shift may not be found (in case of more rapid rotations). If it is too large, too much time is wasted in the search. Empirically, $\delta = \frac{w}{10}$ gives good results, but this depends very much on the expected inter-frame rotations.

After calibrating the camera, i.e. establishing a relation between pixels and the camera's field of view, the rotation can be expressed in degrees.

In comparison to the approach described by [Milford and Wyeth, 2008] we slightly changed the method of estimating the rotation by using two measurement fields, one to the left of the center of optic flow expanse, one to the right of it (see Fig. 7.1 for illustration). If the flow calculated by these two fields points into the same direction, we are rotating. Otherwise, if the flows point in different directions, we are moving straight ahead. Further own work on how sparse optical flow information can be computed using image profiles speeded up by integral images was published in [Sünderhauf and Protzel, 2009].

7.1.2 Place Recognition Using BRIEF-Gist

The ability to recognize known places is an essential competence of any intelligent system that operates autonomously over longer periods of time. Approaches that rely on the visual appearance of distinct scenes have recently been developed and applied to large scale SLAM scenarios. Reliable place recognition is a hard problem, especially in

large-scale environments. Repetitive structure and sensory ambiguity constitute severe challenges for any place recognition system. Recent developments in appearance-based place recognition therefore aimed at reaching a high recall rate at 100% precision, i.e. they concentrated on preventing false positives. This of course leads to computationally involved, very complex systems.

Given the robust SLAM back-end introduced in this thesis, the need of reaching a precision of 100% during the data association (i.e. place recognition) process is eliminated. The place recognition system in the front-end can therefore be kept simple and focused on a high recall rate, since a reasonable number of false positive loop closures is acceptable.

In [Sünderhauf and Protzel, 2011] we therefore proposed BRIEF-Gist, an appearance-based place recognition system that builds upon the BRIEF descriptor by Calonder et al. [Calonder et al., 2010]. BRIEF-Gist is a simple and light-weight place recognition with low computational costs. Our evaluation in [Sünderhauf and Protzel, 2011] showed that it can compete with state-of-the-art appearance-based place recognition systems like FAB-Map [Cummins and Newman, 2008]. In contrast to FAB-Map, BRIEF-Gist does not require a learning phase to acquire a vocabulary of visual words.

The BRIEF Descriptor

BRIEF (*Binary Robust Independent Elementary Features*) has been introduced as an efficient descriptor for feature points (or keypoints) by Calonder et al. [Calonder et al., 2010]. It was found to be superior to the established SIFT [Lowe, 2004] or SURF [Bay et al., 2006] descriptors, both in recognition performance and runtime behaviour.

The BRIEF-descriptor is a bit-vector (e.g. of length 256) that is built by simple binary tests on a subset of the pixels surrounding the keypoint center. Calonder et al. [Calonder et al., 2010] suggest using a simple comparison of pixel intensity values: For a descriptor of length n (e.g. $n = 256$), n pixel-pairs $(p_{k,1}, p_{k,2})$ are chosen in the local neighborhood (e.g. 48×48) of the keypoint center. The k-th bit in the descriptor is set to 1 if $p_{k,1} < p_{k,2}$ and set to 0 otherwise. This way, the descriptor can be built very efficiently. Notice that the same neighboring pixels will be chosen for all descriptors.

Comparing two descriptors D_1 and D_2, i.e. determining their similarity, can be performed very efficiently using the Hamming distance (which is the L_1 norm). As the descriptors are simple bit-vectors, their Hamming distance can be calculated by

$$\|D_1 - D_2\|_H = \text{bitsum}(D_1 \oplus D_2) \tag{7.5}$$

where $\oplus$ is the binary XOR operation and bitsum$(\cdot)$ counts the set bits in a bit-vector.

The BRIEF-Gist Scene Descriptor

The good recognition performance of BRIEF on local keypoints reported by [Calonder et al., 2010] inspired us to use BRIEF as a holistic descriptor for a complete image. We call this approach *BRIEF-Gist*.

The implementation is very straight-forward: To calculate the BRIEF-Gist descriptor, we first downsample the image to a suitable size close to the descriptor patch size

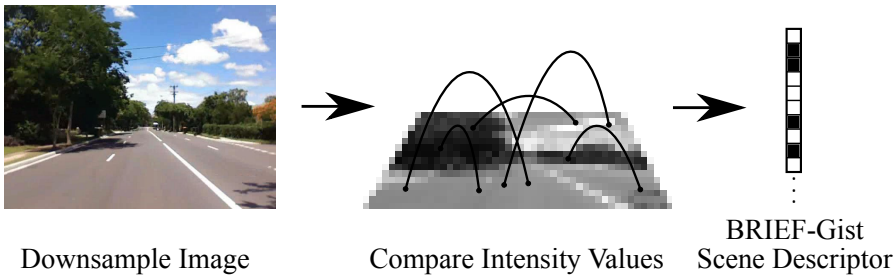


Figure 7.2: *The BRIEF-Gist descriptor is a very simple and fast scene descriptor that can be applied in appearance-based place recognition. It is calculated over a complete image. First the image is downsampled to a comparably small size of e.g.* 60×60*. Then the BRIEF descriptor [Calonder et al., 2010] is calculated on that downsampled image by performing comparisons on the intensity values of* n *randomly chosen pixel pairs* $(p_{k,1}, p_{k,2})$ *. This results in a bit vector of length* n *where the* i*-th bit is set if* $p_{k,1} < p_{k,2}$*.*

(e.g. 60×60 pixel). Then we calculate the BRIEF descriptor around the center of the downsampled image using OpenCV's [Bradski, 2000] implementation. Fig. 7.2 illustrates the concept. Another idea is to partition the image in $m \times m$ equally sized tiles. This tiled BRIEF-Gist descriptor is calculated by downsampling the image to a size of $m \cdot s \times m \cdot s$ pixel, where s is the descriptor patch size, e.g. $s = 48$. Then a BRIEF descriptor is calculated for each of the m^2 tiles separately, resulting in m^2 bit-vectors that are stacked to gain the final descriptor vector.

BRIEF-Gist descriptors can be calculated and compared extremely fast: Using a standard desktop PC (Core 2 Duo) and OpenCV 2.2, calculating the 64 bytes long BRIEF-Gist descriptor takes only 1 ms, including the necessary image downsampling and color conversion. The calculation of the BRIEF descriptor itself takes only 0.05 ms. Calculating the similarity between two descriptors according to (7.5) is performed in 0.001 ms.

The similarity between two scenes, respectively their distance in the descriptor space is given by the distance of their BRIEF-Gist descriptors as defined in (7.5). Notice that depending on how the scene similarity information is processed further, it can be thresholded to gain a binary decision on whether two scenes are identical and thus a loop closure constraint has to be introduced between them. In the context of the proposed robust back-end, a thresholding would not be necessary, instead the distance between two scene descriptors D_i and D_j can be used to specify the strength of the resulting loop closure constraint, e.g. by adapting the corresponding $\mathbf{\Xi}_{ij}$. However, this has not been conducted and is left for further work. In the results presented next, a threshold has been applied to decide whether to introduce a loop closure constraint between two scenes or not.

A detailed evaluation of the BRIEF-Gist descriptor can be found in [Sünderhauf and Protzel, 2011]. Table 7.1 shortly summarizes the results and comparisons against the FAB-MAP system [Cummins and Newman, 2008]. Exemplary true positive and false

Table 7.1: *Feature comparison between BRIEF-Gist and FAB-MAP*

Invariancy	BRIEF-Gist	FAB-Map
lighting conditions	yes	yes
traversal direction	no	yes
small / large rotations	yes / no	yes / yes
small / large displacement	yes / no	yes / yes
Requirements		
learning phase	no	yes
complex implementation	no	yes
Results		
Oxford City Dataset	recall 32%	recall 16% / 31% / 37%
large-scale SLAM	yes	yes

positive place recognitions of BRIEF-Gist can be regarded in Fig. 7.3 and Fig. 7.4 respectively.

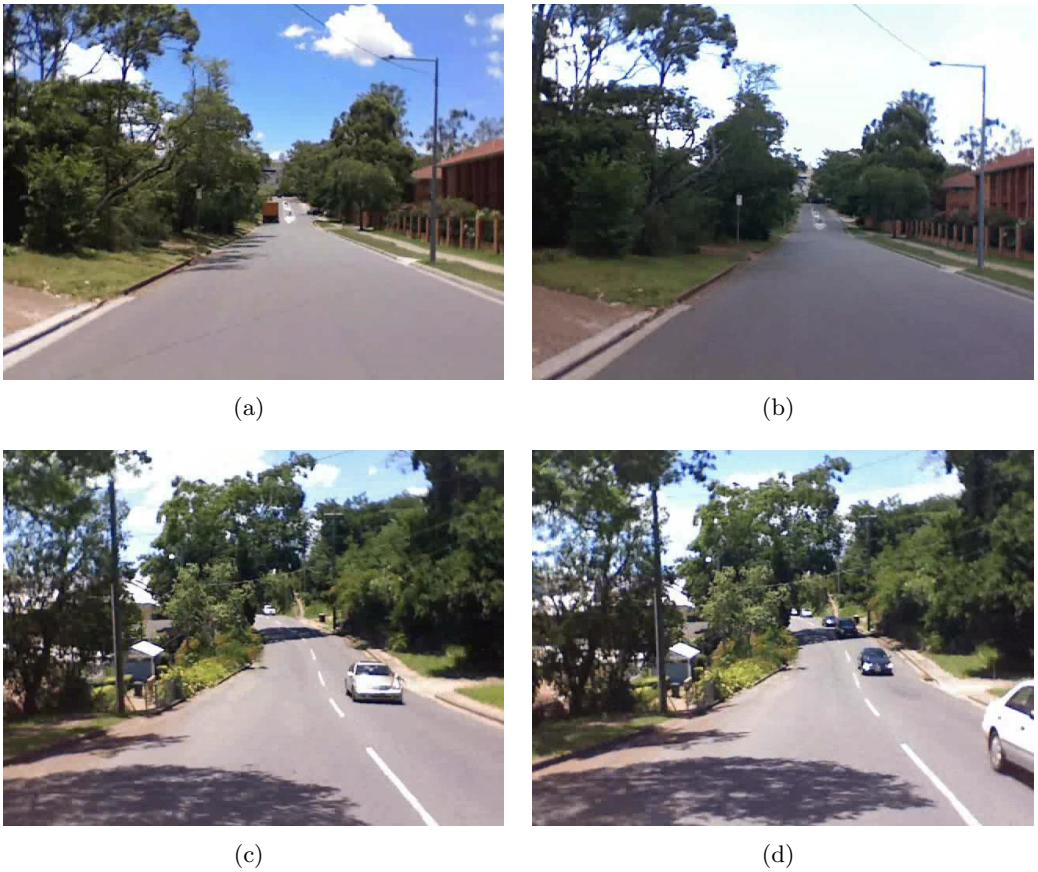
7.2 Results of the Complete SLAM System on the St. Lucia Dataset

The BRIEF-Gist descriptor provides a simple measure of scene similarity that was used to perform place recognition. The front-end described in the previous section also provided coarse visual odometry measurements. Therefore, the necessary information for the back-end are available to solve for the maximum a posteriori trajectory estimate. The GTSAM/ iSAM implementation of the proposed robust back-end was used to conduct the calculations since it easily allowed to perform incremental processing at the time the experiments were done. Notice that meanwhile, the g^{2}o framework is available which can perform incremental updates as well.

The dataset had to be solved incrementally, e.g. feeding 200 poses into the optimizer at a time. In batch processing, i.e. optimizing all 57858 poses all at once, neither iSAM nor g^{2}o could converge to a correct solution. The reason for this behaviour is supposedly the very coarse odometry which leads to a very bad initial guess of the trajectory.

Two different sets of loop closure constraints were used in the experiments that used different settings of the BRIEF-Gist place recognition algorithm. Fig. 7.5 shows the resulting maps after performing SLAM on the whole dataset for these two sets of loop closure constraints. As ground truth information are not available, we can only provide a qualitative analysis of the results. A coarse structure of the road network which was manually derived from Google Maps is shown as well for qualitative comparison.

It is apparent that the general structure of the environment has been correctly captured by the SLAM back-end. The front-end based on BRIEF-Gist identified many false positive



(a) (b)

(c) (d)

Figure 7.3: *Examples for correctly matched scenes from the St. Lucia dataset. Despite the significant change in appearance (lighting conditions, moved cars), BRIEF-Gist is able to correctly recognize these scenes and matched the images on the left with those on the right side.*

(a) (b)

(c) (d)

Figure 7.4: *Examples for erroneously matched scenes (false positives) from the St. Lucia dataset. BRIEF-Gist incorrectly matched scenes from the left column with those on the right.*

loop closures that were rejected by the back-end during the optimization process. These false positive loop closures are visible as grey links in the map of Fig. 7.5. A small number of loops have not been closed, these are false negative loop closures. In these cases, BRIEF-Gist was not able to recognize the scenes.

The vast majority of the loop closures in the dataset was correctly recognized. This is especially impressive as the scenes over the dataset are visually very similar and ambiguous and even for a human observer it is hard to identify all loop closings without mistake. Fig. 7.3 shows a number of exemplary true positive place recognitions.

This real-world dataset shows why a naive approach that deactivates loop closure constraints based upon a simple error threshold cannot work: Some of the loops (especially the first and the 5th) are very large and the accumulated odometry errors already exceed several hundred meters. In order to accept these loop closure, any threshold would have to be set to such high values that most of the wrong loop closure candidates (that require shorter loops) would be erroneously accepted.

7.3 Summary

As we have seen, the proposed robust back-end performed well on a very demanding large-scale real-world dataset. Despite only very rough odometry information were available and – depending on the parametrization of the front-end – a large number of false positive loop closings were detected, the general structure of the environment could be successfully recovered. Although the map is not absolutely metrically correct, it can be considered *semi-metric*, which means that it captures the local metric properties and topology of the environment. The results achieved here are comparable to those of [Milford and Wyeth, 2008; Milford, 2008] that applied RatSLAM, a biologically inspired SLAM system, to filter false positive loop closure constraints.

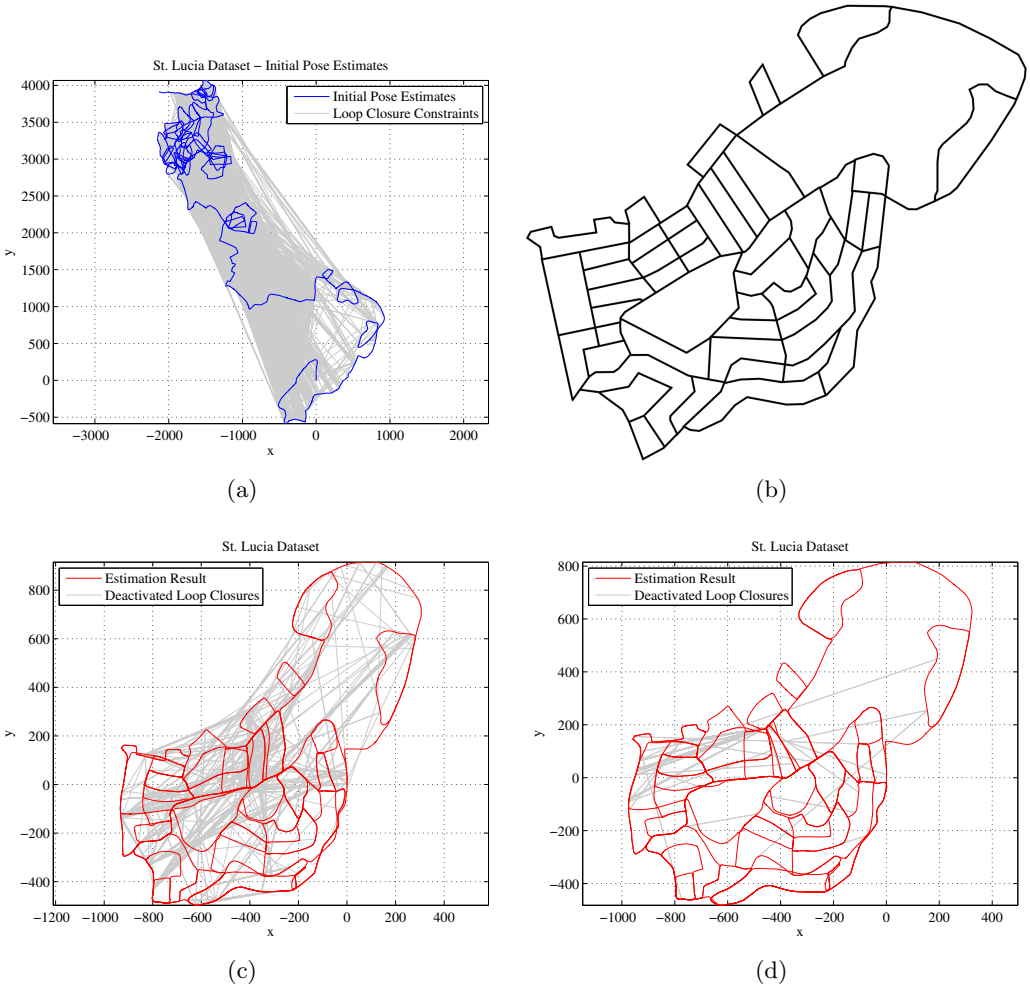


Figure 7.5: *The St. Lucia dataset [Milford and Wyeth, 2008] consists of video footage recorded from a car moving through a network of urban streets. (a) Shows the initial guess of the* 66 km *long trajectory according to visual odometry along with loop closure constraints provided by the place recognition system based on BRIEF-Gist. (b) illustrates the general layout of the driven streets. Notice that this was manually derived from Google Maps. (c) and (d) are the trajectories estimated by the proposed robust back-end for two different sets of loop closure constraints, gained from two parametrizations of the place recognition system. Notice that the general topology of the environment was correctly recovered despite the bad odometry and place recognition information.*

8

Applications Beyond SLAM – Multipath Mitigation in GNSS-based Localization Problems using the Robust Back-End

The previous chapters evaluated the proposed robust back-end in a variety of SLAM scenarios and demonstrated its feasibility. This chapter and the next discuss the applicability of the robust graph-based optimization to other fields, different from SLAM. As we will see, the general idea of altering the topology of a factor graph during the optimization (or the equivalent probabilistic interpretation of adapting the information matrices associated with some of the constraints during the optimization), is rather universal and can possibly be applied in other optimization-based problem domains where outliers can occur.

Transfering the concepts found and approved in one problem domain to another, different domain is worthwhile and underlines the overall value of the found original solution. This chapter therefore explores the application of the proposed robust optimization scheme for outlier mitigation in a GNSS[40]-based localization problem. The outliers in this scenario are miscalculated pseudoranges that are caused by so called *multipath* effects. As we are going to see, these multipath affected pseudorange measurements play the same critical role in GNSS-based localization as false positive loop closure constraints played in the SLAM scenario.

In the following, the GNSS-based localization problem is shortly introduced, before I demonstrate how it can be expressed as a least squares optimization problem and modelled using a factor graph. The reader unfamiliar with GNSS-based localization and the related problems is referred to the broad apparatus of literature dedicated to

[40]Global Navigation Satellite System, e.g. GPS, Galileo, GLONASS.

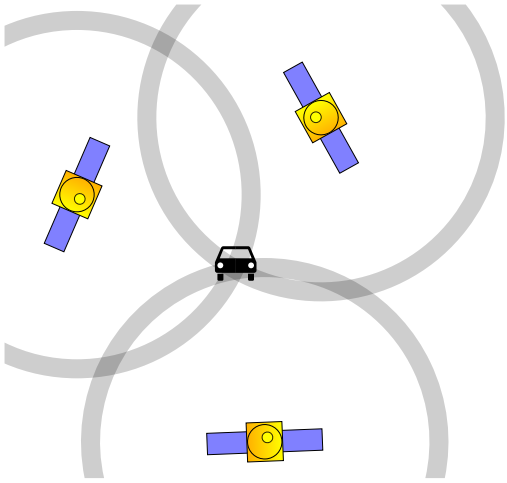


Figure 8.1: *Illustration of GNSS-based localization. In simple words, each pseudorange measurement ρ_{ij} creates a sphere with radius ρ_{ij} around the known satellite position in space. In the ideal case, the receiver's location is determined to be the intersection point of the n observed spheres.*

this field for more in-depth information. For the introductory explanations given in the following, [Grewal et al., 2007] was the most useful source.

8.1 GNSS-based Localization – A Gentle Introduction

From a roboticist's or SLAM researcher's perspective, GNSS-based localization is a 3D localization problem with range-only observations to distant known landmarks. The landmarks in this scenario (see Fig. 8.1) are the satellites which are uniquely identifiable via their transmitted PRN[41] code. The positions of the observed satellites / landmarks in space are known since each satellite transmits ephemeris parameters which describe its orbit. However, the ranges from the receiver to the satellites are not observed directly, but are rather calculated from the signal transit time. This is done by comparing the timestamp that is included in the received signal and specifies when the signal was sent from the satellite, with the local time at the receiver in the moment the signal is received:

$$\rho = c \cdot (t_{\text{recieve}} - t_{\text{transmit}}) \tag{8.1}$$

where c is the speed of light. The quantity that is to be estimated from these *pseudoranges* ρ is the location of the receiver in 3D space, $\mathbf{x} = (x, y, z)^\mathsf{T}$. Since the state space has three degrees of freedom, one would expect three observations to be sufficient to solve the problem. However, four observations are necessary for a unique solution in general. This will become clear in a moment.

[41] Pseudo Random Number / Pseudo Random Noise

Among other sources of error, receiver clock errors have probably the largest implications on the design and working principles of GNSS systems. They occur because it is not possible (or at least not feasible for economic reasons) to keep the receiver clocks exactly synchronised to the transmitter clocks: GNSS satellites like the GPS satellites are equipped with highly accurate atomic clocks and are monitored from the ground segment to correct for even the slight time drifts that occur with these high-precision atomic clocks. On the other side, the receivers have to be cheap and lightweight and therefore the clocks used in these devices are comparably inaccurate. The dilemma is solved elegantly by including the unknown receiver clock error into the state space that is to be estimated: $\mathbf{x} = (x, y, z, \delta_{\text{clock}})^{\mathsf{T}}$. This way, the correction term δ_{clock} covers for the differences between the GPS time used by the satellites and the local receiver time. Since now there are four unknowns that have to be estimated, a minimum of four pseudorange satellite observations are necessary.

Each of the observation therefore gives rise to an equation of the following type:

$$\rho_{tj} = \|\mathbf{x}_t^{\text{x,y,z}} - \mathbf{x}_{tj}^{\text{SAT}}\| + \mathbf{x}_t^{\delta^{\text{Clock}}} + \delta_t \tag{8.2}$$

$$= \sqrt{(x_t^{\text{RECV}} - x_{tj}^{\text{SAT}})^2 + (y_t^{\text{RECV}} - y_{tj}^{\text{SAT}})^2 + (z_t^{\text{RECV}} - z_{tj}^{\text{SAT}})^2} + \delta_t^{\text{Clock}} + \delta_t \tag{8.3}$$

With four measurements ρ_{tj}, the system of equations can be solved for the sought vehicle state vector $\mathbf{x}_t$. If more than the necessary four measurements are available, the system is overdetermined and a least squares solution is applied to solve for $\mathbf{x}_t$. Notice that the term δ_t covers for other systematic errors I am going to shortly describe below. Notice further that the receiver clock error δ_t^{Clock} is given in meters instead of seconds (by multiplying the clock error given in seconds by c, the speed of light).

8.1.1 Systematic Errors

Besides the receiver clock errors explained above, other important sources of systematic error are:

- ionospheric propagation errors
- tropospheric propagation errors
- satellite clock errors
- ephemeris errors

The ionospheric and tropospheric errors occur because the propagation speed of the signal is not equal to the vacuum speed of light. Furthermore, the real speed is not constant when passing through the atmosphere, but is rather influenced by certain ionospheric conditions such as ionisation due to solar radiation and tropospheric conditions such as humidity, temperature and pressure. While ionospheric effects are known to be frequency dependent and can thus be corrected by using two different signal carrier frequencies, tropospheric effects cannot be mitigated this way.

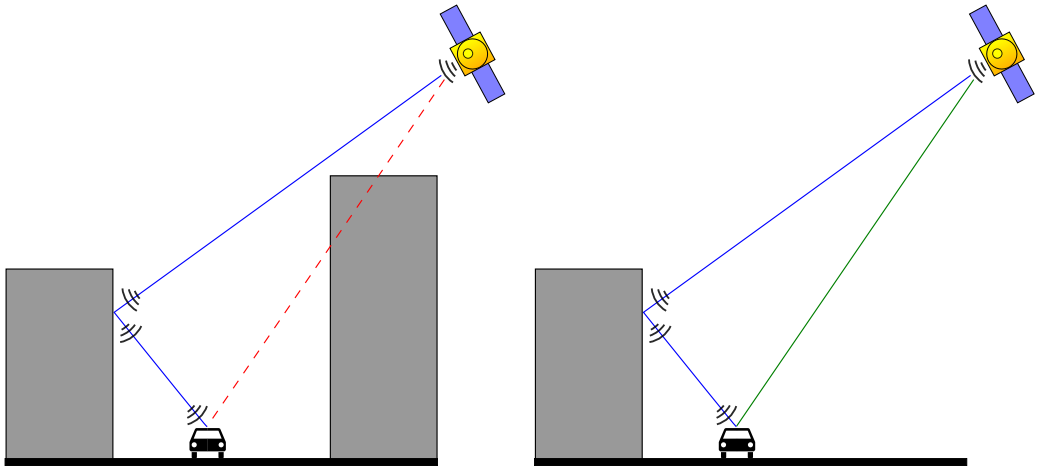


Figure 8.2: *The multipath problem in an urban canyon: (a) The direct line of sight (red dashed line) from the satellite to the receiver on the ground is blocked by a building. The signal reaches the receiver via a reflection (blue line), causing a range error of the observed pseudorange. (b) Although the satellite can be observed directly without occlusions (green line), the signal is received a second time via a reflection on a nearby building (blue line). In both cases, the resulting position estimate can be severely biased.*

Satellite Based Augmentation Systems (SBAS) like WAAS[42] in North America or EGNOS[43] in Europe provide additional information that allow the receivers on the ground to correct e.g. for ionospheric delays. Ephemeris and satellite clock errors are corrected by the ground segment which closely tracks and monitors the satellite's orbits and internal clocks and uploads correction data which is then broadcast by the satellites.

8.1.2 Multipath Errors

A common challenge for GNSS-based localization is the multipath problem, that occurs for instance in urban areas with high buildings blocking the direct line of sight to at least some of the available satellites. This scenario is also referred to as *urban canyon*. Fig. 8.2(a) illustrates the basic problem: Although the direct line of sight to a satellite is blocked, its signal may still reach the receiver on the ground via one or several reflections on building structures or the ground. Since the signal path is longer for the reflected signal, ranging errors occur that can either prolongate the observed pseudorange or, due to correlation effects, shorten it[44]. Multipath effects can also occur when the direct line

[42] Wide Area Augmentation System
[43] European Geostationary Navigation Overlay Service
[44] That seems counter-intuitive at first, see [Grewal et al., 2007, ch. 5.5] for an explanation.

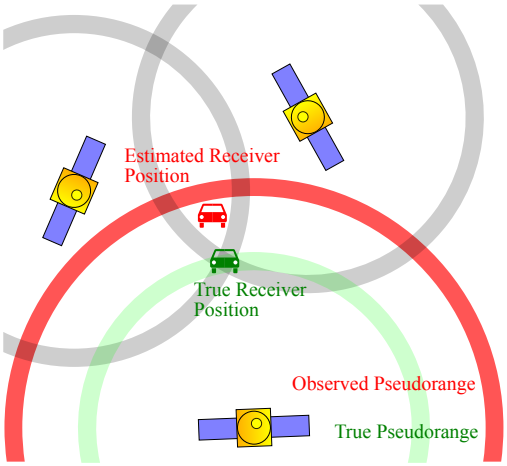


Figure 8.3: *Effects of multipath errors on the receiver's position estimate: A pseudorange measurement that is subject to multipath ranging errors can severely bias the least squares position estimate.*

of sight is free, such as in Fig. 8.2(b). In this situation, the signal is received directly, but is also reflected on a building or another structure in the vicinity of the receiver. Hence the signal is received multiple times and interferes with itself at the receiver's antenna, leading to correlation errors.

In a sense, those observations that are subject to multipath effects can be considered outliers that can severely bias the least squares estimate of the receiver's position on the ground. Fig. 8.3 illustrates this effect. Comparably to what was illustrated in Fig. 3.10(b) in the chapter about least squares optimization in the presence of outliers, a single multipath measurement can lead to a defective position estimate. The problem gets worse if one considers that in urban environments not only one, but several satellite observations might be affected by multipath effects.

8.2 Multipath Identification and Mitigation – Related Work

Different approaches for multipath mitigation are found in the literature. [Grewal et al., 2007] divides the approaches into different strategies: Spatial processing techniques try to optimize the receiver antenna design (e.g. using choke ring antennas or antenna arrays) to decrease the possibility of receiving a reflected signal or incorporate information gained by long-term observations (spanning from one day to another). The second type of techniques mentioned by [Grewal et al., 2007] are time-domain approaches that try to identify multipath errors by post-processing and evaluating the received signals from the satellite in the receiver. These approaches are rather low-level and operate on the radio signal level.

It is curious that RANSAC-like algorithms (see section 3.6) seem to have only recently

found their way into the GNSS-community [Schroth et al., 2008; Tu et al., 2011]. Roughly similar approaches have been summarized under the term RAIM[45] but appear to have mostly expected only a *single* outlier among the satellite observations [Tu et al., 2011] which is inadequate given the increasing number of usable satellites, especially when considering multi-constellation applications. [Qiang et al., 2007] however discusses the application of RAIM in the occurrence of several simultaneous satellite failures. [Meguro et al., 2009] proposes to actively determine occluded satellites with the help of an omnidirectional infrared camera mounted on the vehicle.

[Obst et al., 2011; Bauer, 2011] proposed to identify multipath observations by using information on the local building structure, i.e. a database of building positions and dimensions. This way, given an estimate on the receiver's position on the ground, raytracing and similar proposed approaches can determine whether the direct line of sight to a received satellite is blocked by a building (and thus the signal was received via a reflection, causing multipath range errors). Clearly this method is well suited for multipath signal rejection, but requires considerable additional knowledge about the environment and a good initial guess of the receiver's position on the ground to perform the raytracing.

8.3 Modelling the GNSS-based Localization Problem as a Factor Graph

The least squares optimization problem that has to be solved when estimating the receiver position from a number of satellite observations can be easily modelled as a factor graph. In the most simple formulation, the vehicle position estimates are treated as conditionally independent. However, this notation can be extended by introducing motion models or other state transition relations.

Fig. 8.4 illustrates possible layouts of the factor graph for the GNSS-based localization problem. Remember that the large nodes represent the unknown variables, hence the sought state estimates and the small nodes represent the probabilistic factors that govern these variables. The vehicle state nodes and the different possible factors are explained in the following.

8.3.1 The Vehicle State Vertices

The state space contains at least the 3D position of the vehicle as well as the receiver clock error, leading to a state space that is at least 4-dimensional:

$$\mathbf{x} \in \mathbb{R}^4 = \begin{pmatrix} x & y & z & \delta^{\text{clock}} \end{pmatrix}^{\mathsf{T}} \tag{8.4}$$

This state space may be extended by jointly estimating the vehicle orientation θ, velocity v, rotation rate ω or the clock error drift $\dot{\delta}^{\text{clock}}$, depending on the requirements and which other sensors are used. Estimating the vehicle acceleration a or road curvature $1/r$ would also be possible.

[45] Receiver Autonomous Integrity Monitoring

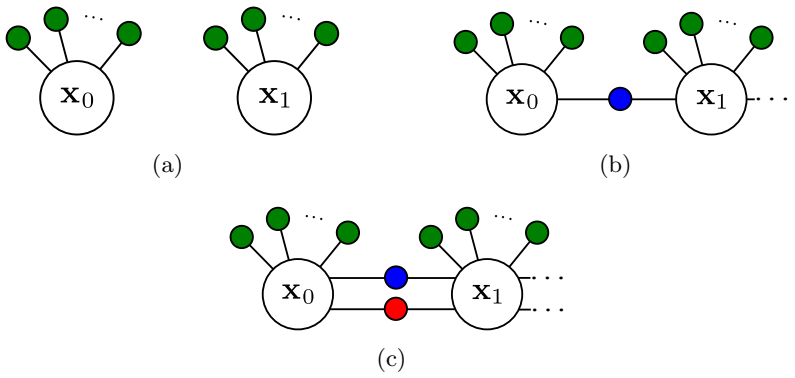


Figure 8.4: *Two vehicle state nodes with their associated pseudorange factors (green). (a) in the most general graphical model, there are no connections between the vehicle state nodes. In (b), a state transition factor joins two successive vehicle nodes. In (c), an additional motion model is incorporated.*

8.3.2 The Pseudorange Factor

A number of satellites are observed from every vehicle state $\mathbf{x}_t$, each providing a pseudorange measurement ρ_{tj}. Given the receiver position $\mathbf{x}_t^{x,y,z}$ and the position of the observed satellite $\mathbf{x}_{tj}^{\text{SAT}}$, the expected pseudorange measurement is given by the measurement function

$$h(\mathbf{x}_t, j) = \|\mathbf{x}_{tj}^{\text{SAT}} - \mathbf{x}_t^{x,y,z}\| + \delta^{\text{EarthRotation}} + \delta^{\text{Atmosphere}} + \mathbf{x}_t^{\delta^{\text{clock}}} \tag{8.5}$$

Notice that we have seen this type of equation before in (8.3). The terms $\delta^{\text{EarthRotation}}$ and $\delta^{\text{Atmosphere}}$ correct ranging effects caused by the earth's rotation and atmosphere (ionospheric and tropospheric propagation errors). $\delta^{\text{EarthRotation}}$ is given by

$$\delta^{\text{EarthRotation}} = \omega^{\text{Earth}} \frac{x_{tj}^{\text{SAT}} \cdot y_t - y_{tj}^{\text{SAT}} \cdot x_t}{c} \tag{8.6}$$

with ω^{Earth} the earth's rotation rate and c the speed of light.

If we assume the measured pseudorange ρ_{tj} is given by the measurement function $h(\mathbf{x}_t, j)$ plus a zero-mean Gaussian error term, thus

$$\rho_{tj} = h(\mathbf{x}_t, j) + \mathcal{N}(\mathbf{0}, \Sigma_{tj}) \tag{8.7}$$

then the error function of a single pseudorange factor is given as

$$\|\mathbf{e}_{tj}^{\text{pr}}\|_{\mathbf{\Sigma}_{tj}}^2 = \|h(\mathbf{x}_t, j) - \rho_{tj}\|_{\mathbf{\Sigma}_{tj}}^2 \tag{8.8}$$

with $\mathbf{\Sigma}_{tj}$ the covariance associated to the pseudorange measurement ρ_{tj}. Notice that minimizing above error over $\mathbf{x}_t$ corresponds to maximizing the likelihood function $L(\rho_{tj}|\mathbf{x}_t) \sim \mathcal{N}(h(\mathbf{x}_t, j), \mathbf{\Sigma}_{tj})$.

8.3.3 Additional Factors

Besides the obligatory pseudorange factors, additional factors can be modelled to incorporate more information or sensor data.

The State Transition Factor

A possible way to account for the receiver clock error is to model it as either constant over time, i.e. $\delta_{t+1}^{\text{Clock}} = \delta_t^{\text{Clock}} + \lambda$ where λ is a zero-mean Gaussian. Another possibility is to use a constant drift model, i.e.

$$\delta_{t+1}^{\text{Clock}} = \delta_t^{\text{Clock}} + \dot{\delta}_t^{\text{Clock}} \Delta t + \mathcal{N}\left(0, \sigma_t^{\text{Clock}}\right) \tag{8.9}$$

$$\dot{\delta}_{t+1}^{\text{Clock}} = \dot{\delta}_t^{\text{Clock}} + \mathcal{N}\left(0, \sigma_t^{\text{ClockDrift}}\right) \tag{8.10}$$

For the latter case, the error function associated with the state transition factor is

$$\|\mathbf{e}_t^{\text{st}}\|_{\mathbf{\Sigma}_t^{\text{st}}}^2 = \left\| \begin{pmatrix} \delta_t^{\text{Clock}} + \dot{\delta}_t^{\text{Clock}} \Delta t \\ \dot{\delta}_t^{\text{Clock}} \end{pmatrix} - \begin{pmatrix} \delta_{t+1}^{\text{Clock}} \\ \dot{\delta}_{t+1}^{\text{Clock}} \end{pmatrix} \right\|_{\mathbf{\Sigma}_t^{\text{st}}}^2 \tag{8.11}$$

$\mathbf{\Sigma}_t^{\text{st}} = \text{diag}(\sigma_t^{\text{Clock}}, \sigma_t^{\text{ClockDrift}})$ is, as usual, the covariance matrix associated with the state transition factor at time t.

The Motion Model Factor

A variety of motion models can be applied in the context of vehicle localization or motion estimation. For instance, [Schubert et al., 2011] lists and evaluates six different types. As an example, the *constant velocity and turn rate* model (CTRV) is used here to formulate a motion model factor. Notice however that any other model from [Schubert et al., 2011] could be used as well.

For the CTRV model, the vehicle state space has to be extended to include the vehicle orientation θ, the velocity v and the turn rate ω. Following [Schubert et al., 2011], the motion model operates in 2D space only, thus not affecting the z coordinate of the vehicle. Therefore, the orientation is specified by only one angle, instead of three angles or a quaternion.

With the motion model function f^{CTRV}, the vehicle state $\mathbf{x}_t$ evolves as

$$\mathbf{x}_{t+1} = f^{\text{CTRV}}(\mathbf{x}_t) + \mathcal{N}\left(\mathbf{0}, \Sigma_t^{\text{mm}}\right) \tag{8.12}$$

Given this, we can define the motion model factor's error function as

$$\|\mathbf{e}_t^{\text{mm}}\|_{\mathbf{\Sigma}_t^{\text{mm}}}^2 = \|f^{\text{CTRV}}(\mathbf{x}_t) - \mathbf{x}_{t+1}\|_{\mathbf{\Sigma}_t^{\text{mm}}}^2 \tag{8.13}$$

with $\mathbf{x}_t = (x, y, z, \delta^{\text{Clock}}, \theta, v, \omega)^\mathsf{T}$ and f^{CTRV} defined as:

$$\mathbf{x}_{t+1} = f^{\text{CTRV}}(\mathbf{x}_t) = \mathbf{x}_t + \begin{cases} \begin{pmatrix} \frac{\mathbf{x}_t^v\left(\sin(\mathbf{x}_t^\theta + \mathbf{x}_t^\omega \Delta t) - \sin(\mathbf{x}_t^\theta)\right)}{\mathbf{x}_t^\omega} \\ \frac{\mathbf{x}_t^v\left(\cos(\mathbf{x}_t^\theta)\right) - \cos(\mathbf{x}_t^\theta + \mathbf{x}_t^\omega \Delta t)}{\mathbf{x}_t^\omega} \\ 0 \\ 0 \\ \mathbf{x}_t^\omega \Delta t \\ 0 \\ 0 \end{pmatrix} & \text{,if } \mathbf{x}_t^\omega \neq 0 \\ \begin{pmatrix} \mathbf{x}_t^v \cos(\mathbf{x}_t^\theta)\Delta t \\ \mathbf{x}_t^v \sin(\mathbf{x}_t^\theta)\Delta t \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & \text{,if } \mathbf{x}_t^\omega = 0 \end{cases} \tag{8.14}$$

The State Prior Factor

With reliable odometry information (i.e. forward velocity and yaw rate in this scenario) available from the vehicle's internal sensors, we can incorporate them using state prior factors. For instance, if the state is $\mathbf{x}_t = (x, y, z, \delta^{\text{Clock}}, \theta, v, \omega)^\mathsf{T}$ as defined above, we can define the state prior factor to be

$$\|\mathbf{e}_t^{\text{stp}}\|^2_{\mathbf{\Sigma}_t^{\text{stp}}} = \|\mathbf{x}_t - \boldsymbol{\zeta}_t\|^2_{\mathbf{\Sigma}_t^{\text{stp}}} \tag{8.15}$$

Where $\boldsymbol{\zeta}_t$ contains the available prior information for the vehicle state at time t. Notice that if only some of the entries in the prior $\boldsymbol{\zeta}_t$ are actually available (e.g. only v and ω), the entries in the information matrix associated to the unavailable entries can simply be set 0 so that they will not have any influence during the optimization.

8.3.4 Solving for the Maximum a Posteriori Solution

When only the pseudorange measurements are given, the maximum a posteriori solution for a single vehicle state $\mathbf{x}_t$ is found by solving the least squares problem

$$\mathbf{x}_t^* = \underset{\mathbf{x}_t}{\text{argmin}} \sum_j \|\mathbf{e}_{tj}^{\text{pr}}\|^2_{\mathbf{\Sigma}_{tj}} \tag{8.16}$$

Similarly, we can solve for a set of vehicle states $X = \{\mathbf{x}_t\}$:

$$X^* = \underset{X}{\text{argmin}} \sum_{tj} \|\mathbf{e}_{tj}^{\text{pr}}\|^2_{\mathbf{\Sigma}_{tj}} \tag{8.17}$$

Any additional factors that account for further measurements and sensor data can be easily incorporated by extending the error function. For instance to incorporate motion model and state transition factors, we solve

$$X^* = \underset{X}{\operatorname{argmin}} \sum_{tj} \|\mathbf{e}_{tj}^{\mathrm{pr}}\|^2_{\mathbf{\Sigma}_{tj}} + \|\mathbf{e}_{t}^{\mathrm{mm}}\|^2_{\mathbf{\Sigma}_{t}^{\mathrm{mm}}} + \|\mathbf{e}_{t}^{\mathrm{st}}\|^2_{\mathbf{\Sigma}_{t}^{\mathrm{st}}} \tag{8.18}$$

and so forth.

In its general structure, the GNSS-based least squares localization problem is not different to any of the problems we encountered in the SLAM chapters. In fact, it shares the same inherent sparsity we found in the SLAM problem and can therefore be solved efficiently by applying the same tools, like g^{2}o or GTSAM.

A key difference to the SLAM problem however is that GNSS-based localization is usually understood as an *online* problem, i.e. it has to be solved while new measurements and observations arrive. In SLAM, we are sometimes satisfied with an *offline* or *batch* solution, after all the data has been gathered. However, since efficient methods for incremental optimization-based smoothing are available (especially iSAM and iSAM2 [Kaess et al., 2011]), we can solve the GNSS-based localization problem online if it is required and still keep the factor graph representation to apply the robust approach that I proposed for the SLAM problem. How this can be accomplished is shown in the next section.

8.4 Towards a Problem Formulation Robust to Multipath Errors

Since we consider some of the pseudorange measurements to be outliers, the same idea that was proposed before to mitigate false positive loop closures in the SLAM context is applied: We extend the problem formulation given above by introducing the switch variables and associating each pseudorange observation with one of these switches.

This way, the pseudorange measurements that represent multipath observations can be removed from the factor graph representation in the same way as false positive loop closures were removed before. Notice that apart from this topological interpretation, we can as well apply the probabilistic interpretation like we did in the SLAM context: In this interpretation the switch variables allow the optimizer to adapt the covariances associated with the multipath measurements and drive them towards representing a high uncertainty.

8.4.1 The Switched Pseudorange Factor

By combining the pseudorange factor from section 8.3.2 with the switch variables, we gain the *switched* pseudorange factor:

$$\|\mathbf{e}_{tj}^{\mathrm{spr}}\|^2_{\mathbf{\Sigma}_{tj}} = \|\Psi(s_{tj}) \cdot (h(\mathbf{x}, j) - \rho_{tj})\|^2_{\mathbf{\Sigma}_{tj}} \tag{8.19}$$

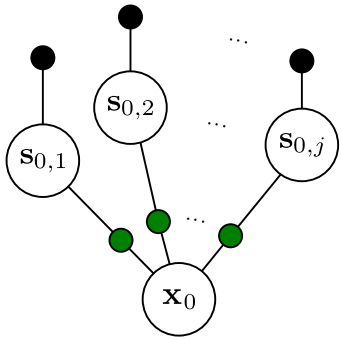


Figure 8.5: *A vehicle state vertex with three switched pseudorange factors (green), the associated switch variables and their prior factors (black).*

We already know from the considerations in Chapter 5 that an additional switch prior factor is needed. It is defined as given before:

$$\|\mathbf{e}_{tj}^{\text{sp}}\|_{\boldsymbol{\Xi}_{tj}}^2 = \|s_{tj} - \gamma_{tj}\|_{\boldsymbol{\Xi}_{tj}}^2 \tag{8.20}$$

Combining these two factors leads to the extended robust problem formulation:

$$X^* = \underset{X}{\operatorname{argmin}} \sum_{tj} \|\mathbf{e}_{tj}^{\text{spr}}\|_{\boldsymbol{\Sigma}_{tj}}^2 + \|\mathbf{e}_{tj}^{\text{sp}}\|_{\boldsymbol{\Xi}_{tj}}^2 \tag{8.21}$$

Fig. 8.5 illustrates this extended formulation for a single vehicle state variable. Notice how each pseudorange measurement is associated with its own switch variable.

8.4.2 The Switch Transition Factor

In contrast to the switch variables in the pose graph SLAM problem, the switch variables in the GNSS-based localization problem are not independent: If a satellite j is observed from two successive vehicle locations $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, then s_{tj} is likely to be equal to $s_{t-1,j}$. We can capture this conditional dependence and model $P(s_{ij}|s_{t-1,j})$ as a Gaussian with

$$P(s_{tj}|s_{t-1,j}) \sim \mathcal{N}\left(s_{t-1,j}, \boldsymbol{\Sigma}_{tj}^{\text{swt}}\right) \tag{8.22}$$

which leads us to the *switch transition* factor

$$\|\mathbf{e}_{tj}^{\text{swt}}\|_{\boldsymbol{\Sigma}_{tj}^{\text{swt}}}^2 = \|s_{tj} - s_{t-1,j}\|_{\boldsymbol{\Sigma}_{tj}^{\text{swt}}}^2 \tag{8.23}$$

Fig. 8.6 illustrates the general idea using a factor graph with switched pseudorange factors and switch transition factors between the switch variables. A more complex graph is shown in Fig. 8.7 where the factor graph contains additional state transition factors, state priors and motion model factors.

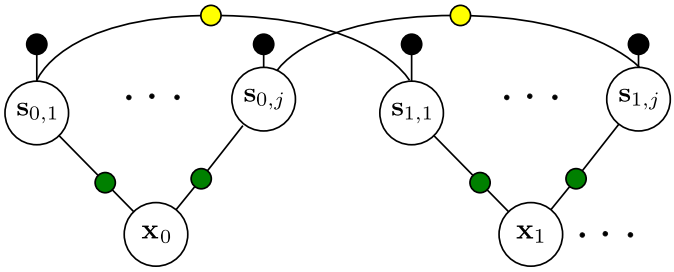


Figure 8.6: *A more complex factor graph, containing two vehicle states and switch transition factors (yellow) between the switch variables.*

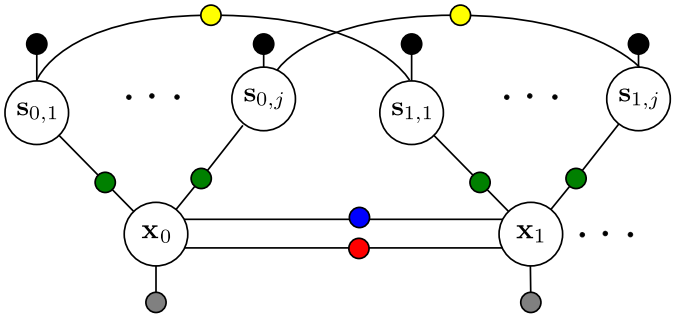


Figure 8.7: *Illustration of the most complex factor graph used in the evaluation for multipath mitigation: There are two types of variables (vehicle states $\mathbf{x}_t$ and switch variables s_{tj}. The switch variables are associated with pseudorange factors (green) and connected by switch transition factors (yellow). The switch priors are depicted in black. The vehicle state variables are connected by state transition (blue) and motion model (red) factors and furthermore governed by prior factors (grey).*

8.5 Multipath Mitigation in a Real-World Urban Scenario

While the previous section explained the different factors necessary for multipath mitigation, the approach is now evaluated using data collected in a real-world urban scenario. We will see how the raw GPS pseudoranges are affected by multipath effects supposedly caused by the tall buildings next to the area where the data was collected (see Fig. 8.8 for a satellite image of the area overlaid with the building structure and ground truth trajectory). The evaluation will show that the proposed scheme for robust optimization is able to mitigate these effects and decrease the overall estimation errors.

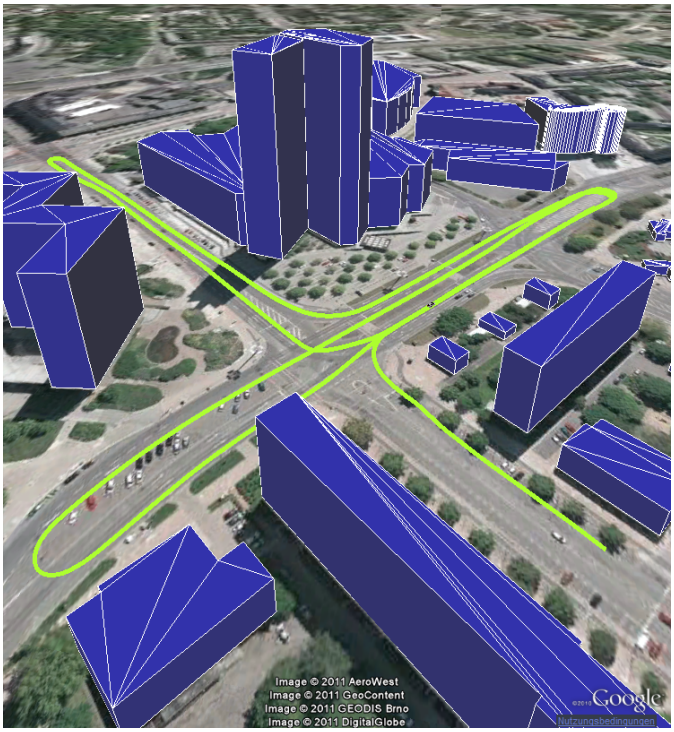


Figure 8.8: *Overview of the urban scenario used in the evaluation. The ground truth path of the vehicle is marked in green. Notice the tall buildings close to the streets. Image courtesy of Sven Bauer, [Bauer, 2011].*

8.5.1 The Chemnitz City Dataset

The necessary data for the evaluation was collected in the city center of Chemnitz, Germany, using the Carai concept vehicle [Schubert et al., 2010] of the Chair of Communications Engineering of our university[46]. The vehicle was driven over a road junction several times for approximately 36 minutes. During that time 90314 satellite observations were taken from 8574 individual positions. Fig. 8.8 visualizes the road layout, the ground truth trajectory, and the tall buildings nearby that caused a high number of GPS signal occlusions and reflections.

Among other sensor systems, the Carai vehicle is equipped with a high-precision differential GPS and inertial measurement unit that allows to determine the vehicle's position with a precision of 2 cm [Bauer, 2011]. The position estimates of this high-precision unit were used as ground truth for the following analysis. In addition to the high-precision GPS unit, a consumer-class device provided the pseudorange measurements that served as inputs for the optimization framework. Odometry information (velocity

[46]I want to thank the colleagues from the chair of Communications Engineering and especially Marcus Obst for providing the data and helping with its analysis and preprocessing.

Table 8.1: *Collected sensor information used for the evaluation.*

Data	Sensor
GPS pseudoranges	consumer-class GPS receiver
SBAS correction data	using EGNOS
ground truth trajectory	NovAtel DGPS with RTK and IMU
velocity and yaw rate	internal vehicle sensors

Table 8.2: *Parameters used in the evaluation.*

Parameter	Value	Description
Ψ	Ψ_1^{lin}	switch function
γ_{tj}	1.0	switch prior value
$\boldsymbol{\Xi}_{tj}$	1.0	switch prior covariance
$\boldsymbol{\Sigma}_{tj}$	$(10\,\text{m})^2$	covariance of pseudorange measurements
$\boldsymbol{\Sigma}_{tj}^{\text{swt}}$	0.05^2	switch transition covariance
$\boldsymbol{\Sigma}_{tj}^{\text{st}}$	$\text{diag}(0.001\,\text{s}, 0.25\,\text{s/s})^2$	state transition covariance
σ_{xy}^2	$(2.5\,\text{m})^2$	variance for the x and y component in $\boldsymbol{\Sigma}_t^{\text{mm}}$
σ_θ^2	$(10\,^\circ)^2$	variance for the θ component in $\boldsymbol{\Sigma}_t^{\text{mm}}$
σ_v^2	$(0.0407\,\text{m/s})^2$	variance for the velocity measurement in $\boldsymbol{\Sigma}_t^{\text{stp}}$
σ_ω^2	$(0.00253\,\text{rad/s})^2$	variance for the yaw rate measurement in $\boldsymbol{\Sigma}_t^{\text{stp}}$

and yaw rate) could be extracted from the vehicle's internal sensors and were used as observations for the state prior factors in those trials where the motion model was used. Table 8.1 summarizes the collected sensor information.

8.5.2 Methodology

In total, seven different problem representations were constructed, using different combinations of the factors described in the previous section. Five of these seven representations contained the switched pseudorange factors and thus are supposed to be robust or at least more robust against multipath errors. To compare the estimation results with the ground truth provided by the high-precision GPS and IMU-devices from the Carai vehicle, an euclidean error metric was used. Notice that it only operated on the x and y component of the position estimates, given in the UTM coordinate frame.

The optimization problems represented by the constructed factor graphs were solved using g^{2}o [Kümmerle et al., 2011b], after the different factors described above were implemented for this framework.

Used Parameters

Table 8.2 lists the different parameters that were used in the implementation. The values in the upper part of the table correspond to the same parameters of the robust back-end we encountered in the SLAM context in the previous chapters. They were chosen to have the same values as before, which underlines that the proposed approach is generic and domain-independent.

The values for the parameters in the lower part of the Table 8.2 are problem specific and were chosen empirically, except for the last two parameters σ_v^2 and σ_ω^2 which were given in [Bauer, 2011].

8.5.3 Results

The seven test cases (using different factor graph representations) and the gained results in terms of position estimation errors are summarized in Table 8.3. I am going to discuss each of these seven trials and compare the results in the following.

Table 8.3: *Position estimation errors and convergence time for different trials on the Chemnitz City dataset.*

Method	Used Factors	Median [m]	Mean [m]	Max [m]	Time [s]
non-robust	$\mathbf{e}^{\text{pr}}$	25.28	32.85	171.64	1.2
	$\mathbf{e}^{\text{pr}}$, $\mathbf{e}^{\text{st}}$, $\mathbf{e}^{\text{mm}}$, $\mathbf{e}^{\text{stp}}$	25.86	31.77	140.24	73.3
robust	$\mathbf{e}^{\text{spr}}$, $\mathbf{e}^{\text{sp}}$	3.66	17.91	202.87	84.3
	$\mathbf{e}^{\text{spr}}$, $\mathbf{e}^{\text{sp}}$, $\mathbf{e}^{\text{st}}$	2.79	14.08	274.49	102.8
	$\mathbf{e}^{\text{spr}}$, $\mathbf{e}^{\text{sp}}$, $\mathbf{e}^{\text{swt}}$	2.69	8.10	128.55	46.2
	$\mathbf{e}^{\text{spr}}$, $\mathbf{e}^{\text{sp}}$, $\mathbf{e}^{\text{swt}}$, $\mathbf{e}^{\text{st}}$	2.45	2.96	16.31	66.9
	$\mathbf{e}^{\text{spr}}$, $\mathbf{e}^{\text{sp}}$, $\mathbf{e}^{\text{swt}}$, $\mathbf{e}^{\text{st}}$, $\mathbf{e}^{\text{mm}}$, $\mathbf{e}^{\text{stp}}$	2.56	2.86	9.35	35.4

Conventional Least Squares The first line of Table 8.3 corresponds to the non-robust, conventional least squares solution, using only the pseudorange factors $\mathbf{e}^{\text{pr}}$. Due to several multipath effects, reflections and occlusions, the error values are very high, with a median error of over 25 m. The plot in Fig. 8.9(a) illustrates how the position estimates are strongly biased compared to the ground truth solution.

Switched Pseudorange Factors When the normal pseudorange factors $\mathbf{e}^{\text{pr}}$ are replaced by their switched counterparts $\mathbf{e}^{\text{spr}}$, the estimation error measures drop significantly: As can be seen in the third line of Table 8.3, the median error now decreased to 3.66 m. Fig. 8.9(b) shows that the bias from the estimates has been removed, although a significant amount of noise in the estimation is still visible. However, the overall quality of the position estimates has been clearly improved with the proposed robustified pseudorange factors.

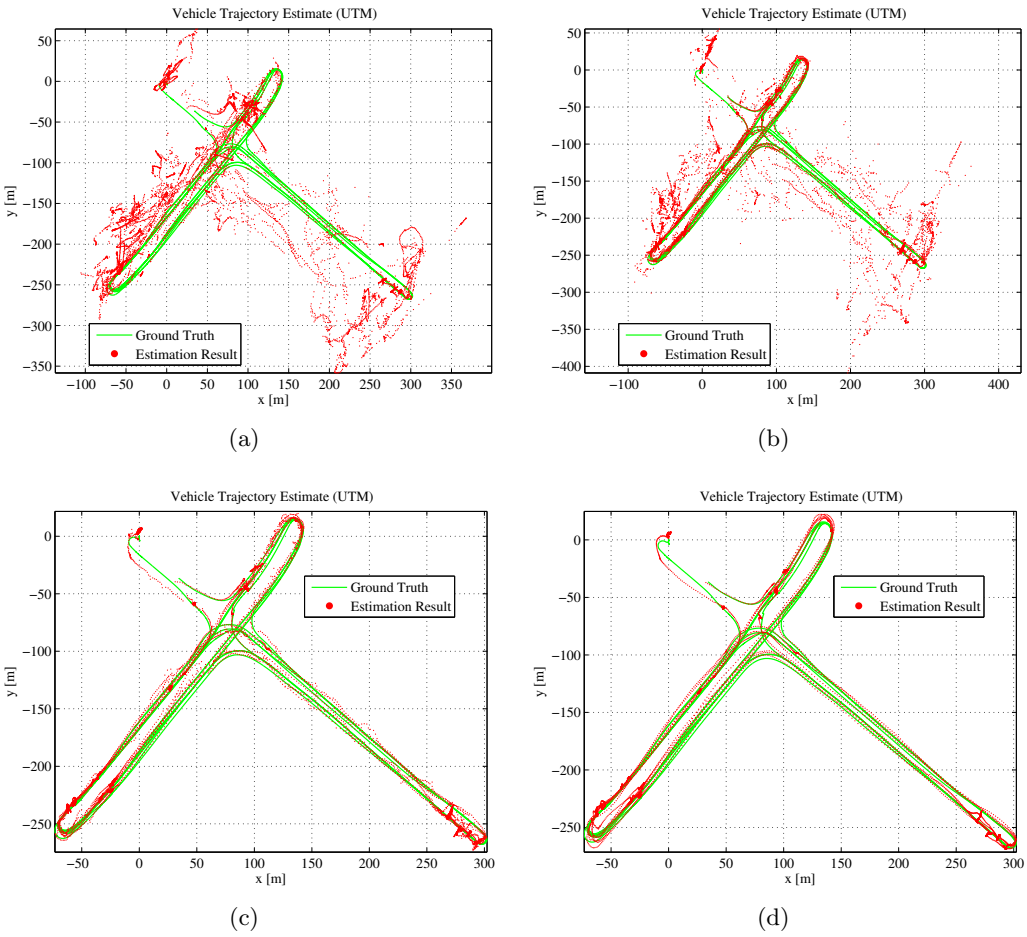


Figure 8.9: *Four different solutions for the Chemnitz City dataset. In (a) the ground truth (green) is compared to the conventional least squares solution (red) which shows significant position biases. The solutions in (b), (c) and (d) were estimated using the switched pseudorange constraints and are thus more robust against multipath effects. The biases have been visually reduced. In (c), the switch transition and state transition factors have been used in addition to the switched pseudorange factor. The solution shown in (d) additionally features the motion model factors and naturally provides the best results compared to the ground truth.*

Switched Pseudorange and State Transition Factors In the next trial, I combined the switched pseudorange factor with the state transition factor $\mathbf{e}^{\mathrm{st}}$. The result was a decrease in median error and RMSE, with the median error down to 2.79 m. However, the maximum deviation is even increased and measures 274.49 m in this scenario.

Switched Pseudorange and Switch Transition Factors When using the switched pseudorange factors $\mathbf{e}^{\mathrm{spr}}$ in combination with the switch transition factors $\mathbf{e}^{\mathrm{swt}}$, the quality of the estimated trajectory increases further. The median error drops to 2.69 m. Despite the low median and mean errors, the maximum deviation from the ground truth of 128.55 m is still very high, although significantly less than in the previous trial when using the state transition factor.

Switched Pseudorange with State and Switch Transition Factors As we saw from the last two trials, both the switch transition factors $\mathbf{e}^{\mathrm{swt}}$ and state transition factors $\mathbf{e}^{\mathrm{st}}$ improve the median and mean position estimation errors. When both are combined, the results improve further (see the second last line in Table 8.3. This way, not only the mean errors stay low at 2.45 m, but also the maximum error is decreased from the unsatisfactory > 100 m to bearable 16.31 m. Fig. 8.9(c) illustrates the resulting trajectory estimate.

Best Robust Solution including the Motion Model The best quality of course is gained when incorporating all available information and using the state transition, state prior and motion model factors ($\mathbf{e}^{\mathrm{st}}$, $\mathbf{e}^{\mathrm{stp}}$, and $\mathbf{e}^{\mathrm{mm}}$) as well as the switched pseudorange, switch prior, and switch transition factors ($\mathbf{e}^{\mathrm{spr}}$, $\mathbf{e}^{\mathrm{sp}}$, and $\mathbf{e}^{\mathrm{swt}}$). This corresponds to the factor graph layout illustrated in Fig. 8.7.

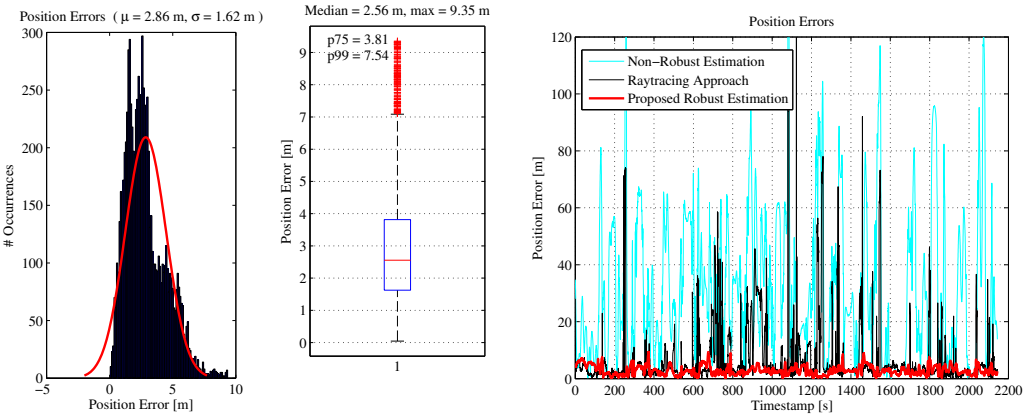
The last line in Table 8.3 reveals that using this setup results in a median position error of 2.56 m while at the same time decreasing the maximum error to 9.35 m. Fig. 8.9(d) illustrates how close the estimated trajectory follows the ground truth solution. The distribution of the individual position errors is shown in Fig. 8.10(a).

These results are remarkable, especially if we compare them against the results gained by a raytracing approach to multipath detection [Obst et al., 2011; Bauer, 2011]. Despite the large amount of additional information[47] about the environment that was used in the raytracing method to decide whether satellites are visible from a certain point on the ground, the results of the proposed robust optimization reach better results[48]. This also visible from Fig. 8.10(b) that shows the position error for each vehicle pose for the proposed robust estimation, the conventional least squares method and the raytracing approach of [Obst et al., 2011].

Fig. 8.11 illustrates the distribution of the individual switch values s_{ti} after the optimization. It is apparent that most switch variables have been assigned values of

[47] Their approach requires a database of building positions and dimensions.

[48] The median, mean and maximum estimation errors of the raytracing method were 2.92 m, 6.83 m, and 509.12 m respectively. The extremely large maximum error was supposedly caused by inaccuracies in the building database that led to a number of gross errors in the visibility check.



(a) (b)

Figure 8.10: *(a) Illustrates the distribution of the estimation errors for the best factor graph representation (using all available factors, including the motion model). In (b), the individual position errors of the proposed robust optimization (red) are compared with the conventional least squares (blue) and the raytracing method of [Obst et al., 2011] (black). Notice that the errors of the proposed robust optimization are constantly low while the other two approaches show significant spikes where the position estimation failed due to unhandled multipath effects.*

approximately 0 or 1. This supports the understanding that the optimization could clearly recognize the outlier measurements (multipath observations) and distinguish them from the inliers ("good" observations).

Tracking the values of the switch variables over time results in further interesting insights: Fig. 8.12 illustrates how some of the switch variables associated with a specific satellite evolve through time. The variables associated to different satellites are shown in different colors, so it is possible to see how the observations of a satellite are estimated to be outliers (thus subject to multipath effects) at one point in time and inliers later on. Hence the switch values for most satellites oscillate between 1 and 0 as the satellite may be occluded by a building at one point in time, but clearly visible at another and so forth. The same behaviour was observed when not using state transition and motion model factors, thus only the switched pseudorange factors and the switch transition factors. The behaviour without the switch transition appeared less coherent, which would be expected.

Conventional Solution with Motion Model and State Transition To ensure that the good performance of the last problem solution using all available factors is not primarily due to the motion model, the second line of Table 8.3 lists the error measurements for a trial where the motion model, state transition and state prior factors ($\mathbf{e}^{\mathrm{mm}}$, $\mathbf{e}^{\mathrm{stp}}$ and

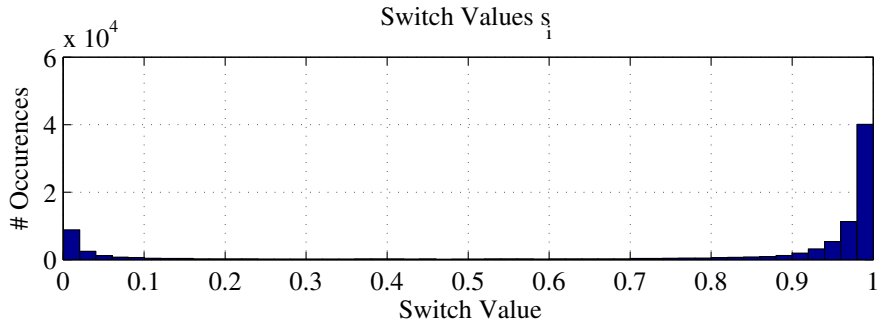


Figure 8.11: *Histogram over the switch values s_{tj} after the optimization for the best factor graph representation (using all mentioned factors). Notice that most switch variables are either approximately 1 or 0, but hardly any intermediate values exist. This means that the optimizer very clearly "decided" whether a satellite observation should be considered an outlier or an inlier. Using a splitting threshold of* 0.5*, 19315 of 90314 observations or* 21.4% *have been declared an outlier.*

$\mathbf{e}^{\text{st}}$) were used in conjunction with the unswitched, conventional pseudorange factors $\mathbf{e}^{\text{pr}}$. Median and mean errors are approximately equal to the the case where neither motion model nor state transition factors were used. Only the maximum error was reduced, presumably due to the smoothing influence of the motion model.

8.6 Interpretation and Summary

As we have seen, since finding the solution for the position of the GNSS receiver on the ground is a least squares problem, it can be conveniently modelled as a factor graph. Furthermore, the evaluation revealed that the general idea of making the topology of the factor graph subject to the optimization process developed in this thesis can be beneficially applied in the GNSS-based localization domain. By associating a switch variable to each of the pseudorange measurements, the optimization is able to identify and remove multipath observations that would otherwise severely bias the position estimate. Like before, the alternative, probabilistic interpretation holds as well: The covariances associated with multipath observations are driven towards high values during the optimization, so that their associated measurements do not or only hardly influence the estimation result. In a way, this corresponds to estimating the optimal covariances for the measurements given.

Given the results presented above, we can summarize the findings in the following conclusions:

Result 8.1. *The GNSS-based localization problem can be modelled using a factor graph and solved using the state of the art solvers like* g^{2}o *. Additional sensor information can*

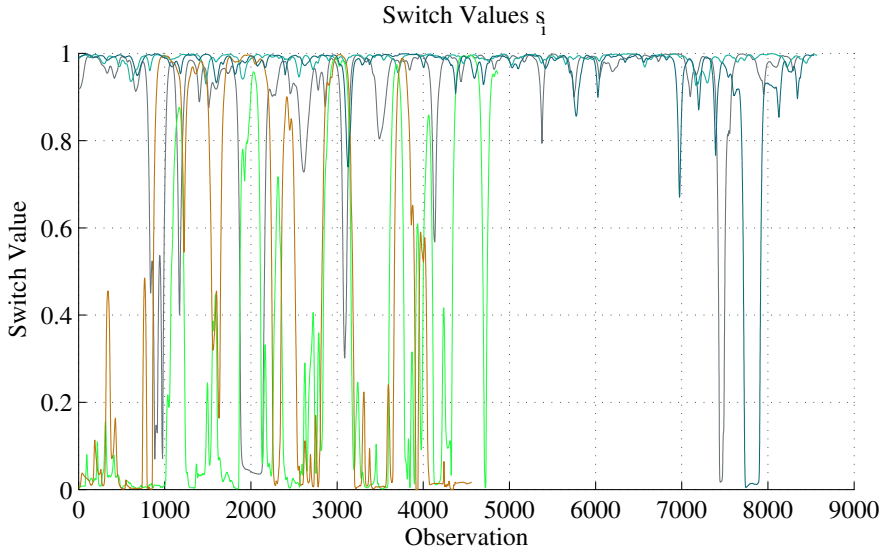


Figure 8.12: *Evolution of the switch values associated with different satellite observations over time. The colors code the observed satellites. Notice how the switch values oscillate between values of 1 and 0, indicating that the satellites are alternately considered outliers and inliers at different points of time.*

be incorporated using additional factors (e.g. for state transition or a suitable motion model).

Result 8.2. *Using the switched pseudorange factor instead of the conventional least squares solution drastically decreased the mean and maximum error of the position estimates. The proposed method does not require an additional pre-processing step or additional knowledge or models of the environmental structure or the surrounding buildings. It outperforms the conventional non-robust least squares solution but also a sophisticated and computationally involved raytracing approach for multipath detection.*

Result 8.3. *The proposed robust optimization approach is able to detect and reject multipath measurements during the optimization process. The switch values oscillated over time and reflected that individual satellites were considered outliers (subject to multipath effects) at some points in time, but treated as inliers at other times.*

Result 8.4. *The proposed scheme for a robust back-end for SLAM has been successfully ported to the domain of GNSS-based localization. It was shown that the mechanisms for robust optimization-based estimation proposed in this thesis can be beneficially applied in this problem domain.*

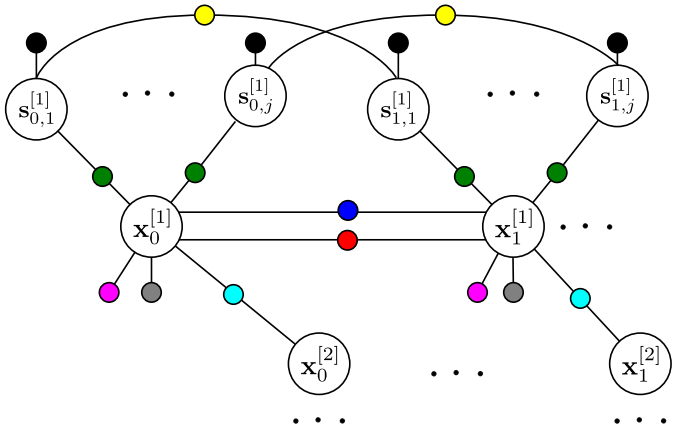


Figure 8.13: *By extending the factor graph illustrated in Fig. 8.7, one can incorporate factors between the vehicle variable and a known map (pink). It would even be possible to exchange information between several other vehicles in the surrounding (using vehicle-to-vehicle communication) and model these additional constraints with inter-vehicle factors (cyan).*

8.7 Outlook

The application of efficient and robust optimization-based approaches to the problem of GNSS-based localization may have strong potential that should be actively pursued in future research. As we have seen in this chapter, factor graphs are a powerful tool that allows convenient modelling of vehicle states and satellite observations and the probabilistic constraints between them. Additional information, like from a priori known maps can be easily incorporated into this framework by introducing additional factors. Also in the context of multi-vehicle or cooperative localization where information is exchanged between vehicles or additional roadside devices, factor graphs and efficient robust optimization-based solvers that perform incremental smoothing may be a feasible alternative to filter approaches commonly used today.

Fig. 8.13 illustrates an exemplary factor graph with two vehicles $\mathbf{x}_t^{[1]}$ and $\mathbf{x}_t^{[2]}$. Both vehicles can exchange information by means of vehicle-to-vehicle communication in order to perform cooperative localization. This mutual information exchange (e.g. mutual distance measurements by radar or visual sensors or shared pseudorange information) can be incorporated by additional inter-vehicle factors. Furthermore, since high-resolution maps of the road layout are readily available, this a priori information can be incorporated as well using additional map factors. In the most naive approach, these factors can penalize a position estimate if the vehicle is located off a drivable road. Preliminary results from simulation showed promising behaviour in that the map factors constrained vehicle position estimates to lie on the drivable road surfaces. These promising ideas will have to be pursued further and evaluated in future work.

Finally, the question of whether an optimization-based smoothing approach bears any

advantages over the filtering approaches commonly used in the GNSS problem domain today leads us to the more general question of filtering vs. smoothing for information fusion that will be discussed in the following chapter.

9

An Outlook on Robust Optimization for Sensor Fusion and Calibration

The last chapter concluded with remarks that pointed out how the basic factor graph representation of the GNSS-localization problem can be conveniently extended to incorporate inter-vehicle information or map information. This already pointed the way towards typical sensor fusion or data fusion problems, where a number of different sources of information have to be incorporated to retrieve the optimal (according to a suitable metric) estimate of the sought variables.

This chapter does not present more results or experiments. It is rather a short outlook on further applications beyond SLAM where, according to the author's opinion, the robust optimization approach could be applied beneficially. However, future work is needed to explore and evaluate these ideas in depth.

9.1 Sensor Fusion by Robust Optimization

Filter approaches (especially EKF, UKF, and particle filters) are widely used in the scientific literature dealing with data fusion problems. *Smoothing* approaches for such problems have been developed early [Rauch et al., 1965] and are still under development and consideration today (e.g. [Deisenroth and Ohlsson, 2011]). Remember that, in general, if the quantity to be estimated is the system state at time t, i.e. we seek an estimate of $\mathbf{x}_t$ and the available measurement data spans from time $0 \dots T$ with $T > t$ we perform smoothing. If $T < t$, it is prediction and when $T = t$, the estimation is called filtering [Gelb, 1974, ch. 1]. As [Russell and Norvig, 2010, ch. 15] points out, smoothing gains better estimates than filtering, since smoothing can always incorporate more information.

So called *fixed-lag* smoothers seem to have the largest practical applications, especially when incorporating delayed or out of sequence sensor information [Bar-Shalom, 2002; Ranganathan et al., 2007]. They estimate $P(\mathbf{x}_{T-d:T}|\mathbf{z}_{0:T})$ with d being the fixed delay or fixed lag.

Coming back to the scope of this dissertation, we remember that the optimization-approaches we encountered in the chapters about SLAM essentially performed smoothing over the complete time interval $0 \dots T$, i.e. using all available data over the complete timespan of the dataset. The same is true for the GNSS-based problem of the last chapter. However, in sensor fusion applications, incremental processing of the incoming sensor data is necessary. The sensor information have to be fused as soon as they are available and the result is immediately used by another component of the overall robot system, e.g. by a controller, path planner etc. I already stated that the available back-end frameworks like iSAM, GTSAM, and g^{2}o are capable of this *incremental* smoothing. That is, we can efficiently add a new state $\mathbf{x}_t$ and new observations $\mathbf{z}_t$ as time progresses. We can furthermore either keep *all* the old data and use it to solve for $X^* = \text{argmax}_{\mathbf{x}} P(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})$ or shift the interval of remembered data (like a sliding window approach) and solve for $X^* = \text{argmax}_{\mathbf{x}} P(\mathbf{x}_{t_{\text{old}}:t}|\mathbf{z}_{t_{\text{old}}:t})$. Smoothing by optimization can be expected to be more efficient than the conventional forward-backward smoothers. This was shown by [Ranganathan et al., 2007] where a connection between the conventional fixed-lag smoothing that originated in the field of data fusion, and the SLAM-centered approaches of $\sqrt{\text{SAM}}$ [Dellaert and Kaess, 2006] and iSAM [Kaess et al., 2008] is provided. The later two use essentially the same mechanisms but are more efficient than the conventional forward-backward techniques for smoothing like [Rauch et al., 1965].

Especially in the case of nonlinear systems, we can expect the optimization-based smoothing approach to yield more accurate results than filtering methods, due to the incremental nature of the optimization [Sibley et al., 2006] that allows repeated linearization. A second advantage is that *delayed* sensor messages can be incorporated very easily in an optimization-based sensor fusion. Imagine the system is already at time t. Due to some delay (e.g. due to communication or pre-processing), a sensor message arrives which contains information the sensor recorded at time $t-2$. Filtering approaches would have large difficulties in correctly incorporating this measurement, since they only keep track of the most recent state estimate and would have to filter backwards to time $t-2$, correct the prediction with the new sensor data and filter forwards again to reach t. In an optimization-based framework, the process is much more straightforward. Incorporating that newly arrived sensor data corresponds to inserting an additional node along with its factors into the factor graph representation of the problem, which can be done efficiently online [Ranganathan et al., 2007].

Processing such delayed information can potentially become more important if we think of decentralized sensor networks, or vehicle-to-vehicle, or vehicle-to-infrastructure communication in the automotive domain. All these systems have to incorporate information from external sensors that have to be transmitted over a communication channel of some kind and therefore varying communication delays have to be expected.

These arguments support the idea of sensor fusion based on nonlinear least squares

optimization in general. The robust optimization approach that was proposed in this thesis can provide further advantages since we can expect it to identify and suppress outlier measurements which would otherwise greatly disturb the sensor fusion result. In future work, this could be explored in depth based on the dataset that was used in Chapter 8 to perform GPS multipath detection. It would be interesting to see how the system performs when it is applied online and incrementally. Another immediate area of application would be our quadrotor UAVs[49] that are equipped with an IMU and visual odometry. Data from both systems have to be fused, taking communication delays into account, and expecting the odometry system to fail from time to time or to produce outlier measurements [Lange et al., 2011; Wunschel et al., 2012].

9.2 Sensor Calibration by Robust Optimization

Least squares optimization is ubiquitous in many sensor calibration applications. Especially in the vision domain, the intrinsic parameters of a camera (i.e. the camera matrix $\mathbf{K}$ and the distortion parameters) are usually determined by optimization after several images of a known calibration target (e.g. a checkerboard) have been acquired [Hartley and Zisserman, 2004; Zhang, 1999].

Cross-calibration of several sensors or sensor-actor calibration is another domain where least squares approaches can be applied beneficially. Examples for sensor-sensor calibration are the determination of the extrinsic parameters of a stereo camera system (e.g. translation and rotation between the two cameras) or the calibration between camera and IMU [Fleps et al., 2011; Kelly and Sukhatme, 2011; Hol et al., 2010; Lobo and Dias, 2007], camera and laser range finder [Vasconcelos et al., 2012; Scaramuzza et al., 2007] or multiple different sensors and manipulators in general [Wagner et al., 2011; Pradeep et al., 2010].

If such calibration tasks are executed in an uncontrolled environment where outliers in the various measurements have to be expected, the proposed robust back-end might prove beneficial. This is especially true if the calibration has to be performed automatically, i.e. without human help and interference or without an operator controlling the results. Future work will have to show the applicability of the proposed robust optimization back-end to this domain.

[49] Unmanned Aerial Vehicle

10 Conclusions

This thesis started with describing how modern SLAM systems are divided into a front-end and a back-end part and how these two parts depend on each other in order to successfully solve the problem of simultaneous localization and mapping. After an introduction to SLAM and least squares optimization was given in Chapters 2 and 3, Chapter 4 provided the main motivation for the following original work and the primary contributions of this thesis.

Since the back-ends solve the SLAM problem using least squares optimization techniques, they are, like all least squares methods, prone to outliers. In the context of SLAM, outliers are mostly expected to be data association errors and especially false positive loop closure requests. Since data association happens in the front-end, state of the art back-ends expect the front-end to identify and reject all potential outliers. The factor graph representation that is passed from the front-end to the back-end is expected to be free of outliers, thus topologically correct. Any error in the graph, hence any error in the least squares problem formulation would result in a corrupted, wrong solution to SLAM. Chapter 4 provided several citations from the current scientific literature which underlined that the problem is in general acknowledged by the community but no concise solution on the back-end side has yet been provided. Instead, several approaches to prevent outliers on the front-end side have been proposed, some of them very computationally involved, and none can guarantee to reject all potential outliers. This marked the starting point for the original contributions of the thesis at hand. These contributions are summarized in the following.

10.1 What has been Achieved – Contributions of this Thesis

Chapter 5 introduced a novel approach that allows the back-end to identify and remove outlier loop closure constraints *during* the optimization. The underlying insight is that outliers correspond to erroneous edges in the problem's factor graph representation. Hence Chapter 5 proposed that the topology of the graph should be subject to the optimization instead of keeping it fixed. This way, the optimizer can literally remove erroneous edges from the graph.

Starting from this rather informal idea, Chapter 5 developed a concise mathematical formulation and introduced the novel switch variables. These additional hidden variables act as weights on the factors that potentially represent outliers. It was shown that the solution to the augmented optimization problem containing the switched loop closure constraints represents the maximum a posteriori solution to the problem if all factors are modelled as Gaussians. Furthermore, besides the topological interpretation of removing edges from the graph, Chapter 5 showed that a probabilistic interpretation of the switch variable's influence is equivalent: The switches act as weights on the information matrices associated with the switched loop closure constraints. Therefore these information matrices can be adapted during the optimization. By driving the resulting weights towards zero, the information matrices are driven towards zero as well, indicating that the associated loop closure constraint contains no information, or equally, infinite uncertainty.

Chapter 6 evaluated the proposed novel approach using several synthetic and real-world datasets in both 2D and 3D. The specific datasets were chosen because they are commonly used as benchmarks in the SLAM community and thus the results presented in the evaluation can be reproduced by other researchers. Furthermore, the standard datasets ensure that alternative approaches to robust back-ends for SLAM can be easily compared and benchmarked against the approach proposed here. The data used for the evaluation will thus be made available to the SLAM community to foster future work on the subject.

The evaluation proved the feasibility of the proposed approach and indicated that the robust back-end can identify even large amounts of outliers and reject them, except for one dataset with a very unfavourable structure.

Chapter 7 presented results on a large-scale real-world dataset: The St. Lucia dataset has been collected in a suburban environment and consists of video footage of a 66 km long trajectory through the streets, including a large number of loop closures of different sizes. Since except for a simple webcam no other sensors have been used, all necessary information for SLAM had to be extracted from the image stream. A simple method for calculating rough visual odometry has been explained and the novel BRIEF-Gist approach for visual place recognition that was developed in parallel work has been presented. Despite the simplicity of the front-end that produces many false positive loop closures, the proposed robust-back end was able to solve the SLAM problem on this very demanding dataset with satisfying accuracy.

Chapter 8 demonstrated that the proposed approach to a robust back-end is versatile and can be ported from SLAM into other problem domains where least squares

optimization problems are solved but outliers can be present in the underlying data. The GNSS-based localization problem served as an example. At its core, the receiver position on the ground is found by solving a least squares problem that contains the known positions of the satellites and the observed pseudoranges. These pseudoranges are subject to a number of error sources and especially in urban environments the multipath problem severely biases the estimation results. Such multipath measurements are the outliers of GNSS-based localization and have to be identified and rejected before the least squares solution is calculated. Chapter 8 showed how the GNSS problem can be formulated and represented using a factor graph and how the switch variables can be adopted to govern the pseudorange measurements. The evaluation using a real-world dataset collected in an urban environment revealed that the proposed approach to robust optimization can be beneficially applied to this domain.

Finally, Chapter 9 discussed further applications of the proposed approach to robust optimization for general sensor fusion and calibration problems where smoothing instead of filtering might be beneficial.

10.2 Open Questions – An Outlook on Further Work

Like in supposedly all dissertations and scientific work in general, at the end of the day, open questions remain. These unanswered questions and issues can be taken as a guideline for further work on the approach for robust SLAM that was proposed in this work but are also of more general interest.

10.2.1 Parameters of the Proposed Robust Formulation

One open question regards the choice for the two free parameters of the proposed robust formulation of optimization-based pose graph SLAM. These parameters are the switch function Ψ and the covariance $\boldsymbol{\Xi}_{ij}$ of the switch priors. Both have been more or less set and chosen empirically. However, a reduction of empiricism in the choice of these parameters should be pursued.

The switch function Ψ is an integral part of the proposed system. As we remember, $\omega_{ij} = \Psi(s_{ij})$ provided a mapping from the continuous switch variables $s_{ij} \in \mathbb{R}$ to the domain of the weight factors $\omega_{ij} \in [0, 1]$. Three different possible switch functions have been mentioned in Chapter 5, namely Ψ^{step}, Ψ^{sigmoid} and Ψ_a^{lin}. The latter one has been the switch function of choice for the evaluations and applications conducted in Chapters 6, 7 and 8. Other suitable functions may be possible as well. An elaborate analysis of the influence of the switch function on the robustness, the convergence behaviour etc. has not yet been conducted. Furthermore it would be worthwhile to actually *derive* what switch function is the best choice for a certain problem domain, according to a suitable measure, instead of choosing a suitable function more or less empirically. The same is true for the parameters of the actual switch function, e.g. the parameter a in Ψ_a^{lin}, which was empirically set to $a = 1$, after a rough evaluation showed its influence to be neglectable.

The switch prior covariance $\boldsymbol{\Xi}_{ij}$ controls the amount of penalty the system gains

for driving a switch variable s_{ij} away from its initial value γ_{ij}. As I described in the evaluation, a suitable value for $\boldsymbol{\Xi}_{ij}$ was determined to be 1. Although this value was derived empirically using one dataset, it proved to be a feasible value for all other tested datasets and even for the application of the robust back-end in the GNSS localization problem. This is a bit surprising, since the choice of the parameter directly influences how easy it is for the back-end to reject a constraint as an outlier. If $\boldsymbol{\Xi}_{ij}$ is too small, the optimizer would rather accept a high error from other constraints instead of driving s_{ij} away from γ_{ij} and thus reject the associated constraint. So although for the evaluation conducted in this thesis setting $\boldsymbol{\Xi}_{ij} = 1$ did not pose any problems, the choice of $\boldsymbol{\Xi}_{ij}$ might be more delicate in other situations. An exact method for deriving the value of $\boldsymbol{\Xi}_{ij}$ or adapting it during the optimization is therefore desirable.

10.2.2 Convergence Behaviour and the Dangers of Local Minima

It is generally understood that the least squares formulation of SLAM leads to non-convex objective functions. Therefore, local minima have to be expected to occur. If the initial guess from where the incremental optimization is started is far from the global optimum, the whole optimization process is likely to converge towards a *local* minimum which can represent a false solution of the problem at hand. Global optimization is a large problem domain on its own and several techniques can be applied in order to circumvent the problems of local minima, like for instance simulated annealing or starting the optimization from different initial guesses.

The influences of the newly introduced switch variables and switch prior constraints on the convexity properties of the error function that is minimized during the optimization should be regarded in more depth in future work. From the results found in the evaluation, we can conclude that although the resulting overall objective function is not convex, it is still benign enough to allow convergence to a reasonably good solution which appears to be at least close to the global optimum.

Even more desirable would be to combine the proposed robust approach to SLAM with recently developed linear approximations of the SLAM problem like [Carlone et al., 2011a; Carlone et al., 2011b] that can be solved directly in closed form and thus are not endangered of stepping into local minima.

From what can be extracted from the few publications that pursue such developments, the approaches are not yet fully developed, but further progress is to be expected. This might either lead to a better understanding of the convexity structure of the SLAM problem in its current nonlinear formulation [Huang et al., 2010; Huang et al., 2012] or to directly solvable linear approximations of SLAM that can at least serve as a first processing step and produce adequate initial guesses that can then be refined using the more exact nonlinear problem solver.

10.2.3 Further Applications of the Robust Approach

In Chapter 8 I demonstrated that the proposed approach for robust nonlinear least squares optimization is not limited to the SLAM problem. It was demonstrated that the

GNSS-based localization problem that contains outliers in the form of multipath satellite observations can be successfully solved with the proposed novel technique.

The approach of adding a switch variable to each potential outlier constraint is universal and can be applied to other domains that can be formulated as least squares problems and where outliers have to be expected, but cannot be totally rejected *before* the optimization commences. I would like to evaluate the feasibility of my proposed approach for applications in such other domains. Since many different problems regarding sensor calibration and data fusion can be formulated as least squares problems and solved using smoothing or incremental smoothing approaches, the ideas developed here might be benefitial for numerous applications.

In the end, I hope that the ideas presented in this thesis provide a valuable contribution to the field of SLAM and maybe also beyond. Quite some time has passed since I first came into contact with SLAM in 2004, and a lot of ideas and methods have been developed since then. Looking back, it was very fascinating to learn about some of these ideas and also to contribute a little.

Not everything I learned during these past years finally found its way into this thesis and the quest for its topic was not always straight forward. It was exciting anyway, since there was *a lot* to learn – there still is.

A

Filter Algorithms for SLAM

The online SLAM filtering problem, i.e. estimating $P(\mathbf{x}_t, M|U_{t-1}, Z_t)$, has been dealt with from the very beginning of SLAM research. Due to its incremental nature, this problem is usually approached using different flavours of Bayesian filtering algorithms.

In this appendix, we are going to explore several of these filters, using a small sandbox example of a robot moving in a one-dimensional world. The robot's position is therefore given by the one dimensional state variable x_t. We will first start with a discrete world, i.e. $x_t \in \{0, 1, \ldots, 6\}$ and later expand the example to a continuous space. The robot is equipped with a sensor that measures the distance to an obstacle at the boundary of the world. Fig. A.1 illustrates the discrete world and the robot in it. We will furthermore simplify the examples by omitting the mapping part of SLAM, thus we only estimate the robot's position.

A.1 The General Bayes Filter

We will start with the general Bayes filter, that is the foundation of all the other filters that follow. The filter consists of two main steps, which are executed in every iteration:

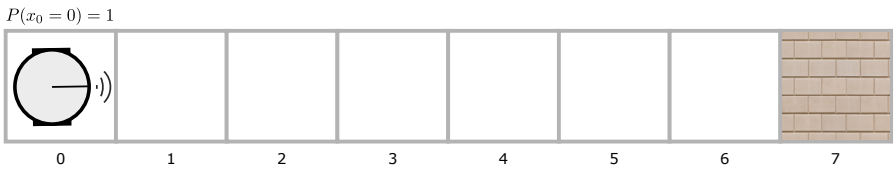


Figure A.1: *A simple, discrete world used as an example for robot localization with the Bayes filter.*

- The *prediction* step uses a state transition model to calculate a prediction of the new state given the old state and the control inputs.
- The *correction* or *measurement update* corrects this prediction using the sensor measurements and a sensor model.

This way, all the available sensor data is incorporated to form the *posterior* estimate of the system state. In its most general form, the central filter equation of the Bayes filter is given by

$$\underbrace{P(\mathbf{x}_t|U_{t-1}, Z_t)}_{\text{posterior}} = \eta \cdot \underbrace{P(\mathbf{z}_t|\mathbf{x}_t)}_{\substack{\text{measurement}\\ \text{probability}}} \cdot \overbrace{\int \underbrace{P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})}_{\text{state transition}} \cdot \underbrace{P(\mathbf{x}_{t-1}|U_{t-2}, Z_{t-1})}_{\text{prior}} \, \mathrm{d}\mathbf{x}_{t-1}}^{\text{prediction}} \tag{A.1}$$

We see that the sought *posterior* distribution is found by multiplying the *prediction* with the *measurement probability* or *measurement likelihood*. The prediction step calculates $P(\mathbf{x}_t|U_{t-1}, Z_{t-1})$ which is found by applying the state transition model to all possible prior states and integrating over them. This prediction distribution is then *corrected* by incorporating the current sensor measurement $\mathbf{z}_t$ using the measurement probability model. The factor η is a normalizer that ensures the integral over the posterior equals 1.

The whole calculation can be continued in a recursive manner, when the posterior of time step t becomes the prior when estimating the new posterior at $t+1$. In order to start the recursion, the first prior $P(\mathbf{x}_0)$ has to be known. Apart from this initial prior, the state transition and measurement models have to be given.

Notice that in the case of a discrete state space, the integral in above equation is replaced by a summation over the individual states.

Example A.1. *We will now apply the Bayes filter to the sandbox example of Fig. A.1.*

The initial position of the robot is known to be in the first cell, hence the initial prior is $P(x_0 = 0) = 1$. *The state transition and measurement models are given in Table A.1. The state transition model expresses that if the robot is in cell* a *at time* $t-1$, *hence* $x_{t-1} = a$, *and the control* $u_{t-1} = 1$ *is applied, the robot will be in cell* $a+1$ *at the next time step* t *with a probability of (only)* 0.6. *With probability* 0.2 *it will remain in the same field* a *or move too far into* $a+2$.

The measurement model describes the range sensor that measures the number of free cells in front of the robot. Notice that if the robot is in cell a, *the correct measurement is* $\hat{z} = 6 - a$. *With probability* 0.8, *the sensor returns this correct measurement, while only with probability* 0.1 *the sensor is influenced by noise and returns a measurement that is too long or too short.*

At time $t = 0$, *the robot starts in cell* 0 *and a control input of* $u_0 = 1$ *is applied. By applying the state transition model, we calculate the following probabilities for the predicted state:* $P(x_1 = 0|u_0) = 0.2$, $P(x_1 = 1|u_0) = 0.6$, *and* $P(x_1 = 2|u_0) = 0.2$. *This prediction is now corrected by incorporating the sensor measurement that was* $z_1 = 5$.

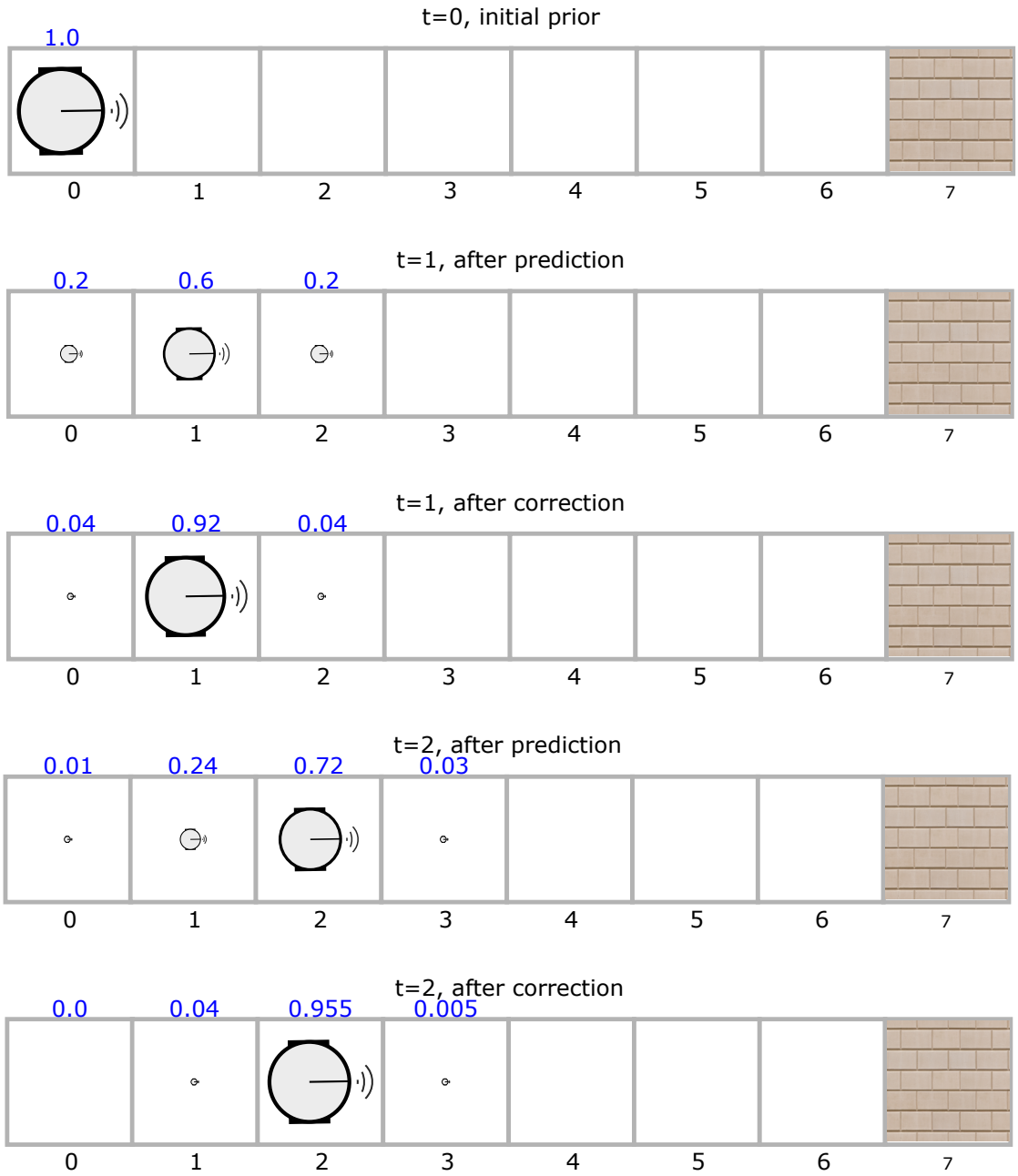


Figure A.2: *Localization using the discrete Bayes filter. The blue numbers above the cells give the probability that the robot is inside the corresponding cell, which is also indicated by the size of the robot. See example A.1 for further explanations.*

Table A.1: *Discrete state transition and measurement model.*

State Transition Model	Measurement Model
$P(x_t = a+0 \mid x_{t-1} = a, u_{t-1} = 1) = 0.2$	$P(z_t = 6-a-1 \mid x_t = a) = 0.1$
$P(x_t = a+1 \mid x_{t-1} = a, u_{t-1} = 1) = 0.6$	$P(z_t = 6-a+0 \mid x_t = a) = 0.8$
$P(x_t = a+2 \mid x_{t-1} = a, u_{t-1} = 1) = 0.2$	$P(z_t = 6-a+1 \mid x_t = a) = 0.1$

From the measurement model we know that $p(z_1 = 5|x_1 = 0) = p(z_1 = 5|x_1 = 2) = 0.1$ *and* $p(z_1 = 5|x_1 = 1) = 0.8$*. Therefore, the resulting posteriors are:*

$$\begin{aligned} p(x_1 = 0|u_0, z_1) &= \eta \cdot p(z_1 = 5|x_1 = 0) \cdot P(x_1 = 0|x_0 = 0, u_0 = 1) \cdot P(x_0) \\ &= \eta \cdot 0.1 \cdot 0.2 \cdot 1 \end{aligned} \tag{A.2}$$

$$\begin{aligned} p(x_1 = 1|u_0, z_1) &= \eta \cdot p(z_1 = 5|x_1 = 1) \cdot P(x_1 = 1|x_0 = 0, u_0 = 1) \cdot P(x_0) \\ &= \eta \cdot 0.8 \cdot 0.6 \cdot 1 \end{aligned} \tag{A.3}$$

$$\begin{aligned} p(x_1 = 2|u_0, z_1) &= \eta \cdot p(z_1 = 5|x_1 = 2) \cdot P(x_1 = 2|x_0 = 0, u_0 = 1) \cdot P(x_0) \\ &= \eta \cdot 0.1 \cdot 0.2 \cdot 1 \end{aligned} \tag{A.4}$$

Since the normalizer $\eta = \sum_a P(x_1 = a|u_0, z_1) = 0.52$ *, the normalized posterior probabilities are*

$$p(x_1 = 0|u_0, z_1) = 0.04 \tag{A.5}$$

$$p(x_1 = 1|u_0, z_1) = 0.92 \tag{A.6}$$

$$p(x_1 = 2|u_0, z_1) = 0.04 \tag{A.7}$$

The normalized results of the prediction and correction steps for the first two time steps are summarized in Table A.2 and depicted graphically in Fig. A.2. Here, the size of the robots corresponds to the probabilities that the robot is in the corresponding cell. Notice how the uncertainty of the robot's position reduces after the prediction step due to the relatively noisy state transition model. However, the sensor measurements can correct the prediction, leading to a comparably narrow posterior distribution.

Table A.2: *Estimation results for the discrete example.*

		Cell					
Time	Step	0	1	2	3	4	5
$t = 0$	initial prior	1	0	0	0	0	0
$t = 1$	prediction	0.2	0.6	0.2	0	0	0
	correction	0.04	0.92	0.04	0	0	0
$t = 2$	prediction	0.01	0.24	0.72	0.03	0	0
	correction	0	0.04	0.955	0.005	0	0

For the sake of completeness, the calculation of $P(x_2 = 2|U_1, Z_2)$ *shall be given below. Assume that the sensor measurement* z_2 *was* $z_2 = 4$*, then we have according to (A.1):*

$$P(x_2 = 2|U_1, Z_2) = \eta \cdot P(z_2 = 4|x_2 = 2) \cdot \sum_a P(x_2|x_1 = a, u_1 = 1) \cdot P(x_1 = a|u_0, z_1) \tag{A.8}$$

$$= \eta \cdot 0.8 \cdot \Big(P(x_2 = 2|x_1 = 1, u_1 = 1) \cdot P(x_1 = 1|u_0, z_1) + P(x_2 = 2|x_1 = 2, u_1 = 1) \cdot P(x_1 = 2|u_0, z_1) \Big) \tag{A.9}$$

$$= \eta \cdot 0.8 \cdot (0.6 \cdot 0.92 + 0.2 \cdot 0.04) \tag{A.10}$$

$$= 0.955 \tag{A.11}$$

A.2 Gaussian Filters

The Gaussian filter is a specialization of the more general Bayes filter, where all probability distributions are Gaussians. That is, they all follow a (usually multivariate) Gaussian distribution of the form

$$P(\mathbf{x}) = |2\pi\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\mathsf{T} \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})) \tag{A.12}$$

We will also write this shorter as

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{A.13}$$

where the symbol $\mathcal{N}(\cdot, \cdot)$ stands for *normal distribution*, which is an alternative name for the Gaussian distribution. Such a multivariate (multidimensional) distribution is characterized by its mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

A.2.1 The Kalman Filter

The Kalman filter is a Gaussian filter with *linear* state transition and measurement function. It is credited to [Kalman, 1960] and [Swerling, 1958] and has been first applied to radar tracking of airborne targets, but used in a large number of other estimation problems ever since. The Kalman filter and its various derivatives are almost ubiquitous in sensor fusion applications.

The Kalman filter can be applied when the system behaviour is linear and continuous, i.e. it cannot be used for discrete problems. Since it is derived from the general Bayes filter, the two main steps, prediction and update or correction, are found in the Kalman filter as well: If the system state at time $t-1$ is given as $\mathbf{x}_{t-1} \sim \mathcal{N}\left(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}\right)$, the *predicted* state $\bar{\mathbf{x}}_t \sim \mathcal{N}\left(\bar{\boldsymbol{\mu}}_t, \bar{\boldsymbol{\Sigma}}_t\right)$ in the Kalman filter is given by:

$$\bar{\boldsymbol{\mu}}_t = \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{B}\mathbf{u}_{t-1} \tag{A.14}$$

$$\bar{\boldsymbol{\Sigma}}_t = \mathbf{A}\boldsymbol{\Sigma}_{t-1}\mathbf{A}^\mathsf{T} + \mathbf{R}_{t-1} \tag{A.15}$$

Notice how the matrices $\mathbf{A}$ and $\mathbf{B}$ form the linear state transition function $\bar{\boldsymbol{\mu}}_t = f(\boldsymbol{\mu}_{t-1}, \mathbf{u}_{t-1})$. The covariance matrix $\mathbf{R}$ in the second equation corresponds to the *process noise* and controls the increase of uncertainty due to the prediction. Notice further how the original covariance matrix $\boldsymbol{\Sigma}_{t-1}$ is propagated through this linear function by multiplying it with $\mathbf{A}$ from both sides[50].

After the prediction step, the *update* or correction step follows, just as in the general Bayes filter. A correction term is added to the predicted new mean $\bar{\boldsymbol{\mu}}_t$ to gain the mean of the posterior distribution:

$$\underbrace{\boldsymbol{\mu}_t}_{\substack{\text{posterior}\\\text{mean}}} = \underbrace{\bar{\boldsymbol{\mu}}_t}_{\substack{\text{predicted}\\\text{mean}}} + \underbrace{\mathbf{K}_t}_{\substack{\text{Kalman}\\\text{Gain}}} \underbrace{\left(\underbrace{\mathbf{z}_t}_{\substack{\text{real}\\\text{measurement}}} - \underbrace{\mathbf{C}_t\bar{\boldsymbol{\mu}}_t}_{\substack{\text{predicted}\\\text{measurement}}} \right)}_{\text{error of predicted measurement}} \tag{A.16}$$

The matrix $\mathbf{C}_t$ captures the linear sensor model, which is given by the linear *measurement function* $\mathbf{z}_t = h(\mathbf{x}_t) = \mathbf{C}_t\mathbf{x}_t + \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$. Notice the zero-mean Gaussian noise term with covariance $\mathbf{Q}$.

In the Kalman filter, the so called *Kalman gain* plays an important role in the update step. In simple words, it controls whether the filter puts more trust into the prediction or the sensor measurement $\mathbf{z}_t$: If $\mathbf{K}$ is small, then the posterior mean will be close to the predicted mean, hence the prediction is more trusted than the sensor measurements and vice versa. The Kalman gain is calculated as:

$$\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t\mathbf{C}_t^\mathsf{T}(\mathbf{C}_t\bar{\boldsymbol{\Sigma}}_t\mathbf{C}_t^\mathsf{T} + \mathbf{Q}_t)^{-1} \tag{A.17}$$

Finally, after the posterior mean has been calculated, the final step in the Kalman filter is to calculate the updated posterior covariance. It is given as

$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C}_t)\bar{\boldsymbol{\Sigma}}_t \tag{A.18}$$

This way, the Kalman filter calculates the new posterior $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ from the old posterior $\mathbf{x}_{t-1}$, the control input $\mathbf{u}_{t-1}$ and the sensor data $\mathbf{z}_t$.

Example A.2. *In order to demonstrate the Kalman filter, we have to adapt the one-dimensional sandbox localization example and make it continuous. The robot therefore is no longer bound to reside inside the discrete cells, but can rather move along the one-dimensional continuum. Fig. A.3 illustrates the example.*

Like in the discrete example, we have to define the state transition and measurement models. Remember that for the Kalman filter, both have to be linear and work on Gaussian distributions. The linear state transition function f is given as follows:

$$\begin{aligned} x_t &= f(x_{t-1}, u_{t-1}) \\ &= Ax_{t-1} + Bu_{t-1} + \mathcal{N}(0, R_{t-1}) \\ &= x_{t-1} + u_{t-1} + \mathcal{N}(0, 0.25) \end{aligned} \tag{A.19}$$

[50] This is a general concept: Given a linear function $f(\mathbf{x}) = \mathbf{A}\mathbf{x}$, any Gaussian $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be propagated through it according to $f(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})) = \mathcal{N}(\mathbf{A}\boldsymbol{\mu}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\mathsf{T})$.

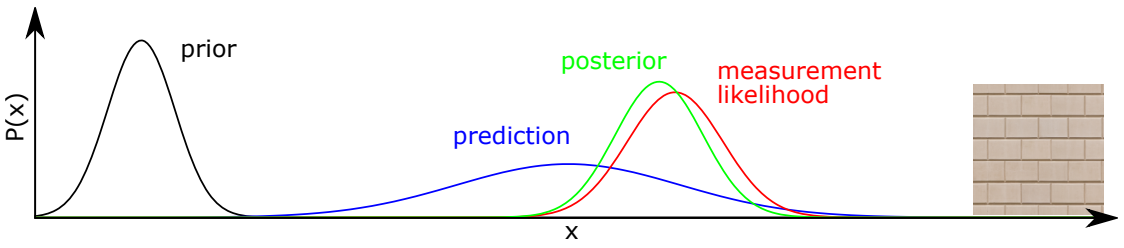


Figure A.3: *Simple localization example using the Kalman filter. Notice how the probability distributions used to represent the different quantities and estimates in the filter are all Gaussians. Starting from the prior, the robot moves towards the right. Its new position is predicted according to the state transition model and the control input and represented by the blue Gaussian. Notice that due to the noise in the control inputs, the prediction is rather uncertain about the robot's position. After measuring the distance to the wall on the far right side, the Kalman filter can correct the predicted position by incorporating the measurement likelihood (red). Notice how the green posterior is much closer to the measurement likelihood than to the prediction, because the measurement uncertainty is smaller than the uncertainty of the prediction. Further explanations are given in example A.2.*

with $A = 1$ *and* $B = 1$. *We let* $C = 1$, *hence the measurement function is defined very simply as*

$$\begin{aligned} z_t &= h(x_t) \\ &= Cx_t + \mathcal{N}(0, Q_t) \\ &= x_t + \mathcal{N}(0, 0.2) \end{aligned} \tag{A.20}$$

Given these models, we need the first prior to initialize the filter: $\mathbf{x}_0 \sim \mathcal{N}(1, 0.1)$.

We can now let the Kalman filter recursively estimate the system states as time advances. Suppose the control input u_0 *is given as* $u_0 = 4$ *and the associated process noise* R_0 *is* $R_0 = 1.0$. *To predict the system state at time* $t = 1$, *we perform the state prediction and get:*

$$\begin{aligned} \bar{x}_1 &= \mathcal{N}\left(A\mu_0 + Bu_0, A\Sigma_0 A^\mathsf{T} + R\right) \\ &= \mathcal{N}(1 \cdot 1 + 1 \cdot 4, 1 \cdot 0.1 \cdot 1 + 1.0) \\ &= \mathcal{N}(5, 1.1) \end{aligned} \tag{A.21}$$

The Kalman gain needed for the update step is found by calculating

$$\begin{aligned} K_1 &= \bar{\Sigma}_1 C_1^\mathsf{T} (C_1 \bar{\Sigma}_1 C_1^\mathsf{T} + Q)^{-1} \\ &= 1.1 \cdot (1.1 + 0.2)^{-1} \\ &= 1.1 \cdot \frac{1}{1.3} \\ &\approx 0.85 \end{aligned} \tag{A.22}$$

The update is then found by incorporating the measurement. Let us assume the sensor measured a value of $z_1 = 6$. Then we have

$$\begin{aligned} \mu_1 &= \bar{\mu}_1 + K_1(z_1 - C_1 \bar{\mu}_1) \\ \mu_1 &= 5 + \frac{1.1}{1.3} \cdot (6 - 5) \\ \mu_1 &\approx 5.85 \end{aligned} \tag{A.23}$$

The associated covariance is finally calculated by

$$\begin{aligned} \Sigma_t &= (1 - K_1 C_1)\bar{\Sigma}_1 \\ &= (1 - \frac{1.1}{1.3}) \cdot 1.1 \\ &\approx 0.17 \end{aligned} \tag{A.24}$$

Notice that the posterior mean $\mu_1 = 5.85$ is closer to the value proposed by the measurement likelihood (which was 6) than to that of the prediction (which was 5). This is because the measurement uncertainty $Q_1 = 0.2$ is much smaller than the uncertainty of the prediction ($\bar{\Sigma}_1 = 1.1$). Notice further that the resulting posterior uncertainty expressed in Σ_1 is smaller than both the prediction and measurement uncertainties. This is a characteristic behaviour of Gaussian sensor fusion, the fused posterior uncertainty is always smaller than the single uncertainties, expressing that "every piece of information contributes".

A.2.2 The Extended Kalman Filter

The extended Kalman filter (EKF) follows the same principles as the conventional Kalman filter (KF), but allows to use *nonlinear* state transition and measurement functions.

This way, the state prediction becomes:

$$\bar{\boldsymbol{\mu}}_t = f(\boldsymbol{\mu}_{t-1}, \mathbf{u}_{t-1}) \tag{A.25}$$

The predicted mean is simply propagated through the nonlinear state transition function f. To propagate the covariances however, f has to be linearized, i.e. the derivatives with respect to its parameters have to be taken. The derivative (also called the *Jacobian* matrix) with respect to $\boldsymbol{\mu}$ is denoted $\mathbf{J}_{\boldsymbol{\mu}}$, and the derivative with respect to $\mathbf{u}$ is denoted $\mathbf{J}_\mathbf{u}$. The overall Jacobian is then $\mathbf{J} = (\mathbf{J}_\mu, \mathbf{J}_\mathbf{u})$.

Since $\mathbf{J}$ is the linearization of f at the current $\boldsymbol{\mu}_{t-1}$ and $\mathbf{u}_{t-1}$, (the time index in $\mathbf{J}$ has been omitted to increase readability) the normal propagation rule can be applied, which was used before in the prediction step of the KF to propagate the covariances:

$$\bar{\boldsymbol{\Sigma}}_t = \mathbf{J}_{\boldsymbol{\mu}} \boldsymbol{\Sigma}_{t-1} \mathbf{J}_{\boldsymbol{\mu}}^\top + \mathbf{J}_{\mathbf{u}} \mathbf{R}_{t-1} \mathbf{J}_{\mathbf{u}}^\top \tag{A.26}$$

Notice that this time, $\mathbf{R}$ is given in terms of the control inputs and therefore has to be propagated through f as well.

Once the concept of linearization has been understood, it is easy to see how the measurement update or correction step is performed. The measurement model is a nonlinear function $\mathbf{z}_t = h(\mathbf{x}_t)$, and its derivative is denoted $\mathbf{H}$. Then we can calculate the Kalman gain and the posterior as

$$\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^\top \left(\mathbf{H}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^\top + \mathbf{Q}_t \right)^{-1} \tag{A.27}$$

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t \left(\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t) \right) \tag{A.28}$$

$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \bar{\boldsymbol{\Sigma}}_t \tag{A.29}$$

Extended Kalman Filters have been the first filtering algorithms used to solve the SLAM problem [Smith and Cheeseman, 1987; Moutarlier and Chatila, 1989; Moutarlier and Chatila, 1990]. Especially landmark-based SLAM is well suited for EKF-based approaches, since the robot pose and the landmark positions can both be maintained in a single state vector. That state vector however will grow over time as the robot explores its environment and adds more and more landmarks to the map. Since the runtime of EKF is quadratical in the size of the state vector (due to the associated covariance matrix that is growing as well), scalability is an issue and EKF SLAM is only feasible for problems with a limited number of landmarks[51]. Workarounds have been for instance using an information filter that works on the information matrix instead of the covariance matrix [Thrun et al., 2004a], or sub-map techniques that split the problem into several smaller problems that are tackled separately [Guivant et al., 2004; Bailey, 2002; Bosse et al., 2004].

Example A.3. *While the previous two examples dealt with localization only, the example illustrated in Fig. A.4 demonstrates how the EKF can be used to perform SLAM with landmarks. We will see how the robot discovers two landmarks, inserts them into the map and performs a loop closing using one of the landmarks later on. Only the current pose is part of the state vector that is maintained by the filter, not the complete trajectory. This is a major difference to the optimization-based SLAM approaches considered in the main part of this thesis. The current robot pose is depicted with the red Gaussian curve in Fig. A.4.*

At time $t = 0$, we start observing the robot and its filter. Notice how the peak of the red Gaussian distribution which represents the position estimate does not exactly coincide with the ground truth position, indicated by by the small robot. This is a common behaviour

[51] in the order of a thousand landmarks, according to [Thrun et al., 2005] and [Thrun, 2002]

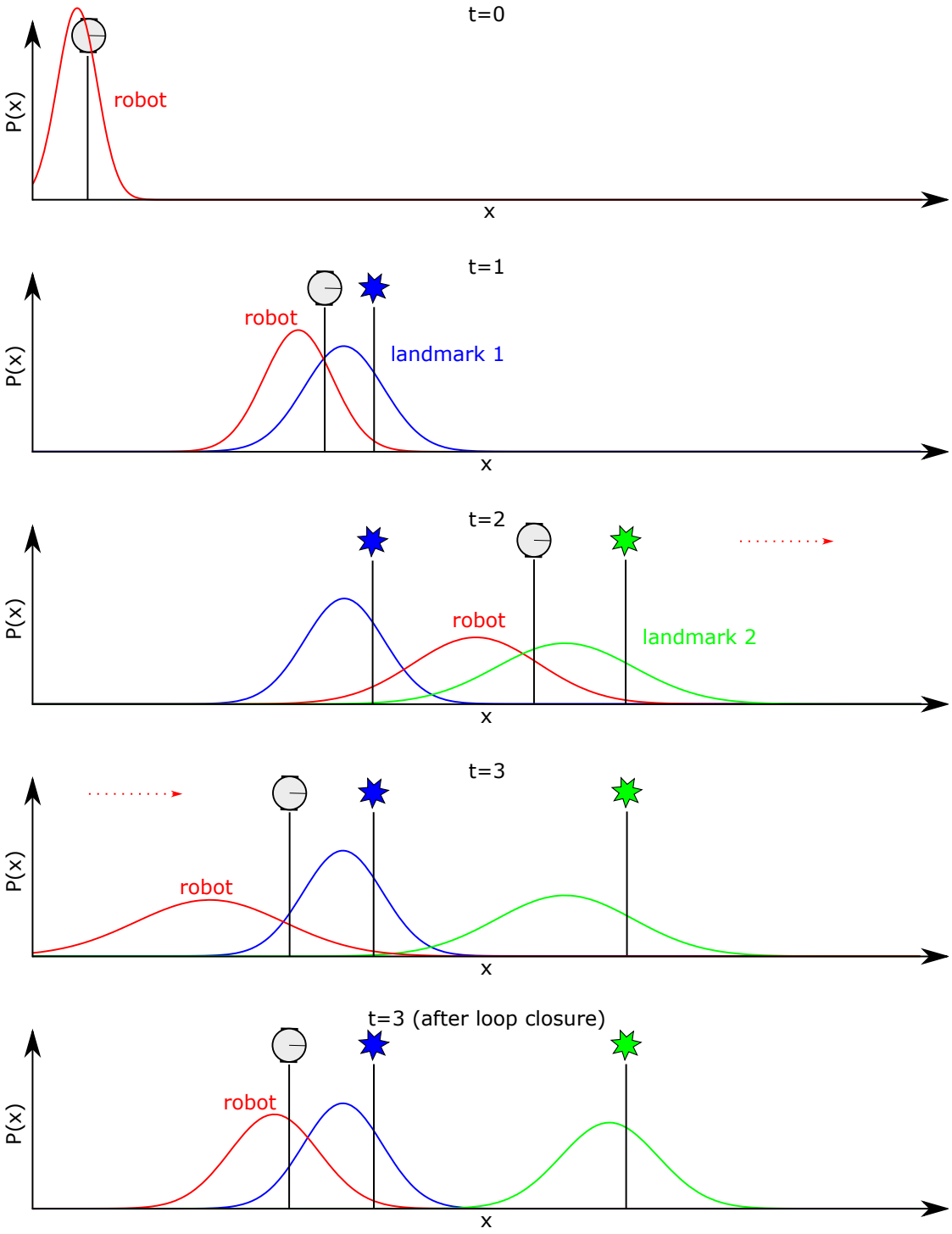


Figure A.4: *SLAM with an extended Kalman filter in a one-dimensional environment. See example A.3 in the text for further explanation.*

and regarded as uncritical as long as the ground truth is within 3σ bounds of the mean of the estimation. Otherwise, the filter would be considered failed and diverged.

At time $t = 1$, we see how the robot moved to the right. Due to the process noise, the uncertainty in its position estimate increased which is visible by the broadened Gaussian. The blue landmark is discovered and added to the state vector, which now contains both the robot's and the landmark's position: $\mathbf{x}_t = (x_t, l_1)^\mathsf{T}$. The covariance matrix has to be updated as well, and it is extended from a scalar to a 2×2 matrix $\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_x \sigma_{l_1} \\ \sigma_{l_1} \sigma_x & \sigma_{l_1}^2 \end{pmatrix}$. The off-diagonal entries capture the correlations between the robot pose x and the landmark position l_1 and hence are usually non-zero. This is an important characteristic of EKF filters: The covariance matrices are dense. Notice from Fig. A.4 that both the estimates of the robot pose and the landmark position are shifted towards the left, compared to the ground truth, but still within 3σ bounds.

After the next time step, at $t = 2$, the robot has moved further and discovered the second landmark, shown in green. Notice how the uncertainty of its position increased further and how the position of the second landmark can only be initialized with a comparably high uncertainty as well. The estimate of the blue landmark has not changed, since no further observations have been made. However, the state vector and covariance matrices have been extended again, to include the second landmark. Now the state vector is $\mathbf{x}_t = (x_t, l_1, l_2)^\mathsf{T}$ and the covariance matrix Σ is a densely occupied 3×3 matrix.

After that, the robot leaves the one-dimensional scenario towards the right and re-enters the scene on the left side. We assume the world to be circular, thus the robot simply re-appears in the world at the opposite end. Now at $t = 3$, the robot is about to re-observe the first landmark and hence perform a loop closing. This is an important step. Notice how much the mean of the robot pose estimate (red curve) differs from the ground truth position due to the accumulated errors. After the loop closure has been performed however, the position has been corrected to about the same deviation it had when the landmark was first discovered. At the same time, the uncertainty of the robot pose estimate has been reduced as well. The second remarkable thing happening is that due to the correlations built up between the two landmarks and the robot position, the second (green) landmark has been corrected as well.

A.2.3 The Unscented Kalman Filter

Remember that in the extended Kalman filter, the derivatives of the state transition and measurement functions were used to propagate the process noise during the prediction step. A problem of EKFs are the linearization errors that occur when Gaussian probability distributions are propagated through these linearized nonlinear functions. These errors can have severe effects on the estimation results and the filter consistency. [Julier and Uhlmann, 2001] and [Bailey and Durrant-Whyte, 2006] discuss the inevitable inconsistencies of EKF SLAM in even the simplest scenarios. A flavour of Kalman Filters that address this problem and aim at reducing linearization errors are the Unscented Kalman Filters (UKFs) [Julier and Uhlmann, 1997] or sigma-point Kalman filters (SPKF) [van der Merwe and Wan, 2004]. These filters are derivative-free, i.e. it is not necessary to calculate

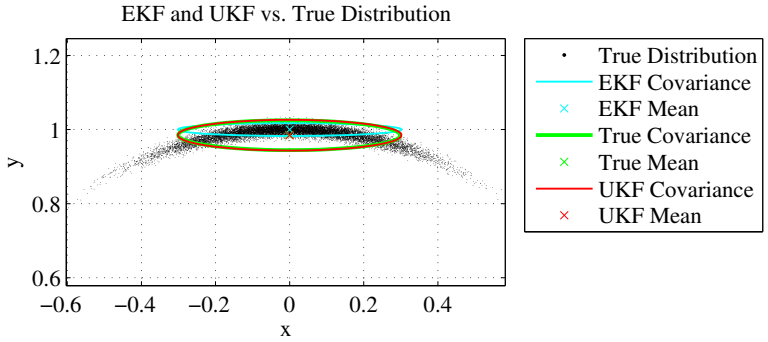


Figure A.5: *SLAM with an extended Kalman filter in a one-dimensional environment. See example A.3 in the text for further explanation.*

Jacobians of the state and measurement equations, which can sometimes be tedious.

The so called *unscented transform* is the heart of the UKF and is the main difference to the EKF algorithm. Its general idea is to select so called *sigma points* from the original (Gaussian) distribution and propagate these sigma points through the nonlinear function in order to calculate the mean and covariance of the propagated Gaussian: Consider we are given a n-dimensional Gaussian variable $\mathbf{x}$ with $x \sim \mathcal{N}(\mathbf{x}, \mathbf{\Sigma})$. This variable $\mathbf{x}$ is propagated through a nonlinear function f. The EKF would linearize the function f at the current estimate of μ and apply the resulting Jacobian matrix $\mathbf{J}$ to propagate $\mathbf{\Sigma}$ according to $\bar{\mathbf{\Sigma}} = \mathbf{J}\mathbf{\Sigma}\mathbf{J}^\mathsf{T}$. For the UKF, no Jacobian has to be calculated. Instead, a small number of $2n+1$ sigma points $\S_i$ are picked, according to a deterministic selection scheme. These sigma points are then propagated through the function, yielding $\dagger_i = f(\S_i)$. The mean and covariance of the propagated Gaussian are then restored from these $\dagger_i$.

Since the unscented transform and the UKF plays no further role in this thesis, details on the sigma point selection scheme and how exactly the mean and covariance are restored from the propagated sigma points $\mathcal{Y}_i$ are omitted here. The reader is referred to the original papers [Julier and Uhlmann, 1997; van der Merwe and Wan, 2004] or secondary literature like [Thrun et al., 2005, ch. 3.4]. In previous work [Sünderhauf et al., 2007], we applied a UKF to monocular SLAM [Davison, 2003; Solà, 2007] for an autonomous aerial vehicle.

Example A.4. *Fig. A.5 rather illustrates the effects of the unscented transform applied to a nonlinear function. Here, the Gaussian input variable was a control input* $\mathbf{u} = (v, \omega)^\mathsf{T}$ *and the nonlinear function was the constant turn rate and velocity motion model* f^{CTRV}. *The black particles in the figure indicate the true resulting distribution of* $\mathbf{x}_t = (x, y)^\mathsf{T}$ *when the motion model is applied:* $\mathbf{x}_t = f^{CTRV}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$. *The initial position was chosen to be* $\mathbf{x}_0 = (0, 0)^\mathsf{T}$ *and the control input was* $\mathbf{u}_0 = (1, 0)^\mathsf{T}$. *Notice that the resulting true distribution* $P(\mathbf{x}_1)$ *is clearly non-Gaussian. Since in the domain of Gaussian filtering, we are forced to express everything with Gaussians (even non-Gaussian distributions), the EKF approximates the true Gaussian distribution using the linearization-and-propagation*

scheme explained before. The result is given with the cyan covariance ellipse and the mean marker. Notice how the EKF estimate diverges from the true Gaussian given in green. The unscented transform (red) is able to reduce the linearization effects and closely follows the true Gaussian representation. However, since the UKF is a Gaussian filter as well, it is not able to capture the true, non-Gaussian nature of the involved distributions.

We have seen that the UKF with the unscented transform has a number of advantages over the EKF. It can significantly reduce the linearization effects that occur when propagating Gaussians through nonlinear functions, it is derivative free and the runtime requirements are of the same order as the EKF or KF. However, since the UKF is a Gaussian filter as well, it is not able to capture the true, non-Gaussian nature of the involved distributions. Other techniques like the particle filter shortly reviewed next are better suited to cope with such non-Gaussian quantities.

A.3 The Particle Filter

The general idea behind particle filters is to represent the posterior distribution in a nonparametric way. This means that instead of using a Gaussian that is characterized by its parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, the posterior distribution is represented by a population of samples. These samples are called *particles*. The main advantage of such a representation is that particle populations can represent arbitrary probability densities and are not limited to certain parametric models. This is especially useful when originally Gaussian distributions are propagated through nonlinear functions. As we have seen before, the resulting posteriors are non-Gaussian and any Gaussian representation can only approximate the true distribution, inevitably introducing errors.

Example A.5. *Fig. A.6 for instance shows how a population of particles can represent the state of a robot that moves according to the CTRV motion model in a 2D world. The robot starts at* $(0, 0)$ *and moves to the right, then turns and so on. The particles belonging to different time steps are drawn in different colors. Notice how the particles spread out and how the particle population changes its outer form over time.*

Each of the particles $\mathbf{x}_t^{[i]}$ *represents the complete robot state, i.e.* $\mathbf{x}_t^{[i]} = (x, y, \theta)^\mathsf{T}$ *at time* t*. The nonlinear motion model* f^{CTRV} *is applied separately to each particle to calculate its new state at time* $t + 1$*, given the control input* $\mathbf{u}_t = (v_t, \omega_t)^\mathsf{T}$*:*

$$\mathbf{x}_{t+1}^{[i]} = f^{CTRV}(\mathbf{x}_t^{[i]}, \mathbf{u}_t^{[i]}) \tag{A.30}$$

with

$$\mathbf{u}_t^{[i]} \sim \mathcal{N}\left(\begin{pmatrix} v_t \\ \omega_t \end{pmatrix}, \boldsymbol{\Sigma}\right) \tag{A.31}$$

Since each $\mathbf{u}_t^{[i]}$ *is an actual sample drawn from the normal distribution, each particle evolves differently through time and the whole population of particles approximates the true distribution* $P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$*.*

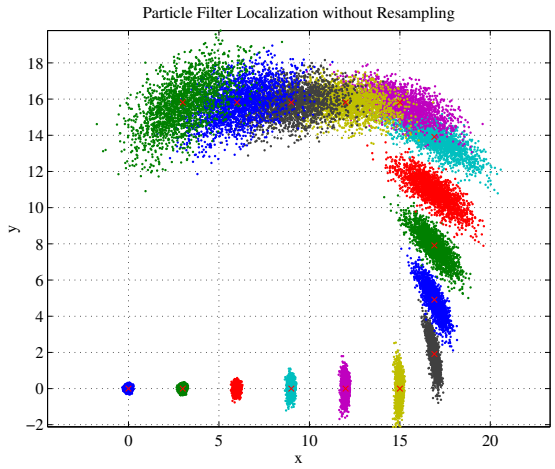


Figure A.6: *Using a population of particles to represent the probability distribution over a robot's position when the robot moves according to a nonlinear constant turn rate and velocity model. The particles for each time step are drawn in a different color. See example A.5 for further explanation.*

Particle filters typically comprise two separate steps: *sampling* and *resampling* (also called *importance sampling*). The *sampling* step has been described in example A.5 and illustrated in Fig. A.6. From the given particle population $\mathcal{X}_t = \{\mathbf{x}_t^{[i]}\}$, the new population $\mathcal{X}_{t+1}$ is built by sampling $\mathbf{x}_{t+1}^{[i]} \sim P(\mathbf{x}_{t+1}|\mathbf{x}_t^{[i]}, \mathbf{u}_t)$ as described above. In addition, in order to incorporate a sensor measurement $\mathbf{z}_t$, each particle is assigned a *weight* $w_t^{[i]} = P(\mathbf{z}_t|\mathbf{x}_t^{[i]})$ and stored in the intermediate particle set $\bar{\mathcal{X}}_{t+1}$

This weight is then used in the second step, the *resampling*: Here the final particle set $\mathcal{X}_{t+1}$ is built from the intermediate set. This is achieved by randomly drawing particles from the intermediate set $\bar{\mathcal{X}}_{t+1}$ according to a probability that is proportional to the weights of the particles. That means, those particles with a high weight (i.e. with a high measurement likelihood since $w_t^{[i]} = P(\mathbf{z}_t|\mathbf{x}_t^{[i]})$) are drawn with higher probability and have a higher chance to "survive". Notice that single particles can be drawn multiple times.

Example A.6. *The sampling and resampling has been combined in Fig. A.7. Here a landmark is present at position* $(10, 10)^\mathsf{T}$ *and illustrated by the red square. The range to the landmark can be observed by the robot. The same motion model and parameters as in Fig. A.6 were used to perform the sampling. Due to the resampling however, particles that are far away from the ground truth position are pruned away (since they have only a small chance to survive the resampling) and the particle population clusters tightly around the true robot position.*

Particle filters, especially their Rao-Blackwellized flavour, have been a popular measure to address the SLAM problem since they bear the advantage that they can cope with

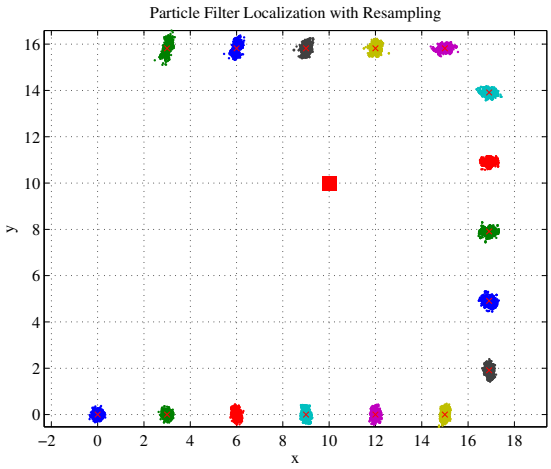


Figure A.7: *Localization using a particle filter. The robot moves around a landmark (red square) and can observe its distance. Through the resampling step, particles with a low measurement likelihood have only a small chance to survive. The ground truth position is indicated by the red cross. Further explanations are given in example A.6*

multimodal non-Gaussian distributions: FastSLAM [Montemerlo et al., 2002; Thrun et al., 2004b] and its enhancement FastSLAM 2.0 [Montemerlo et al., 2003] have been successfully applied to both occupancy grids and landmark based maps. Other popular particle filter SLAM systems are GMapping [Grisetti et al., 2007] or DP-SLAM [Eliazar and Parr, 2004]. In previous work we demonstrated visual FastSLAM using SURF features [Neubert et al., 2007]. Particle filters are also well suited for localization only, which has been demonstrated e.g. by [Fox et al., 1999]. Further details on particle filters and their application for SLAM can be found in [Thrun et al., 2005, ch. 4].

A.4 Summary

This appendix reviewed the most important filtering techniques that have been used for SLAM and related problems in the past and are still used today. The review was held relatively short, since filtering is not in the scope of this thesis. For the same reason, a branch of filtering techniques called the information filter and certain derived techniques [Thrun et al., 2004a] were omitted completely, as were Gaussian sum filters [Alspach and Sorenson, 1972]. Also, the data association problem was not considered so far, since the landmarks in the example were assumed to be identified correctly. In real-world applications, the unknown or at least uncertain association from measurements to landmarks has to be considered in the filter but complicates the explanation.

Translations

TECHNISCHE UNIVERSITÄT CHEMNITZ

Robuste Optimierung für simultane Lokalisierung und Kartierung

von der Fakultät für Elektrotechnik und Informationstechnik
der Technischen Universität Chemnitz

genehmigte

Dissertation

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von

Dipl.-Inf. Niko Sünderhauf

geboren am 10. April 1981 in Zwickau

eingereicht am 14. Februar 2012

Gutachter: Prof. Dr.-Ing. Peter Protzel
Prof. Dr.-Ing. Jozef Suchý

Tag der Verleihung: 19. April 2012

Zusammenfassung

Im Forschungsbereich der mobilen Robotik beschreibt das sog. SLAM-Problem das Kartieren einer noch unbekannten Umgebung durch einen Roboter. Das Acronym *SLAM* steht dabei für den englischen Ausdruck *Simultaneous Localization and Mapping* und betont, dass die Kartierung der Umgebung und die Lokalisierung des Roboters innerhalb dieser Umgebung *gleichzeitig* stattfinden müssen, um eine korrekte und genaue Repräsentation der Umwelt zu erhalten.

Seit das Problem am Ende der 80er Jahre zum ersten Mal beschrieben wurde, wurden viele verschiedene Algorithmen und Lösungsverfahren entwickelt. Über lange Zeit basierten diese Ansätze auf Bayes'schen *Filtern*, wie zum Beispiel dem Kalman-Filter und seinen verschiedenen Varianten, oder dem Partikel-Filter. In den letzten Jahren lässt sich jedoch in der wissenschaftlichen Literatur ein neuer Trend beobachten. Der Fokus neuer Arbeiten auf dem Gebiet verschiebt sich mehr und mehr in Richtung optimierungsbasierter Verfahren, die SLAM als nichtlineares quadratisches Optimierungsproblem formulieren. Obwohl derartige Lösungsansätze ihre Leistungsfähigkeit und ihre Vorteile gegenüber den Filter-Verfahren bereits unter Beweis gestellt haben, teilen sie ein generelles Problem aller quadratischen Optimierungsverfahren: Sie sind nicht robust gegenüber Fehlern wie zum Beispiel sog. Ausreißern in den zugrundeliegenden Daten. Solche Ausreißer entstehen vor allem durch eine fehlerhafte Zuordnung von Sensordaten zu bereits bekannten Strukturen in der Karte bzw. durch eine falsche Erkennung von sog. Schleifenschlüssen (Loop Closures) innerhalb der Trajektorie des Roboters. Das eigentliche Optimierungsverfahren muss sich daher darauf verlassen, dass in einem Vorverarbeitungsschritt sämtliche Fehler und Ausreißer erkannt und entfernt wurden. Andernfalls ist die geschätzte Kartenstruktur mit enormen Fehlern behaftet und nicht nutzbar.

Die vorliegende Arbeit konzentriert sich darauf, den Optimierungskern von SLAM *robust* gegenüber Ausreißern und Datenfehlern zu gestalten und damit unabhängig von der Leistung evntl. vorgeschalteter Algorithmen zu machen. Dies wird erreicht, in dem die Topologie der Graph-basierten Reräsentation des Optimierungsproblems selbst während der Optimierung verändert werden kann. Damit kann der Optimierer einzelne Datenelemente während der Lösungssuche als fehlerhaft identifizieren und aus der Problemformulierung entfernen. Im Ergebnis kann das vorgeschlagene Verfahren auch dann gegen eine korrekte Lösung konvergieren, wenn zahlreiche Fehler und Ausreißer vorhanden sind. Die Leistungsfähigkeit des vorgeschlagenen robusten SLAM-Verfahrens wird anhand großer synthetischer und realer Datensätze gezeigt und bewertet.

Die Arbeit zeigt weiterhin, dass das vorgeschlagene Verfahren nicht auf SLAM beschränkt ist, sondern darüber hinaus auch in anderen Problemfeldern eingesetzt werden kann, in denen nichtlineare quadratische Optimierungsprobleme zu lösen sind, aber mit Ausreißern in den zugrundeliegenden Daten zu rechnen ist. Dies kann erfolgreich für die satellitengestützte Positionsbestimmung demonstriert werden. Die Datenfehler oder Ausreißer in diesem Problem entstehen besonders in städtischen Umgebungen durch Signalreflexionen an Gebäuden und sog. Mehrwegeempfangsfehler. Es wird anhand eines realen Datensatzes gezeigt, dass mit Hilfe des entwickelten Systems trotz zahlreicher derartiger Fehler eine genaue Positionierung des Empfängers möglich ist.

Inhaltsverzeichnis

1 Einführung **13**
Ziele und Beiträge der Arbeit 18
Überblick der Arbeit 18

2 SLAM – Gleichzeitige Kartographierung und Lokalisierung **21**
2.1 Die Teile von SLAM 21
2.1.1 Kartieren 23
2.1.2 Lokalisieren 32
2.1.3 Gemeinsames Schätzen von Karte und Position 35
2.2 Graphen als Repräsentationen für SLAM 41
2.2.1 Dynamische Bayes'sche Netzwerke 41
2.2.2 Faktor Graphen 42
2.2.3 Markov-Netzwerke 43
2.3 SLAM als nichtlineares Optimizierungproblem 44
2.3.1 Pose Graph SLAM 44
2.3.2 Ableitung einer Formulierung für ein nichtlineares Optimierungsproblem 45
2.3.3 Eine anschauliche Analogie für die nichtlineare Optimierung 48
2.3.4 SLAM als Optimierungsproblem – Literaturüberblick 49
2.4 Zusammenfassung 51

3 Optimierung durch die Methode der kleinsten Quadrate **53**
3.1 Einführung 53
3.1.1 Eine Taxonomie von Optimierungsproblemen 54
3.1.2 Quadratische Optimierungsprobleme 55
3.2 Lineare quadratische Probleme 55
3.2.1 Lösen von linearen quadratischen Optimierungsproblemen 56
3.2.2 Beispiele für quadratische Optimierungsprobleme 57
3.3 Nichtlineare quadratische Optimierungsprobleme 59
3.3.1 Die Methode des Gradientenabstiegs 60
3.3.2 Newtons Methode 61
3.3.3 Der Gauss-Newton-Algorithmus 63
3.3.4 Der Algorithmus nach Levenberg und Marquardt 64
3.3.5 Zusammenfassung 65

3.4 Gewichtete nichtlineare quadratische Probleme . . . 66
3.5 Quadratische Optimierung für SLAM . . . 68
3.5.1 Ein einfaches Beispiel . . . 69
3.5.2 Gründe der Effizienz von Optimierungs-basierten SLAM-Algorithmen 72
3.6 Quadratisches Optimieren mit Ausreißern . . . 74
3.6.1 Methoden zum Erkennen und Unterdrücken von Ausreißern . . . 75
3.6.2 Robuste Kostenmaße . . . 77
3.7 Zusammenfassung . . . 79

4 Die Grenzen von SLAM – Motivation 81
4.1 Datenkorrespondezfehler und ihre Auswirkungen . . . 82
4.2 Aktuelle Ansätze für Erkennung und Vermeidung von Ausreißern . . . 83
4.2.1 Vermeiden von Ausreißern im Front-End . . . 84
4.2.2 Verminderung der Effekte von Ausreißern im Back-End . . . 86
4.3 Zusammenfassung . . . 87

5 Ein robustes Back-End für SLAM 89
5.1 Eine robuste Formulierung für Pose Graph SLAM . . . 90
5.1.1 Erste Ansätze einer mathematischen Formulierung . . . 90
5.1.2 Einführung der Switch-Variablen und Bestimmen einer geeigneten Switch Function . . . 92
5.1.3 Einführung der Switch Priors . . . 95
5.1.4 Zusammenfügen des Systems . . . 96
5.2 Diskussion . . . 97
5.2.1 Der Einfluss von s_{ij} auf die Informationsmatrix Λ_{ij} . . . 98
5.2.2 Die Verbindung zur maximum-a-posteriori-Lösung . . . 100
5.2.3 Einfluss der zusätzlichen Variablen auf die Größe des Optimierungsproblems . . . 102
5.2.4 Einfluss der zusätzlichen Variablen auf die Struktur des Optimierungsproblems . . . 102
5.2.5 Einfluss der zusätzlichen Variablen auf die Konvergenzeigenschaften der Optimierung . . . 105
5.3 Zusammenfassung und ein erstes Beispiel . . . 105

6 Bewertung des vorgeschlagenen Systems 109
6.1 Fehlermetriken für SLAM . . . 110
6.1.1 Der mittlere absolute Fehler RMSE . . . 110
6.1.2 Der relative Positionsfehler . . . 111
6.1.3 Statistiken über Genauigkeit und Trefferquote . . . 112
6.2 Benutzte Datensätze für die Systembewertung . . . 113
6.3 Allgemeine Vorgehensweise . . . 114
6.3.1 Systematische Erzeugung von Datenfehlern zur Untersuchung des Systemverhaltens . . . 115
6.3.2 Angenommene Eigenbewegung . . . 117

6.3.3 Die Switch-Funktion . . . 119
6.3.4 Benutzte Software und Implementierung . . . 119
6.4 Einfluss von Ξ_{ij} auf das Ergebnis der Schätzung . . . 119
6.4.1 Vorgehensweise . . . 120
6.4.2 Ergebnisse und Interpretation . . . 120
6.5 Untersuchung der Robustheit des Systems . . . 121
6.5.1 Vorgehensweise . . . 121
6.5.2 Ergebnisse und Interpretation . . . 121
6.5.3 Diskussion der Fehlerfälle . . . 125
6.6 Laufzeit- und Konvergenzverhalten . . . 129
6.6.1 Vorgehensweise . . . 129
6.6.2 Ergebnisse und Interpretation . . . 129
6.7 Verhalten bei Fehlerfreiheit . . . 130
6.7.1 Vorgehensweise . . . 130
6.7.2 Ergebnisse und Interpretation . . . 132
6.8 Einfluss der Switch-Funktion Ψ . . . 133
6.9 Zusammenfassung der Systembewertung und erste Schlussfolgerungen . . . 134

7 Anwendung des robusten Back-Ends in einem vollständigen SLAM-System und Test auf einem realen Datensatz 137
7.1 Das Front-End . . . 138
7.1.1 Visuelle Odometrie durch Bildprofile . . . 138
7.1.2 Ortserkennung mit BRIEF-Gist . . . 139
7.2 Ergebnisse des Gesamtsystems . . . 142
7.3 Zusammenfassung . . . 145

8 Weitere Anwendungen – Identifizierung von Mehrwegeempfang bei der satellitengestützen Positionierung 147
8.1 Einführung in die satellitengestützte Positionierung . . . 148
8.1.1 Systematische Fehler . . . 149
8.1.2 Mehrwegefehler . . . 150
8.2 Aktuelle Ansätze zur Erkennung von Mehrwegefehlern . . . 151
8.3 Modellierung des satellitengestützten Lokalisierungsproblems als Faktor-Graph . . . 152
8.3.1 Die Systemzustandsknoten . . . 152
8.3.2 Der Pseudostrecken-Faktor . . . 153
8.3.3 Weitere Faktoren . . . 154
8.3.4 Errechnen der maximum-a-posteriori-Lösung . . . 155
8.4 Entwickeln einer robusten Problemformulierung . . . 156
8.4.1 Verbindung der Switch-Variablen mit dem Pseudostrecken-Faktor . 156
8.4.2 Der Switch-Transition-Faktor . . . 157
8.5 Demonstration der Abschwächung von Mehrwegeempfangsfehlern in städtischen Umgebungen . . . 158
8.5.1 Der verwendete Datensatz – Chemnitz Innenstadt . . . 159

8.5.2 Vorgehensweise . . . 160
8.5.3 Ergebnisse . . . 161
8.6 Interpretation und Zusammenfassung . . . 165
8.7 Ausblick . . . 167

9 Weitere Anwendungen im Bereich Sensorfusion und Sensorkalibrierung 169
9.1 Sensorfusion durch robuste Optimierung . . . 169
9.2 Kalibrierung von Sensoren durch robuste Optimierung . . . 171

10 Zusammenfassung 173
10.1 Beiträge der Arbeit und erreichte Ziele . . . 174
10.2 Offene Fragen und Ausblick auf weitere Arbeiten . . . 175
10.2.1 Parameter der vorgeschlagenene robusten Problemformulierung . . . 175
10.2.2 Konvergenzverhalten und die Gefahren von lokalen Minima . . . 176
10.2.3 Weitere Anwendungen des Systems . . . 176

Bibliography

[Agrawal and Konolige, 2008] Agrawal, M. and Konolige, K. (2008). CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching. In *ECCV*.

[Alspach and Sorenson, 1972] Alspach, D. and Sorenson, H. (1972). Nonlinear bayesian estimation using gaussian sum approximations. *Automatic Control, IEEE Transactions on*, 17(4):439 – 448.

[Angeli et al., 2008] Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2008). Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words. *Robotics, IEEE Transactions on*, 24(5):1027 –1037.

[Bailey, 2002] Bailey, T. (2002). *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. PhD thesis, Australian Centre for Field Robotics, University of Sydney.

[Bailey and Durrant-Whyte, 2006] Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous Localisation and Mapping (SLAM): Part II State of the Art. *IEEE Robotics and Automation Magazine*, 13(3):108–117.

[Balakrishnan and Childs, 2001] Balakrishnan, N. and Childs, A. (2001). Outlier. In Hazewinkel, M., editor, *Encyclopaedia of Mathematics*. Springer.

[Bar-Shalom, 2002] Bar-Shalom, Y. (2002). Update with out-of-sequence measurements in tracking: exact solution. *Aerospace and Electronic Systems, IEEE Transactions on*, 38(3):769 – 777.

[Bauer, 2011] Bauer, S. (2011). Modellierung von GPS-Mehrwegeausbreitungen mit Hilfe von digitalen Karten und 3D-Modellen (Diplomarbeit). Master's thesis, Hochschule für Technik und Wirtschaft, Dresden, Germany.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Gool, L. V. (2006). Surf: Speeded up robust features. In *Proceedings of the ninth European Conference on Computer Vision*.

[Blake and Zisserman, 1987] Blake, A. and Zisserman, A. (1987). *Visual Reconstruction*. MIT Press, Cambridge, USA.

[Borenstein et al., 1996] Borenstein, J., Everett, H. R., and Feng, L. (1996). Where am I? – Systems and Methods for Mobile Robot Positioning. `http://www-personal.engin.umich.edu/~johannb/position.htm/`.

[Borenstein et al., 1991] Borenstein, J., Koren, Y., and Member, S. (1991). The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7:278–288.

[Bosse et al., 2004] Bosse, M., Newman, P., Leonard, J., and Teller, S. (2004). Slam in large-scale cyclic environments using the atlas framework. *Artificial Intelligence*, 23(12):1113–1139.

[Boyd and Vandenberghe, 2004] Boyd, S. P. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[Bradski, 2000] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[Braitenberg, 1986] Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. MIT Press.

[Burgard et al., 2009] Burgard, W., Stachniss, C., Grisetti, G., Steder, B., Kümmerle, R., Dornhege, C., Ruhnke, M., Kleiner, A., and Tardós, J. D. (2009). A comparison of SLAM algorithms based on a graph of relations. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA.

[Cadena et al., 2010] Cadena, C., Gálvez-López, D., Ramos, F., Tardós, J. D., and Neira, J. (2010). Robust Place Recognition with Stereo Cameras. In *IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[Cadena et al., 2011] Cadena, C., McDonald, J., Leonard, J. J., and Neira, J. (2011). Place Recognition using Near and Far Visual Information. In *Proc. of the 18th IFAC World Congress*.

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision (ECCV)*. Springer.

[Carlone et al., 2011a] Carlone, L., Aragues, R., Castellanos, J., and Bona, B. (2011a). A linear approximation for graph-based simultaneous localization and mapping. In *Proc. of Robotics: Science and Systems, RSS*.

[Carlone et al., 2011b] Carlone, L., Aragues, R., Castellanos, J. A., and Bona, B. (2011b). A first-order solution to simultaneous localization and mapping with graphical models. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1764 –1771.

[Censi, 2008] Censi, A. (2008). An ICP Variant Using a Point-to-Line Metric. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 19 –25.

[Chow and Liu, 1968] Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462 – 467.

[Chum and Matas, 2005] Chum, O. and Matas, J. (2005). Matching with prosac – progressive sample consensus. In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR*, volume 1, pages 220–226. IEEE.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. Number 978-0262033848. MIT Press, ISBN 978-0262033848.

[Cummins and Newman, 2008] Cummins, M. and Newman, P. (2008). FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665.

[Cummins and Newman, 2009] Cummins, M. and Newman, P. (2009). Highly Scalable Appearance-Only SLAM – FAB-MAP 2.0. In *Robotics Science and Systems*.

[Davis, 2006] Davis, T. A. (2006). *Direct Methods for Sparse Linear Systems*, volume 75. SIAM.

[Davison, 2003] Davison, A. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the ninth international Conference on Computer Vision ICCV'03, IEEE Computer Society Press.*, Nice, France.

[Dean and Kanazawa, 1988] Dean, T. and Kanazawa, K. (1988). Probabilistic Temporal reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence AAAI88*, pages 524–528. American Association for Artificial Intelligence.

[Deisenroth and Ohlsson, 2011] Deisenroth, M. P. and Ohlsson, H. (2011). A General Perspective on Gaussian Filtering and Smoothing: Explaining Current and Deriving New Algorithms. In *Proceedings of the 2011 American Control Conference (ACC)*.

[Dellaert and Kaess, 2006] Dellaert, F. and Kaess, M. (2006). Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *Intl. J. of Robotics Research, IJRR*, 25(12).

[Diosi and Kleeman, 2007] Diosi, A. and Kleeman, L. (2007). Fast laser scan matching using polar coordinates. *The International Journal of Robotics Research*, 26(10):1125–1153.

[Duckett et al., 2002] Duckett, T., Marsland, S., and Shapiro, J. (2002). Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE Robotics and Automation Magazine*, 13(2):99–110.

[Durrant-Whyte et al., 1995] Durrant-Whyte, H. F., Rye, D., and Nebot, E. (1995). *Localisation of automatic guided vehicles*, volume 25, pages 613–625. Springer Verlag.

[Eade and Drummond, 2009] Eade, E. and Drummond, T. (2009). Edge landmarks in monocular slam. *Image and Vision Computing*, 27(5):588–596.

[Eliazar and Parr, 2004] Eliazar, A. and Parr, R. (2004). DP-SLAM: Fast, Robust Simultainous Localization and Mapping Without Predetermined Landmarks. In *Procof the IntConfon Artificial Intelligence IJCAI*, pages 1135–1142. IEEE.

[Engelhard et al., 2011] Engelhard, N., Endres, F., Hess, J., Sturm, J., and Burgard, W. (2011). Real-time 3D visual SLAM with a hand-held RGB-D camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, Vasteras, Sweden.

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

[Fleps et al., 2011] Fleps, M., Mair, E., Ruepp, O., Suppa, M., and Burschka, D. (2011). Optimization based imu camera calibration. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3297 –3304.

[Folkesson and Christensen, 2004] Folkesson, J. and Christensen, H. (2004). Graphical SLAM – A Self-Correcting Map. In *IEEE International Conference on Robotics and Automation 2004 Proceedings ICRA 04 2004*, volume 1, pages 383–390. Ieee.

[Fox et al., 1999] Fox, D., Burgard, W., Dellaert, F., and Thrun, S. (1999). Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence*.

[Frese, 2010] Frese, U. (2010). Interview: Is SLAM Solved? *KI - Künstliche Intelligenz*, 24:255–257.

[Frese et al., 2005] Frese, U., Larsson, P., and Duckett, T. (2005). A multilevel relaxation algorithm for simultaneous localization and mapping. In *IEEE Transactions on Robotics*, volume 21, pages 196–207. IEEE.

[Galvez-Lopez and Tardos, 2011] Galvez-Lopez, D. and Tardos, J. D. (2011). Real-time loop detection with bags of binary words. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 51 –58.

[Gauss, 1809] Gauss, C. F. (1809). *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Hamburg.

[Gelb, 1974] Gelb, A. (1974). *Applied optimal estimation*. MIT Press.

[Glover et al., 2010] Glover, A., Maddern, W., Milford, M., and Wyeth, G. (2010). FAB-MAP + RatSLAM : Appearance-Based SLAM for Multiple Times of Day. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA2010)*, Anchorage, Alaska.

[Golfarelli et al., 1998] Golfarelli, M., Maio, D., and Rizzi, S. (1998). Elastic correction of dead-reckoning errors in map building. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 905–911 vol.2.

[Grewal et al., 2007] Grewal, M. S., Weill, L. R., and Andrews, A. P. (2007). *Global Positioning Systems, Inertial Navigation, and Integration.* Wiley Interscience, ISBN 9780470041901, second edition.

[Grisetti et al., 2010] Grisetti, G., K, R., Stachniss, C., and Hertzberg, C. (2010). Hierarchical optimization on manifolds for online 2d and 3d mapping. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 273–278. IEEE.

[Grisetti et al., 2011] Grisetti, G., Kümmerle, R., Strasdat, H., and Konolige, K. (2011). g^2*o*: A general Framework for (Hyper) Graph Optimization. Technical report.

[Grisetti et al., 2007] Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46.

[Grisetti et al., 2009] Grisetti, G., Stachniss, C., and Burgard, W. (2009). Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3).

[Guivant et al., 2004] Guivant, J., Nebot, E., Nieto, J., and Masson, F. (2004). Navigation and mapping in large unstructured environments. *The International Journal of Robotics Research*, 23(4):449–472.

[Guivant et al., 2002] Guivant, J. E., Masson, F. R., and Nebot, E. M. (2002). Simultaneous localization and map building using natural features and absolute information. *Robotics and Autonomous Systems*, 40(2-3):79–90.

[Gutmann and Konolige, 1999] Gutmann, J. and Konolige, K. (1999). Incremental mapping of large cyclic environments. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, page 318–325, Monterey, California.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference 1988*, pages 147–151.

[Hartley and Zisserman, 2004] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition.

[Henry et al., 2010] Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox., D. (2010). RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. In *Proc. of International Symposium on Experimental Robotics, ISER10.*

[Hertzberg et al., 2011] Hertzberg, C., Wagner, R., Frese, U., and Schröder, L. (2011). Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion.*

[Hol et al., 2010] Hol, J. D., Schön, T. B., and Gustafsson, F. (2010). Modeling and calibration of inertial and vision sensors. *International Journal of Robotics Research*, 29(2):231–244.

[Huang et al., 2010] Huang, S., Lai, Y., Frese, U., and Dissanayake, G. (2010). How far is SLAM from a linear least squares problem? In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[Huang et al., 2012] Huang, S., Wang, H., Frese, U., and Dissanayake, G. (2012). On the Number of Local Minima to the Point Feature Based SLAM Problem. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA)*. under review.

[Huber, 1973] Huber, P. J. (1973). Robust regression: Asymptotics, conjectures and monte carlo. *The Annals of Statistics*, 1(5):799–821.

[Itti et al., 1998] Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20.

[Julier and Uhlmann, 1997] Julier, S. and Uhlmann, J. (1997). A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*.

[Julier and Uhlmann, 2001] Julier, S. and Uhlmann, J. (2001). A counter example to the theory of simultaneous localization and map building. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4238–4243.

[Jung and Lacroix, 2005] Jung, I.-K. and Lacroix, S. (2005). Simultaneous localization and mapping with stereovision. In Dario, P. and Chatila, R., editors, *Robotics Research*, volume 15 of *Springer Tracts in Advanced Robotics*, pages 315–324. Springer Berlin / Heidelberg.

[Kaess et al., 2010] Kaess, M., Ila, V., Roberts, R., and Dellaert, F. (2010). The Bayes Tree: An Algorithmic Foundation for Probabilistic Robot Mapping. In *Proc. of Intl. Workshop on the Algorithmic Foundations of Robotics*.

[Kaess et al., 2011] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., and Dellaert, F. (2011). iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*.

[Kaess et al., 2012] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., and Dellaert, F. (2012). iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. *International Journal of Robotics Research*.

[Kaess et al., 2008] Kaess, M., Ranganathan, A., and Dellaert, F. (2008). iSAM: Incremental Smoothing and Mapping. *IEEE Transactions on Robotics*, 24(6).

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal Of Basic Engineering*, 82(Series D):35–45.

[Kelly and Sukhatme, 2011] Kelly, J. and Sukhatme, G. S. (2011). Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *International Journal of Robotics Research*, 30(1):56–79.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5:90–98.

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan.

[Konolige, 2004] Konolige, K. (2004). Large-scale map-making. In *Proceedings of the National Conference on AI (AAAI)*.

[Konolige and Bowman, 2009] Konolige, K. and Bowman, J. (2009). Towards Lifelong Visual Maps. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1156 –1163.

[Konolige et al., 2010a] Konolige, K., Bowman, J., Chen, J. D., Mihelich, P., Calonder, M., Lepetit, V., and Fua, P. (2010a). View-based maps. *International Journal of Robotics Research (IJRR)*, 29(10).

[Konolige et al., 2010b] Konolige, K., Grisetti, G., Kümmerle, R., Burgard, W., Limketkai, B., and Vincent, R. (2010b). Efficient sparse pose adjustment for 2d mapping. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[Krauthausen et al., 2006] Krauthausen, P., Kipp, A., and Dellaert, F. (2006). Exploiting locality in slam by nested dissection. In *Robotics: Science and Systems*.

[Kschischang et al., 2001] Kschischang, F., Frey, B., and Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

[Kümmerle et al., 2011a] Kümmerle, R., Grisetti, G., and Burgard, W. (2011a). Simultaneous Calibration, Localization, and Mapping. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3716 –3721.

[Kümmerle et al., 2011b] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011b). g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*.

[Kümmerle et al., 2009] Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., and Kleiner, A. (2009). On Measuring the Accuracy of SLAM Algorithms. *Auton. Robots*, 27:387–407.

[Lacroix et al., 2008] Lacroix, S., Lemaire, T., and Berger, C. (2008). More vision for slam. In Kragic, D. and Kyrki, V., editors, *Unifying Perspectives in Computational and Robot Vision*, volume 8 of *Lecture Notes in Electrical Engineering*, pages 129–147. Springer US.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc 18th International Conf on Machine Learning*, volume CONF 18, pages 282–289. Morgan Kaufmann, San Francisco, CA.

[Lange et al., 2011] Lange, S., Sünderhauf, N., Neubert, P., Drews, S., and Protzel, P. (2011). Autonomous Corridor Flight of a UAV Using a Low-Cost and Light-Weight RGB-D Camera. In *Proc. of Intl. Symposium on Autonomous Mini Robots for Research and Edutainment (AMiRE11)*.

[Lavalle and Kuffner, 1999] Lavalle, S. M. and Kuffner, J. J. (1999). Randomized kinodynamic planning. *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, 1:473–479 vol.1.

[Lemaire et al., 2007] Lemaire, T., Berger, C., Jung, I.-K., and Lacroix, S. (2007). Vision-based slam: Stereo and monocular approaches. *International Journal of Computer Vision*, 74(3):343–364.

[Lemaire and Lacroix, 2007] Lemaire, T. and Lacroix, S. (2007). Monocular-vision based slam using line segments. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2791 –2796.

[Lobo and Dias, 2007] Lobo, J. and Dias, J. (2007). Relative pose calibration between visual and inertial sensors. *The International Journal of Robotics Research*, 26(6):561–575.

[Lourakis and Argyros, 2004] Lourakis, M. and Argyros, A. (2004). The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece. Available from `http://www.ics.forth.gr/~lourakis/sba`.

[Lourakis and Argyros, 2005] Lourakis, M. and Argyros, A. (2005). Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1526 –1531 Vol. 2.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. In *International Journal of Computer Vision, 60, 2*.

[Lu and Milios, 1997] Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349.

[Maddern et al., 2009] Maddern, W., Glover, A., Milford, M., and Wyeth, G. (2009). Augmenting RatSLAM using FAB-MAP-based Visual Data Association. In *Proceedings of Australasian Conference on Robotics and Automation (ACRA09)*.

[Maddern et al., 2011] Maddern, W., Milford, M., and Wyeth, G. (2011). Continuous Appearance-based Trajectory SLAM. In *International Conference on Robotics and Automation (ICRA)*.

[Madsen et al., 2004] Madsen, K., Nielsen, H. B., and Tingleff, O. (2004). Methods for non-linear least squares problems, 2nd edition. Technical Report.

[Makarenko et al., 2002] Makarenko, A., Williams, S., Bourgault, F., and Durrant-Whyte, H. (2002). An Experiment in Integrated Exploration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 534 – 539 vol.1.

[Mar et al., 2010] Mar, L., Pini, P., and G, D. (2010). CI-Graph Simultaneous Localization and Mapping for Three-Dimensional Reconstruction of Large and Complex Environments Using a Multicamera System. *Journal of Field Robotics*, 27(5):561–586.

[Matas et al., 2002] Matas, J., Chum, O., Martin, U., and Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of British Machine Vision Conference*, volume 1, pages 384–393, London.

[McDonald et al., 2011] McDonald, J., Kaess, M., Cadena, C., Neira, J., and Leonard, J. J. (2011). 6-DOF Multi-session Visual SLAM using Anchor Nodes. In *Proc. of European Conference on Mobile Robots, ECMR*, pages 69–76.

[Meguro et al., 2009] Meguro, J.-i., Murata, T., Takiguchi, J.-i., Amano, Y., and Hashizume, T. (2009). GPS Multipath Mitigation for Urban Area Using Omnidirectional Infrared Camera. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):22 –30.

[Milford, 2008] Milford, M. J. (2008). *Robot Navigation from Nature*. Springer Verlag.

[Milford et al., 2005] Milford, M. J., Prasser, D., and Wyeth, G. F. (2005). Experience Mapping: Producing Spatially Continuous Environment Representations Using RatSLAM. In *Proc. of Australasian Conference on Robotics and Automation*, Sydney, Australia.

[Milford and Wyeth, 2008] Milford, M. J. and Wyeth, G. F. (2008). Mapping a Suburb with a Single Camera using a Biologically Inspired SLAM System. *IEEE Transactions on Robotics*, 24(5).

[Montemerlo et al., 2002] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). FastSLAM: A Factored Solution to Simultaneous Localization and Mapping. In *Proc. of the National Conference on Artificial Intelligence AAAI*, pages 593–598.

[Montemerlo et al., 2003] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2003). Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico.

[Montiel et al., 2006] Montiel, J., Civera, J., and Davison, A. (2006). Unified inverse depth parametrization for monocular slam. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA.

[Moravec and Elfes, 1985] Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *Proc of the IEEE International Conference on Robotics and Automation*, pages 116–121.

[Moutarlier and Chatila, 1989] Moutarlier, P. and Chatila, R. (1989). Stochastic multisensory data fusion for mobile robot location and environment modeling. *5th Int Symposium on Robotics Research*, 1(2):85–94.

[Moutarlier and Chatila, 1990] Moutarlier, P. and Chatila, R. (1990). An experimental system for incremental environment modelling by an autonomous mobile robot. In Hayward, V. and Khatib, O., editors, *Experimental Robotics I*, volume 139 of *Lecture Notes in Control and Information Sciences*, pages 327–346. Springer Berlin / Heidelberg. 10.1007/BFb0042528.

[Murillo and Kosecka, 2009] Murillo, A. and Kosecka, J. (2009). Experiments in place recognition using gist panoramas. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 2196 –2203.

[Neira and Tardos, 2001] Neira, J. and Tardos, J. (2001). Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897.

[Neubert et al., 2007] Neubert, P., Sünderhauf, N., and Protzel, P. (2007). Fastslam using surf features: An efficient implementation and practical experiences. In *Proceedings of the International Conference on Intelligent and Autonomous Vehicles, IAV07*, Tolouse, France.

[Newcombe et al., 2011] Newcombe, R. A., Lovegrove, S., and Davison, A. J. (2011). Dtam: Dense tracking and mapping in real-time. In *ICCV*, pages 2320–2327.

[Nister, 2003] Nister, D. (2003). Preemptive ransac for live structure and motion estimation. In *Proceedings of the 9th International Conference on Computer Vision, Nice*.

[Nister et al., 2004] Nister, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, pages 652–659.

[Nister and Stewenius, 2006] Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Proceedings of the IIEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168. IEEE Computer Society.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering.

[Nüchter et al., 2005] Nüchter, A., Lingemann, K., Hertzberg, J., and Surmann, H. (2005). Heuristic-based laser scan matching for outdoor 6d slam. In *KI 2005: Advances in Artificial Intelligence. Proceedings of 28th Annual German Conference on AI.*, Koblenz, Germany.

[Obst et al., 2011] Obst, M., Bauer, S., and Wanielik, G. (2011). Vorhersage von GNSS-Mehrwegeausbreitung in städtischen Gebieten mit digitalen Karten und 3D-Umgebungsmodellen. In *In Proc. of DGON POSNAV ITS Symposium*.

[Oliva and Torralba, 2001] Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3).

[Oliva and Torralba, 2006] Oliva, A. and Torralba, A. (2006). Building the gist of a scene: The role of global image features in recognition. *Visual Perception, Progress in Brain Research*, 155.

[Olson et al., 2000] Olson, C., Matthies, L., Schoppers, M., and Maimon, M. (2000). Robust stereo ego-motion for long distance navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-00)*, pages 453–458, Los Alamitos. IEEE.

[Olson, 2009] Olson, E. (2009). Recognizing places using spectrally clustered local matches. *Robotics and Autonomous Systems*, 57(12):1157–1172.

[Olson et al., 2006] Olson, E., Leonard, J., and Teller, S. (2006). Fast iterative optimization of pose graphs with poor initial estimates. In *Inl. Conf. on Robotics and Automation, ICRA*.

[Olson et al., 2005] Olson, E., Walter, M., Teller, S., and Leonard, J. (2005). Single-cluster spectral graph partitioning for robotics applications. In *Robotics: Science and Systems (RSS)*.

[Paul and Newman, 2010] Paul, R. and Newman, P. (2010). FAB-MAP 3D : Topological Mapping with Spatial and Visual Appearance. In *Proc. of IEEE Intl. Conf. on Robotics and Automation, ICRA*.

[Pintér, 2006] Pintér, J. D. (2006). *Global Optimization*, volume 85 of *Nonconvex Optimization and Its Applications*. Springer.

[Powell, 1970] Powell, M. J. D. (1970). A hybrid method for nonlinear equations. In Rabinowitz, P., editor, *Numerical Methods for Nonlinear Algebraic Equations*, pages 87–114. Gordon & Breach, London.

[Pradeep et al., 2010] Pradeep, V., Konolige, K., and Berger, E. (2010). Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. In *International Symposium on Experimental Robotics (ISER)*, New Delhi, India.

[Qiang et al., 2007] Qiang, Z., Xiaolin, Z., and Xiaoming, C. (2007). Research on RAIM Algorithm under the Assumption of Simultaneous Multiple Satellites Failure. In *Proc. of ACIS Intl. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*.

[Raguram et al., 2009] Raguram, R., Frahm, J.-M., and Pollefeys, M. (2009). Exploiting uncertainty in random sample consensus. In *ICCV*, pages 2074–2081.

[Ramos et al., 2007] Ramos, F., Fox, D., and Durrant-Whyte, H. (2007). CRF-matching: Conditional random fields for feature-based scan matching. In *Proc. of Robotics: Science and Systems*. Citeseer.

[Ramos et al., 2008] Ramos, F., Kadous, W., and Fox, D. (2008). Learning to Associate Image Features with CRF-Matching. In *Proc. of the 11th International Symposium on Experimental Robotics, ISER*.

[Ranganathan et al., 2007] Ranganathan, A., Kaess, M., and Dellaert, F. (2007). Fast 3d pose estimation with out-of-sequence measurements. In *In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[Rauch et al., 1965] Rauch, H. E., Striebel, C. T., and Tung, F. (1965). Maximum Likelihood Estimates of Linear Dynamic Systems. *Journal of the American Institute of Aeronautics and Astronautics*, 3(8):1445–1450.

[Rosen et al., 2012] Rosen, D., Kaess, M., and Leonard, J. (2012). An incremental trust-region method for robust online sparse least-squares estimation. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, St. Paul, MN.

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443.

[Rusinkiewicz and Levoy, 2001] Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *INTERNATIONAL CONFERENCE ON 3-D DIGITAL IMAGING AND MODELING*.

[Russell and Norvig, 2010] Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education.

[Rusu, 2009] Rusu, R. B. (2009). *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. phd, Technische Universitätet München, Munich, Germany.

[Rusu and Cousins, 2011] Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *International Conference on Robotics and Automation*, Shanghai, China.

[Scaramuzza et al., 2007] Scaramuzza, D., Harati, A., and Siegwart, R. (2007). Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 4164 –4169.

[Schindler et al., 2007] Schindler, G., Brown, M., and Szeliski, R. (2007). City-scale location recognition. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–7.

[Schroth et al., 2008] Schroth, G., Ene, A., Blanch, J., Walter, T., and Enge, P. (2008). Failure detection and exclusion via range consensus. In *Proc. of European Navigation Conference*.

[Schubert et al., 2011] Schubert, R., Adam, C., Obst, M., Mattern, N., Leonhardt, V., and Wanielik, G. (2011). Empirical evaluation of vehicular models for ego motion estimation. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 534 –539.

[Schubert et al., 2010] Schubert, R., Richter, E., Mattern, N., Lindner, P., and Wanielik, G. (2010). A concept vehicle for rapid prototyping of advanced driver assistance systems. In Meyer, G. and Valldorf, J., editors, *Advanced Microsystems for Automotive Applications 2010*, VDI-Buch, pages 211–219. Springer Berlin Heidelberg.

[Se et al., 2002] Se, S., Lowe, D. G., and Little, J. (2002). Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. In *International Journal of Robotics Research, 21, 8*, pages 735–758.

[Sedgewick and Wayne, 2011] Sedgewick, R. and Wayne, K. (2011). *Algorithms*. Addison-Wesley, ISBN 978-0321573513, 4th edition.

[Siagian and Itti, 2007] Siagian, C. and Itti, L. (2007). Rapid biologically-inspired scene classification using features shared with visual attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):300–312.

[Sibley et al., 2006] Sibley, G., Sukhatme, G., and Matthies, L. (2006). The iterated sigma point kalman filter with applications to long range stereo. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA.

[Siciliano and Khatib, 2008] Siciliano, B. and Khatib, O., editors (2008). *Springer Handbook of Robotics*. Springer, Berlin, Heidelberg.

[Sivic and Zisserman, 2003] Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*.

[Smith et al., 2006] Smith, P., Reid, I., and Davison, A. (2006). Real-time monocular slam with straight lines. In *Proc. of British Machine Vision Conference, BMVC*.

[Smith and Cheeseman, 1987] Smith, R. C. and Cheeseman, P. (1987). On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5(4):56–68.

[Solà, 2007] Solà, J. (2007). *Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilistic Approach.* PhD thesis, LAAS, Toulouse.

[Solà et al., 2005] Solà, J., Devy, M., Monin, A., and Lemaire, T. (2005). Undelayed initialization in bearing only slam. In *IEEE International Conference on Intelligent Robots and Systems*.

[Solà et al., 2011] Solà, J., Vidal-Calleja, T., Civera, J., and Montiel, J. (2011). Impact of landmark parametrization on monocular ekf-slam with points and lines. *International Journal of Computer Vision*, online first:1–30. 10.1007/s11263-011-0492-5.

[Sünderhauf et al., 2005] Sünderhauf, N., Konolige, K., Lacroix, S., and Protzel, P. (2005). Visual Odometry using Sparse Bundle Adjustment on an Autonomous Outdoor Vehicle. In Levi, Schanz, Lafrenz, and Avrutin, editors, *Tagungsband Autonome Mobile Systeme 2005*, Reihe Informatik aktuell, pages 157–163. Springer-Verlag.

[Sünderhauf et al., 2007] Sünderhauf, N., Lange, S., and Protzel, P. (2007). Using the Unscented Kalman Filter in Mono-SLAM with Inverse Depth Parametrization for Autonomous Airship Control. In *Proc. of IEEE International Workshop on Safety Security and Rescue Robotics, SSRR 2007*.

[Sünderhauf et al., 2010] Sünderhauf, N., Neubert, P., and Protzel, P. (2010). The Causal Update Filter – A Novel Biologically Inspired Filter Paradigm for Appearance Based SLAM. In *Proc. of the IEEE International Conference on Intelligent Robots and Systems (IROS)*.

[Sünderhauf and Protzel, 2006] Sünderhauf, N. and Protzel, P. (2006). Towards Using Bundle Adjustment for Robust Stereo Odometry in Outdoor Terrain. In *Proceedings of Towards Autonomous Robotic Systems (TAROS06)*.

[Sünderhauf and Protzel, 2007] Sünderhauf, N. and Protzel, P. (2007). Stereo Odometry – A Review of Approaches. Technical report, Chemnitz University of Technology.

[Sünderhauf and Protzel, 2009] Sünderhauf, N. and Protzel, P. (2009). Using Image Profiles and Integral Images for Efficient Calculation of Sparse Optical Flow Fields. In *Proceedings of the International Conference on Advanced Robotics*.

[Sünderhauf and Protzel, 2010a] Sünderhauf, N. and Protzel, P. (2010a). Beyond RatSLAM: Improvements to a Biologically Inspired SLAM System. In *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*.

[Sünderhauf and Protzel, 2010b] Sünderhauf, N. and Protzel, P. (2010b). Learning from Nature: Biologically Inspired Robot Navigation and SLAM – A Review. In *Künstliche Intelligenz (German Journal on Artificial Intelligence), Special Issue on SLAM*. Springer Verlag, Heidelberg.

[Sünderhauf and Protzel, 2011] Sünderhauf, N. and Protzel, P. (2011). BRIEF-Gist – Closing the Loop by Simple Means. In *Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[Sünderhauf and Protzel, 2012] Sünderhauf, N. and Protzel, P. (2012). Towards a Robust Back-End for Pose Graph SLAM. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

[Swerling, 1958] Swerling, P. (1958). A proposed stagewise differential correction procedure for satellite tracking and prediction. Technical report, RAND Corporation, Boston, MA, USA.

[Thrun et al., 2005] Thrun, Burgard, and Fox (2005). *Probabilistic Robotics*. The MIT Press, Cambridge, Massachusetts, London, England.

[Thrun, 2002] Thrun, S. (2002). Robotic mapping: A survey. *Science*, 298:1–35.

[Thrun et al., 2004a] Thrun, S., Koller, D., Ghahramani, Z., Durrant-Whyte, H., and Ng, A. Y. (2004a). Simultaneous mapping and localization with sparse extended information filters: Theory and initial results. *Algorithmic Foundations of Robotics V*, 23(7-8):693–716.

[Thrun and Montemerlo, 2005] Thrun, S. and Montemerlo, M. (2005). The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6):403–430.

[Thrun et al., 2004b] Thrun, S., Montemerlo, M., Koller, D., Wegbreit, B., Nieto, J., and Nebot, E. (2004b). Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data association. *Journal of Machine Learning Research*.

[Torr and Zisserman, 2000] Torr, P. and Zisserman, A. (2000). Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:138–156.

[Triggs et al., 2000] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment – A modern synthesis. In Triggs, W., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag.

[Tu et al., 2011] Tu, X., Gu, D., Yi, D., and Zhou, H. (2011). Evaluation of GNSS Receiver Autonomous Integrity Monitoring for multiple outliers with a smart random sample consensus strategy. In *Proc. of 19th International Conference on Geoinformatics*.

[van der Merwe and Wan, 2004] van der Merwe, R. and Wan, E. (2004). Sigma-point kalman filters for integrated navigation. In *Proceedings of the 60th Annual Meeting of The Institute of Navigation (ION)*, Dayton, OH.

[Vasconcelos et al., 2012] Vasconcelos, F., Barreto, J., and Nunes, U. (2012). A minimal solution for the extrinsic calibration of a camera and a laser-rangefinder. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PP(99):1.

[Wagner et al., 2011] Wagner, R., Birbach, O., and Frese, U. (2011). Rapid Development of Manifold-Based Graph Optimization Systems for Multi-Sensor Calibration and SLAM. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS11*, pages 3305–3312. IEEE.

[Wunschel et al., 2012] Wunschel, D., Lange, S., and Protzel, P. (2012). Motion Estimation for Autonomous Quadrocopter Indoor Flight. In *Proc. of the IEEE Intl. Multi-Conference on Systems, Signals and Devices, SSD12*.

[Wurm et al., 2010] Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., and Burgard, W. (2010). OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA.

[Zhang, 1999] Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 666 –673 vol.1.

List of Figures

1.1 SLAM at the intersection of localization and mapping 14
1.2 Loop Closing in SLAM 15
1.3 Division into a front-end and a back-end 16

2.1 A taxonomy of the SLAM problem 22
2.2 Occupancy grid maps 24
2.3 Point clouds and 3D grid maps 25
2.4 Feature maps 26
2.5 Different landmark observations 27
2.6 Visual features for SLAM 28
2.7 RGB-D cameras provide color and depth information 28
2.8 Triangulation of bearing-only landmarks 29
2.9 The problem of small motion parallax 29
2.10 The general concept of a pose graph 30
2.11 Different pose graphs for SLAM in 2D and 3D 31
2.12 The visual place recognition problem 33
2.13 Training phase of an appearance-based place recognition system 34
2.14 A motion model for a skid steered robot 37
2.15 Comparison of ideal and more realistic measurement models 38
2.16 The problem of data association 39
2.17 The SLAM problem represented as a dynamic Bayesian network 42
2.18 Factor graph representation of the SLAM problem with landmarks 43
2.19 Factor graph representation of the pose graph SLAM problem 43
2.20 Markov random field representation of the SLAM problem with landmarks 44
2.21 A small simulated dataset for graph-based SLAM 46
2.22 Pose graph map for SLAM before and after the optimization 48
2.23 Spring-mass model of the optimization for pose graph SLAM 49

3.1 Functions with global and local minima 54
3.2 A simple example to curve fitting with noisy input points. 57
3.3 Least squares solution to the simple curve fitting problem 58
3.4 Fitting a polynomial of 2^{nd} degree 59
3.5 Comparison of gradient descent and Newton's Method 62
3.6 Fitting parameters of a nonlinear function 66
3.7 The error function for a nonlinear fitting problem 67

3.8 Factor graph representation for a one-dimensional example 69
3.9 Jacobian and Hessian matrices 73
3.10 Data fitting in the presence of outliers 75
3.11 Two steps of the RANSAC algorithm 77
3.12 The Huber and pseudo-Huber cost functions 78

4.1 Failed place recognition due to perceptual aliasing 82
4.2 A small simulated dataset with data association errors 84

5.1 Illustration of the proposed problem augmentation 90
5.2 A first idea how to disable constraint edges during the optimization . . . 92
5.3 Three possible choices for the switch function Ψ 93
5.4 Factor graph representation of the augmented pose graph SLAM problem 94
5.5 Factor graph representation of the proposed robust back-end 97
5.6 Influence of the weight factors on the covariances 100
5.7 Jacobian and Hessian for an example SLAM problem 104
5.8 Histogram over the final switch variable values after the optimization . . . 107
5.9 A simple synthetic example for robust pose graph SLAM 108

6.1 Two versions of the Manhattan dataset used for the evaluation 115
6.2 Two more synthetic datasets used during the evaluation 116
6.3 Two real-word datasets: Intel and Parking Garage 117
6.4 Illustration of the different outlier policies 118
6.5 The influence of $\Xi_{ij} = \xi$ on the estimation quality 120
6.6 RPE measures against outliers for various datasets 123
6.7 RPE measures for the robust and conventional back-end 124
6.8 Precision-recall statistics for the various datasets 125
6.9 Examples for good estimation results despite many outliers 126
6.10 Two failure cases for the Sphere and Manhattan datasets 127
6.11 The parking garage dataset is particular difficult 128
6.12 Convergence time for different outlier policies 130
6.13 Convergence behaviour for all datasets and all four policies 131

7.1 A typical scene from the St. Lucia dataset 139
7.2 Illustration of the BRIEF-Gist descriptor 141
7.3 Examples for correctly matched scenes from the St. Lucia dataset 143
7.4 Examples for erroneously matched scenes from the St. Lucia dataset . . . 144
7.5 Initial guess and estimated trajectory for the St. Lucia dataset 146

8.1 Illustration of GNSS-based localization 148
8.2 The multipath problem in an urban canyon 150
8.3 Effects of multipath errors on the receiver's position estimate 151
8.4 Two vehicle state nodes with their associated pseudorange factors 153
8.5 A vehicle state vertex with three switched pseudorange factors 157
8.6 Complex factor graph with switch transition factors 158

8.7 Illustration of the most complex factor graph used in the evaluation . . . 158
8.8 Overview of the urban scenario used in the evaluation 159
8.9 Four different solutions for the Chemnitz City dataset 162
8.10 Comparison with the conventional and raytracing solutions 164
8.11 Histogram over the switch values s_{tj} after the optimization 165
8.12 Evolution of the switch values over time 166
8.13 Extended factor graph for cooperative localization 167

A.1 A discrete robot localization problem 179
A.2 Localization using the discrete Bayes filter 181
A.3 Simple localization example using the Kalman filter 185
A.4 SLAM with an extended Kalman filter in a one-dimensional environment 188
A.5 SLAM with an extended Kalman filter in a one-dimensional environment 190
A.6 Using a particle population to represent a probability distribution 192
A.7 Localization using a particle filter 193

List of Tables

2.1 Different probabilistic estimation techniques. 41
2.2 Available frameworks and optimization-based SLAM back-ends. 49

3.1 Steps for different iterative strategies for nonlinear least squares problems. 65
3.2 A sandbox example for 1D SLAM: Observed values. 69
3.3 A sandbox example for 1D SLAM: Robot position according to odometry measurements and ground truth. 69

6.1 The datasets used during the evaluation. 114
6.2 Initial errors for the various datasets. 114
6.3 Overall RPE_{pos} metric for different datasets 122
6.4 RPE_{pos} metric for the g^2o version of the Manhattan dataset 123
6.5 Performance comparison for the outlier-free case 133
6.6 Performance comparison for the outlier-free case. 133

7.1 Feature comparison between BRIEF-Gist and FAB-MAP 142

8.1 Collected sensor information used for the evaluation. 160
8.2 Parameters used in the evaluation. 160
8.3 Position estimation errors and convergence time for the Chemnitz City dataset. 161

A.1 Discrete state transition and measurement model. 182
A.2 Estimation results for the discrete example. 182

Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Weitere Personen waren an der Abfassung der vorliegenden Arbeit nicht beteiligt. Die Hilfe eines Promotionsberaters habe ich nicht in Anspruch genommen. Weitere Personen haben von mir keine geldwerten Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Chemnitz, den 14. Februar 2012

Niko Sünderhauf

Theses

1. Currently, state of the art SLAM systems can be divided into a front-end and a back-end. The front-end is responsible for processing the available sensor data and constructing and maintaining a graphical representation of the SLAM problem. The back-end solves the least squares optimization problem represented by that factor graph, exploiting the sparse structure inherent in the SLAM problem.

2. Like all least squares methods, current SLAM back-ends are prone to outliers. Such outliers are for instance data association errors like false positive loop closure detections. Any outlier in the problem formulation will lead to a corrupted solution of the SLAM problem. This is commonly known and accepted in the literature.

3. Current back-ends therefore rely heavily on the front-ends to identify and reject all outliers before the graphical representation is constructed or updated. The front-ends are expected to construct a topologically correct factor graph, hence operating at 100% precision. This leads to computationally involved front-ends.

4. Despite different efforts taken in the scientific literature, the front-ends cannot guarantee that all outliers can be identified and rejected. Thus SLAM systems are always endangered to fail due to an undetected data association error in the front-end.

5. This dissertation proposed a method to identify and reject outliers in the back-end.

6. This is achieved *during* the optimization process, instead of in an additional pre-processing step *before* the optimization.

7. The core idea of the proposed approach is that the topology of the factor graph representation is subject to the optimization, instead of being fixed. This way, the optimizer can alter the graph topology and remove edges that represent outlier constraints.

8. This is achieved by introducing another type of hidden variable into the problem formulation: A switch variable is associated with each factor that could potentially represent an outlier. The optimization now works on an augmented problem, searching for the joint optimal configuration of the original variables and the newly introduced switch variables, hence searching the optimal graph topology.

9. The influence of the switch variables can be described and understood in two equal ways: In the topological interpretation, a switch can enable or disable the constraint edge it is associated with, thus literally remove it from the graph topology. In the probabilistic interpretation, the switch variable influences the information matrix of the factor it is associated with and can drive it from its original value to zero, thus increasing the covariance associated with this factor until infinity. It has been shown that both interpretations are equivalent.

10. Although the newly added switch variables enlarge the problem, the problem formulation remains sparse and thus can be solved efficiently.

11. The feasibility of the proposed approach has been demonstrated and evaluated on a number of standard datasets that are commonly used as benchmarks in the community. This allows comparability and reproducibility of the proposed methods and findings by other researchers.

12. The robust back-end successfully solved SLAM problems with a large number – up to 1000 – outlier loop closure constraints, both in 2D and 3D, using synthetic and real-world datasets. The proposed approach was shown to outperform state of the art approaches by orders of magnitude, since these are not able to cope with outlier constraints.

13. It was shown that the proposed approach is not limited to SLAM problems, but can be beneficially applied to other domains where least squares problems have to be solved but outliers have to be suspected. This has been demonstrated using the problem of multipath mitigation in GNSS-based localization.

Curriculum Vitae

Personal Information

Name	Niko Sünderhauf
Date of Birth	April 10^{th} 1981
Place of Birth	Zwickau, Germany
Academic Degree	Diplom-Informatiker (Dipl.-Inf.)

Research Experience

May 2006 – present	Research associate at Chemnitz University of Technology
Sept 2004 – March 2005	Research internship at LAAS/CNRS, Toulouse, France

Refereed Publications

1 journal article, 18 conference papers, 1 workshop contribution

Education

Sept 2000 – April 2006	Computer Science with major in Artificial Intelligence Chemnitz University of Technology Degree: Diplom-Informatiker (Dipl.-Inf.)
Sept 1992 – May 1999	Gymnasium Am Sandberg, Wilkau-Haßlau, Germany Degree: Abitur