Universitat Politècnica de Catalunya

DOCTORAL PROGRAMME

AUTOMATIC CONTROL, ROBOTICS AND COMPUTER VISION

Ph.D. thesis

Information Metrics for Localization and Mapping

Joan Vallvé Navarro

Supervisors:
Juan Andrade-Cetto and Joan Solà

November 9, 2018

Information Metrics for Localization and Mapping

by Joan Vallvé Navarro

Doctoral programme:
Automatic Control, Robotics and Computer Vision

The work presented in this thesis has been carried out at:
Institut de Robòtica i Informàtica Industrial (CSIC-UPC).

Advisors:
Juan Andrade-Cetto
Joan Solà Ortega

Dissertation Committee:
Udo Frese
Javier Civera
Francesc Moreno-Noguer

The last version of this thesis and additional audiovisual material can be found at www.joanvallve.com.

Exploring the unknown requires tolerating uncertainty.

Brian Greene

Abstract

Information Metrics for Localization and Mapping

by Joan Vallvé Navarro

Decades of research have made possible the existence of several autonomous systems that successfully and efficiently navigate within a variety of environments under certain conditions. One core technology that has allowed this is simultaneous localization and mapping (SLAM), the process of building a representation of the environment while localizing the robot in it.

State-of-the-art solutions to the SLAM problem still rely, however, on heuristic decisions and options set by the user. In this thesis we search for principled solutions to various aspects of the localization and mapping problem with the help of information metrics.

One such aspect is the issue of scalability. In SLAM, the problem size grows indefinitely as the experiment goes by, increasing computational resource demands. To maintain the problem tractable, we develop methods to build an approximation to the original network of constraints of the SLAM problem by reducing its size while maintaining its sparsity. In this thesis we propose three methods to build the topology of such approximated network, and two methods to perform the approximation itself.

In addition, SLAM is a passive application. It means, it does not drive the robot. The problem of driving the robot with the aim of both accurately localizing the robot and mapping the environment is called active SLAM. In this problem two normally opposite forces drive the robot, one to new places discovering unknown regions and another to revisit previous configurations to improve localization. As opposed to heuristics, in this thesis we pose the problem as the joint minimization of both map and trajectory estimation uncertainties, and present four different active SLAM approaches based on entropy-reduction formulation.

All methods presented in this thesis have been rigorously validated in both synthetic and real datasets.

Resum

Information Metrics for Localization and Mapping

by Joan Vallvé Navarro

Dècades de recerca han fet possible l'existència de nombrosos sistemes autònoms que naveguen eficaçment i eficient per varietat d'entorns sota certes condicions. Una de les principals tecnologies que ho han fet possible és la localització i mapeig simultanis (SLAM), el procés de crear una representació de l'entorn mentre es localitza el robot en aquesta.

De tota manera, els algoritmes d'SLAM de l'estat de l'art encara basen moltes decisions en heurístiques i opcions a escollir per l'usuari final. Aquesta tesi persegueix solucions fonamentades per a varietat d'aspectes del problema de localització i mappeig amb l'ajuda de mesures d'informació.

Un d'aquests aspectes és l'escalabilitat. En SLAM, el problema creix indefinidament a mesura que l'experiment avança fent créixer la demanda de recursos computacionals. Per mantenir el problema tractable, desenvolupem mètodes per construir una aproximació de la xarxa de restriccions original del problema d'SLAM, reduint així el seu tamany a l'hora que es manté la seva naturalesa dispersa. En aquesta tesi, proposem tres métodes per confeccionar la topologia de l'approximació i dos mètodes per calcular l'aproximació pròpiament.

A més, l'SLAM és una aplicació passiva. És a dir que no dirigeix el robot. El problema de guiar el robot amb els objectius de localitzar el robot i mapejar l'entorn amb precisió es diu SLAM actiu. En aquest problema, dues forces normalment oposades guien el robot, una cap a llocs nous descobrint regions desconegudes i l'altra a revisitar prèvies configuracions per millorar la localització. En contraposició amb mètodes heurístics, en aquesta tesi plantegem el problema com una minimització de l'incertesa tant en el mapa com en l'estimació de la trajectòria feta i presentem quatre mètodes d'SLAM actiu basats en la reducció de l'entropia.

Tots els mètodes presentats en aquesta tesi han estat rigurosament validats tant en sèries de dades sintètiques com en reals.

Acknowledgements

I would like to express my gratitude to all people that made this thesis possible. First of all, I want to thank my supervisors Juan and Joan for their guidance, motivation and valuable help in the course of the development of this thesis.

I also want to thank to the people I had the pleasure to work with during these years. To Andreu, along all years working together (sometimes with considerable pressure) he always had a kind word, a wise advice and a severe and contagious positive attitude. To Àngel, Martí and Fernando for their kind and valuable assistance specially at my very beginning at IRI. I want to extend my gratitude to all IRI staff, specially to the *tupper crew* for the ridiculous number of jokes, fun chats, silly Google searches and Agustin Awards. Going to work becomes easy counting on these colleagues.

Thanks to Pau, Berta, Marta and specially my mother and my father. Growing in a home full of artistic, political and philosophical stimuli has definitely made me who I am. I also want to specifically thank them as well as Mercè and Jordi, for the outrageous amount of hours taking care of Greta while I'm writing this thesis. It just would had been impossible without that help.

To my beloved Clara for her inconditional support and motivation along all these years, I am very grateful. And finally, a very special thanks to Greta, for showing me that this thesis is insignificant compared to the important things in life. Fortunately, it was revealed when my PhD was almost finished.

This work has been partially supported by the EU Project CargoAnts (FP7-605598) and the EU H2020 Project LOGIMATIC (H2020-Galileo-2015-1-687534).

Contents

List of Figures

List of Tables

Abbreviations

AMD	Approximate Minimum Degree
A-space	Action space
BA	Bundle Adjustment
CG	Conjugated Gradient
CLT	Chow-Liu Tree
COLAMD	COLumn Approximate Minimum Degree
C-space	Configuration space
dMI	downdated Mutual Information
EDE	Entropy Decrease Estimation
EIF	Extended Information Filter
EKF	Extended Kalman Filter
eKLD	expected Kullback-Liebler Divergence
EMD	Exact Minimum Degree
FD	Factor Descent
FFD	First Factor Descent cycle
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
ICP	Iterative Closest Point
IMU	Intertial Measurement Unit
IP	Interior Point
iSAM	incremental Smoothing And Mapping
KLD	Kullback-Liebler Divergence
LM	Levenberg-Marquardt
MAP	Maximum A Posteriori
MI	Mutual Information
MLE	Maximum Likelihood Estimation
ncFD	non-cyclic Factor Descent
NLS	Non-linear Least Squares
ODB	Off-Diagonal Block
ODD	Off-Diagonal block Determinant
PCG	Preconditioned Conjugated Gradient
PQN	Projected Quasi-Newton
RBPF	Rao-Blackwellized Particle Filter
RMSE	Root Mean Squared Error
RRT	Rapidly-exploring Random Tree
SAM	Smoothing And Mapping

$\sqrt{\textbf{SAM}}$	Square-root Smoothing And Mapping
SG	Sub-Graph
SPA	Sparse Pose Adjustment
SPCG	Sub-graph Preconditioned Conjugated Gradient
SLAM	Simultaneous Localization And Mapping
SRT	Sensor-based Random Tree

Nomenclature

Operators

$rk(\cdot)$	Matrix rank.
$tr(\cdot)$	Matrix trace.
$\lvert \cdot \rvert$	Matrix determinant.
$(\cdot)^{1/2}$	Any matrix factorization such that $\mathbf{A} = (\mathbf{A}^{1/2})^\top \mathbf{A}^{1/2}$.
$\lVert \cdot \rVert^2$	Squared norm of a vector $\lVert \mathbf{a} \rVert^2 = \mathbf{a}^\top \mathbf{a}$.
$\lVert \cdot \rVert^2_{\boldsymbol{\Sigma}}$	Squared Mahalanobis norm of a vector $\lVert \mathbf{a} \rVert^2_{\boldsymbol{\Sigma}} = \mathbf{a}^\top \boldsymbol{\Sigma}^{-1} \mathbf{a} = \lVert \boldsymbol{\Sigma}^{-1/2} \mathbf{a} \rVert^2$.
$\mathbf{A} \succ 0$	Positive definite matrix.

Notation

General convention:

a	Scalar.
$\mathbf{a}$	Vector.
$\mathbf{A}$	Matrix.
$a(\cdot)$	Function.

Specific notation:

$P(\cdot)$	Probability function.
$p(\cdot), q(\cdot)$	Probability density functions.
$\boldsymbol{\mu}$	Mean of multi-variate Gaussian distribution.
$\boldsymbol{\Sigma}$	Covariance matrix of multi-variate Gaussian distribution.
$\boldsymbol{\eta}$	Information vector of multi-variate Gaussian distribution.
$\boldsymbol{\Lambda}$	Information matrix of multi-variate Gaussian distribution.
$\mathbf{x}$	State.
$\mathbf{z}_k$	Measurement.
$h_k(\mathbf{x})$	Measurement model.
$g_k(\mathbf{x}, \mathbf{z})$	Inverse measurement model.
$\mathbf{e}_k$	Measurement error.
$\mathbf{J}_k$	Measurement Jacobian.
$\boldsymbol{\Omega}_k$	Measurement noise information matrix.
$\mathbf{r}_k$	Residual.
$\mathbf{J}$	All measurements Jacobian.
$\boldsymbol{\Omega}$	Block diagonal matrix containing all measurement information matrices.
$\mathbf{A}$	Residuals Jacobian.

To Clara and Greta.

1 Introduction

The capability of navigating is, in itself, an expression of intelligence. Being capable of creating a representation of the environment and moving within it requires perception, motion planning and control skills.

In our aim to create intelligent artificial devices, science and technology made progress from different disciplines taking place in robotics. Specifically, research on mobile robotics focuses on the development of mechanisms with the capability of autonomous navigation, among others.

Robots have been widely and successfully deployed in industry during the last decades. However, oftentimes their physical capabilities were exploited (load, productivity, accuracy) without developing cognitive capabilities to enable higher degree of autonomy. In other words, adopting the environment to the robot capabilities was preferred rather than over equipping robots with the ability to adapt themselves to the environment. Then, industrial manipulation robots have been normally deployed inside a cage where people are not allowed to stay. And analogously, autonomous ground vehicles systems normally count on several supporting infrastructures such as wire or tape guided routes, laser reflectors, etc..

The research advances in the last decades are making arise a new paradigm in which robots (or autonomous devices) and humans cohabit in the same space. Autonomous driving and industry 4.0 are two exponents of this new paradigm in which autonomous navigation has a key role. Behind autonomous navigation, different problems take place such as localization and mapping. One needs a representation of the environment and knowing its location in order to navigate within it.

Nowadays, we count on several localization and mapping systems. Some of them, however, are human-based tools to complement our capabilities such as GNSS (GPS, Galileo, GLONAS...) or Google maps.

Regarding mobile robotics, there are several applications in which offline maps are not suitable. Trivially, navigation within unmapped environments such as search and rescue or any unmapped indoors environment.

Furthermore, the dynamic nature of human environments depends on the scale that is considered. For instance, in high-level navigation such as route planning, cities can be assumed to be static allowing the use of offline maps and GNSS. However, navigating in such environments requires finer spatial resolution and human environments become more and more dynamic as the scale decreases: Buildings, shops, trees, ads, construction works, light conditions, cars, furniture, people.

Hence, even if the whole world could be mapped at one instance, autonomous navigation could not fully rely on such a map at certain scales. Therefore, mapping is necessary to be performed simultaneously to localization in several cases. Moreover, there are other applications that also require online localization and mapping algorithms beyond navigation such as augmented reality or online 3D object modeling.

In mobile robotics, simultaneous localization and mapping (SLAM) has been a largely explored research topic. More than 30 years after the firsts works were published, the research community have come a long way. Nowadays, several online robotic systems are already capable of autonomously navigating in specific environments under certain conditions. However, several open issues are still requesting further research.

Additionally, despite the researchers interest, autonomous navigation is not the ultimate task we want the robots to do. Rather, it shall be a background non-costly process that will provide the robot with basic capabilities to do more sophisticated tasks such as fetching goods, guiding humans, etc. Robotics is just starting to show its potential by demonstrating the feasibility of basic capabilities such as autonomous navigation. In the following years, robustness and efficiency of these basic capabilities will give raise to more complex applications.

1.1 Motivation

Three of the most researched topics in mobile robotics are localization, mapping and path planning. Mapping can be defined as the problem of generating a representation of the environment from the sensory output of a robot that is following a known trajectory [Moravec, 1988; Elfes, 1989]. Localization [Dellaert et al., 1999; Thrun et al., 2000] is the problem of estimating the pose of such robot within a given a map whilst moving. And, path planning [Kavraki and Latombe, 1994; LaValle, 1998] is the problem of computing a feasible trajectory from an initial configuration to a final one, given a map and full certainty about the robot location within it.

The intersection of these three problems generates four new problems (Figure 1.1). Combining localization and mapping, simultaneous localization and mapping (SLAM) arises as the problem of generating a map and localizing the robot in it while it is being driven [Smith and Cheeseman, 1986; Durrant-Whyte, 1988; Thrun et al., 2004; Kaess et al., 2012; Ila et al., 2017]. Alternatively, the problem of planning a trajectory that will localize better the robot given a map, i.e. the combination of localization and path planning, is called active localization [Fox et al., 1998; Corominas-Murtra et al., 2008]. Exploration [Yamauchi, 1997; Shade

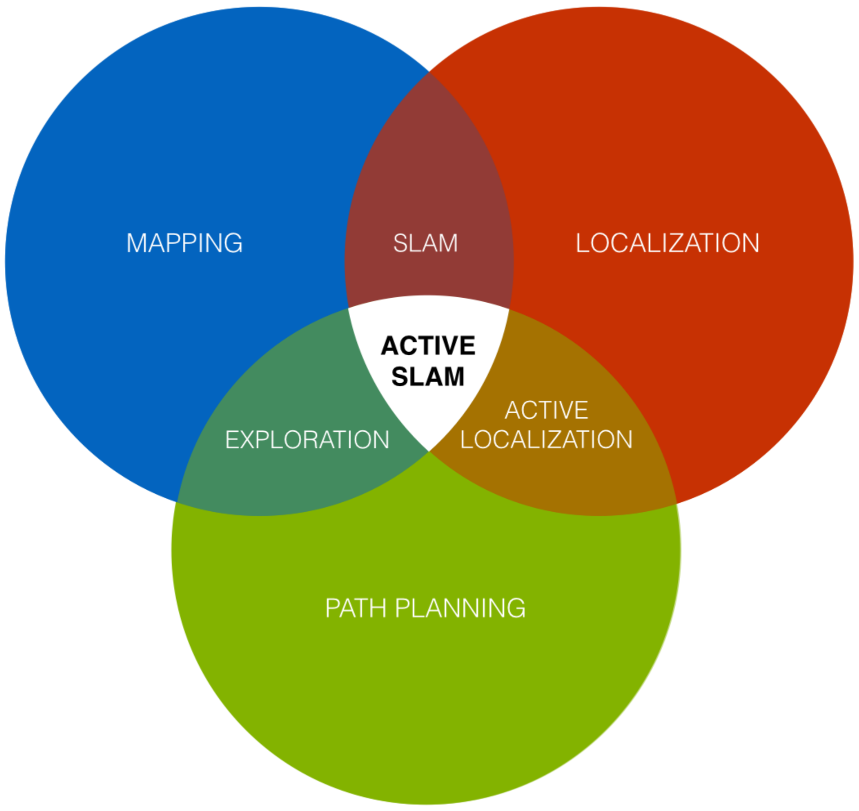


Figure 1.1: Combining the three main problems in mobile robotics, four new problems arise.

and Newman, 2011], the combination of mapping and path planning, is the problem of driving a (localized) robot in order to build a map of an environment.

Finally, we call the combination of all three initial problems as Active Simultaneous Localization and Mapping (Active SLAM) [Stachniss et al., 2005; Valencia et al., 2012], and we define it as the problem of finding the best path to drive a robot to build a map of a previously unknown environment and simultaneously localizing itself in it.

As introduced before, in such a dynamic world, offline maps are just suitable for applications that do not require accurate localization or in special cases in which the static environment assumption is rigorously fulfilled. Then, there are plenty of cases in which SLAM is required.

Moreover, in some cases such as search and rescue or indoors navigation, methods to autonomously drive the robot to map the environment (i.e. exploration) are useful applications. In these cases active SLAM methods are able to produce more accurate maps than exploration, since by not decoupling the localization from the problem the robot is better localized while building the map. This thesis is focused on both applications: SLAM and active SLAM.

In SLAM, the problem grows as time increases, receiving more sensor data, increasing the trajectory of the robot to be estimated, enlarging the map, etc. Hence, addressing the scalability of SLAM poses big challenges.

Significant advances have been made. We can find now state of the art algorithms that accurately localize a robot in real time for a restricted number of domains. However, new application demands require these algorithms to be just a small non-costly module of a more complex task or to be running in systems with limited computational power (cell phones, unmanned aerial vehicles, embedded computing systems, etc.). So finding scalable solutions to the problem is always a valid concern for researchers.

Active SLAM pursues two different and normally opposite aims: exploring the environment which drives the robot to unknown areas, and improving robot localization which means going to already visited places. The latter has been systematically decoupled from the problem of exploration assuming independence between the localization performance and the robot trajectory. This assumption remains far from reality in most cases.

Several active SLAM methods in the literature propose the alternation between exploratory and relocalization goals decoupling path planning. Thus, the potential exploration and/or relocalization that can be performed along the planned path is not considered. Furthermore, both the alternation and the selection of goals is usually based on heuristics.

Formalization of some SLAM and active SLAM mechanisms should provide a principled way to tackle these problems. In the present thesis we recall on information theory to formalize some of these problems deriving in new methods as opposed to heuristic mechanisms.

1.2 Objectives and scope

The main objective of this thesis is to propose and formalize the use of information gain metrics to tackle the scalability concern for the SLAM problem and devise new active SLAM methods.

As introduced before, in SLAM the reduction of the problem size is oftentimes necessary. However, it normally results in a reduced approximated problem posing a compromise between accuracy and computational resources. Information metrics can be used to measure the amount of information loss in the approximated solution.

In the case of Active SLAM, the decision of which paths to take have a direct influence on both the total travelled distance by the robot, hence time, and the quality of the resulting map. The information encoded in the trajectory estimate and the ensuing map would allow us to evaluate different path candidates and even give some clues on how to create the path candidate set.

Summarizing, our objectives are the use of information metrics for the various parts of the SLAM and Active SLAM problems:

- Develop and formalize information-based formulations to be used in existing state-of-art SLAM methods to ease the time and space computational burden of the algorithms.
- Develop information-based Active SLAM methods, formalizing how to measure the amount of information encoded in a path in order to choose the one that most efficiently and correctly maps the environment.

1.3 Thesis overview and summary of contributions

The present thesis is organized in six chapters. After this chapter, the SLAM problem is described introducing the formulation of the state-of-art non-linear least squares approach and deriving its variants (Chapter 2). Additionally, an overview of other SLAM approaches and formulations in the literature is provided establishing connections between them. The drawbacks and benefits of all presented SLAM methods is also discussed.

Chapter 3 includes a brief introduction to information theory. Some relevant information metrics for localization and mapping applications are presented and formulated in the multi-variate Gaussian distribution case.

Later, the use of information metrics in graph SLAM sparsification is explored in Chapter 4. Sparsification allows reduction of the problem size while maintaining its sparsity and relinearization capability to keep the solution accuracy and efficiency.

Finally, the active SLAM problem is approached making use of entropy to measure the uncertainty reduction both in map and robot trajectory estimation (Chapter 5). Active SLAM approaches produce significantly better maps than exploration methods since the latter decouples localization which it is not independent from the robot trajectory.

The following is a brief overview of this thesis contributions and the related scientific publications of the thesis author.

- **Chapter 1:** The scope and motivation of the present thesis have been presented introducing the need of formalization on some SLAM and active SLAM decisions and methods. The use of information metrics to do this has been proposed.

- **Chapter 2:** A general formulation for the SLAM problem is developed from the non-linear least squares approach. Furthermore, links and connections with other SLAM approaches in the literature are explored providing a general formalization of the SLAM problem. The general problem formulation and intuitive understanding is the basis on which the next chapters are developed.

- **Chapter 3:** From the localization and mapping perspective, an overview of information theory is presented. The main metrics are introduced, providing both their intuitive meaning and the specific formulation for the multi-variate Gaussian case used in SLAM. The potential applications to mobile robotics applications of each metric are initially explored.

- **Chapter 4:** Sparsification aim is to approximate a dense and not relinearizable sub-graph resulting of a node (or nodes) marginalization with a new sparse and relinearizable sub-graph. In this problem, all processes involved have been examined in order to propose new alternatives making use of information metrics. The topology of the new sub-graph is critical to reach a good approximation. How its population is set and how it is built is one line of work presented in this chapter. Given the topology, new methods to compute the best new factors' parameters are also proposed. All new approaches are rigorously validated and compared with the state-of-art sparsificaiton methods. This chapter includes the work of three publications [Vallvé et al., 2017, 2018a,b].

- **Chapter 5:** The specific active SLAM approach using a pose-graph SLAM and occupancy maps is exhaustively explored in this chapter. The use of entropy to measure the uncertainty in both trajectory and map estimations is developed to devise new methods. Two main lines of research are included. First, the problem is posed as a search in the configuration space looking for the most informative robot location to drive the robot. Secondly, as a search in the action space looking for the most informative path candidate efficiently building and evaluating large action sets. Four publications are related with the work presented in this chapter [Vallvé and Andrade-Cetto, 2013, 2014, 2015a,b].

- **Chapter 6:** A summary of the work presented in the thesis and the closing remarks are stated. Some lines of research to be considered in future work are also proposed.

In the course of realization of the work presented in this thesis, the author collaborated in the development of other scientific publications [Valencia et al., 2014; Corominas-Murtra et al., 2016]. Despite they are slightly related with the work presented in this thesis, the inclusion of these works would have diverged its scope. Therefore, this work has not been incorporated to the present thesis. Section 6.2 encloses a list containing all scientific publications performed during the realization of this thesis.

2 Simultaneous Localization and Mapping

2.1 Introduction

In mobile robotics, simultaneous localization and mapping (SLAM) is the problem of building a representation of the environment (i.e. a map) while localizing the robot in this map at the same time. The fundamentals of this problem were introduced in the seminal work of Smith, Self and Cheeseman [Smith and Cheeseman, 1986; Smith et al., 1988, 1990] as well as the work of Durrant-Whyte and Leonard [Durrant-Whyte, 1988; Leonard et al., 1992; Durrant-Whyte et al., 1996].

Initially, the problem was posed as concurrently estimating the localization of a robot and a set of environment landmarks. However, several different environment representations have been proposed in the literature. From landmark-based to denser representations such as occupancy grids [Moravec, 1988; Elfes, 1989] or 3D RGB semi-dense [Engel et al., 2013] or dense [Newcombe et al., 2011] reconstructions.

Furthermore, the so-called pose-graph SLAM [Lu and Milios, 1997; Olson et al., 2006; Ila et al., 2010; Konolige, 2010] only estimates the robot trajectory establishing geometrical restrictions due to sensor data alignment, for instance. It could be argued that pose-graph SLAM is not strictly SLAM since it does not estimate any map representation. However, most of the pose SLAM methods are capable of easily rendering a map from the sensory data and the trajectory estimate.

Graphs have been widely used in the SLAM literature for problem representation and derivation of new methods. Some examples of the latter are graphical SLAM [Folkesson and Christensen, 2004; Folkesson et al., 2005] which elaborates a SLAM algorithm from a graphical representation. In [Paskin, 2003], Junction Trees are exploited to derive a new SLAM method. A similar graph is presented in iSAM2 [Kaess et al., 2012], the so-called

Bayes tree encodes the incremental QR-factorization algorithm for a non-linear least squares. Graphs are also used to guide data association search such as the co-visibility graph in ORB-SLAM [Mur-Artal et al., 2015].

Different ways of representing the SLAM problem in a graph are devised in the literature. The Bayes net or belief net [Montemerlo and Thrun, 2007; Dellaert and Kaess, 2006] is a directed uncyclic graph that represents the conditional independence of variables (each variable only depends on its predecessors). Alternatively, the factor graph [Kschischang et al., 2001] represents which subsets of variables are indirectly observed by the measurements. The Markov Random Field also encodes this information but without an explicit representation of the measurements: it only depicts which nodes are involved in each observation.

In fact, linear algebra is strongly linked to graph theory being most of the times analogous representations of the same problem. Actually, most of the main processes in linear algebra have their equivalent process in graph theory. For instance, variable elimination resulting in a Bayes net is equivalent to QR or Cholesky factorization of a linear problem. Also, most of the variable ordering methods to improve the factorization efficiency are derived from graph theory.

Nowadays, factor graph is a popular SLAM representation. It is formally related with the probabilistic formulation of the problem. A factor graph (see Figure 2.1) is a bipartite graph in which the SLAM problem is represented using two types of nodes: variable nodes and factors nodes. Variable nodes (from now on just *nodes*) represent estimated variables such as robot or landmark configurations. Factor nodes (from now on just *factors*) represent geometrical constraints according to sensor measurements.

Considering that, we realize that SLAM is a very common estimation problem that has been faced from very different disciplines. Probably, geodesy was the first discipline to formulate a similar problem: computing the coordinates of several points on the earth surface from a large set of (noisy) measurements. Actually, Cholesky factorization [Benoït, 1924] was proposed with the purpose to solve it. Nowadays, Cholesky is a largely used factorization for solving SLAM problems. Also, in [Golub and Plemmons, 1980], a very similar approach to the state-of-art SLAM methods was introduced.

The Bundle Adjustment (BA) problem, in computer vision, is also equivalent to the SLAM problem. Given a set of images, a 3D representation of the environment and the camera trajectory are estimated. Its initial application was aerial cartography [Brown, 1958]. As in geodesics, it early derived to non-linear least squares formulations [Mikhail and Ackermann, 1976].

A particularity of SLAM is that it has been normally faced as an online mobile robot application. An early least squares approach for a 2D SLAM using laser scans was presented in [Lu and Milios, 1997]. However, it was applied to very small problems. Probably due to the online specification, SLAM methods did not consider least squares until the last decade [Dellaert and Kaess, 2006; Konolige, 2010] due to its computational burden. In Section 2.2 and the following, the extensively used Gauss-Newton least squares formulation is derived. Afterwards, Section 2.5 provides an overview of past and present alternative SLAM methods.

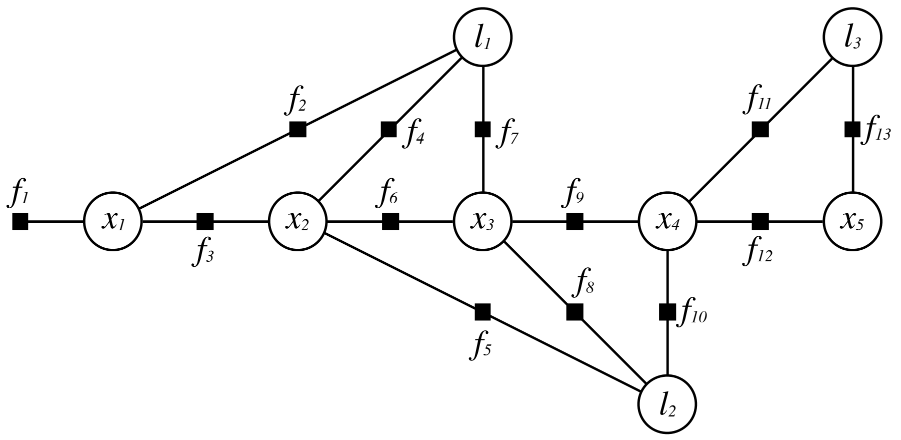


Figure 2.1: Factor graph example. White nodes represent the state variables including the robot trajectory (x_1,x_2,x_3,x_4,x_5) and the set of landmarks (l_1, l_2, l_3). Black nodes (factors) represent observations, i.e. constraints, including odometry constraints (f_3, f_6, f_9, f_{12}), landmark observations (f_2, f_4, f_5, f_7, f_8, f_{10}, f_{11}, f_{13}) and a prior for the first robot pose (f_1).

2.2 Smoothing and mapping

Classically, the SLAM problem was formulated as a filtering problem. In filtering, only the present state is estimated. Old observations are concurrently marginalized in a prior on the current state, and its estimation is updated according to recent observations.

Nowadays, SLAM is formulated as a smoothing problem. In smoothing, all the observations of the state are taken into account to estimate the solution. Therefore, the smoothing approach for SLAM includes in the state $\mathbf{x}$ all variables that have been observed. This is widely known as smoothing and mapping (SAM). In Section 2.5, the benefits of smoothing over filtering are defended.

Normally, the SAM state includes poses of the vehicle along its trajectory exclusively (i.e. pose-graph SLAM) or together with some environment landmarks (i.e. full SLAM). Thus, pose-graph SLAM observations are created between robot poses after finding relationships between corresponding sensor measurements. Full SLAM also includes observations that relate vehicle poses and some relevant environment landmarks that have been observed. The state can also include some other variables that are wanted to be estimated such as sensor intrinsic parameters (IMU biases, camera intrinsic parameters, etc.) or extrinsic ones (sensor position and orientation).

2.2.1 Probabilistic approach

Dealing with uncertainty is crucial for the proper modeling of the SLAM problems, and hence for finding well-balanced (*i.e.*, optimal in some sense) solutions. This can be done by approaching the problem from a probabilistic viewpoint.

From the point of view of Bayesian inference, we estimate the conditional probability of the state given all the observations

$$p(\mathbf{x}|Z) = p(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m) = \frac{p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m)}.$$

where $\mathbf{x}$ is the state to estimate and $Z = (\mathbf{z}_1, \cdots, \mathbf{z}_n)$ is the set of all measurements.

As shown above, the Bayes rule allows us to express this conditional $p(\mathbf{x}|Z)$ from its opposite $p(Z|\mathbf{x})$, which can be easily formulated from the measurement models, as described further down.

2.2.2 Multi-variate Gaussian distribution

From the very first approaches, SLAM methods uses multi-variate Gaussian distribution to describe the conditional probability density $p(\mathbf{x}|Z)$. It can be represented in the canonical form $p(\mathbf{x}|Z) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ or alternatively, in the information form $p(\mathbf{x}|Z) \sim \mathcal{N}^{-1}(\boldsymbol{\eta}, \boldsymbol{\Lambda})$ [Thrun et al., 2004]. These are related by,

$$\begin{aligned} P(\mathbf{x}|Z) &\propto \exp\big(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\big) \\ &= \exp\big(-\tfrac{1}{2}\mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \tfrac{1}{2}\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\big), \end{aligned}$$

and since $\boldsymbol{\mu}^\top \boldsymbol{\Sigma} \boldsymbol{\mu}$ is constant,

$$\begin{aligned} P(\mathbf{x}|Z) &\propto \exp\big(-\tfrac{1}{2}\mathbf{x}^\top \underbrace{\boldsymbol{\Sigma}^{-1}}_{\boldsymbol{\Lambda}}\mathbf{x} + \mathbf{x}^\top \underbrace{\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}_{\boldsymbol{\eta}}\big) \\ &= \exp\big(-\tfrac{1}{2}\mathbf{x}^\top \boldsymbol{\Lambda}\mathbf{x} + \mathbf{x}^\top \boldsymbol{\eta}\big). \end{aligned} \tag{2.1}$$

The information form defines a multi-variate Gaussian distribution using the information vector $\boldsymbol{\eta}$ [(i)] and the information matrix $\boldsymbol{\Lambda}$ that can be expressed in terms of the canonical form

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} \quad \text{and} \quad \boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}. \tag{2.2}$$

2.2.3 Maximum A Posteriori or Maximum Likelihood Estimation

In the context of Bayesian inference, the Maximum a Posteriori (MAP) estimates the mode of the posterior conditional probability $p(\mathbf{x}|Z)$. In case of a multi-variate Gaussian distribution, it is its mean, $\boldsymbol{\mu} = \mathbf{x}_{\mathrm{MAP}}$. Then,

$$\begin{aligned} \boldsymbol{\mu} = \mathbf{x}_{\mathrm{MAP}} &= \arg\max_{\mathbf{x}} p(\mathbf{x}|Z) \\ &= \arg\max_{\mathbf{x}} \frac{p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m)}. \end{aligned}$$

[(i)] Some papers in the literature define the information vector as a row vector. However, the induced differences in the following derivations are trivial.

The Maximum Likelihood Estimation (MLE) is a special case of the Maximum a Posteriori (MAP) estimation in which there is no prior distribution on the state (i.e. the prior is a uniform distribution). As already stated, this is the case of smoothing. Hence, $p(\mathbf{x})$ above is a uniform distribution, thus constant w.r.t. the state. So is the denominator, which does not depend on $\mathbf{x}$. Furthermore, considering independent measurements, the conditional probability of all measurements given the state, $p(\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n|\mathbf{x})$, is the product of conditionals of each measurement given the state. Considering these facts, the problem can be re-written as,

$$\boldsymbol{\mu} = \arg\max_{\mathbf{x}} \prod p(\mathbf{z}_k|\mathbf{x}).$$

Thus, the mean of the conditional distribution $p(\mathbf{x}|Z)$ (or the MAP or the MLE, equivalently) is the maximization of the product of the conditional probability densities corresponding to all observations. Hence the name *factor* to refer to the effect of each observation, since each observation $\mathbf{z}_k$ leads to a factor of this product of conditionals, $p(\mathbf{z}_k|\mathbf{x})$.

Each factor represents a geometrical constraint between some state variables. Deriving from sensory data, control laws or a dynamic model, each factor is posed as a measurement $\mathbf{z}_k$ of the state through a measurement model $h_k(\mathbf{x})$ and affected by a Gaussian noise $\mathbf{v}_k \sim \mathcal{N}(0, \boldsymbol{\Omega}_k^{-1})$, that is,

$$\mathbf{z}_k = h_k(\mathbf{x}) + \mathbf{v}_k. \tag{2.3}$$

Being the noise Gaussian, the conditional probability factor for this observation can be written as

$$p(\mathbf{z}_k|\mathbf{x}) = \mathcal{N}(h_k(\mathbf{x}), \boldsymbol{\Omega}_k^{-1}) = \exp\Big(-\frac{1}{2}\big(\mathbf{z}_k - h_k(\mathbf{x})\big)^\top \boldsymbol{\Omega}_k \big(\mathbf{z}_k - h_k(\mathbf{x})\big)\Big).$$

Then, substituting and considering that the natural logarithm is a monotonically increasing function,

$$\begin{aligned}\boldsymbol{\mu} &= \arg\max_{\mathbf{x}} \log \Big(\prod p(\mathbf{z}_k \mid \mathbf{x})\Big) \\ &= \arg\max_{\mathbf{x}} \sum \log\exp\Big(-\frac{1}{2}\big(\mathbf{z}_k - h_k(\mathbf{x})\big)^\top \boldsymbol{\Omega}_k \big(\mathbf{z}_k - h_k(\mathbf{x})\big)\Big).\end{aligned}$$

Defining the measurement error as $\mathbf{e}_k = h_k(\mathbf{x}) - \mathbf{z}_k$, and the factor residual as $\mathbf{r}_k = \boldsymbol{\Omega}_k^{1/2}\mathbf{e}_k$[ii], we can transform the problem into the more tractable form,

$$\begin{aligned}&= \arg\min_{\mathbf{x}} \sum \|\mathbf{e}_k\|^2_{\boldsymbol{\Omega}_k^{-1}} \\ &= \arg\min_{\mathbf{x}} \sum \|\mathbf{r}_k\|^2.\end{aligned}$$

That is, the mean of the conditional distribution of the state given the observations is the one that minimizes the summation of the squared Mahalanobis distance of all measurements

[ii]We define $\boldsymbol{\Omega}_k^{1/2}$ as whichever factorization such that $\boldsymbol{\Omega}_k = (\boldsymbol{\Omega}_k^{1/2})^\top \boldsymbol{\Omega}_k^{1/2}$.

errors $\mathbf{e}_k$. Or alternatively, the one that minimizes the sum of squared residuals $\mathbf{r}_k$. These constitute equivalent but alternative forms of the problem: the least squares forms.

2.2.4 Smoothing by Gauss-Newton non-linear least squares

Since measurement models $h_k(\mathbf{x})$ are rarely linear, graph-SLAM problems are normally posed as a Nonlinear Least Squares (NLS) optimization and solved via Gauss-Newton,

$$\Delta\boldsymbol{\mu}^* = \arg\min_{\Delta\boldsymbol{\mu}} \sum_k \|\mathbf{r}_k(\boldsymbol{\mu} + \Delta\boldsymbol{\mu})\|^2.$$

The residuals are linearized taking its first order Taylor expansion

$$\begin{aligned} \mathbf{r}_k(\boldsymbol{\mu} + \Delta\boldsymbol{\mu}) &= \mathbf{r}_k(\boldsymbol{\mu}) + \left.\frac{\partial \mathbf{r}_k(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}} \Delta\boldsymbol{\mu} \\ &= \mathbf{r}_k(\boldsymbol{\mu}) + \boldsymbol{\Omega}_k^{1/2}\Big(\underbrace{\left.\frac{\partial h_k(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}}}_{\mathbf{J}_k} - \cancelto{0}{\left.\frac{\partial \mathbf{z}_k}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}}} \Big)\Delta\boldsymbol{\mu}, \end{aligned} \tag{2.4}$$

where $\mathbf{J}_k$ is the Jacobian of the k-th measurement model evaluated at the current state estimation $\mathbf{x} = \boldsymbol{\mu}$ [(iii)]. Then the minimization (2.4) becomes

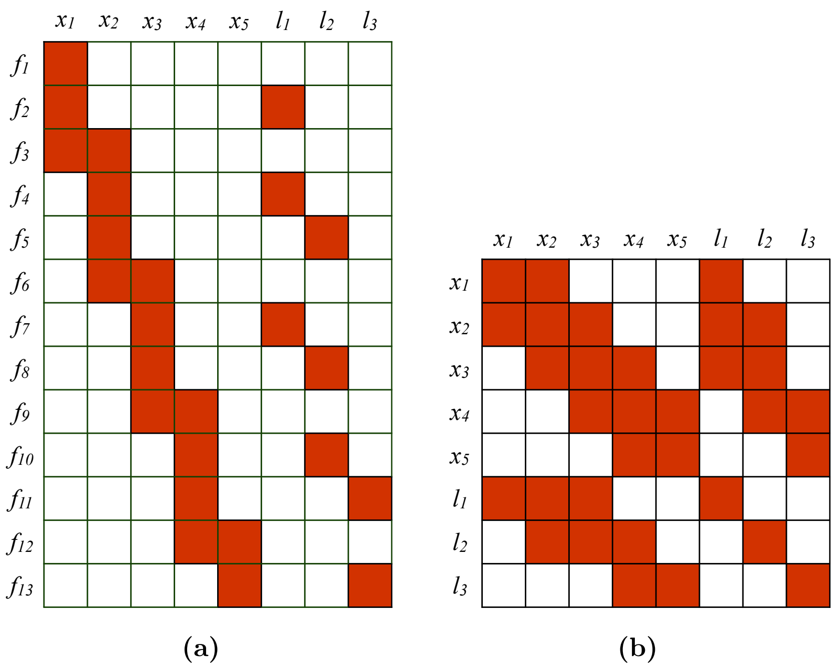
$$\begin{aligned} \Delta\boldsymbol{\mu}^* &= \arg\min_{\Delta\boldsymbol{\mu}} \sum_k \|\mathbf{r}_k(\boldsymbol{\mu}) + \boldsymbol{\Omega}_k^{1/2}\mathbf{J}_k\Delta\boldsymbol{\mu})\|^2 \\ &= \arg\min_{\Delta\boldsymbol{\mu}} \left\| \underbrace{\begin{bmatrix} \vdots \\ \mathbf{r}_k(\boldsymbol{\mu}) \\ \vdots \end{bmatrix}}_{\mathbf{r}} + \underbrace{\begin{bmatrix} \vdots \\ \boldsymbol{\Omega}_k^{1/2}\mathbf{J}_k \\ \vdots \end{bmatrix}}_{\mathbf{A}} \Delta\boldsymbol{\mu} \right\|^2 \end{aligned}$$

After each iteration, the state estimate is updated with $\Delta\boldsymbol{\mu}^*$ and the problem is relinearized and built to be solved again. This requires, re-evaluating the residuals vector $\mathbf{r}$ and measurements Jacobians $\mathbf{J}_k$ at the new linearization point to build the residuals Jacobian $\mathbf{A}$. The process is repeated until convergence.

Additionally, the squared residuals Jacobian is the information matrix of the linearized conditional probability density of the state given the observations $\mathbf{A}^\top\mathbf{A} = \boldsymbol{\Lambda}$, $p(\mathbf{x}|Z) \sim \mathcal{N}^{-1}(\boldsymbol{\eta}, \boldsymbol{\Lambda})$ (see Prop. 1 in Appendix A). In other words, the problem encodes the parameters of the multi-variate Gaussian distribution of the state, updating the information matrix in its square-root form and directly estimating the mean

$$\Delta\boldsymbol{\mu}^* = \arg\min_{\Delta\boldsymbol{\mu}} \|\mathbf{A}\Delta\boldsymbol{\mu} + \mathbf{r}\|^2. \tag{2.5}$$

[(iii)] In case of manifolds, the error becomes $\mathbf{e}_k(\mathbf{x}) = h_k(\mathbf{x}) \ominus \mathbf{z}_k \oplus \mathbf{v}_k$ and $\mathbf{J}_k = \partial(h_k(\mathbf{x}) \ominus \mathbf{z}_k)/\partial\mathbf{x}$. The $\oplus$ and $\ominus$ are the addition and subtraction operators on the manifold, as described in [Smith et al., 1990].



x_1 x_2 x_3 x_4 x_5 l_1 l_2 l_3
f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 f_9 f_{10} f_{11} f_{12} f_{13}
(a)
x_1 x_2 x_3 x_4 x_5 l_1 l_2 l_3
(b)

Figure 2.2: Sparisty pattern of the residuals Jacobian $\mathbf{A}$ (a) and the information matrix $\mathbf{\Lambda}$ (b) corresponding to the factor graph example (Figure 2.1).

2.2.5 Sparsity

Since the factors usually involve very few nodes of the state, i.e., the measurements observe very few state variables (two robot poses, a robot pose and a landmark, etc.), the Jacobians $\mathbf{J}_k$ are usually very sparse, as well as the residuals Jacobian $\mathbf{A}$ and the information matrix $\mathbf{\Lambda}$. In Figure 2.2, we depict the sparsity pattern of the residuals Jacobian $\mathbf{A}$ and the information matrix $\mathbf{\Lambda}$ corresponding to the factor graph example (Fig 2.1). Only the non-zero matrix blocks are color filled.

The state-of-art methods for solving (2.5) take profit of this sparsity to speed up the process, as explained in the following sections.

2.3 Batch methods

The first SAM methods for solving SLAM [Dellaert and Kaess, 2006; Kümmerle et al., 2011] problem where designed to solve the whole problem (2.5) from scratch. Note that the initial guess is critical for convergence since the problem is highly non-linear.

In an offline scenario, the problem is built and solved iteratively until convergence. Alternatively, in online SLAM applications, the problem is built and solved periodically following a specific heuristic (after each measurement, time-based, after adding a new node, etc.). Since the prior used is the previous problem solution, normally only a single iteration is executed.

There are mainly two different methods to batch solve the NLS problem (2.5). They are presented in the following subsections.

2.3.1 Cholesky factorization

The Gauss-Newton step solves (2.5) imposing null derivative of the squared norm term w.r.t. the state estimate update $\Delta\boldsymbol{\mu}$.

$$\begin{aligned}\frac{\partial\|\mathbf{A}\Delta\boldsymbol{\mu}+\mathbf{r}\|^2}{\partial\Delta\boldsymbol{\mu}} &= \frac{\partial(\Delta\boldsymbol{\mu}^\top\mathbf{A}^\top\mathbf{A}\Delta\boldsymbol{\mu}+2\Delta\boldsymbol{\mu}^\top\mathbf{A}^\top\mathbf{r}+\mathbf{r}^\top\mathbf{r})}{\partial\Delta\boldsymbol{\mu}} \\ &= 2\underbrace{\mathbf{A}^\top\mathbf{A}}_{\boldsymbol{\Lambda}}\Delta\boldsymbol{\mu}+2\underbrace{\mathbf{A}^\top\mathbf{r}}_{\boldsymbol{\nu}}=0. \end{aligned} \tag{2.6}$$

Imposing null derivative, the solution of (2.5) is

$$\boldsymbol{\Lambda}\Delta\boldsymbol{\mu}^* = -\boldsymbol{\nu}. \tag{2.7}$$

Note that the Gauss-Newton method is equivalent to Newton's method by approximating the Hessian cost by the squared Jacobian ($2\mathbf{A}^\top\mathbf{A}$), since the gradient of the summation of the squared residuals is $2\boldsymbol{\nu}$.

Solving (2.7) requires the inversion of $\boldsymbol{\Lambda}$ which may be a large matrix (and growing along the SLAM experiment). To address it, the Cholesky factorization can be used $\boldsymbol{\Lambda} = \mathbf{R}^\top\mathbf{R}$ being $\mathbf{R}$ an upper triangular matrix [iv]. Then, (2.7) becomes

$$\mathbf{R}^\top\mathbf{R}\Delta\boldsymbol{\mu}^* = -\boldsymbol{\nu}, \tag{2.8}$$

which can be solved using a forward-substitution followed by a back-substitution

$$\mathbf{R}^\top\mathbf{y} = -\boldsymbol{\nu}, \tag{2.9}$$

$$\mathbf{R}\Delta\boldsymbol{\mu}^* = \mathbf{y}. \tag{2.10}$$

As previously stated, the information matrix $\boldsymbol{\Lambda}$ is very sparse. Efficient Cholesky factorization implementations are suitable for sparse matrices [Davis, 2006; Chen et al., 2008]. Furthermore, the resulting factorization matrix $\mathbf{R}$ is normally sparse speeding up the forward and back-substitution processes.

Exploiting landmark-based sparsity structure

As commonly done in Bundle Adjustment methods since [Konolige, 2010], the sparsity structure of landmark-based SLAM can be exploited to further improve computational efficiency [Kümmerle et al., 2011]. Consider that the robot poses $\mathbf{x}_P$ and landmarks $\mathbf{x}_L$ are ordered in blocks. Then (2.7) can be rewritten as

$$\begin{bmatrix}\boldsymbol{\Lambda}_{PP} & \boldsymbol{\Lambda}_{PL} \\ \boldsymbol{\Lambda}_{PL}^\top & \boldsymbol{\Lambda}_{LL}\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{\mu}_P \\ \Delta\boldsymbol{\mu}_L\end{bmatrix} = -\begin{bmatrix}\boldsymbol{\nu}_P \\ \boldsymbol{\nu}_L\end{bmatrix}. \tag{2.11}$$

[iv] or equivalently, $\boldsymbol{\Lambda} = \mathbf{L}\mathbf{L}^\top$ being $\mathbf{L}$ a lower triangular matrix: $\mathbf{L} = \mathbf{R}^\top$

It can be posed as two separate equations

$$\boldsymbol{\Lambda}_{PP}\Delta\boldsymbol{\mu}_P + \boldsymbol{\Lambda}_{PL}\Delta\boldsymbol{\mu}_L = -\boldsymbol{\nu}_P \tag{2.12}$$
$$\boldsymbol{\Lambda}_{PL}^\top\Delta\boldsymbol{\mu}_P + \boldsymbol{\Lambda}_{LL}\Delta\boldsymbol{\mu}_L = -\boldsymbol{\nu}_L. \tag{2.13}$$

Then, solving for $\boldsymbol{\mu}_L$ in (2.13) and substituting in (2.12) we obtain a new smaller linear problem

$$(\boldsymbol{\Lambda}_{PP} - \boldsymbol{\Lambda}_{PL}\boldsymbol{\Lambda}_{LL}^{-1}\boldsymbol{\Lambda}_{PL}^\top)\Delta\boldsymbol{\mu}_P = -\boldsymbol{\nu}_P + \boldsymbol{\Lambda}_{PL}\boldsymbol{\Lambda}_{LL}^{-1}\boldsymbol{\nu}_L. \tag{2.14}$$

To build it, the inversion of $\boldsymbol{\Lambda}_{LL}$ is required but it only has linear cost, since the information matrix part relating to the landmark variables $\boldsymbol{\Lambda}_{LL}$ is block diagonal (see the bottom-left block corner in Figure 2.2.b). Then, solving the NLS can be done via Cholesky factorization of the significantly smaller matrix $\boldsymbol{\Lambda}_{PP} - \boldsymbol{\Lambda}_{PL}\boldsymbol{\Lambda}_{LL}^{-1}\boldsymbol{\Lambda}_{PL}^\top$. Once $\Delta\boldsymbol{\mu}_P$ is obtained, $\Delta\boldsymbol{\mu}_L$ is taken from (2.13) with no need to invert or factorize any other matrix.

This is a widely used strategy in Bundle Adjustment. In problems in which the amount of landmarks are much higher than camera poses the reduction of the problem is significant. However, in [Agarwal et al., 2010] the scalability of this reduction is questioned for very large datasets. Actually, (2.14) performs a marginalization of all landmarks inducing a quite dense problem.

2.3.2 Square-root SAM

Alternatively to Cholesky method, the problem can be solved in its square-root form. The $\sqrt{\text{SAM}}$ method [Dellaert and Kaess, 2006], solves the square-root problem using the QR decomposition of $\mathbf{A}$

$$\mathbf{A} = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\0\end{bmatrix}, \tag{2.15}$$

being $\mathbf{Q}$ an orthogonal matrix ($\mathbf{Q}^{-1} = \mathbf{Q}^\top$) and $\mathbf{R}$ an upper triangular matrix. Applying it to (2.5) leads to

$$\Delta\boldsymbol{\mu}^* = \arg\min_{\Delta\boldsymbol{\mu}} \left\| \mathbf{Q}\begin{bmatrix}\mathbf{R}\\0\end{bmatrix}\Delta\boldsymbol{\mu} + \mathbf{r}\right\|^2$$

Since it is orthogonal, pre-multiplying by $\mathbf{Q}$, the squared norm remains the same, obtaining

$$\begin{aligned}
&= \arg\min_{\Delta\boldsymbol{\mu}} \left\| \mathbf{Q}^\top\mathbf{Q}\begin{bmatrix}\mathbf{R}\\0\end{bmatrix}\Delta\boldsymbol{\mu} + \mathbf{Q}^\top\mathbf{r}\right\|^2\\
&= \arg\min_{\Delta\boldsymbol{\mu}} \left\| \begin{bmatrix}\mathbf{R}\\0\end{bmatrix}\Delta\boldsymbol{\mu} + \begin{bmatrix}\mathbf{b}\\\mathbf{c}\end{bmatrix}\right\|^2\\
&= \arg\min_{\Delta\boldsymbol{\mu}} \|\mathbf{R}\Delta\boldsymbol{\mu} + \mathbf{b}\|^2 + \|\mathbf{c}\|^2.
\end{aligned}$$

Since the second term is constant, the solution of (2.5) is

$$\mathbf{R}\Delta\boldsymbol{\mu}^* = -\mathbf{b} \tag{2.16}$$

which can be solved by back-substitution. Note that the second term is the summation of the squared Mahalanobis norm of all linearized factors errors of the optimal solution. In other words, $\|\mathbf{c}\|^2$ is the minimum residual achievable of the linearized problem.

QR decomposition can be computed via several methods such as Givens rotations [Givens, 1958], Householder transformations [Householder, 1958], or Gram-Schmidt process [Gram, 1883; Schmidt, 1907].

Note that for the same problem the matrix $\mathbf{R}$ is exactly the same either from Cholesky factorization of $\boldsymbol{\Lambda}$ or from QR decomposition of $\mathbf{A}$ since

$$\boldsymbol{\Lambda} = \mathbf{A}^\top\mathbf{A} \underset{QR}{=} \mathbf{R}^\top\mathbf{Q}^\top\mathbf{Q}\mathbf{R} = \mathbf{R}^\top\mathbf{R} \underset{Chol.}{=} \boldsymbol{\Lambda}. \tag{2.17}$$

2.3.3 Augmenting the state

When a new measurement produces a new node in the graph (an odometry measurement or observing a new landmark), at the time of rebuilding the problem, the state size should be augmented. The information matrix $\boldsymbol{\Lambda}$ rows and columns should be augmented by the new node state size. And the residuals Jacobian $\mathbf{A}$ only has to be augmented in its columns. In both cases the matrix new entries are filled with zeros since the Jacobians of old measurements w.r.t. the new variables are null.

The estimation mean can be initialized using the measurement inverse model $g_k(\boldsymbol{\mu}, \mathbf{z}_k)$[(v)] if available

$$\boldsymbol{\mu}' = \begin{bmatrix} \boldsymbol{\mu} \\ g_k(\boldsymbol{\mu}, \mathbf{z}_k) \end{bmatrix}.$$

Note that in this case, the measurement error $\mathbf{e}_k$ is zero. Therefore, if only this factor has been added, building entirely and solving the problem is not required, just the state resize and adding this factor.

2.3.4 Fill-in, sparsity and variable ordering

The fill-in of $\mathbf{R}$ directly affects on the computational cost of solving the forward and/or back-substitution processes of (2.10) and (2.16). As previously stated, given the high sparsity of $\boldsymbol{\Lambda}$ and $\mathbf{A}$, the factorization $\mathbf{R}$ is normally sparse and the computational complexity of solving (2.7) is reduced dramatically.

Moreover, altering the order of the variables induces different factorizations. The $\mathbf{R}$ fill-in can be further reduced altering the order of the state variables. Despite computing the optimal variable ordering given the sparsity pattern of $\boldsymbol{\Lambda}$ or $\mathbf{A}$ is NP-complete [Yannakakis, 1981], there are some known efficient algorithms that provide variable orderings that significantly reduce the resulting factorization fill-in. These algorithms compute good variable orderings given the sparsity pattern of $\mathbf{A}$ or $\boldsymbol{\Lambda}$ based on the Minimum Degree ordering.

(v) The measurement inverse model $g_k(\boldsymbol{\mu}, \mathbf{z}_k)$ is a function such that $h_k([\boldsymbol{\mu}^\top g_k(\boldsymbol{\mu}, \mathbf{z})^\top]^\top) = \mathbf{z}_k$

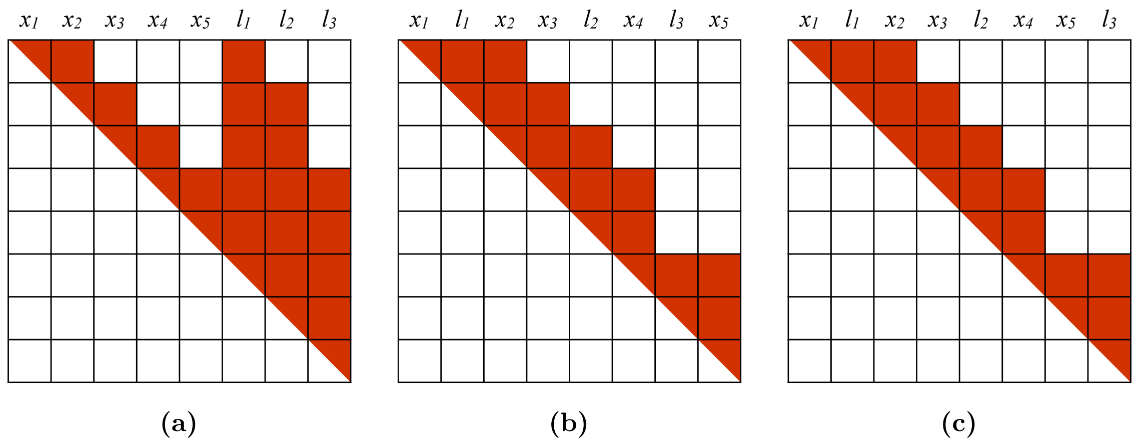

x_1 x_2 x_3 x_4 x_5 l_1 l_2 l_3
x_1 l_1 x_2 x_3 l_2 x_4 l_3 x_5
x_1 l_1 x_2 x_3 l_2 x_4 x_5 l_3
(a)
(b)
(c)

Figure 2.3: Fill-in induced in $\mathbf{R}$ without variable reordering (a) and using COLAMD reordering method taking as input the residuals Jacobian $\mathbf{A}$ (b) and the information matrix $\boldsymbol{\Lambda}$ (c); corresponding to the factor graph example (Figure 2.1).

Exact Minimum Degree (EMD) [Tinney and Walker, 1967] and Approximated Minimum Degree (AMD) [Amestoy et al., 2004] methods can be applied only on symmetric matrices, so they are suitable if using Cholesky factorization since they require $\boldsymbol{\Lambda}$. Column Approximated Minimum Degree (COLAMD) [Davis et al., 2004] is able to work with rectangular matrices, being suitable in both Cholesky and QR factorizations. METIS [Karypis and Kumar, 1995] and NESDIS [Davis] methods use graph partitioning and then compute the Minimum Degree ordering of the partitioned graphs. Despite being based on approximations and heuristics, all mentioned ordering methods produce very similar fill-in ratios and its computational cost is significantly smaller than the solve time [Agarwal and Olson, 2012].

Figure 2.3 depicts an example of the improvement in the $\mathbf{R}$ fill-in by using COLAMD variable reordering. As already explained, COLAMD can be used with both residuals Jacobian $\mathbf{A}$ and information matrix $\boldsymbol{\Lambda}$. However the resulting ordering can be slightly different (see the variables order at top of the figure).

2.4 Incremental methods

As introduced before, SLAM is normally perceived as an online robotic application. In this case, the graph grows adding more factors and nodes as the experiment goes by. Building the problem (2.5) from scratch (and computing the new Cholesky or QR factorization to solve it) every time a new node and/or factor is added to the graph is far from optimal.

Incremental smoothing methods reuse the previously factorized problem and incrementally update it with new nodes and/or factors directly into the factorized form. However, it implies that the variable ordering is not altered and the previous factors are not relinearized with the new state estimate.

Note that, no matter which factorization is used, if no relinearization is performed after solving the problem and no new factors are added, the linearized problem (2.5) remains the same. This means that the optimal estimation update is zero $\Delta\boldsymbol{\mu}^* = 0$. Then, considering (2.8) and (2.16), $\boldsymbol{\nu} = 0$ and $\mathbf{b} = 0$, respectively, if the residuals are not relinearized.

2.4.1 Incremental Cholesky

An incremental version of Cholesky factorization was introduced in [Polok et al., 2013]. As previously mentioned, the measurements only constrain a few nodes of the whole state, and hence its Jacobians are very sparse. Let J_k be the right Jacobian column-block starting from the left-most column that contains any non-zero element $\mathbf{J}_k = [0 \; J_k]$. After an update, only the bottom-right corner of the information matrix is subject to change

$$\mathbf{\Lambda}' = \mathbf{\Lambda} + \mathbf{J}_k^\top \mathbf{\Omega}_k \mathbf{J}_k = \begin{bmatrix} \mathbf{\Lambda}_{aa} & \mathbf{\Lambda}_{ab} \\ \mathbf{\Lambda}_{ba} & \mathbf{\Lambda}_{bb} + J_k^\top \mathbf{\Omega}_k J_k \end{bmatrix} \tag{2.18}$$

Comparing the old and the new Cholesky factorizations

$$\begin{aligned}
\mathbf{\Lambda} &= \mathbf{R}^\top \mathbf{R} \\
\begin{bmatrix} \mathbf{\Lambda}_{aa} & \mathbf{\Lambda}_{ab} \\ \mathbf{\Lambda}_{ba} & \mathbf{\Lambda}_{bb} \end{bmatrix} &= \begin{bmatrix} \mathbf{R}_{aa}^\top & 0 \\ \mathbf{R}_{ab}^\top & \mathbf{R}_{bb}^\top \end{bmatrix} \begin{bmatrix} \mathbf{R}_{aa} & \mathbf{R}_{ab} \\ 0 & \mathbf{R}_{bb} \end{bmatrix} \\
\begin{bmatrix} \mathbf{\Lambda}_{aa} & \mathbf{\Lambda}_{ab} \\ \mathbf{\Lambda}_{ba} & \mathbf{\Lambda}_{bb} \end{bmatrix} &= \begin{bmatrix} \mathbf{R}_{aa}^\top \mathbf{R}_{aa} & \mathbf{R}_{aa}^\top \mathbf{R}_{ab} \\ \mathbf{R}_{ab}^\top \mathbf{R}_{aa} & \mathbf{R}_{ab}^\top \mathbf{R}_{ab} + \mathbf{R}_{bb}^\top \mathbf{R}_{bb} \end{bmatrix} \\
\mathbf{\Lambda}' &= \mathbf{R}'^\top \mathbf{R}' \\
\begin{bmatrix} \mathbf{\Lambda}_{aa} & \mathbf{\Lambda}_{ab} \\ \mathbf{\Lambda}_{ba} & \mathbf{\Lambda}_{bb} + J_k^\top \mathbf{\Omega}_k J_k \end{bmatrix} &= \begin{bmatrix} \mathbf{R}'^\top_{aa} \mathbf{R}'_{aa} & \mathbf{R}'^\top_{aa} \mathbf{R}'_{ab} \\ \mathbf{R}'^\top_{ab} \mathbf{R}'_{aa} & \mathbf{R}'^\top_{ab} \mathbf{R}'_{ab} + \mathbf{R}'^\top_{bb} \mathbf{R}'_{bb} \end{bmatrix}
\end{aligned} \tag{2.19}$$

we see that the upper part of the factorization remains unchanged $\mathbf{R}'_{aa} = \mathbf{R}_{aa}, \mathbf{R}'_{ab} = \mathbf{R}_{ab}$, and the Cholesky factorization has to be performed only for the bottom-right corner

$$\mathbf{R}'^\top_{bb} \mathbf{R}'_{bb} = \mathbf{\Lambda}_{bb} + J_k^\top \mathbf{\Omega}_k J_k - \mathbf{R}_{ab}^\top \mathbf{R}_{ab}. \tag{2.20}$$

Then, the new factorized matrix $\mathbf{R}'$ is obtained from the previous $\mathbf{R}$ by only changing its bottom-right corner to $\mathbf{R}'_{bb}$. Finally, the problem can be solved with the corresponding forward and back substitution

$$\mathbf{R}'^\top \mathbf{R}' \Delta \boldsymbol{\mu} = \boldsymbol{\nu}'. \tag{2.21}$$

Remember that since no relinearization of previous factors is performed, only the last factor residual is not null $\boldsymbol{\nu}' = \mathbf{J}_k^\top \mathbf{\Omega}_k \mathbf{e}_k(\boldsymbol{\mu})$.

2.4.2 Incremental QR

Kaess et al. [2008] proposed iSAM, an incremental version of $\sqrt{\text{SAM}}$. Posteriorly, a more sophisticated version iSAM2 was presented [Kaess et al., 2012]. Since QR decomposition used in iSAM and iSAM2 is based on an orthogonal matrix, the problem can be updated and factorized directly from the previous factorized problem:

$$\begin{aligned}\left\|\begin{bmatrix}\mathbf{R}\\ \boldsymbol{\Omega}_k^{1/2}\mathbf{J}_k\end{bmatrix}\Delta\boldsymbol{\mu}+\begin{bmatrix}0\\ \boldsymbol{\Omega}_k^{1/2}\mathbf{e}_k(\boldsymbol{\mu})\end{bmatrix}\right\|^2 &= \left\|\mathbf{Q}'\begin{bmatrix}\mathbf{R}'\\ 0\end{bmatrix}\Delta\boldsymbol{\mu}+\begin{bmatrix}0\\ \boldsymbol{\Omega}_k^{1/2}\mathbf{e}_k(\boldsymbol{\mu})\end{bmatrix}\right\|^2\\ &= \left\|\begin{bmatrix}\mathbf{R}'\\ 0\end{bmatrix}\Delta\boldsymbol{\mu}+\mathbf{Q}'^{\top}\begin{bmatrix}0\\ \boldsymbol{\Omega}_k^{1/2}\mathbf{e}_k(\boldsymbol{\mu})\end{bmatrix}\right\|^2\\ &= \|\mathbf{R}'\Delta\boldsymbol{\mu}+\mathbf{b}'\|^2+\|\mathbf{c}'\|^2. \end{aligned} \tag{2.22}$$

The computational cost of the new decomposition is extremely reduced compared with the batch decomposition of the entire residuals Jacobian $\mathbf{A}$ since only the last rows corresponding to the new factor do not follow the upper triangular form.

As in the batch case, if the new measurement implies a state augmentation, a new block column of zeros is added to the matrix $\mathbf{R}$ before the incremental update.

2.4.3 Relinearization and variable ordering in incremental methods

The main advantage of smoothing with respect to filtering is the capability of relinearization. It produces higher accuracy preventing the estimation from becoming inconsistent. Additionally, variable ordering highly affects on the resulting factorized matrix $\mathbf{R}$ fill-in which is related with the computational costs of the forward and/or back substitution.

However, incremental methods described above do not relinearize the old factors already encoded in matrix $\mathbf{R}$ nor change the variable ordering appending new variables at the end. Carrying incremental methods endlessly entails avoiding relinearization and variable reordering. Therefore, if one wants to exploit the SAM benefits, every so often the problem should be rebuilt (completely or partially) relinearizing factors, reordering variables and finally re-computing the factorization and solving.

Some heuristics are normally used to decide when the problem should be rebuilt. In iSAM [Kaess et al., 2008], the problem is rebuilt entirely periodically after n new nodes are added to the state. However, more elaborate mechanisms can be devised. In iSAM2 [Kaess et al., 2012] a new graph structure is used called Bayes Tree. It encodes the non-zero entries of the equivalent $\mathbf{R}$ matrix. Also, it introduces a method to partially rebuild the problem (relinearize and reordering) depending on how much the linearization point diverged from the current estimate. It makes a difference from iSAM in terms of computational time and estimation accuracy.

Furthermore, variable reordering can speed-up the incremental factorization process as well. Both incremental factorizations are faster as bigger is the left-most block of zeros of the Jacobian. In other words, the righter the variables involved in a measurement are, the faster the incremental factorization is. Hence, the efficiency of future incremental factorizations depends on the current variable ordering.

As noted in [Kaess et al., 2012], in SLAM applications new factors mostly involve the most recently observed nodes. Landmarks and/or poses that have been observed in the recent past are more likely to be observed again. Therefore, apart from the induced fill-in in $\mathbf{R}$, a good variable ordering for incremental methods should consider keeping the most recently observed nodes (its variables) at the right-most part of the state. Constrained COLAMD (CCOLAMD) ordering method allows to constrain the ordering by defining ordered groups of variables. In [Kaess et al., 2012] CCOLAMD is used each time a variable reordering is performed imposing the last observed nodes to be at the right-most part of the state.

2.5 Before and beyond Gauss-Newton

The Gauss-Newton NLS formulation previously presented has two main drawbacks. The first is the computational resources demand, not only computational time but also memory to allocate and operate with large matrices. For the last decade, due to technology advances it has progressively been a minor drawback for most of average robotics applications, but the issue made arise several alternative methods to face online SLAM applications such as filtering or relaxation methods.

Secondly, Gauss-Newton may have convergence issues due to a bad initial guess and non-linearities. The Gauss-Newton step $\Delta\boldsymbol{\mu}^*$ is computed by linearizing the problem in the current estimate. However, the quadratic form of the NLS is only approximately true near the optimal solution. Otherwise, the Gauss-Newton step can diverge. Furthermore, Gauss-Newton approximates the quadratic Newton step by approximating the Hessian by the squared residuals Jacobian $2\mathbf{A}^\top\mathbf{A}$. This also contributes to the divergence of the resulting step.

In the literature, other SLAM methods have been proposed to cope with the mentioned Gauss-Newton approach drawbacks. This section pretends to be a brief summary of these alternative SLAM methods.

2.5.1 Filtering

Bayes filters constitute a family of methods historically applied to solve the SLAM problem. This problem is assumed to be a Markov process (i.e. probability of events depend only on the state previous to each event) and sensor measurements are observations of the state.

Bayes filters have in their fundamentals the aim of estimating the present value of a state. It means, the previous state value is recursively used to build a prior on the current value: *prediction*. Trivially, concatenating noisy predictions over time accumulates uncertainty and the estimation of the state rapidly diverges from the actual state. Independent observations of the state reduce its estimation uncertainty: *update*.

Applied to SLAM, the traditional Bayes filter approach only includes the current robot configuration and a set of landmarks in the state. It implies maintaining a (relatively) reduced problem only growing due to the size of the map.

The prediction only affects the part of the state $\mathbf{x}^R$ referring to the robot pose due to the motion model f:

$$\mathbf{x}^R_{k+1} = f(\mathbf{x}^R_k) + \mathbf{w}_{k+1}, \tag{2.23}$$

where $\mathbf{w}_k$ is some noise normally assumed to be Gaussian $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{\Sigma}_{\mathbf{w}_k})$. This is normally performed with a dynamic model (e.g. constant velocity) or any proprioceptive sensor measurement (e.g. wheel encoders or IMU). The second case would be equivalent to the measurement inverse model as defined in Sec. 2.3.3, $f(\mathbf{x}_k) = g(\boldsymbol{x}_k, \mathbf{z}_k)$.

In the update step, other sensor measurements observe some geometrical relation between state variables,

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k. \tag{2.24}$$

where $\mathbf{v}_k$ is also a noise normally assumed to be Gaussian $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{\Sigma}_{\mathbf{v}_k})$. Note the equivalence with (2.3). Smoothing only considers observations since by keeping old robot poses in the state, it does not encode causality.

Extended Kalman Filter

The Extended Kalman Filter (EKF) is the adaptation of the Kalman Filter [Kalman, 1960] for non-linear transition and measurement models. It iteratively estimates the state and its uncertainty modeling it as a multi-variate Gaussian distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$. Both prediction and update steps directly operate on the state mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$.

Whichever motion model $\boldsymbol{f}(\mathbf{x}), \mathbf{w}$ is used to perform the prediction step, in the EKF it consists of an uncertainty propagation,

$$\boldsymbol{\mu}_{k|k-1} = f(\boldsymbol{\mu}_{k-1|k-1}) \tag{2.25}$$

$$\mathbf{\Sigma}_{k|k-1} = \mathbf{F}_k \mathbf{\Sigma}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{\Sigma}_{\mathbf{w}_k}, \tag{2.26}$$

where the Jacobian of the motion model evaluated at the current state estimation is used to propagate the covariance:

$$\mathbf{F}_k = \left. \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\boldsymbol{\mu}_{k-1|k-1}}.$$

Note that since the motion model only depends on the last robot state (2.23), the Jacobian $\mathbf{F}_k$ will be full of zeros except for the variables corresponding to the robot pose.

The update step reduces the uncertainty of the estimation and also corrects the mean due to the observation $\mathbf{z}_k$:

$$\boldsymbol{\mu}_{k|k} = \boldsymbol{\mu}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - h(\boldsymbol{\mu}_{k|k-1})) \tag{2.27}$$

$$\mathbf{\Sigma}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{J}_k) \mathbf{\Sigma}_{k|k-1} \tag{2.28}$$

being $\mathbf{J}_k$ the measurement model Jacobian evaluated at the current state estimation $\boldsymbol{\mu}_{k|k-1}$ and $\mathbf{K}_k$ the so-called optimal Kalman gain

$$\mathbf{K}_k = \mathbf{\Sigma}_{k|k-1} \mathbf{J}_k^\top (\mathbf{J}_k \mathbf{\Sigma}_{k|k-1} \mathbf{J}_k^\top + \mathbf{\Sigma}_{\mathbf{v}_k})^{-1}. \tag{2.29}$$

The EKF has been widely used for SLAM applications from the very first approaches [Ayache and Faugeras, 1989; Leonard et al., 1992] to present applications [Li and Mourikis, 2013].

However, while the prediction step is computationally cheap due to the mentioned Jacobian sparsity, the update step computational cost is important. Since the covariance matrix $\mathbf{\Sigma}$ is dense, the complexity of the update is at least $\mathcal{O}(n^{2.37})$ due to the matrix products in (2.28) and (2.29).

Moreover, the prediction step actually performs a marginalization of previous robot state. Then, it avoids the relinearization of old observations leading to inconsistency due to the non-linearity of SLAM problems [Julier and Uhlmann, 2001].

Extended Information Filter

As introduced in Section 2.2, a multi-variate Gaussian distribution can be used in canonical or information form $\mathbf{x} \sim \mathcal{N}^{-1}(\boldsymbol{\eta}, \mathbf{\Lambda})$. The Extended Information Filter (EIF) is equivalent to the EKF switched to the information form. Analogously, it operates directly over the information vector and matrix $\boldsymbol{\eta}, \mathbf{\Lambda}$.

Then, the prediction step is obtained by applying (2.2) to the EKF prediction (2.25) (2.26)

$$\mathbf{\Lambda}_{k|k-1} = (\mathbf{F}_k \mathbf{\Lambda}_{k-1|k-1}^{-1} \mathbf{F}_k^\top + \mathbf{\Sigma}_{\mathbf{w}_k})^{-1} \tag{2.30}$$

$$\boldsymbol{\eta}_{k|k-1} = \mathbf{\Lambda}_{k-1|k} f(\boldsymbol{\mu}_{k-1|k-1}) \tag{2.31}$$

And the update step can be obtained from (2.28) via Woodbury identity (see Prop. 2 in Appendix A)

$$\mathbf{\Lambda}_{k|k} = \mathbf{\Lambda}_{k|k-1} + \mathbf{J}_k^\top \mathbf{\Sigma}_{\mathbf{v}_k}^{-1} \mathbf{J}_k \tag{2.32}$$

$$\boldsymbol{\eta}_{k|k} = \boldsymbol{\eta}_{k|k-1} + \mathbf{J}_k^\top \mathbf{\Sigma}_{\mathbf{v}_k}^{-1} \big(\mathbf{z}_k - h(\boldsymbol{\mu}_{k|k-1}) + \mathbf{J}_k \boldsymbol{\mu}_{k|k-1}\big). \tag{2.33}$$

Note that the estimation mean is required in (2.31) and (2.33). It can be easily updated after the prediction step using the motion model (2.25). After the update step, however, $\mathbf{\Lambda}_{k|k}$ should be inverted to recover $\boldsymbol{\mu}_{k|k}$ using (2.2). Nevertheless, the inverted matrix can be reused in the next prediction step (2.30).

Since EIF and EKF are equivalent filters only using different Gaussian parameterizations, EIF also suffers from inconsistency due to the bad linearization points. In contrast to the EKF, the EIF update step is computationally cheap due to the Jacobian sparsity and the prediction step is the one computationally expensive due to the mentioned matrix inversion.

The main benefit of EIF is the information matrix sparsity nature. As opposed to the canonical representation, the information form encodes conditional relations instead of correlation. In other words, it only has non-zero entries if the corresponding variables have been observed (whether directly measured or as a result of marginalizing old nodes). It allows the use of sparse algebra implementations to efficiently perform the operations.

However, after a few prediction steps the information matrix sparsity is lost. To maintain the sparsity in $\mathbf{\Lambda}$, in [Thrun et al., 2004] some conditionals are removed by setting the corresponding off-diagonal blocks to zero.

Alternatively, [Eustice et al., 2005] proposed to keep all robot trajectory in the state (i.e. avoiding the prediction step) maintaining the sparsity in $\mathbf{\Lambda}$ without performing any approximation. In [Ila et al., 2010], the same idea was adopted. To mitigate the inconsistency issue that normally ends in a overconfident estimation, the method discards the least informative measurements. However, despite maintaining all trajectory in the state, none of the methods proposed relinearizing the factors. In other words, they adopted the drawbacks from both smoothing and filtering: large resources demands and inconsistency.

Actually, Thrun et al. [2005] proposed the so-called full SLAM defining it as a batch information filter that keeps all variables in the state. It allows relinearization to be iteratively solved. It is, in other words, the Gauss-Newton SAM approach described in the previous sections.

Particle filters

In most SLAM methods, the probability density is modeled as a multi-variate Gaussian distribution. The non-linearities of the SLAM problem (due to rotations, for example) make this assumption very inaccurate.

Particle filters or sequential Montecarlo filters represent the probability density using a finite number of weighted samples. They are based on the principle that any probability density function can be approximated as the sum of weighted Dirac deltas, corresponding to each sample, for a number of samples tending to infinite.

In mobile robotics, particle filters have been applied to robot localization [Dellaert et al., 1999; Thrun et al., 2000]. However, for highly dimensional problems such as SLAM, the amount of particles needed to consistently describe the probability density grows exponentially. The Rao-blackwellized particle filter (RBPF) marginalizes out part of the problem to reduce the dimensionality of the probability distribution represented by the samples. In exchange, the load of each particle is increased since each one encodes the marginalized subproblem. Therefore, RBPF are capable of describing more complex probability distributions with less samples.

Murphy and Doucet introduced the use of RBPF for SLAM [Murphy, 2000; Doucet et al., 2000] in which the particle filter represents the robot trajectory and each sample contains the resulting grid-based map corresponding to such trajectory. FastSLAM [Montemerlo, 2002] proposed a similar approach with a landmark-based map using an EKF inside each particle. Grisetti et al. [2005, 2007] proposed some improvements for grid-based RBPF SLAM resulting in the GMapping algorithm that became very popular within the community and it is still widely used nowadays.

2.5.2 Relaxation methods

After posing SLAM as a NLS problem (2.5), relaxation methods were firstly seen as alternatives to the costly matrix inversion or factorization required using Gauss-Newton. Recalling (2.7), relaxation methods aim to solve the problem

$$\mathbf{\Lambda}\Delta\boldsymbol{\mu} = -\boldsymbol{\nu}$$

without inverting nor factorizing $\mathbf{\Lambda}$.

The computational resources demands of relaxation methods are minor. In exchange, a large number of iterations is necessary to obtain a solution approximately equal to a single Gauss-Newton step. The convergence rate mainly depends on how much diagonal-dominant is the problem. In other words, how much independent is each variable from the rest.

Two main relaxation methods have been adopted to solve SLAM problems, Jacobi and Gauss-Seidel.

Jacobi method

The most simple relaxation is the Jacobi method. It decomposes the matrix in two terms: the diagonal and the remainder $\mathbf{\Lambda} = \mathbf{\Lambda}_D + \mathbf{\Lambda}_R$. Then, the solution is iteratively found

$$\Delta\boldsymbol{\mu}^{(i+1)} = \mathbf{\Lambda}_D^{-1}(-\boldsymbol{\nu} - \mathbf{\Lambda}_R\Delta\boldsymbol{\mu}^{(i)}).$$

The main benefit of the method is that it can be performed element by element without the need of performing, storing nor inverting the decomposition matrices

$$\Delta\mu_j^{(i+1)} = \frac{1}{\Lambda_{jj}}\Big(-\boldsymbol{\nu} - \sum_{l\neq j}\Lambda jl\Delta\mu_l^{(i)}\Big). \tag{2.34}$$

Here, Λ_{jl} denotes the (j,l) entry of $\mathbf{\Lambda}$ and $\Delta\mu_j$ the j-th entry of $\Delta\boldsymbol{\mu}$.

At each iteration, Jacobi relaxation computes the optimal value of each variable if the rest of variables where fixed. To compute it, the old values $\Delta\boldsymbol{\mu}^{(i)}$ are always used even if a newer value has been computed for some variables. This allows the parallelization of the process requiring at least two vectors to store $\Delta\boldsymbol{\mu}^{(i)}$ and $\Delta\boldsymbol{\mu}^{(i+1)}$.

In [Thrun et al., 2004], in order to recover the (approximated) mean $\boldsymbol{\mu}$ in an EIF, a constant number of Jacobi relaxation iterations is performed following (2.2).

Gauss-Seidel method

Gauss-Seidel relaxation is similar to Jacobi relaxation. This time, the matrix is decomposed in a lower matrix and a strictly upper matrix (i.e. its diagonal is zero) $\mathbf{\Lambda} = \mathbf{\Lambda}_L + \mathbf{\Lambda}_U$. The solution then can be found element by element

$$\Delta\boldsymbol{\mu}^{(i+1)} = \mathbf{\Lambda}_L^{-1}(-\boldsymbol{\nu} - \mathbf{\Lambda}_U\Delta\boldsymbol{\mu}^{(i)}).$$

This can be solved by forward-substitution without need of actually computing the descomposition nor the inversion of $\mathbf{\Lambda}_L$

$$\Delta\mu_j^{(i+1)} = \frac{1}{\Lambda_{jj}}\Big(-\boldsymbol{\nu} - \sum_{l<j} \Lambda jl \Delta\mu_l^{(i+1)} - \sum_{l>j} \Lambda jl \Delta\mu_l^{(i)}\Big). \tag{2.35}$$

Gauss-Seidel relaxation can be understood as a variation of Jacobi relaxation that directly updates the values in the same vector. Thus, it uses the most recent state estimation improving the convergence but avoiding parallelization.

Duckett and Howard introduced relaxation methods to solve SLAM problems [Duckett et al., 2000, 2002; Howard et al., 2001]. Their proposed iterative methods were described as "move each node to where its neighbors think it should be" which can be interpreted as a block-Gauss-Seidel relaxation. Afterwards, the same approach was proposed by Frese et al. [2005] at multiple resolution levels in order to improve convergence.

2.5.3 Line search

To overcome convergence issues of Gauss-Newton NLS, two different algorithm families are devised in the different non-linear optimization literature: line search and trust region. They can be considered as dual strategies. While trust region methods decide the maximum step size a priori and afterwards its best direction is computed, line search methods compute the direction of the step first and posteriorly finds its best size.

Depending on how the step direction and the search are performed, different line search methods are devised. Line search algorithms can be generalized as finding the solution as

$$\Delta\boldsymbol{\mu} = -\alpha \underbrace{\mathbf{H}^{-1}\boldsymbol{\nabla}}_{\mathbf{d}}, \tag{2.36}$$

where ∇ is the gradient of the cost, i.e. the sum of squared residuals in SAM. Also, $\mathbf{H}$ can be either the actual Hessian (i.e. Newton method), an approximation of the Hessian (i.e. quasi-Newton methods) or the Identity matrix (i.e. gradient descent).

Normally, line search is applied in gradient descent or quasi-Newton methods such as DFP [Davidon, 1959; Fletcher and Powell, 1963] BFGS [Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970] or LBFGS [Nocedal, 1980]. However, it can be applied to the Gauss-Newton step as well.

All previously mentioned quasi-Newton methods estimate the Hessian (or its inverse) according to the gradient change by imposing the so-called *secant equation*: $\Delta\boldsymbol{\nabla} = \mathbf{H}\Delta\boldsymbol{\mu}$. The main benefit of most quasi-Newton methods is that the inverse of the Hessian matrix can be directly estimated instead of the Hessian. It avoids the computational cost of the Hessian inversion.

Once the step direction $\mathbf{d}$ is computed, the step size is determined by finding α. It can be either *exactly* computed or *loosely* found by inexact line search.

Inexact line search iteratively searches for a value of α until some conditions are fulfilled, such as the Armijo-Goldstein [Armijo, 1966] or Wolfe conditions [Wolfe, 1969, 1971]. The former only checks that the step decreases sufficiently the total residual, the latter also checks for the slope change.

Posing the SLAM NLS problem (2.5) in terms of α and the computed direction $\mathbf{d}$,

$$\mathbf{r}(\alpha) = \|\mathbf{A}\alpha\mathbf{d} + \mathbf{r}\|^2,$$

null derivative w.r.t. α can be imposed obtaining the exact optimal step size

$$\alpha = -\frac{\mathbf{d}^\top \boldsymbol{\nu}}{\|\mathbf{A}\mathbf{d}\|^2}. \tag{2.37}$$

For instance, in gradient descent the chosen direction is the negative gradient $\mathbf{d} = -\boldsymbol{\nabla}$. Then applying (2.37), the optimal step is

$$\Delta\boldsymbol{\mu}_C^* = -\frac{\|\boldsymbol{\nabla}\|^2}{\|\mathbf{A}\boldsymbol{\nabla}\|^2}\boldsymbol{\nabla}. \tag{2.38}$$

which is called the Cauchy point. However, gradient descent has a very slow convergence specially when having narrowed curved valleys leading to zig-zag behaviour.

Conjugate Gradient

The Conjugate gradient (CG) [Hestenes and Stiefel, 1952] method alternates conjugate directions with respect to $\mathbf{H}$ importantly improving the convergence of gradient descent. Two vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ are said to be conjugate w.r.t. a matrix $\mathbf{H}$ if and only if $\mathbf{d}_1^\top\mathbf{H}\mathbf{d}_2 = 0$. Differently from gradient descent, CG takes an $\mathbf{H}$-conjugate directions instead of the gradient itself. This can be done by taking the gradient direction and subtracting out some of the previous search direction

$$\mathbf{d}_k = -(\boldsymbol{\nabla}_k - \beta_k\mathbf{d}_{k-1}) \quad \text{with} \quad \beta_k = \frac{\|\boldsymbol{\nabla}_k\|^2}{\|\boldsymbol{\nabla}_{k-1}\|^2}. \tag{2.39}$$

The negative gradient is taken as the first direction $\mathbf{d}_0 = -\boldsymbol{\nabla}_0$.

CG provides the exact solution of the linear problem after a finite number of iterations lower than the problem size. However, it can provide a good solution (below a defined tolerance) in a quite smaller number of iterations. Convergence speed depends on the distribution of eigenvalues of $\mathbf{H}$, the most uniformly distributed the fastest the convergence. Thus, the condition number of the Hessian matrix is a good indicator

$$\kappa(\mathbf{H}) = \frac{|\lambda_{max}|}{|\lambda_{min}|}, \tag{2.40}$$

being λ_{max} and λ_{min} the maximum and the minimum eigenvalues of $\mathbf{H}$, respectively. Hence, the convergence of CG is faster as the condition number come closer to one.

Note that CG does not require storing the Hessian nor even the residuals Jacobian $\mathbf{A}$ but only evaluate the gradient and storing the last direction and gradient norm (2.39). This entails the main advantages for using CG in SLAM: low memory requirements and low computational cost of an iteration. On exchange, the main disadvantage is the slow convergence due to the usually high condition numbers in SLAM problems.

However, the convergence can be significantly improved by preconditioning. It also can be understood as re-parameterizing the problem. A good preconditioner should decrease the condition number. The best preconditioner is the inverse of the Hessian, however it is exactly what CG method aims to avoid (2.5). Therefore, finding a good preconditioner is a trade-off between the cost of computing it and the convergence improvement achieved.

Preconditioned conjugate gradient (PCG) has been widely applied in SLAM and Bundle Adjustment (BA) applications. The first PCG method for SLAM was presented in [Konolige, 2004] using incomplete Cholesky Factorization [Meijerink et al., 1977] as preconditioner. Dellaert et al. [2010] posteriorly proposed to use a SLAM sub-graph as preconditioner. A simple sub-graph is easily solved using QR and the whole problem is re-parametrized (or preconditioned) in terms of the sub-graph variables to be solved by CG.

In [Jian et al., 2012, 2013], the sub-graph preconditioner was shown to perform worse than Jacobi for 3D BA problems and an algorithm to greedy build sub-graphs to precondition the CG was proposed. In [Jian and Dellaert, 2014], an hybrid approach is presented which uses iSAM to solve a sub-graph and only when the error is too high, PCG is used to solve the whole problem. Afterwards, prior factors are added to the iSAM sub-graph to inject the PCG solution until the next PCG solution.

PCG has been also explored to solve large BA problems either for solving the NLS problem or the damped Levenberg-Marquardt problem (see next sub-section). Also, in [Byröd and Åström, 2010; Agarwal et al., 2010] PCG is shown to be a good alternative in which Cholesky factorization is infeasible or too resource demanding.

2.5.4 Trust region

Conversely to line search, trust region algorithms first defines a region where the quadratic model is trusted and afterwards the optimal direction is found within it. Trust region algorithms can be posed as a constrained version of the original problem (2.5):

$$\Delta\mu^* = \arg\min_{\Delta\boldsymbol{\mu}} \|\mathbf{A}\Delta\boldsymbol{\mu} + \mathbf{r}\|^2 \tag{2.41}$$

$$\text{s.t.} \quad \|\mathbf{D}\Delta\boldsymbol{\mu}\| \leq r \tag{2.42}$$

where r is the maximum radius allowed and $\mathbf{D}$ is any matrix to define a metric on the state domain, such as the identity or the Hessian diagonal.

Several trust region algorithms and variants have been proposed in non-linear optimization literature. Two of the most suited trust-region algorithms are described below.

Levenberg-Marquardt

The Levenberg-Marquardt (LM) [Levenberg, 1944; Marquardt, 1963] method imposes the trust region constraint by adding a radius term in the minimization cost

$$\Delta\mu^*_{LM} = \arg\min_{\Delta\boldsymbol{\mu}} \|\mathbf{A}\Delta\boldsymbol{\mu} + \mathbf{r}\|^2 + \lambda\|\mathbf{D}\Delta\boldsymbol{\mu}\|^2. \tag{2.43}$$

The convergence is controlled by dynamically changing the damping factor λ at each iteration depending on the resulting step. For high values of λ, the solution of (2.43) tends to the gradient descent direction. On the contrary, for λ tending to zero, the algorithm gets closer to the original Gauss-Newton in (2.5). Thus, λ is a parameter that inversely refers to the trust-region maximum radius r.

The damping factor is updated after each iteration due to how similar is the residual reduction w.r.t. the expected reduction. If the residual reduction was significantly bigger than the expected reduction, the trust region should be increased (decreasing the damping factor) and vice-versa.

One drawback of LM is that it can produce a step that does not reduce the cost, due to non-linearities, if the damping factor becomes too small. In this case, the damping factor is increased balancing to gradient descent behaviour but the problem should be solved again.

LM can be easily applied to Cholesky and QR methods described in the previous sections by altering the information matrix and the residuals Jacobian respectively

$$\begin{aligned} \boldsymbol{\Lambda} &\mapsto \boldsymbol{\Lambda} + \lambda\mathbf{D} \\ \mathbf{A} &\mapsto \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda}\mathbf{D}^{1/2} \end{bmatrix}. \end{aligned}$$

For instance, Sparse Pose Adjustment (SPA) proposed by Konolige et al. [2010] implements a LM optimization for 2D SLAM problems using Cholesky factorization. Kümmerle et al. [2011] presented g^{2}o using the same approach for solving 3D SLAM problems. g^{2}o is still widely used as back-end even in recent SLAM applications [Mur-Artal et al., 2015].

Dogleg

The dogleg method [Powell, 1970], computes the step by finding the point that remains inside the trust region $\|\mathbf{D}\Delta\boldsymbol{\mu}(\tau)\| \leq r$ of the polygonal path, parametrized by τ, that goes from initial point to Newton step $\Delta\boldsymbol{\mu}^*_N$ thru Cauchy step $\Delta\boldsymbol{\mu}^*_C$

$$\Delta\boldsymbol{\mu}^*_{DL}(\tau) = \begin{cases} \tau\Delta\boldsymbol{\mu}^*_C & 0 \leq \tau \leq 1 \\ \Delta\boldsymbol{\mu}^*_C + (\tau - 1)(\Delta\boldsymbol{\mu}^*_N - \Delta\boldsymbol{\mu}^*_C) & 1 \leq \tau \leq 2. \end{cases} \tag{2.44}$$

As previously stated, for big problems, the Hessian may be very expensive to be computed. Therefore, normally a Gauss-Newton step is adopted instead of Newton's. Also, any quasi-Newton method can also be used as the quadratic step if the approximated Hessian is positive definite $\mathbf{H} \succ 0$.

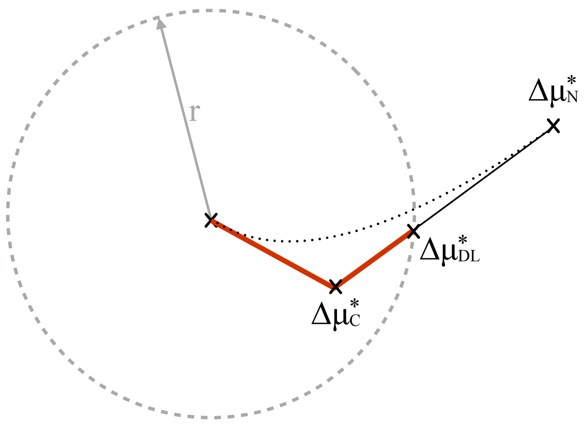


Figure 2.4: Dogleg step example. In red, the resulting step $\|\mathbf{D}\Delta\boldsymbol{\mu}^*\| = r$. In dotted black, the LM step path. The trust region is depicted in dashed gray.

In the same manner as LM, the trust region is dynamically resized depending on the ratio of the resulting and the expected residual reduction. Analogously, depending on how far is the quadratic step from the trust region, the method balances to gradient descent (see Figure 2.4). Actually, the dogleg method can be interpreted as an approximation of LM.

The dogleg method has some advantages over LM due to the polygonal parameterization. Firstly, instead of defining the trust region by the LM damping factor, it is directly defined by r which is more intuitive. Secondly, the polygonal point search avoids unsuccessful steps that can take place in LM, allowing resizing the trust region without the need of solving the problem again.

In exchange, dogleg requires to additionally compute the Cauchy step and perform the search of τ which requires some residual evaluations. Despite of the extensive use of LM in BA, dogleg has been proved to be faster in some cases [Lourakis and Argyros, 2005; Börlin and Grussenmeyer, 2013].

2.6 Discussion

SLAM is a pretty mature research topic in mobile robotics. Given the online nature of the SLAM problem, it has been solved in several different ways according to the computational power available at each time. Nowadays, it appears to be a consensus on posing SLAM as a NLS optimization problem also known as graph-based optimization or SAM.

In this chapter, the state-of-art Gauss-Newton formulation was extensively developed along with an overview of alternative SLAM methods whether used in the past or currently being proposed. These alternative methods have been focused on the convergence or the computational resources required to solve the problem. Apart from the filtering-based approaches, the totality of these alternatives are coherent with the factor graph representation and complementary to the NLS formulation.

As vastly exposed, allowing relinearization is critical in terms of accuracy of the solution. Filtering methods do not allow for relinearization since past observations are marginalized out to build a prior on the state. Despite it can significantly reduce the problem size, it leads to inconsistency of the estimation.

Regarding the Gauss-Newton convergence, any trust-region or line search method can be useful in case of poor state priors. This is the case of batch solving which is hardly the case in a mobile robotics application. However, poor priors can also take place in some applications in which no proprioceptive sensor is available such as visual odometry or visual SLAM.

Although apparently the SLAM back-end can be considered as a quite closed line of research, there are still several open issues that require further investigation.

Large scale applications reveal the same computational demands problem faced by the robotics research community in the past. For very large problems, the state-of-art methods can be computationally unfeasible making PCG an appealing alternative to be explored. But apart from approaching the problem strictly from the back-end point of view, one can also look for ways to reduce the problem size. *Sparsification* faces the problem of reducing the problem size without avoiding relinearization, i.e. without undermining the accuracy of the solution (see Chapter 4). Interestingly, graph sparsification can be performed after an accurate solution has already been found from a large and very connected graph.

Moreover, SLAM is a *passive* algorithm, that is, it does not drive the robot but only builds a map and localizes it. The extended problem of autonomously driving the robot in order to build a map while maintaining a good localization can be considered. This is called active SLAM in the literature (see Chapter 5).

Several further open issues are still under research such as data association, outlier rejection or loop closure detection, to mention a few. These are not covered in this thesis. Contrary to the common belief, in our opinion there is still a long way to go in the SLAM research to be explored in the next years.

3 Information theory overview

3.1 Introduction

Information theory quantifies and formalizes the analysis of all processes related with information (communication, storage, encryption, decoding...). Its fundamentals were established in the seminal work of Shannon [Shannon, 1948] applied to signal processing. However, information theory has been applied in several disciplines to solve a variety of problems, from neurobiology to the understanding of black holes.

Information theory is based on a simple idea: The less known some topic is, the more information one can get about it. In other words, the information content of this chapter is negligible for people who are familiar with information theory and significant for the rest. Equivalently in a more appropriate example, the same satellite-based localization measurement is highly informative in case of a coarse robot localization but it has low information content in case of having an accurate localization. Intuitively, in the words often attributed to Claude Shannon, information can be understood as the “resolution of uncertainty”.

As previously stated, several decisions and processes in localization and mapping applications require metrics to avoid simple heuristics. Information theory has been widely applied to localization and mapping problems by the robotics research community. It provides a formal manner to take decisions concerning the goodness of the robot localization or environment representation.

As described in the previous chapter, in SLAM the state is represented as a multi-variate Gaussian distribution. The following is a collection of some relevant information metrics for mobile robot localization and mapping applications and its form in the case of multi-variate Gaussian distribution.

3.2 Self-information

As previously described, information is transferred to a receiver in the degree that the receiver has no knowledge about that information a priori. For example, recalling the Monty Python's gag "Nobody expects the Spanish inquisition", each time the inquisitors appear in the scene they provide less and less information.

Therefore, the information of an event x is high if its probability $P(x)$ is low and vice-versa. The self-information metric, also called *surprisal*, measures this effect and is defined as

$$I(x) = \log \frac{1}{P(x)}. \tag{3.1}$$

Different logarithm basis are used. Base 2 is commonly used being the unit of self-information bits. In case the natural logarithm is used, the unit is nats. In this thesis and the following derivations, we use natural logarithms.

Note that self-information is well defined only in discrete probability distributions since the probability of an specific event in a continuous probability distribution is zero.

3.3 Entropy

Entropy is a key measure in information theory. It can be defined from two slightly different perspectives: the expected information content of an event from which we know its probability distribution or also the uncertainty associated to this probability distribution.

As previously stated, the more probable is an event, the less information it provides. Entropy measures the expected information of a future event given the probability distribution of the possible results. For instance, taking randomly a Scrabble tile from the bag has more entropy than casting a dice. Therefore, the more unpredictable an experiment is, the more entropy it has. In other words, entropy measures the uncertainty of an experiment, of a probability distribution.

Shannon defined the entropy H of a discrete random variable X with possible values $\{x_1, \ldots, x_n\}$ and probability mass function $P(X)$ as:

$$H(X) = E[I(X)] = E[-\log(P(X))] = \sum_{1}^{n} P(x_i) I(x_i) = -\sum_{1}^{n} P(x_i) \log P(x_i).$$

It can be extended to continuous random variable, defining entropy as in terms of its probability density function $p(X)$

$$H(X) = -\int_X p(x) \log p(x) dx.$$

For the multi-variate Gaussian distribution $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, as developed in [Ahmed and Gokhale, 1989], the entropy can be computed by

$$H(\mathbf{x}) = \log \left((2\pi e)^{\frac{n}{2}} |\boldsymbol{\Sigma}| \right), \tag{3.2}$$

being n the dimension of the state.

In mobile robotics, entropy becomes the main metric from which several uncertainty-driven methods can be derived.

3.4 Kullback-Leibler divergence

The Kullback-Leibler divergence (KLD), also called *relative entropy*, is an information metric that compares how much a probability distribution diverges from another. It was introduced in [Kullback and Leibler, 1951] and further formalized in [Kullback, 1959] where it was referred as *directed divergence*.

The KLD is well defined both in discrete and continuous probability distributions. The KLD from $p(X)$ to $q(X)$, both continuous probability density functions, is defined as

$$D_{KL}(p\|q) = \int p(x) \log \frac{p(x)}{q(x)} dx. \tag{3.3}$$

For multi-variate Gaussian distributions $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $q(\mathbf{x}) = \mathcal{N}(\breve{\boldsymbol{\mu}}, \breve{\boldsymbol{\Sigma}})$, the KLD from $p(\mathbf{x})$ to $q(\mathbf{x})$ reduces to

$$D_{KL}(p\|q) = \frac{1}{2}\Big(tr(\breve{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}) - \ln|\breve{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}| + \|\breve{\boldsymbol{\mu}} - \boldsymbol{\mu}\|^2_{\breve{\boldsymbol{\Sigma}}} - n \Big), \tag{3.4}$$

where n is the dimension of the distributions and $tr(\cdot)$ denotes the trace operator.

Note that the KLD is not symmetric ($D_{KL}(p\|q) \neq D_{KL}(p\|q)$), thus the term 'directed' used by Kullback. Furthermore, it does not accomplish the triangle inequality so it is not a distance. However, it is non-negative, it is zero if and only if both distributions are equal and last but not least: it is convex.

These properties make the KLD a useful and widely used metric in SLAM applications for maximizing the similarity of a new (simplified) distribution and the original one.

3.5 Mutual information

The mutual information (MI) between two random variables X, Y is a measure of dependence between both variables. The MI quantifies the "amount of information" obtained about X through Y and vice-versa.

For continuous random variables, it is defined as

$$I(X;Y) = \int_Y \int_X p(x,y) \frac{p(x,y)}{p(x)p(y)} dxdy. \tag{3.5}$$

Intuitively, the MI measures how similar is the joint distribution $p(X,Y)$ to the product of the two marginal distributions $p(X)p(Y)$. In fact, the MI is the KLD from the joint distribution to the product of marginal distributions, $I(X;Y) = D_{KL}(p(X,Y)\|p(X)p(Y))$. This can be seen trivially substituting the above into (3.3).

Additionally, the MI can be defined in terms of entropy as the entropy of both marginal distributions minus the joint distribution entropy

$$I(X;Y) = H(X) + H(Y) - H(X,Y). \tag{3.6}$$

In the multivariate Gaussian case, the MI of two subsets of variables $\mathbf{x}_1 \in \mathbf{x}$ and $\mathbf{x}_2 \in \mathbf{x}$ of the distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, due to (3.2) is

$$\begin{aligned}
I(\mathbf{x}_1;\mathbf{x}_2) &= \log\left((2\pi e)^{\frac{n_1}{2}}|\boldsymbol{\Sigma}_{11}|\right) + \log\left((2\pi e)^{\frac{n_2}{2}}|\boldsymbol{\Sigma}_{22}|\right) - \log\left((2\pi e)^{\frac{n_1+n_2}{2}}\begin{vmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{vmatrix}\right)\\
&= \cancel{\left(\frac{n_1}{2} + \frac{n_2}{2} - \frac{n_1+n_2}{2}\right)}\log(2\pi e) + \log|\boldsymbol{\Sigma}_{11}| + \log|\boldsymbol{\Sigma}_{22}| - \log\begin{vmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{vmatrix}\\
&= \log\frac{|\boldsymbol{\Sigma}_{11}||\boldsymbol{\Sigma}_{22}|}{\begin{vmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{vmatrix}},
\end{aligned} \tag{3.7}$$

where $\boldsymbol{\Sigma}_{11}, \boldsymbol{\Sigma}_{12}, \boldsymbol{\Sigma}_{21}$ and $\boldsymbol{\Sigma}_{22}$ are the diagonal and off-diagonal blocks of the covariance matrix $\boldsymbol{\Sigma}$ corresponding to the concerned subsets of variables.

Equivalently, in information form $\mathbf{x} \sim \mathcal{N}^{-1}(\boldsymbol{\eta}, \boldsymbol{\Lambda})$, the MI is

$$\begin{aligned}
I(\mathbf{x}_1;\mathbf{x}_2) =&\log\left((2\pi e)^{\frac{n_1}{2}}|\boldsymbol{\Lambda}_{11} - \boldsymbol{\Lambda}_{12}\boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}|^{-1}\right) + \log\left((2\pi e)^{\frac{n_2}{2}}|\boldsymbol{\Lambda}_{22} - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}|^{-1}\right)\\
&- \log\left((2\pi e)^{\frac{n_1+n_2}{2}}\begin{vmatrix}\boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12}\\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22}\end{vmatrix}^{-1}\right)\\
=&\log\frac{\begin{vmatrix}\boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12}\\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22}\end{vmatrix}}{|\boldsymbol{\Lambda}_{11} - \boldsymbol{\Lambda}_{12}\boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}||\boldsymbol{\Lambda}_{22} - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}|}\\
=&\log\frac{|\boldsymbol{\Lambda}_{11}|}{|\boldsymbol{\Lambda}_{11} - \boldsymbol{\Lambda}_{12}\boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}|} = \log\frac{|\boldsymbol{\Lambda}_{22}|}{|\boldsymbol{\Lambda}_{22} - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}|}.
\end{aligned} \tag{3.8}$$

Note that if $\mathbf{x}_1 \cup \mathbf{x}_2 \neq \mathbf{x}$, the information matrix $\boldsymbol{\Lambda}$ is the result of marginalizing out the rest of variables in $\mathbf{x}$ via the Schur complement.

In the SLAM context, the MI provides a measure of how much a subset of variables injects information about another subset. It is used in different methods for graph simplification in order to identify the most relevant correlations.

3.6 Other metrics

For further completeness, we present other relevant information metrics hereunder.

The conditional entropy $H(X|Y)$, also called *equivocation*, measures the expected information of the outcome of a random variable X given that another random variable Y is known. Not to be confused with the entropy of a conditional probability. Conditional entropy is the average of the entropy of conditional probability $H(X|Y=y)$ for all values y

that Y may take.

$$H(X|Y) = \int_Y p(y)H(X|Y = y)dy = H(X,Y) - H(Y).$$

As well as the MI, the conditional entropy measures how much two random variables are correlated. In fact, it can be expressed in terms of the MI as $H(X|Y) = H(X) - I(X;Y)$.

The cross entropy between two probability distribution $p(X)$ and $q(X)$ is defined as the weighted expectation (over $p(X)$) of the self information of $q(X)$:

$$H(p\|q) = -\int_X p(x)\log q(x)dx.$$

It can be interpreted as the expected *surprisal* when a wrong distribution $q(X)$ is assumed while the data actually follows the distribution $p(X)$. As well as the KLD, it measures how different the distributions $p(X)$ and $q(X)$ are, and it is asymmetrical. Actually, it can be defined in terms of the KLD

$$H(p\|q) = H(p(X)) + D_{KL}(p\|q).$$

Kullback and Leibler also proposed a symmetrized version of KLD

$$D_{KL}(p\|q) + D_{KL}(q\|p). \tag{3.9}$$

Actually, they referred to KLD as *directed divergence* and named the symmetrized KLD as just *divergence.*

Instead of the symmetrized KLD, due to its simpler derivation and considering its convexity, it is the KLD that has been widely adopted for maximizing the similarity of an approximation to a target probability distribution. Due to the massive use of the *directed divergence* in the literature, it finally was named in their honor and lost the *directed* adjective.

4 Information Metrics for Graph SLAM Sparsification

4.1 Introduction

One of the biggest pitfalls of graph SLAM methods is that the problem grows over time: it suffers from scalability. To tackle such growing resource demands, efforts have been invested mainly in two directions: by improving the efficiency of the algorithms, and by reducing the problem size. Despite recent improvements in algorithm efficiency (see Chapter 2), the later is still of concern, as the complexity of the solution is always linked to the size of the problem. Therefore, methods for reducing the problem size while keeping as much of its information as possible are essential, especially for large SLAM experiments.

As already explained, efficient SLAM methods take profit of two important characteristics of SLAM: sparsity and capability of relinearization. Maintaining these features is important so that a computationally efficient and accurate solution to the problem can be found.

In general, node marginalization is the only way of reducing the problem size without loss of information. However, marginalization has the disadvantage of causing loss of sparsity, increasing computational cost, and does not allow for relinearization, damping the accuracy.

In the graph-SLAM context, sparsification is the process of finding the best sparse and relinearizable approximation to the result of marginalization.

4.1.1 Related work

Several SLAM methods include mechanisms to limit the problem size growth. One of the simplest approaches consists in uniformly limiting the number of poses with respect to time or distance traveled, or to marginalize new poses close to old ones, thus growing only with the size of the area being mapped [Johannsson et al., 2013].

Information metrics can also be used to limit the size of the problem. For instance, Pose SLAM [Ila et al., 2010] only keeps new observations and robot poses if their entropy-based information content is significant. Vial et al. [2011] proposed a conservative sparsification based on Kullback-Leibler divergence (KLD) for a filter-based SLAM. The sparsification is directly performed over the information matrix instead of creating a set of new factors. Kretzschmar and Stachniss [2012] presented an information-based compression method for laser-based pose graph SLAM, in which they compute a subset of nodes containing the scans that maximize the mutual information of the map for that subset. Choudhary et al. [2015] also proposed to discard some landmarks depending on their information content using an entropy-based cost function.

Khosoussi et al. [2016] faced the issue from a different perspective resorting to graph theory. Under some assumptions, the weighted number of spanning trees of a graph approximates the determinant of the state covariance which is strongly related with its entropy (3.2). Then it can be used for measurement selection as well as graph pruning.

Different authors approache pose sparsification as a KLD minimization problem [Carlevaris-Bianco et al., 2014; Eckenhoff et al., 2016; Mazuran et al., 2016]. The best sparse approximation is the one with a minimum KLD from the dense distribution resulting of the marginalization. In case of using the simplest topology, i.e. a spanning tree, there exists a closed form for the optimal solution of all factors. However, in the majority of cases a tree topology is too simple to accurately approximate the dense result of node marginalization. For richer (more populated) topologies, an iterative optimization is needed to solve the KLD minimization. This is the focus of the work presented in this chapter.

State of the art methods [Carlevaris-Bianco et al., 2014; Eckenhoff et al., 2016; Mazuran et al., 2016] propose the use of interior point and projected quasi-Newton optimization methods to achieve sparsification. Both methods require the tuning of a set of parameters that strongly affects their convergence and robustness. Our work explores new methods that compete or outperform state-of-art methods in both accuracy and computational cost but do not require parameter tuning.

4.2 Node removal and sparsification in graph SLAM

Usually, the reduction of the SLAM problem size is approached in two different steps: node marginalization and sparsification (see Figure 4.1). These two processes can be decoupled, postponing the second one depending on the available computational resources as proposed by Eckenhoff et al. [2016].

The whole process is faced locally. Once a node is selected for removal, the local problem is constrained over the immediate surroundings of that node, i.e., the node's Markov blanket (all nodes at distance 1) and all its intra-factors (the factors involving only nodes in the Markov blanket). Optionally, this cropped problem can be solved using (2.5). Then, the new solution can be used henceforth, yielding slightly better results especially in on-line cases, according to Mazuran et al. [2016]. The selected node is marginalized via Schur complement, generating a dense information matrix $\mathbf{\Lambda}$.

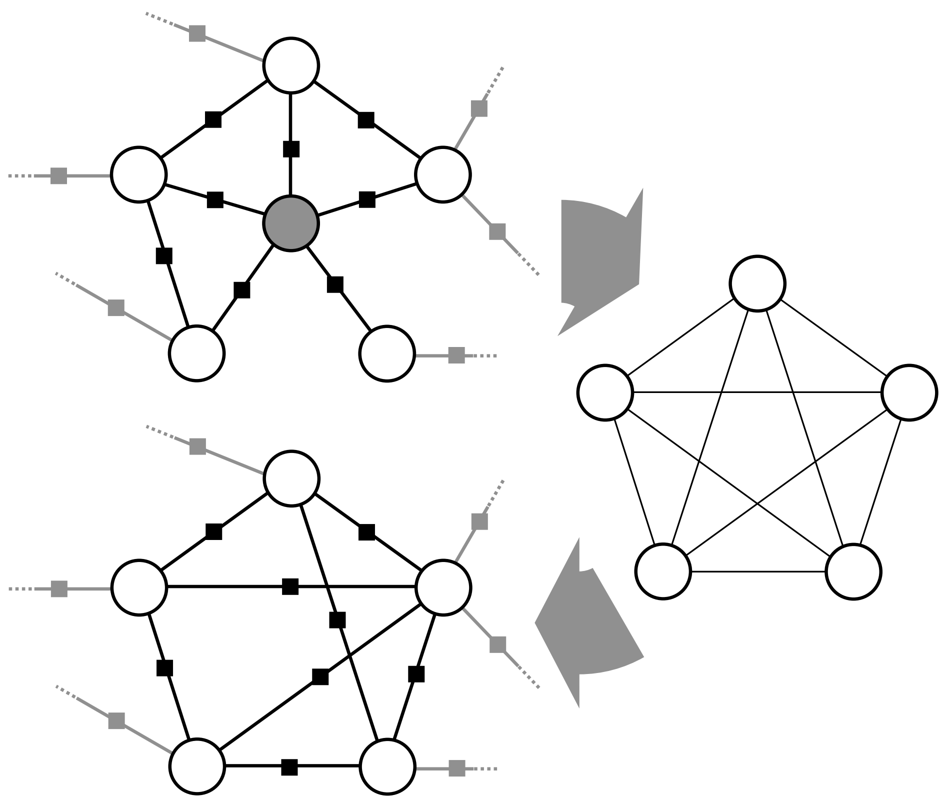


Figure 4.1: Example of marginalization and sparsification of a node (gray). After cropping a local problem, marginalization produces a dense and non-relinearizable sub-graph (right figure). This sub-graph is replaced in the original graph with a sparse approximation (bottom).

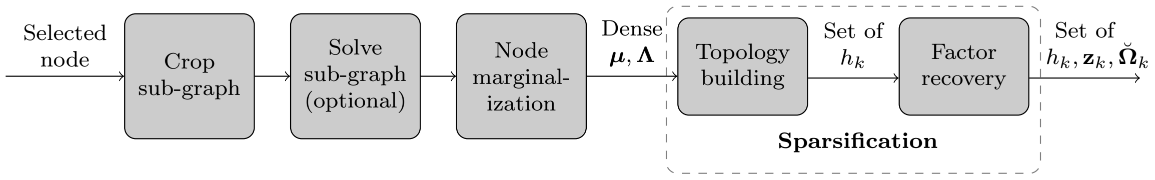

Selected node
Crop sub-graph
Solve sub-graph (optional)
Node marginal-ization
Dense $\boldsymbol{\mu}, \boldsymbol{\Lambda}$
Topology building
Set of h_k
Factor recovery
Set of $h_k, \mathbf{z}_k, \breve{\boldsymbol{\Omega}}_k$
Sparsification

Figure 4.2: Block diagram of the whole marginalization and sparsification process.

The aim of the sparsification process is to approximate the dense distribution, resulting from node marginalization, with a sparse distribution $\breve{\boldsymbol{\Lambda}}$ defined by a new set of factors. Sparsification is also split in two phases: building a topology, i.e., defining a set of new factors with their measurement model $h_k(\mathbf{x})$; and factor recovery, i.e., computing their mean $\mathbf{z}_k$ and information $\breve{\boldsymbol{\Omega}}_k$. Figure 4.2 depicts all the different processes included in marginalization and sparsification. The following sections 4.3 and 4.4 detail factor recovery and topology building processes, respectively.

4.3 Factor recovery: Kullback-Leibler Divergence minimization

Factor recovery computes the mean $\breve{\mathbf{z}}_k$ and information $\breve{\boldsymbol{\Omega}}_k$ of all new factors of a given topology. The new set of factors will produce an approximated local distribution $q(\mathbf{x}) \sim \mathcal{N}(\breve{\boldsymbol{\mu}}, \breve{\boldsymbol{\Sigma}} = \breve{\boldsymbol{\Lambda}}^{-1})$. Its information matrix $\breve{\boldsymbol{\Lambda}}$ will be fixed according to all the new factors $\breve{\boldsymbol{\Lambda}} = \sum_k \breve{\mathbf{J}}_k^\top \breve{\boldsymbol{\Omega}}_k \breve{\mathbf{J}}_k$, being $\breve{\mathbf{J}}_k$ the residuals Jacobian of each new factor.

We look for those mean $\breve{\mathbf{z}}_k$ and information $\breve{\mathbf{\Omega}}_k$ values that make the new approximated distribution $q(\mathbf{x})$ most similar to the dense local distribution $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma} = \mathbf{\Lambda}^{-1})$ resulting after node marginalization. As introduced in Chapter 3, the Kullback-Leibler Divergence (KLD) measures how much a distribution diverges from another. Hence, factor recovery can be posed as finding the measurement mean $\breve{\mathbf{z}}_k$ and information $\breve{\mathbf{\Omega}}_k$ of all new factors of a given topology that minimize the KLD of the sparse approximated distribution $q(\mathbf{x})$ from the dense distribution $p(\mathbf{x})$. Recalling the KLD for multivariate Gaussian distributions (3.4), the problem can be posed as

$$\begin{aligned}
\{\breve{\mathbf{z}}^*_k, \breve{\mathbf{\Omega}}^*_k\}_{\forall k} &= \underset{\{\breve{\mathbf{z}}_k, \breve{\mathbf{\Omega}}_k\}_{\forall k}}{\arg\min}\ D_{KL}(p\|q) \\
&= \underset{\{\breve{\mathbf{z}}_k, \breve{\mathbf{\Omega}}_k\}_{\forall k}}{\arg\min}\ \frac{1}{2}\Big(tr(\breve{\mathbf{\Lambda}}\mathbf{\Sigma}) - \ln|\breve{\mathbf{\Lambda}}\mathbf{\Sigma}| + \|\breve{\boldsymbol{\mu}} - \boldsymbol{\mu}\|^2_{\breve{\mathbf{\Lambda}}^{-1}} - d\Big), \qquad (4.1)
\end{aligned}$$

The Mahalanobis norm term $\|\breve{\boldsymbol{\mu}} - \boldsymbol{\mu}\|^2_{\breve{\mathbf{\Lambda}}^{-1}}$ is the only one that (indirectly) depends on the new factors mean $\breve{\mathbf{z}}_k$. Therefore, it is convenient to set each new factor's mean as the expected measurement considering the dense distribution mean $\breve{\mathbf{z}}^*_k = h_k(\boldsymbol{\mu})$. Doing this, the resulting approximated distribution mean becomes equal to the dense one, $\breve{\boldsymbol{\mu}} = \boldsymbol{\mu}$, and the Mahalanobis norm term $\|\breve{\boldsymbol{\mu}} - \boldsymbol{\mu}\|^2_{\breve{\mathbf{\Lambda}}^{-1}}$ becomes null. This sets the optimal solution for all new factors' means $\breve{\mathbf{z}}^*_k$.

To find all factors' information $\breve{\mathbf{\Omega}}^*_k$, the rest of the expression can be simplified as follows

$$\{\breve{\mathbf{\Omega}}^*_k\}_{\forall k} = \underset{\{\breve{\mathbf{\Omega}}_k\}_{\forall k}}{\arg\min}\ \frac{\cancel{1}}{\cancel{2}}\Big(tr(\breve{\mathbf{\Lambda}}\mathbf{\Sigma}) - \underbrace{\ln|\breve{\mathbf{\Lambda}}\mathbf{\Sigma}|}_{\ln|\breve{\mathbf{\Lambda}}| + \cancel{\ln|\mathbf{\Sigma}|}} - \cancel{d}\Big),$$

where constant terms w.r.t. all new factors' information matrices $\breve{\mathbf{\Omega}}_k$ have been deleted. The information matrix of the approximate distribution can also be expressed as $\breve{\mathbf{\Lambda}} = \breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}}$, being

$$\breve{\mathbf{\Omega}} = \begin{bmatrix} \ddots & & \\ & \breve{\mathbf{\Omega}}_k & \\ & & \ddots \end{bmatrix} \quad \text{and} \quad \breve{\mathbf{J}} = \begin{bmatrix} \vdots \\ \breve{\mathbf{J}}_k \\ \vdots \end{bmatrix}.$$

With all these considerations, the factor recovery solution (3.3) can be written as

$$\begin{aligned}
\breve{\mathbf{z}}^*_k &= h_k(\boldsymbol{\mu}), \quad \forall k \\
\breve{\mathbf{\Omega}}^* &= \underset{\breve{\mathbf{\Omega}}}{\arg\min}\ tr(\breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}} \mathbf{\Sigma}) - \ln|\breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}}|. \\
&\quad \text{s.t.} \quad \breve{\mathbf{\Omega}} \succ 0 \qquad (4.2)
\end{aligned}$$

Since finding all $\breve{\mathbf{z}}^*_k$ is trivial, in the following we are only concerned with finding the optimal information matrix $\breve{\mathbf{\Omega}}^*$.

Note that in order to evaluate the Kullback-Leibler divergence as well as the posterior formulation derived above, the covariance matrix of the dense distribution $\mathbf{\Sigma}$ is required.

However, in some cases such as in original SLAM problems containing only relative measurements, the dense problem resulting from cropping and node marginalization has a rank-deficient information matrix $\mathbf{\Lambda}$, so the covariance matrix $\mathbf{\Sigma}$ is singular. In such case, the KLD is evaluated by projecting both distributions $p(\mathbf{x})$ and $q(\mathbf{x})$ onto a sub-space in which the covariance is well defined. In other words, any projection $\mathbf{\Lambda} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ such that $\mathbf{D}$ is invertible can be used. Then, the KLD minimization in (4.2) is performed in the projected sub-space by substituting

$$\breve{\mathbf{J}} \mapsto \breve{\mathbf{J}}\mathbf{U}, \quad \mathbf{\Sigma} \mapsto \mathbf{D}^{-1}. \tag{4.3}$$

To obtain the projection, one can re-parametrize the problem to relative poses w.r.t an arbitrarily chosen node [Eckenhoff et al., 2016; Carlevaris-Bianco et al., 2014] or use a rank-revealing eigen decomposition [Mazuran et al., 2016]. The first solution is applicable to the pose-graph SLAM case while the latter is generic.

4.3.1 Factor recovery in closed form

According to Mazuran et al. [2016], when the stacked Jacobian $\breve{\mathbf{J}}$ is invertible, the solution to (4.2) can be obtained by imposing a null derivative w.r.t. all factor information matrices,

$$\breve{\mathbf{\Omega}}_k = (\breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1}. \tag{4.4}$$

This is the case, for instance, of the spanning tree topology in pose-graph SLAM using the projection (4.3). The factor recovery is very efficient in this case since it has a closed form solution. However, the tree topology is normally too sparse to accurately approximate the exact dense distribution.

4.3.2 Factor recovery via iterative optimization

Other more populated topologies do not admit a closed form solution for all factors, and (4.2) has to be solved using iterative optimization. The state-of-the-art literature [Carlevaris-Bianco et al., 2014; Eckenhoff et al., 2016; Mazuran et al., 2016] proposes two different optimization methods for the factor recovery problem: Interior Point and Limited-memory Projected Quasi-Newton (PQN) [Schmidt et al., 2009].

Interior point

In the Interior Point method (IP), the positive definiteness constraint is included in the cost function $c(\rho)$ as a log barrier,

$$c(\rho) = tr(\breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}} \mathbf{\Sigma}) - \ln|\breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}}| - \rho \ln|\breve{\mathbf{\Omega}}|. \tag{4.5}$$

A stricter constraint can be applied instead of the log barrier term to also guarantee conservativeness: $\rho \ln|\mathbf{\Lambda} - \breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}}|$ as in [Eckenhoff et al., 2016].

The IP method consists of two nested loops. For each value of the log barrier parameter ρ_i, the resulting problem (4.5) is solved in the inner loop

$$\breve{\mathbf{\Omega}}^{(i)} = \arg\min_{\breve{\mathbf{\Omega}}} c(\rho_i).$$

After inner loop convergence, the outer loop decreases the log barrier parameter, $\rho_{i+1} = \alpha\rho_i$. The outer loop ends after inner loop convergence when ρ_i is 'close enough' to 0.

The inner loop can be solved with Newton's method using the gradient and Hessian of (4.5) provided by Mazuran et al. [2016]:

$$\frac{\partial c(\rho)}{\partial \breve{\mathbf{\Omega}}_k} = \breve{\mathbf{J}}_k\big(\mathbf{\Sigma} - (\breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}})^{-1}\big)\breve{\mathbf{J}}_k^\top - \rho\,\breve{\mathbf{\Omega}}_k^{-1} \tag{4.6}$$

$$\frac{\partial^2 c(\rho)}{\partial \breve{\Omega}_{ij}\,\partial \breve{\mathbf{\Omega}}_k} = \breve{\mathbf{J}}_k(\breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}})^{-1}\breve{\mathbf{J}}^\top \delta_{ij}\,\breve{\mathbf{J}}(\breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}})^{-1}\breve{\mathbf{J}}_k^\top + \rho\,\breve{\mathbf{\Omega}}_k\,\delta_{ij}\,\breve{\mathbf{\Omega}}_k \tag{4.7}$$

where $\breve{\Omega}_{ij}$ is the (i,j) entry of $\breve{\mathbf{\Omega}}$, δ_{ij} is a matrix of zeros except a one in (i,j). Note that to build the gradient $\nabla_{c(\rho)}$ from (4.6) its entries must be stacked into a vector. To do that, an order should be established only including the block diagonal entries of $\breve{\mathbf{\Omega}}$ corresponding to each factor information $\breve{\mathbf{\Omega}}_k$. Furthermore, due to the matrix symmetry only upper or lower triangular entries have to be included.

The tuning of both parameters α and ρ_0, together with the inner loop's end conditions, strongly affect the IP convergence and robustness. To ensure that the positive definite constraint is satisfied, the contributions to the gradient and the Hessian corresponding to the KLD and the log barrier terms have to be balanced. Relaxing the inner loop end conditions or enhancing the decrease factor α lead to a lower contribution of the log barrier. This speeds up the method but may converge to a non positive definite result since the quadratic step 'jumps over' the log barrier. Therefore, tuning the IP parameters is a trade off between convergence velocity and robustness to divergence. Worse, this tuning is oftentimes problem-dependent.

Limited-memory Projected Quasi-Newton

In the Limited-memory Projected Quasi-Newton (PQN) method, the positive definiteness constraint is imposed along the line search through the projection of $\breve{\mathbf{\Omega}}$ onto the positive semi-definite subspace. This projection consists in setting all negative eigenvalues to zero. As with all quasi-Newton methods, PQN does not require the computation and the inversion of the Hessian, but it evaluates the cost function (4.2) several times at each iteration during line search. Since the Markov Blanket is of small size, the cost of evaluating (4.2) is not significantly lower than the Hessian computation and inversion performed in IP. For this reason, IP greatly outperforms PQN in terms of convergence time.

Initial guess

An initial guess in case of relative measurements was proposed by Mazuran and Tipaldi [2014] based on the off-diagonal blocks of the dense information matrix.

$$\breve{\mathbf{\Omega}}_k = \mathbf{J}_{k_1}^{-\top}\mathbf{\Lambda}_{k_1,k_2}\mathbf{J}_{k_2}^{-1} \tag{4.8}$$

being k_1, k_2 the two nodes involved in the factor k (so $\mathbf{J}_{k_1}, \mathbf{J}_{k_2}$ are the non-zero blocks of $\mathbf{J}_k$) and being $\mathbf{\Lambda}_{k_1,k_2}$ the off-diagonal blocks corresponding to the involved nodes. Since (4.8) is normally not symmetric and may be non positive semi-definite, its closest symmetric positive semi-definite approximation [Higham, 1988] should be taken instead. Such initial guess is usually much better than the identity matrix often used by default as will be seen in the results in Section 4.6.

Moreover, and contrary to what is stated in [Mazuran et al., 2016], this initial guess can be suited to IP with the appropriate initial log-barrier parameter ρ_0. For this, the gradient of $c(\rho)$ (4.6) can be split in two terms, the KLD term and the positive definite constraint term

$$\nabla_{c(\rho)} = \nabla_{KLD} + \rho\nabla_{PD}.$$

To balance their respective contributions we propose to take a weighted ratio between both terms' norms, obtaining a warm start

$$\rho_0 = \omega\frac{\|\nabla_{KLD}\|}{\|\nabla_{PD}\|}.$$

4.3.3 Factor recovery with Factor Descent

Inspired in block-coordinate descent methods, we propose an iterative method for factor recovery called Factor Descent (FD). Each iteration consists in solving for a block of variables (those defining one factor's information matrix $\breve{\mathbf{\Omega}}_k$) while fixing the rest. The optimal solution in the factor's subspace is computed analytically, that is, there is no fitting to any linear or quadratic function.

Solving (4.2) only for the k-th factor leads to

$$\begin{aligned}\breve{\mathbf{\Omega}}_k^* = \arg\min_{\breve{\mathbf{\Omega}}_k} & \; tr(\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top\breve{\mathbf{\Omega}}_k\breve{\mathbf{J}}_k\mathbf{\Sigma}) - \ln|\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top\breve{\mathbf{\Omega}}_k\breve{\mathbf{J}}_k| \\ \text{s.t.} & \quad \breve{\mathbf{\Omega}}_k \succ 0\end{aligned} \tag{4.9}$$

where $\breve{\mathbf{\Upsilon}}_k$ is the information matrix constituted by the rest of the factors in the topology,

$$\breve{\mathbf{\Upsilon}}_k = \sum_{i\neq k}\breve{\mathbf{J}}_i^\top\breve{\mathbf{\Omega}}_i\breve{\mathbf{J}}_i \; . \tag{4.10}$$

Descent of the KLD cost is achieved factor by factor, and hence the Factor Descent name. After solving (4.9) for one factor, we iterate over all the factors until convergence.

Some similarities with block-relaxation methods can be established since they also compute the solution of a subset of variables leaving the rest fixed. However, while relaxation methods solve linear problems, FD directly finds the optimal solution of the non-linear problem (4.2) for one factor when the rest of factors are fixed.

Considering all factors other than k are fixed, the optimal $\breve{\mathbf{\Omega}}_k$ can be computed analytically by finding the null derivative of (4.9),

$$\frac{\partial D_{KL}(p\|q)}{\partial\breve{\mathbf{\Omega}}_k} = \frac{\partial\Big(tr(\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top\breve{\mathbf{\Omega}}_k\breve{\mathbf{J}}_k\mathbf{\Sigma}) - \ln|\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top\breve{\mathbf{\Omega}}_k\breve{\mathbf{J}}_k|\Big)}{\partial\breve{\mathbf{\Omega}}_k} = 0. \tag{4.11}$$

This can be done in different manners depending on the particular properties of $\breve{\mathbf{\Upsilon}}_k$ and $\breve{\mathbf{J}}_k$. Consider the following propositions (see proofs in the Appendix A).

Proposition 1 *If $\breve{\mathbf{\Lambda}}$ is invertible and $\breve{\mathbf{J}}_k$ is full rank, the derivative (4.11) becomes null in*

$$\breve{\mathbf{\Omega}}_k = (\breve{\mathbf{J}}_k\mathbf{\Sigma}\breve{\mathbf{J}}_k^\top)^{-1} - L^{-\top}\mathbf{Q}_L\big(\breve{\mathbf{\Upsilon}}_k - \breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top(\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top)^{-1}\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\big)\mathbf{Q}_L^\top L^{-1} \tag{4.12}$$

being the LQ-decomposition[(i)] *of* $\breve{\mathbf{J}}_k = \mathbf{LQ} = \begin{bmatrix} L & 0 \end{bmatrix}\begin{bmatrix}\mathbf{Q}_L \\ \mathbf{Q}_0\end{bmatrix}$.

Proposition 2 *If $\breve{\mathbf{\Upsilon}}_k$ is invertible and $\breve{\mathbf{J}}_k$ is full rank, the derivative (4.11) becomes null in*

$$\breve{\mathbf{\Omega}}_k = (\breve{\mathbf{J}}_k\mathbf{\Sigma}\breve{\mathbf{J}}_k^\top)^{-1} - (\breve{\mathbf{J}}_k\breve{\mathbf{\Upsilon}}_k^{-1}\breve{\mathbf{J}}_k^\top)^{-1}. \tag{4.13}$$

Furthermore, it can be efficiently solved using the Cholesky decomposition of $\breve{\mathbf{\Upsilon}} = \mathbf{R}^\top\mathbf{R}$

$$\breve{\mathbf{\Omega}}_k = (\breve{\mathbf{J}}_k\mathbf{\Sigma}\breve{\mathbf{J}}_k^\top)^{-1} - (\mathbf{\Gamma}\mathbf{\Gamma}^\top)^{-1}, \tag{4.14}$$

where $\mathbf{\Gamma} = \breve{\mathbf{J}}_k\mathbf{R}^{-1}$ is directly obtained by back substitution.

Proposition 3 *If $\breve{\mathbf{\Lambda}}$ is invertible, $\breve{\mathbf{J}}_k$ is full rank and $nul(\breve{\mathbf{\Upsilon}}_k) = rank(\breve{\mathbf{J}}_k)$, the derivative (4.11) becomes null in*

$$\breve{\mathbf{\Omega}}_k = (\breve{\mathbf{J}}_k\mathbf{\Sigma}\breve{\mathbf{J}}_k^\top)^{-1}. \tag{4.15}$$

Prop. 1 applies to all cases, if one takes care to project the sub-graph with (4.3) if necessary, but its computation is more expensive than the other two propositions. Prop. 2 applies to the cases where the sub-graph with the current k factor removed would still be full rank. Otherwise, when the rank loss after removing the k factor equals the factor's degrees of freedom, Prop. 3 applies.

Cases where neither Prop. 2 nor Prop. 3 apply, and therefore we must resort to Prop. 1, include *e.g* landmarks observed from two monocular views (removing one view's factor renders the landmark's depth unobservable) or constrained IMU motion factors (removing the constraining factors renders the IMU biases unobservable). In SLAM problems such that all

(i) LQ-decomposition of a full row-rank matrix is equivalent to the QR-decomposition of its transposed.

Algorithm 1 Factor descent method

```
Input: Dense mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, topology $\mathcal{T} = \{\dots, h_k(), \dots\}$
Output: All factors' mean $\breve{\mathbf{z}}_k$ and information $\breve{\boldsymbol{\Omega}}_k$
  // Precompute constant variables
  for $h_k \in \mathcal{T}$ do
    $\mathbf{J}_k \leftarrow evaluateJacobian(h_k, \boldsymbol{\mu})$
    $\boldsymbol{\Phi}_k \leftarrow (\breve{\mathbf{J}}_k \boldsymbol{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1}$
    $\breve{\mathbf{z}}_k \leftarrow h_k(\boldsymbol{\mu})$
  end for
  $\mathcal{K}_{CF} \leftarrow \emptyset$ // Closed form factors' indexes
  while not $endConditions()$ do
    // Avoid recomputation of closed form factors
    $k = nextFactor()$
    while $k \in \mathcal{K}_{CF}$ do
      $k = nextFactor()$
    end while
    // Factor descent
    $\breve{\boldsymbol{\Upsilon}}_k \leftarrow \sum_{i \neq k} \breve{\mathbf{J}}_i^\top \breve{\boldsymbol{\Omega}}_i \breve{\mathbf{J}}_i$
    switch $rk(\breve{\boldsymbol{\Upsilon}})$
      case $dim(\breve{\boldsymbol{\Upsilon}})$ // Proposition 2
        $\breve{\boldsymbol{\Omega}}_k \leftarrow \boldsymbol{\Phi}_k - (\breve{\mathbf{J}}_k \breve{\boldsymbol{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top)^{-1}$
      case $dim(\breve{\boldsymbol{\Upsilon}}) - rk(\breve{\mathbf{J}}_k)$ // Proposition 3
        $\breve{\boldsymbol{\Omega}}_k \leftarrow \boldsymbol{\Phi}_k$
        $\mathcal{K}_{CF} \leftarrow k$ // Store closed form factors to save iterations
      default // Proposition 1
        $\breve{\boldsymbol{\Omega}}_k \leftarrow \boldsymbol{\Phi}_k - L^{-\top} \mathbf{Q}_L \big(\breve{\boldsymbol{\Upsilon}}_k - \breve{\boldsymbol{\Upsilon}}_k \mathbf{Q}_0^\top (\mathbf{Q}_0 \breve{\boldsymbol{\Upsilon}}_k \mathbf{Q}_0^\top)^{-1} \mathbf{Q}_0 \breve{\boldsymbol{\Upsilon}}_k \big) \mathbf{Q}_L^\top L^{-1}$
    end switch
    // Ensure positive definite solution
    if $\breve{\boldsymbol{\Omega}}_k \preceq 0$ then
      $\mathbf{V}, \boldsymbol{\lambda} \leftarrow eigenDecomposition(\breve{\boldsymbol{\Omega}}_k)$
      $\breve{\boldsymbol{\Omega}}_k \leftarrow \mathbf{V} diag(max(\epsilon, \boldsymbol{\lambda})) \mathbf{V}^\top$
    end if
  end while
```

nodes are of the same dimension and all factors correspond to relative measurements (such as pose-graph SLAM), either Prop. 2 or 3 are applicable.

The overall factor descent method is described in Alg. 1. Since the first term $(\breve{\mathbf{J}}_k \boldsymbol{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1}$ of all three solutions (4.12), (4.14) and (4.15) does not depend on the rest of factors, it can be computed only once at the beginning of the algorithm. This term can be interpreted as the projection, onto the k-th factor's measurement space, of the information of the dense distribution resulting from node marginalization. Analogously, the second term in both cases is the projection, onto the measurement space of the k-th factor, of the information of the rest of factors.

In case of a rank-deficient dense distribution $\boldsymbol{\Lambda}$, the projection (4.3) can be applied as well. Note that the information matrix corresponding to the rest of new factors must be projected as well

$$\breve{\boldsymbol{\Upsilon}}_k \mapsto \mathbf{U}^\top \breve{\boldsymbol{\Upsilon}} \mathbf{U}.$$

In this case, assuming that $\breve{\mathbf{\Lambda}}$ is invertible is equivalent to assuming that the rank is not decreased, $rk(\breve{\mathbf{\Lambda}}) = rk(\mathbf{\Lambda})$. Also, a full rank Jacobian $\breve{\mathbf{J}}_k$ only implies linear independence of all measurement elements. With these considerations, we can ensure that all assumptions are taken without loss of generality.

Closed form factors

Prop. 3 applies in those cases where the second term of (4.12) is null. That is, in those factors whose Jacobians $\breve{\mathbf{J}}_k$ are linearly independent of all the rest of factors' Jacobians of the topology. Therefore, the projection onto the measurement space of the k-th factor, of the information of the rest of factors is null.

In FD, factors in which Prop. 3 applies are only solved once, since the solution does not depend on the rest of the factors and hence is constant. The method, then, iterates over the rest of the topology.

Note that the solution (4.15) is exactly the same as (4.4) applicable in case of an invertible stacked Jacobian $\breve{\mathbf{J}}$ as demonstrated by [Mazuran et al., 2016]. Trivially, a topology such that the Jacobian of each factor is independent from the rest (Prop. 3) produces an invertible stacked Jacobian.

However, in a more general case, in some topologies Prop. 3 may apply to just some factors. In that case, these factors can be solved via closed form (4.4) and the iterative factor recovery (no matter which method is used) has to be performed only for the rest of factors. Therefore, we want to emphasize that Prop. 3 extends the cases demonstrated by Mazuran et al. [2016], in which the the closed form solution (4.4) is applicable.

Positive-definiteness

The solutions (4.12) and (4.14) are based on the KLD derivatives, and no positive definiteness constraint is applied. Therefore, the result may be a non positive definite solution if the second term of (4.12) or (4.14) is larger than the first term. That is, if the projection of the information of the rest of factors has a larger information content than the projection of the dense distribution. In other words, when the approximation made by the rest of factors $\breve{\mathbf{\Upsilon}}_k$ is not conservative in some direction, the optimal k-factor would subtract this excess of information, leading to a negative eigenvalue of $\breve{\mathbf{\Omega}}_k$. In this case, we set all negative eigenvalues to a small positive value ϵ.

Non-cyclic Factor Descent

Factor Descent iterates over all factors cyclically. Clearly, the order in which the factors are optimized can be altered to our benefit. To improve convergence, we propose selecting at each step the factor that will decrease the KLD the most. To find it, we make use of the gradient of the KLD w.r.t. each non-zero element of $\breve{\mathbf{\Omega}}$ (thus, each element of each $\breve{\mathbf{\Omega}}_k$)

$$\frac{\partial D_{KL}(p\|q)}{\partial \breve{\mathbf{\Omega}}_k} = \mathbf{J}_k\big(\mathbf{\Sigma} - (\breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}})^{-1}\big)\breve{\mathbf{J}}_k^\top. \tag{4.16}$$

The gradient vector is formed by different segments corresponding to all the factors. Each one of these vector segments is built appending the upper (or lower) triangular entries of the matrix (4.16). We select the factor with the largest corresponding gradient segment norm as the one that would reduce the KLD the most.

4.4 Topology

Whichever factor recovery method is used to solve (4.2), its result is only the best approximation that can be achieved given the topology used. But different topologies produce different approximations. Therefore, the design of the topology is critical in terms of the accuracy of the sparsification approximation. The accuracy of the solution depends on how much the selected topology can explain the dense distribution.

We focus on the graph SLAM problems in which all factors correspond to relative measurements between pairs of nodes. For simplicity, we consider that all nodes are of the same dimension. As previously stated, the simplest topology using relative measurements is a spanning tree. The Chow-Liu tree (CLT) defines the tree topology that can encode more information from the dense distribution. To do that, it recalls on mutual information (MI) between all pairs of nodes in order to measure which nodes are more correlated.

However, both MI formulations (3.7) and (3.8) require a non-singular dense information matrix $\mathbf{\Lambda}$. Precisely, as previously stated, in graphs containing exclusively relative measurements the resulting dense information matrix is singular. The main objective of computing the MI between pairs of nodes is measuring how a potential factor between each pair would explain the original dense distribution. Then, working on a projected sub-space as in section 4.3 is not a suitable solution since we lose the track of each node.

Carlevaris-Bianco et al. [2014] compute the MI of each pair of nodes using the information form (3.8). Since the dense information matrix $\mathbf{\Lambda}$ is singular, to prevent from null determinants a Tikhonov regularization is used within all determinants

$$\mathbf{I}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} \log \frac{|\tilde{\mathbf{\Lambda}}_{ii} + \epsilon \mathbf{I}|}{|\tilde{\mathbf{\Lambda}}_{ii} - \tilde{\mathbf{\Lambda}}_{ij} \tilde{\mathbf{\Lambda}}_{jj}^{-1} \tilde{\mathbf{\Lambda}}_{ji} + \epsilon \mathbf{I}|}. \tag{4.17}$$

Remember that, as explained in Ch. 3, in order to obtain $\tilde{\mathbf{\Lambda}}_{ii}, \tilde{\mathbf{\Lambda}}_{ij}$ and $\tilde{\mathbf{\Lambda}}_{jj}$, all variables different from i and j should be marginalized out from $\mathbf{\Lambda}$ via Schur complement.

Conversely, Mazuran et al. [2016] compute the MI using the covariance form (3.7). But first, the Tikhonov regularization is performed to get an approximation of the covariance matrix $\hat{\mathbf{\Sigma}} = (\mathbf{\Lambda} + \epsilon \mathbf{I})^{-1}$,

$$\mathbf{I}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} \log \frac{|\hat{\mathbf{\Sigma}}_{ii}||\hat{\mathbf{\Sigma}}_{jj}|}{\begin{vmatrix} \hat{\mathbf{\Sigma}}_{ii} & \hat{\mathbf{\Sigma}}_{ij} \\ \hat{\mathbf{\Sigma}}_{ji} & \hat{\mathbf{\Sigma}}_{jj} \end{vmatrix}}. \tag{4.18}$$

Note that (4.17) and (4.18) are not strictly equivalent since the regularization is performed in different places. No matter how the MI is computed, the CLT tree is built by incrementally taking the pair of nodes with largest MI that do not create a cycle.

However, even being the most informative tree, since a tree topology is the sparsest topology, CLT is usually too simple to approximate the original distribution. For this reason, the so-called sub-graph (SG) topology is proposed by Mazuran et al. [2016]. It is built departing from the CLT and complementing it with more factors, also based on the already computed MI values.

Alternatively, the cliquey topology [Mazuran et al., 2016] takes the CLT and converts pairs of independent factors into one single factor by correlating them. However, it implies breaking the homogeneity of factors creating factors that involve more than two nodes. We propose to use only relative pose measurement factors to maintain the pose-graph SLAM nature.

Differently to the CLT-based methods, Eckenhoff et al. [2016] proposed an ℓ_1-regularized KLD minimization to compute the topology that will encode the most information

$$\breve{\mathbf{\Lambda}}^* = \arg\min_{\breve{\mathbf{\Lambda}}} tr(\breve{\mathbf{\Lambda}}\mathbf{\Sigma}) - \ln|\breve{\mathbf{\Lambda}}| + \lambda\|\breve{\mathbf{\Lambda}}\|_1. \tag{4.19}$$

Then, the topology is built setting relative measurement factors according to the significant off-diagonal blocks of the resulting $\breve{\mathbf{\Lambda}}$.

4.4.1 Topology population

We call population to the number of factors that are contained in a sub-graph. Determining the topology population K is a compromise between sparsity and accuracy. More populated topologies achieve better approximations, however they undermine the sparsity of the resulting graph and also solving the factor recovery becomes computationally more expensive. While using tree topologies allows the use of closed form factor recovery solution, complementing it with some more factors (SG) leads to constrained iterative optimization. Moreover, all mentioned iterative factor recovery methods become slower as more populated are the topologies. While the gradient and Hessian gets larger (IP, PQN), factor descent has more factors to iterate over.

A policy to fix the population of the topology should be adopted. The topology must satisfy two conditions: It has to connect all the nodes, and there should not be two factors between the same pair of nodes since it would be redundant. Thus the topology population K, i.e., the number of factors, for a given Markov blanket of size n should be in the interval

$$K \in \left[n-1, \tfrac{n(n-1)}{2}\right]. \tag{4.20}$$

In [Mazuran et al., 2016] the population is fixed proportionally to the minimum population, i.e. the spanning tree population

$$K = \gamma(n-1), \quad \text{with} \quad \gamma \geq 1. \tag{4.21}$$

Then, while for small Markov blankets the result is quite dense, for bigger ones it becomes very sparse.

In [Eckenhoff et al., 2016], a threshold is used to consider *significant* the resulting off-diagonal values of (4.19). However, the magnitude of the information matrix entries depends on the noise of the original measurements, so the tuning of this threshold is highly problem dependant.

A sparsification application has two opposite aims: sparsity and accuracy of the simplified graph. For this reason, we propose a new population policy proportional to the total amount of possible factors,

$$K = \alpha \frac{n(n-1)}{2} \quad \text{with} \quad \alpha \in (0, 1]. \tag{4.22}$$

The parameter α fixes the fill-in ratio of the resulting information matrix. It is a more intuitive parameter to be tuned by the end user depending on the application specifications. Also, since its quadratic form, the fill-in ratio adapts better than tree-proportional to very different Markov blanket sizes.

Note that both tree-proportional and fill-in policies can provide populations out of the allowed range (4.20). Then, the resulting population of (4.21) and (4.22) must be clipped to an admissible value.

4.4.2 Building populated topologies

As the previous sections formulation shows, the factors of a populated topology affect each other in the computation of the optimal solution. Therefore, it is important to remark that building a sub-graph topology by incrementally taking the most informative factor does not have to produce the most informative sub-graph topology. In consequence, despite being CLT the most informative tree, it does not mean that the most informative sub-graph must contain it. However, optimally solving this combinatorial problem would require significant computational effort.

Three new alternative methods to build an informative topology are proposed below. Figure 4.3 shows an example of the different factor recovery results using the CLT topology, the three topology methods proposed and the best topology found by brute-force search. The information matrix density after node marginalization can be observed as well as how CLT is too simple to approximate it.

Downdated mutual information

As introduced early, the mutual information (MI) between pairs of nodes is a useful metric to measure how much a factor relating those nodes could explain the dense distribution. However, once the CLT is built, the MI between the rest of pairs of nodes computed either using (4.17) or (4.18) may not be a reliable metric about the amount of information that each factor candidate will be able to encode anymore. In other words, the added CLT factors can already explain some of these correlations.

To tackle this, our first proposal is a variation of the MI-based method to complement the CLT. First the CLT is computed in the same manner, using the MI. Afterwards, instead of directly complementing the CLT with the subsequent MI pairs of nodes, the MI of the remaining pairs of nodes are recomputed taking into account the already added CLT factors.

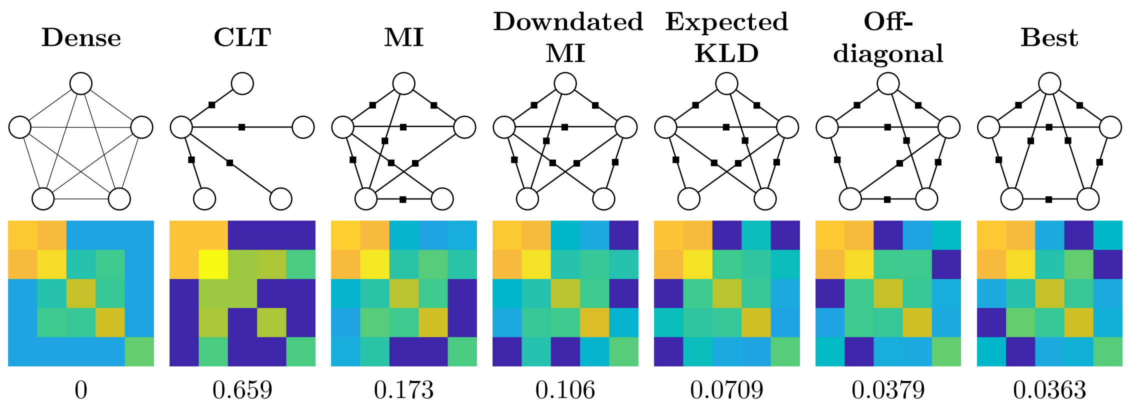


Figure 4.3: Example of factor recovery results using different topology building methods (top). In the middle row, the information pattern of $\breve{\mathbf{\Lambda}}$ (or $\mathbf{\Lambda}$ in the first case), colors depict the log absolute values. In the bottom, the KLD of each factor recovery solution from the dense distribution.

To do that, before recomputing these MIs, we perform a downdate on the regularized covariance $\hat{\mathbf{\Sigma}}$ with the information of all CLT factors. Since we are working in the covariance space, we can rely on the Kalman equations to find an optimal value for the covariance increment,

$$\Delta\hat{\mathbf{\Sigma}} = \sum_{j\in CLT} \hat{\mathbf{\Sigma}}\breve{\mathbf{J}}_j^\top(\breve{\mathbf{\Omega}}_j^{-1} + \breve{\mathbf{J}}_j\hat{\mathbf{\Sigma}}\breve{\mathbf{J}}_j^\top)^{-1}\breve{\mathbf{J}}_j\hat{\mathbf{\Sigma}}. \tag{4.23}$$

Note that equation (4.23) is obtained by substituting the Kalman gain (2.29) into the covariance update (2.28), for each factor of the CLT, and then adding them all up. Contrary to the Kalman update, however, this increment is added to the covariance, removing information and thus constituting a *downdate*, $\hat{\mathbf{\Sigma}} \leftarrow \hat{\mathbf{\Sigma}} + \Delta\hat{\mathbf{\Sigma}}$.

To evaluate (4.23), all CLT factors information matrices $\breve{\mathbf{\Omega}}_j$ are required. Therefore and ideally, factor recovery for all CLT factors should be performed. However, the factor recovery solution for these factors depends on the rest of the factors that will be finally added to the topology, which have not been decided yet. To resolve the situation, we resort to estimating the final CLT factors information $\breve{\mathbf{\Omega}}_j$ as they would be without complementing with more factors. Hence, we approximate each $\breve{\mathbf{\Omega}}_j$ with the closed form factor recovery (4.4).

After downdating the CLT estimated contribution to the regularized covariance $\hat{\mathbf{\Sigma}}$, the MI of the remaining pairs of nodes can be computed again. Hence, we named it downdated mutual information. It provides a measure of where there is still information that has not been explained by the CLT. Finally, until the topology population is achieved, new factors are added to the topology corresponding to the pairs of nodes with most downdated mutual information.

The more complementary factors are added to the topology, the more unreliable the downdated mutual information becomes, for the same reason as the initial MI values. Then, the downdate could be performed once or recursively after adding each new factor. However, in our experience, this does not contribute significantly to improve the resulting topology compared to its computational cost. We propose only doing a single downdate after building the CLT.

Expected KLD decrease

Taking profit of our FD method, we propose a new topology building methodology. Departing from the CLT topology, we incrementally build the topology by finding the factor candidate that will decrease the KLD the most. To compute that, for each remaining factor candidate we compute an estimation of its factor recovery solution. Afterwards, the KLD decrease of including each factor in the topology is computed. The candidate with the most expected KLD decrease is chosen.

In order to compute the factor recovery solution estimate, we perform one single FD instance, without iterating. Since the rest of factors contains at least the CLT factors, $\breve{\boldsymbol{\Upsilon}}_k$ is invertible and then the FD can be computed using Prop. 2.

Despite a single instance of Factor Descent does not provide the final factor recovery solution, it is enough to evaluate how much new information can be provided by each factor candidate. This alternative requires several evaluations of KLD and Factor Descent instances. However, note that the resulting topology already has a good initial guess equivalent to the first Factor Descent cycle.

Off-diagonal block determinant

Taking inspiration from the off-diagonal block initialization (4.8), we propose a third method to build the topology. The off-diagonal block entries of the dense information matrix $\boldsymbol{\Lambda}$ corresponding to each pair of nodes can only be explained with a factor between both nodes. Therefore, a new factor is added to the topology corresponding to the off-diagonal blocks with the most significant values.

Note that in this case, since we are dealing with an information matrix, adding factors does not affect the rest of off-diagonal blocks. Thus, no downdating is needed. The absolute determinant of each off-diagonal block of the information $\boldsymbol{\Lambda}$ is taken as an information measure.

Differently from the other methods proposed, this topology is built from scratch using this measure instead of complementing the CLT. For this reason it has to be build taking care of producing a fully connected topology. To guarantee this, the first $n-1$ pairs of factors are chosen such that they built a spanning-tree. Analogously to the CLT procedure, the pairs of nodes that do not create a cycle with largest off-diagonal block absolute determinant, are incrementally added to the topology. Once a tree topology is built, the no-cycle restriction is not applied anymore and more pairs of nodes are added until reaching the desired population.

4.5 Multi-node marginalization and sparsification

Typically, the marginalization and sparsification procedure is done sequentially, node after node. Once a new node is added to the graph, the marginalization of the previous one is considered.

However, if the marginalization and sparsification of nodes is made periodically after some nodes have been added, different alternative schemes appear. Multi-node marginalization and

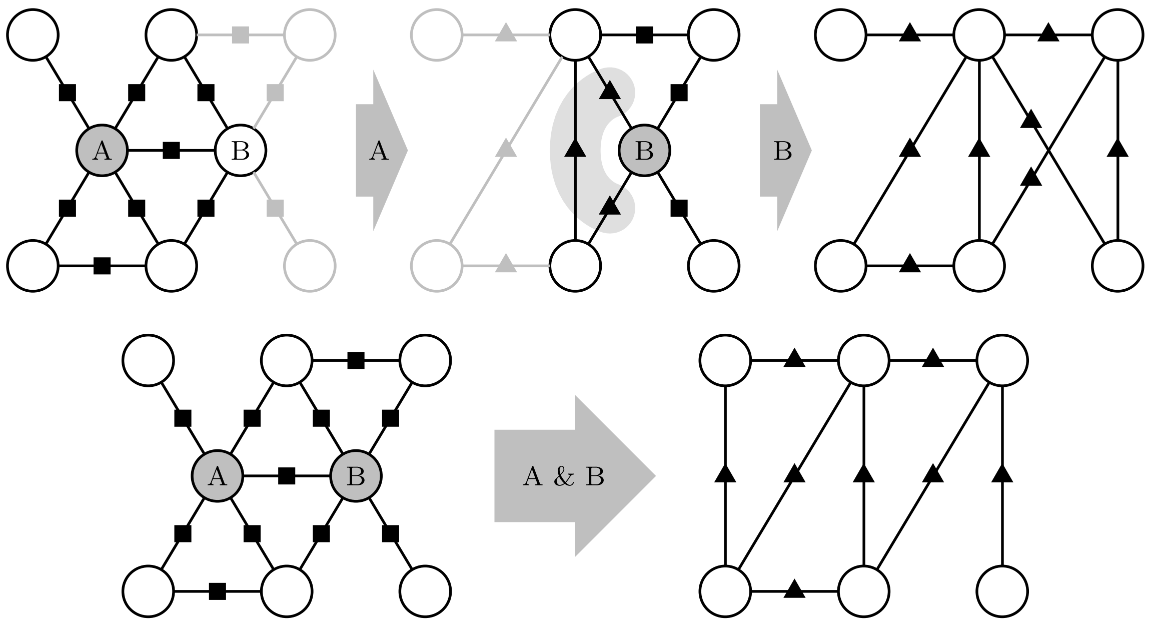


Figure 4.4: Example of single-node (top) and multi-node (bottom) schemes for removing and sparsifiying nodes A and B. Triangles indicate factors resulting of an sparsification.

sparsification would be possible considering groups of nodes at a time —for instance, those selected nodes to be removed that are connected by any factor. In the multi-node scheme, neither the procedure for marginalization nor sparsification suffer any changes. Taking as the Markov blanket the union of all removed nodes' Markov blankets, the marginalization of the group of nodes leads to a dense problem in the exact same way as if removing a single node.

In the single-node procedure, the approximations are accumulated since some factors resulting from a sparsification become intra-factors in the next instance. In contrast, in the multi-node scheme the process is executed only once for the entire group of connected nodes that have been selected for removal.

Despite connected nodes normally share most of their Markov blankets, the multi-node Markov blankets are larger than in the single-node scheme, depending on the sparsity of the graph. This results in bigger sparsification problems and thus higher computational cost to solve them.

The resulting topology in the multi-node scheme is usually different than that of the single-node method. While the first is designed considering the union of Markov blankets as a whole, the second is an accumulation of locally designed topologies. For example, when sparsifying with tree topologies such as CLT, the resulting topology in multi-node is indeed a tree, while the accumulation of trees in the single-node scheme usually yields a more populated final topology. This happens naturally in the general case: the final population of multiple applications of the single-node sparsification is higher than the final population of one multi-node sparsification for the same number of removed nodes.

Figure 4.4 depicts a toy example of the two schemes for the removal of two nodes connected by a factor. Note how in the single-node scheme, three factors (marked with a grey area) resulting from the first sparsification become intra-factors in the second one.

The election of the most appropriate scheme in terms of computational time depends on the graph SLAM morphology. In case of very sparse problems, multi-node Markov blankets do not or just slightly become bigger and the sparsification computational cost is significantly smaller than in single-node. Contrarily, in highly connected graphs, the Markov blanket in the multi-node scheme grows significantly and the factor recovery can take even more time that sparsifying the nodes one by one.

In terms of accuracy, and in the hypothesis of equal final populations, the multi-node scheme would be always a better choice since it does not accumulate approximations. However, for a given population policy, since the final population of the single-node scheme is normally bigger than in multi-node, it can produce better approximations. We want to remark that this is far from an advantage since after some sparsification instances, the results are more populated than as specified.

4.6 Results

Several experiments were realized in order to evaluate and validate the work presented in this chapter. They are organized in five batteries of tests.

A Matlab prototype was implemented to validate the cyclic FD method and test initial guess goodness. The PQN authors' Matlab implementation [Schmidt et al.] was used and IP was implemented using the gradient (4.6) and Hessian (4.7) detailed in [Mazuran et al., 2016]. However, in Matlab the most computationally expensive operations are optimized, and therefore CPU time measures are not reliable. For this reason the two first tests where based on the number of iterations of each factor recovery method. These two tests rapidly discarded PQN as a suitable method for factor recovery due to its slow convergence.

The remaining three groups of tests were evolved into a C++ implementation of Factor Descent (FD), non-cyclic FD (ncFD) as well as Interior Point (IP), in both single-node and multi-node schemes. This allowed us to rigorously compare the convergence rates, the performances in a real SLAM application, and to evaluate the different topology methods proposed.

All five batteries of tests are detailed in the following subsections. In all tests, we used real and synthetic SLAM datasets provided by [Carlone]. Additionally, in order to make a fair comparison available, a *sparsification dataset* was built. This dataset contains several sparsification problems (i.e. node marginalization dense results $\boldsymbol{\mu}, \boldsymbol{\Lambda}$) of different Markov blanket sizes. To build it, an experiment was run for the Manhattan M3500 dataset storing the result of each marginalization for the 80% of the nodes.

4.6.1 Quality of the initial guess

No matter which iterative factor recovery method is used, the closer to the solution is the initial guess the better. A first test was performed to evaluate the goodness of different initial guesses for factor recovery in relation with the problem size. Three different initial guesses were compared: the identity matrix (Id) as used in [Mazuran et al., 2016], the one based on the off-diagonal blocks (ODB) proposed by Mazuran and Tipaldi [2014] and introduced in Sec. 4.3.2, and the first cycle of our FD method (FFD).

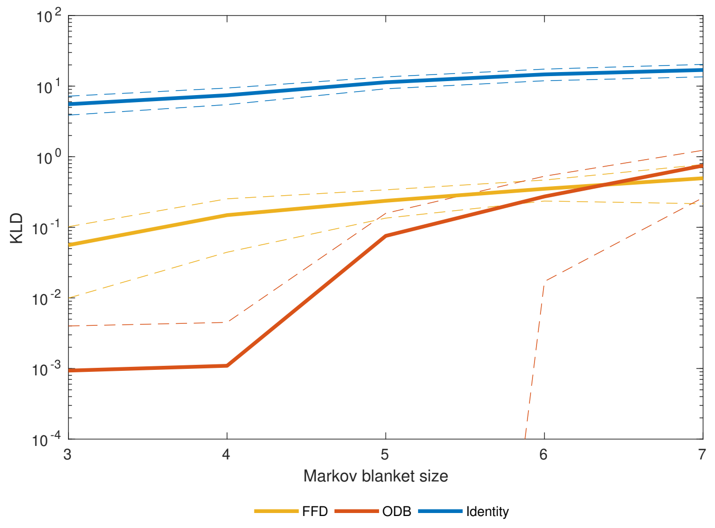


Figure 4.5: KLD mean and standard deviation (dashed line 1-σ) of different initial guesses vs. the Markov blanket size for all problems of the sparsification dataset. Notice the logarithmic KLD scale.

For all problems in the sparsification dataset, the same topology was generated using the MI topology method and the tree-proportional population policy with $\gamma = 2$ (twice as factors as in a tree topology). Then, the KLD of the approximation corresponding to each initial guess from the dense distribution was evaluated and stored. Figure 4.5 depicts the mean and the deviation of the KLD values as a function of the sparsification problem size, i.e. the Markov blanket size. In all cases, smaller Markov blankets have better initial guess approximations. ODB initialization performs better for small Markov blankets whereas the proposed FFD slightly outperforms it in larger problems. The identity initial guess (Id) is significantly inferior than the other two initialization methods (notice the vertical log axis).

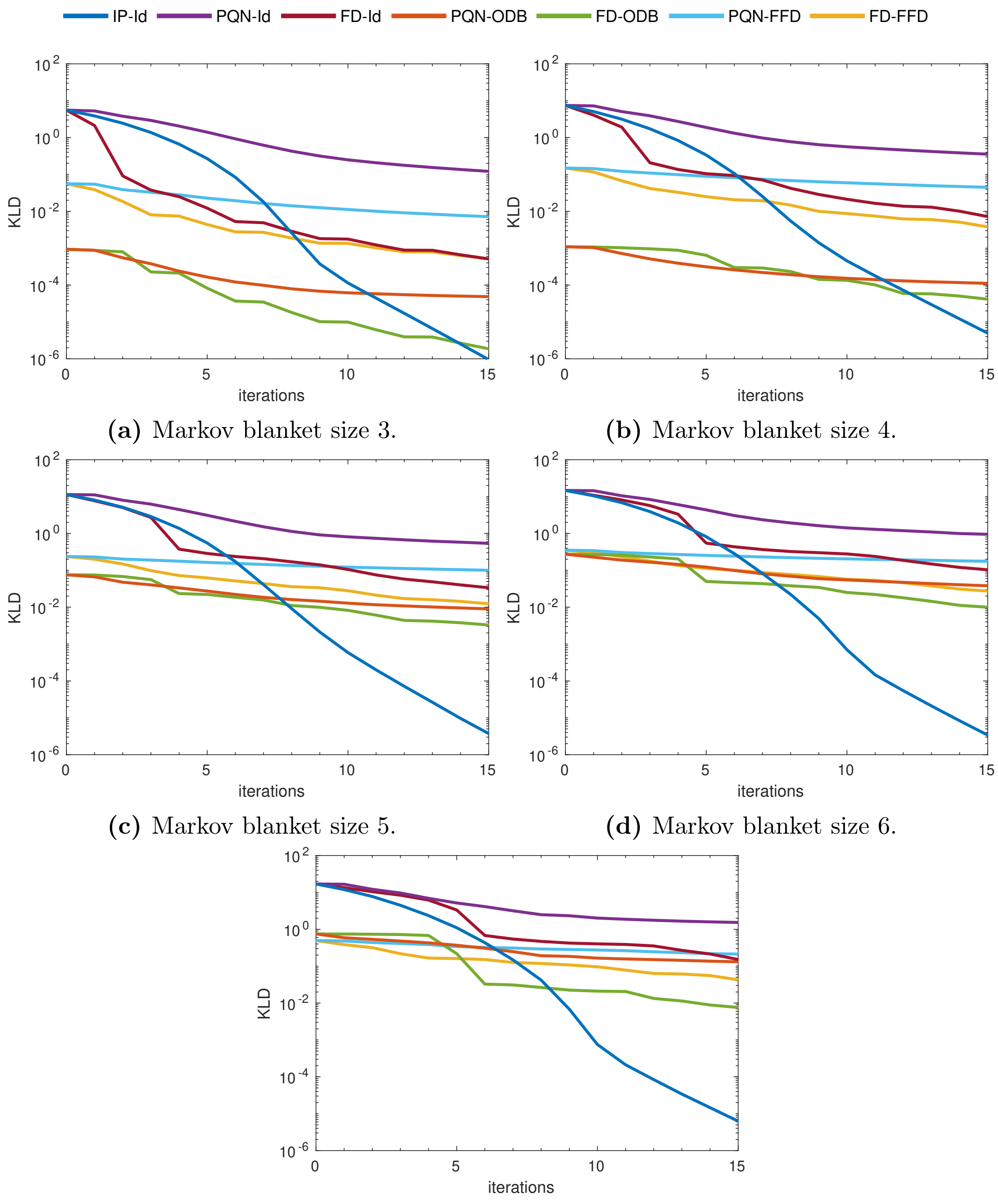
4.6.2 Algorithm prototype validation

To validate the FD method prototype, we ran a test similar to the initial guess test. For all problems in the sparsification dataset, factor recovery is computed for different combinations of initial guess and factor recovery method. Since topology is not the issue in these tests, we use the same topology for all cases (MI with the tree-proportional policy with $\gamma = 2$). In all combinations, after each factor recovery (IP, PQN or FD) iteration, the KLD from the dense distribution is stored.

The methods PQN and FD were combined with three different initial guesses: identity matrix (Id), off-diagonal blocks (ODB) and first FD cycle (FFD). The IP method was only combined with Id.

As with the initial guess, the convergence rate of the methods depends also on the size of the sparsification problem. We show in Figure 4.6 the mean KLD evolution for all problem sizes. While in small problems FD-ODB is the best combination up to 15 iterations, for larger Markov blanket sizes IP becomes the best choice after a smaller amount of iterations.

Evaluating in terms of number of iterations is not a fair comparison since in an iteration each method performs very different computations. However, it is enough to validate the



(a) Markov blanket size 3.

(b) Markov blanket size 4.

(c) Markov blanket size 5.

(d) Markov blanket size 6.

(e) Markov blanket size 7.

Figure 4.6: Mean KLD evolution of all sparsification combinations (methods and initial guesses) for different Markov blanket sizes corresponding to all problems in the sparsification dataset.

algorithm prototype. As expected, PQN is not a suitable method for factor recovery. This problem is very small and the cost evaluation is significant in comparison with the cost of computing the solution of the linearized problem. Indeed, PQN is designed for the opposite case: large problems in which cost evaluation is negligible.

4.6.3 Convergence time

Once the FD prototype has been validated, the IP and both the cyclic factor descent (FD) and its non-cyclic variant (ncFD) were implemented in `C++`. This allowed us to evaluate the

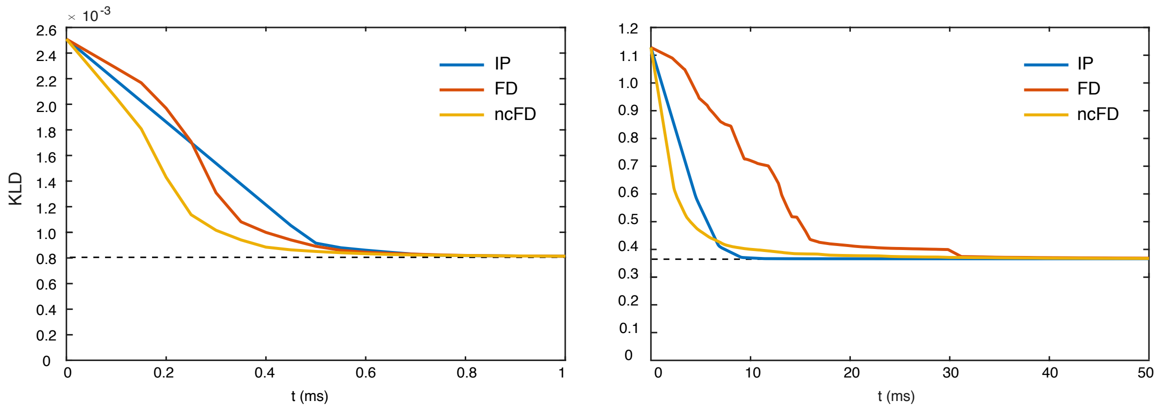


Figure 4.7: Mean KLD temporal evolution of the compared methods, for all problems of Markov blanket size 3 (left) and 8 (right) of the sparsification dataset. Note the different KLD and time scales.

respective CPU time burdens, *i.e*, the real-time convergence rates.

To compare the convergence, the three factor recovery methods solved all problems in the sparsification dataset, using the same topology (MI and tree-proportional with $\gamma = 2$) and taking the off-diagonal blocks (ODB) as initial guess. The temporal evolution of the KLD of each method from the dense distribution was stored.

The IP parameters described in subsection 4.3.2 were set as a result of a delicate tuning process. Specifically, we set the decrease parameter $\alpha = 0.5$. The balance weight $\omega = 0.1$ was used for setting the initial value ρ_0. Also, we imposed a relaxed end condition for the inner loop when the norm of the KLD gradient becomes lower than 1. Conversely, neither factor descent nor non-cyclic factor descent have any internal parameter to be tuned.

Figure 4.7 shows the mean KLD temporal evolution of each sparsification method for all problems of Markov blanket size 3 and 8. As observed in the initial guess test 4.6.1, the ODB initialization is closer to the optimal solution for small problems than for bigger ones. Likewise, all methods converge faster for small problems since IP has smaller Hessian and FD and ncFD have less factors to iterate over. The convergence of FD and ncFD are comparable to IP's. Additionally, the benefits of the non-cyclic strategy are clear, specially in bigger problems.

4.6.4 Application to real SLAM problems

A new battery of tests was made with the purpose of evaluating the performance of each method and the multi-node scheme in a real application.

We tested each method using both single-node and the multi-node scheme on four different datasets with different node reduction levels. Since node selection remains out of the scope of this work, we applied the simple strategy of keeping one node every N.

The typology of the chosen datasets is very different. The Manhattan M3500 sequence is large and dense (i.e. highly connected), which means large Markov blankets. On the contrary, the Killian Court dataset has few loop closures leading to small Markov blankets. The Freiburg Building (FR079) and Intel Research Lab sequences are a compromise between

Dataset	Density	Scheme	Topology population (tree-prop. policy)	Node reduction							
				66.6%		75%		80%		90%	
				#	n	#	n	#	n	#	n
Manhattan	1.558	Single	$\gamma = 1.0$ (CLT)	2332	2.46	2624	2.42	2799	2.41	3149	2.33
			$\gamma = 2.0$	2332	2.71	2624	2.77	2799	2.90	3149	3.10
		Multi	$\gamma = 1.0$ (CLT)	1177	2.91	909	3.18	734	3.45	384	4.21
			$\gamma = 2.0$	1177	2.91	909	3.21	734	3.48	384	4.29
Intel	1.227	Single	$\gamma = 1.0$ (CLT)	818	2.23	920	2.25	981	2.26	1104	2.27
			$\gamma = 2.0$	818	2.29	920	2.34	981	2.42	1104	2.64
		Multi	$\gamma = 1.0$ (CLT)	413	2.41	319	2.57	258	2.75	135	3.19
			$\gamma = 2.0$	413	2.41	319	2.58	258	2.76	135	3.25
FR079	1.232	Single	$\gamma = 1.0$ (CLT)	658	2.22	741	2.20	790	2.20	889	2.20
			$\gamma = 2.0$	658	2.26	741	2.27	790	2.29	889	2.36
		Multi	$\gamma = 1.0$ (CLT)	332	2.33	256	2.37	207	2.42	108	2.65
			$\gamma = 2.0$	332	2.33	256	2.38	207	2.42	108	2.67
Killian	1.025	Single	$\gamma = 1.0$ (CLT)	538	2.02	605	2.04	645	2.04	726	2.07
			$\gamma = 2.0$	538	2.02	605	2.05	645	2.06	726	2.12
		Multi	$\gamma = 1.0$ (CLT)	271	2.04	210	2.08	170	2.08	89	2.19
			$\gamma = 2.0$	271	2.04	210	2.08	170	2.08	89	2.20

Table 4.1: Amount of sparsification problems solved (#) and Markov blanket mean size (n) in single-node and multi-node schemes.

the other two datasets. Much denser datasets such as the city10k constitute a challenge for our and the other methods, and are considered for future work.

Three iterative optimization methods were compared: IP, FD and ncFD. In all cases, the topology was equally built using the MI method and the tree-proportional population policy with $\gamma = 2$ as in [Mazuran et al., 2016]. We applied the same end condition for all methods: when all elements of the KLD gradient become lower than 10^{-3}. Also a maximum time condition was set to 50ms. For further completeness, the CLT topology with closed form factor recovery was added to the comparison.

An independent experiment was ran for each method using our own non-linear least squares SLAM implementation based on Ceres [Agarwal et al.]. Node marginalization and sparsification was performed every 100 nodes. Then, each experiment accumulates the sparsification approximations along the whole dataset. The original SLAM graph without removing any node was taken as a baseline. The global KLD of each method from the baseline was computed using (3.4) for the whole SLAM problem. As in [Mazuran et al., 2016], factors involving previously removed nodes were redirected to the closest existing node. In order not to distort the KLD results, this was also done for the baseline graph. The SLAM problem was relinearized continuously to prevent linearization errors to be confused with sparsification inaccuracy.

Table 4.1 contains the amount of sparsification problems solved and the Markov blanket

Dataset	**Scheme**	**Method**	**Node reduction**											
			66.6%			**75%**			**80%**			**90%**		
			KLD	**RMSE**	**Total time**	**KLD**	**RMSE**	**Total time**	**KLD**	**RMSE**	**Total time**	**KLD**	**RMSE**	**Total time**
Manhattan	Single	CLT	59.39	0.297	0.23 s	45.79	0.607	0.26 s	32.33	0.113	0.28 s	17.32	0.258	0.28 s
		IP	3.72	0.037	8.49 s	2.58	0.026	11.41 s	2.93	0.036	14.77 s	3.17	0.138	25.51 s
		FD	**3.53**	0.044	3.30 s	**2.44**	**0.024**	4.91 s	**2.69**	0.048	7.02 s	**2.73**	0.148	16.25 s
		ncFD	3.56	0.046	2.38 s	2.46	0.024	2.97 s	2.69	0.035	3.90 s	2.75	0.136	13.52 s
	Multi	CLT	60.39	0.245	**0.17 s**	46.73	0.495	**0.16 s**	42.07	0.168	**0.16 s**	36.68	0.096	**0.13 s**
		IP	3.72	0.017	6.12 s	2.76	0.028	6.70 s	3.54	0.038	7.71 s	4.80	0.093	***13.00 s***
		FD	3.70	**0.015**	3.11 s	2.78	0.033	4.24 s	3.54	0.037	5.92 s	4.71	0.094	20.30 s
		ncFD	3.76	0.029	***2.15 s***	2.76	0.030	***2.29 s***	3.45	**0.030**	***3.46 s***	4.75	**0.091**	16.49 s
Intel	Single	CLT	28.77	0.094	0.07 s	29.00	0.048	0.08 s	29.16	0.066	0.09 s	17.47	0.118	0.09 s
		IP	6.04	0.022	1.93 s	5.66	0.044	2.21 s	5.42	0.022	2.85 s	2.51	0.022	4.69 s
		FD	5.45	0.024	1.68 s	6.79	0.049	1.81 s	**5.04**	0.023	2.20 s	2.04	0.023	2.55 s
		ncFD	5.46	0.024	1.33 s	5.87	0.046	1.35 s	5.15	0.024	1.62 s	2.14	0.023	1.59 s
	Multi	CLT	25.33	0.115	**0.04 s**	21.90	0.079	**0.04 s**	22.59	0.043	**0.04 s**	10.42	0.157	**0.02 s**
		IP	5.91	0.023	1.46 s	**4.78**	**0.028**	1.37 s	5.61	**0.015**	1.50 s	1.99	0.018	1.31 s
		FD	**5.27**	**0.021**	1.43 s	4.81	0.032	1.38 s	5.59	0.016	1.53 s	**1.93**	**0.018**	1.49 s
		ncFD	5.34	0.022	***1.11 s***	5.16	0.030	***1.00 s***	5.55	0.016	***0.96 s***	2.05	0.018	***0.92 s***
FR079	Single	CLT	12.93	0.024	0.05 s	13.73	0.015	0.06 s	12.92	0.028	0.07 s	10.11	0.025	0.07 s
		IP	**2.63**	0.008	1.91 s	2.27	0.005	2.15 s	1.66	0.003	2.71 s	1.16	0.009	3.75 s
		FD	2.64	**0.008**	1.81 s	2.33	0.004	2.05 s	**1.63**	0.004	2.54 s	0.98	0.010	3.29 s
		ncFD	2.68	0.008	1.52 s	2.31	0.005	1.59 s	1.71	**0.003**	2.11 s	0.99	**0.009**	2.63 s
	Multi	CLT	14.81	0.022	**0.03 s**	13.05	0.014	**0.03 s**	10.04	0.018	**0.02 s**	5.94	0.012	**0.01 s**
		IP	3.08	0.009	1.18 s	**2.14**	0.003	0.97 s	1.70	0.005	1.20 s	**0.97**	0.017	***0.92 s***
		FD	3.10	0.009	1.18 s	2.16	0.003	1.01 s	1.70	0.006	1.16 s	1.15	0.016	1.11 s
		ncFD	3.17	0.010	***0.92 s***	2.22	**0.003**	***0.78 s***	1.75	0.005	***0.95 s***	1.29	0.018	0.98 s
Killian	Single	CLT	2.48	0.520	0.03 s	6.43	0.290	0.04 s	7.92	1.048	0.04 s	9.51	2.887	0.05 s
		IP	0.37	0.230	0.10 s	0.43	**0.181**	0.22 s	2.18	0.580	0.26 s	0.41	0.473	1.24 s
		FD	0.37	0.230	0.05 s	0.45	0.192	0.07 s	2.19	0.571	0.08 s	0.41	0.459	0.26 s
		ncFD	0.37	0.229	0.04 s	0.42	0.182	0.07 s	2.18	0.569	0.08 s	0.41	0.385	0.26 s
	Multi	CLT	2.15	0.266	**0.02 s**	14.83	0.236	**0.01 s**	3.39	1.337	**0.01 s**	3.39	2.402	**0.01 s**
		IP	**0.08**	0.053	0.07 s	**0.36**	0.181	0.11 s	0.42	0.262	0.09 s	**0.28**	0.327	0.32 s
		FD	0.08	0.053	0.03 s	0.38	0.191	0.04 s	0.41	0.265	0.04 s	0.28	0.352	0.13 s
		ncFD	0.08	**0.053**	***0.03 s***	0.36	0.185	***0.03 s***	**0.41**	**0.255**	***0.03 s***	0.28	**0.304**	***0.10 s***

Table 4.2: Comparison of final global KLD and RMSE and CPU time for all factor recovery methods, different datasets and node reduction levels.

mean size using single-node and multi-node schemes for all datasets-node reduction combinations. As can be observed, the multi-node scheme reduces significantly the amount of sparsifications performed according to the node reduction ratio. That is, for higher reduction ratios, the nodes selected to be removed are more connected leading to less problems. In exchange, there is an increase in the Markov blanket mean size. As expected, this Markov blanket increase is significant for highly connected graphs. With the 90% of node reduction, in Manhattan dataset, the Markov blanket mean size is almost doubled while in Killian and Freiburg cases it remains almost the same.

Table 4.2 includes the final global KLD values after applying each method in both single-node and multi-node schemes in the different datasets for different node reduction ratios. All iterative methods achieve similar KLD and RMSE values in all experiments. However, the approximation of CLT is significantly worse. This confirms once again that the tree topology is too sparse to explain the dense distribution.

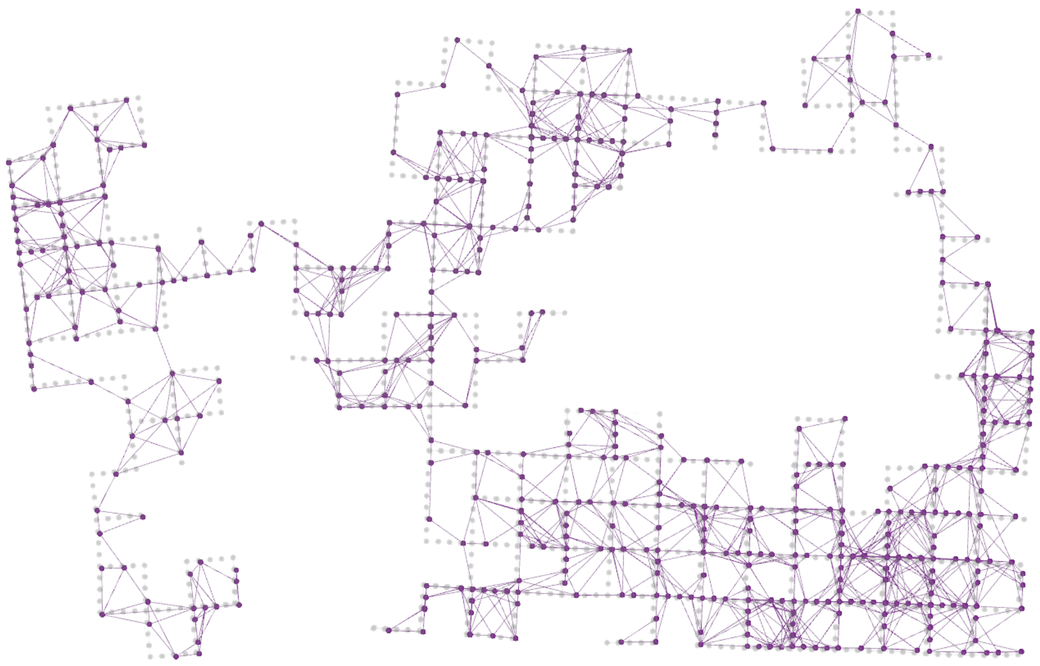


Figure 4.8: Final baseline graph (faded gray) and sparsified graph (purple) in Manhattan experiment with 80% of node reduction.

The optimal parameters of IP are not the same for all datasets and node reduction levels, and a significant effort on tuning was needed to achieve the optimal KLD reduction with as less computational cost as possible. Conversely, the simplicity of the algorithm and the absence of parameters are the main advantages of FD and ncFD. Furthermore, both FD and ncFD outperform IP in computational time in almost all the experiments.

Apart from CLT, ncFD is faster than IP and FD in almost all experiments. As pointed out before, ncFD convergence improvements w.r.t. FD are specially relevant for the case of big Markov blankets. For this reason, the computation time benefits are more significant for the denser datasets. While in the Manhattan dataset the total time spent by ncFD is lower than the cyclic version, in the Killian dataset it is similar.

The multi-node scheme speeds up all methods by reducing the amount of sparsification problems to be solved. However, the Markov blanket growth may explain the different performance in KLD and RMSE depending on the dataset. In the Killian dataset, the multi-node scheme produces more accurate approximations than the single-node scheme for all methods and reduction levels. However, in the FR079 and Intel datasets, using multi-node instead of single-node is not beneficial in any of the cases regarding to KLD and RMSE. The Markov blanket growth in the Manhattan dataset is strong, undermining the approximation accuracy, especially for high node reduction levels.

Figure 4.8 shows the final graph of the Manhattan experiment with 80% of node reduction. The baseline graph is overlapped in faded gray. It can be observed how, despite removing most of the nodes, the estimation remains practically the same.

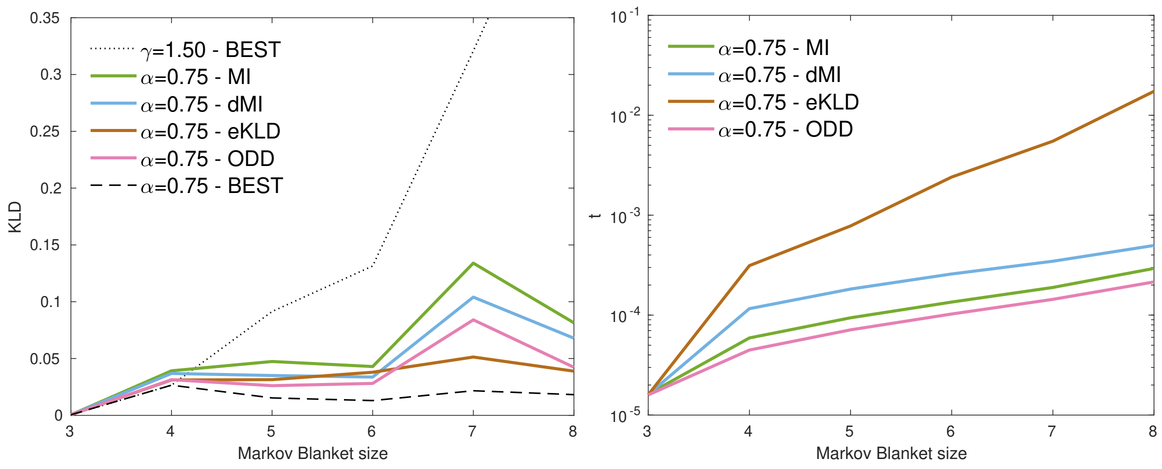


Figure 4.9: Comparison of topology building methods using different fill-in population policies. The plots show KLD of the optimal factor recovery solution (left) and computational cost (right) for different Markov blanket sizes. In dashed black, the best topology, shown as a baseline. In dotted black, the solution of the best topology using the tree-proportional population policy.

4.6.5 Topology

A different set of experiments had the purpose of evaluating the sparsification performance regarding to the topology methods and the population policy. The comparison included all proposed topology methods: decreased mutual information (dMI), expected Kullback-Liebler divergence (eKLD) and off-diagonal determinant (ODD). Also, the state-of-art method which complements the CLT based on mutual information (MI) and the best (BEST) topology found using brute force combinatorial search were added to the comparison.

A first experiment was ran using the sparsification dataset. For each stored dense distribution defined by $\boldsymbol{\mu}, \boldsymbol{\Lambda}$, all topology methods were launched. In each problem, the same topology population was used using a fill-in population policy with $\alpha = 0.75$. The computational cost of the topology building process and the KLD of the optimal factor recovery solution (solved using FD) were stored for each topology method and all problems in sparsification dataset.

The best topology is obtained in a brute-force search, for all possible topology combinations keeping the best set of factors in terms of KLD from the original problem. This alternative is only considered as a baseline for comparison since it is computationally prohibitive.

Figure 4.9 (left) shows the mean KLD of the factor recovery solution from the dense distribution using each of the compared topology methods. The frame to the right in the same figure shows the mean time required to build the topology. It is important to remark that the plot does not include the time required for factor recovery.

The most computationally expensive topology method is eKLD since it requires a Factor Descent instance and a KLD evaluation for all remaining factors after each new factor that is added to the topology. Despite its computational cost, eKLD performs best in terms of KLD for the largest Markov blanket sizes. The dMI method improves significantly the KLD performance of MI by only slightly increasing its computational cost. It confirms that downdating the regularized covariance avoids adding factors between nodes whose mutual information is already explained by the CLT factors. The ODD method is the fastest method given its simplicity and provides topologies that can still accurately approximate the original distribution.

The best topology found with brute force combinatorial search using the fill-in population policy is plotted in dashed black and the best topology using the tree-proportional policy in dotted black. It can be observed how the tree-proportional population policy becomes too sparse for big Markov blankets leading to higher KLD values.

Finally, a last battery of experiments were made on two different datasets: the Manhattan M3500 sequence and the Intel Research Lab sequence. The Manhattan M3500 sequence is a large and highly connected problem. Then, it has large Markov blankets. Contrarily, the Intel Research Lab sequence has very few loop closures and smaller Markov blankets. The Freiburg and Killian datasets were discarded for this test since they provided very few sparsification problems with Markov blankets bigger than 3 nodes. The topology methods and population policies differences show up in higher problem sizes.

Several combinations of population policy and topology methods were compared. Both population policies were used with two different values of the corresponding parameters. For each four population combinations, four topology methods are compared, MI, dMI, ODD and eKLD. Additionally, the CLT topology with closed form factor recovery was also included in the comparative. Note that CLT is exactly equivalent to the MI topology method using the tree-proportional population policy with $\gamma = 1.0$.

For each combination of topology method and population policy, an independent experiment was run. In all experiments, the same 80% volume of nodes were marginalized. As in previous tests, since node selection is out of the scope, we applied the simple strategy of keeping one node every 5. Also, the node marginalization and sparsification is performed every 100 nodes using FD and ODB initial guess in all cases. Therefore, each experiment accumulates the sparsification approximations along the whole dataset showing the differences induced exclusively by the topology differences.

As in previous application tests, the original SLAM graph without removing any node was taken as a baseline. The global KLD of each method from the baseline was again computed using (3.4) for the whole SLAM problem. Again, the factors involving previously removed nodes were redirected to the closest existing node as well. In order not to distort the results, this was also done for the baseline graph. The same end conditions were used in all cases: the KLD gradient norm lower than 10^{-3} and a maximum time of 0.20 ms.

The results of all experiments are contained in Table 4.3. Global KLD and RMSE of the final graph w.r.t. the baseline are reported. Additionally, the table contains the total computational time spent on sparsification and topology building, and the amount of factors of the final graph.

As experienced before, the closed form factor recovery makes the CLT the fastest sparsification alternative and the one that produces the sparsest final graph. However, the simplicity of the topology produces worse approximations leading to higher global KLD and RMSE values. In contrast, for the same population policy, the most populated options

	Population policy		**Topology**	**KLD**	**RMSE**	**Sparsification total time**	**Topology total time**	**Number of factors**
Manhattan	Tree-proportional	$\gamma = 1.0$	MI (CLT)	32.33	0.114	**0.42 s**	0.40 s	1605
		$\gamma = 1.5$	MI	6.99	0.072	10.22 s	0.47 s	2283
			dMI	***6.74***	0.066	10.50 s	0.75 s	2269
			ODD	8.73	0.076	***9.78 s***	***0.40 s***	2231
			eKLD	7.14	0.063	12.11 s	2.85 s	2274
		$\gamma = 2.0$	MI	2.63	0.035	***9.78 s***	0.45 s	2624
			dMI	2.22	***0.023***	10.84 s	0.66 s	2623
			ODD	***2.17***	0.040	11.49 s	**0.40 s**	2619
			eKLD	2.77	0.046	16.62 s	4.32 s	2658
	Fill-in	$\alpha = 0.75$	MI	2.97	0.120	***13.60 s***	0.51 s	2482
			dMI	***2.57***	**0.020**	14.55 s	0.81 s	2482
			ODD	3.30	0.133	16.23 s	***0.42 s***	2519
			eKLD	4.29	0.056	23.74 s	5.95 s	2579
		$\alpha = 0.85$	MI	0.85	0.046	13.90 s	0.48 s	2895
			dMI	**0.73**	***0.026***	***13.63 s***	0.72 s	2898
			ODD	0.81	0.027	15.15 s	***0.43 s***	2937
			eKLD	1.66	0.030	19.66 s	9.12 s	2945
Intel	Tree-proportional	$\gamma = 1.0$	MI (CLT)	29.16	0.066	**0.15 s**	0.13 s	366
		$\gamma = 1.5$	MI	10.00	0.045	3.53 s	0.12 s	492
			dMI	10.65	0.041	4.95 s	0.20 s	492
			ODD	9.58	***0.024***	***3.38 s***	***0.10 s***	488
			eKLD	***5.61***	0.030	4.53 s	0.71 s	509
		$\gamma = 2.0$	MI	5.01	***0.023***	5.50 s	0.11 s	558
			dMI	4.72	0.030	5.21 s	0.16 s	560
			ODD	3.98	0.030	5.31 s	**0.10 s**	563
			eKLD	***3.11***	0.029	5.59 s	0.84 s	577
	Fill-in	$\alpha = 0.75$	MI	4.52	0.034	5.74 s	0.15 s	545
			dMI	3.49	0.023	6.85 s	0.23 s	547
			ODD	***3.39***	0.023	5.77 s	***0.12 s***	549
			eKLD	3.61	***0.015***	8.47 s	1.34 s	559
		$\alpha = 0.85$	MI	2.21	0.016	6.45 s	0.13 s	610
			dMI	**2.07**	0.017	7.13 s	0.19 s	611
			ODD	2.09	0.019	6.70 s	***0.11 s***	611
			eKLD	2.23	**0.013**	7.89 s	1.63 s	618

Table 4.3: Comparison of different combinations of population policy and topology building method using FD factor recovery in different datasets at 80% node reduction.

($\gamma = 2.0, \alpha = 0.85$) produce much more accurate approximations both in terms of KLD and RMSE. Indeed, as repeatedly stated, more populated topologies can better approximate dense distributions.

Comparing dMI and MI, it can be seen how downdating the regularized covariance slightly improves the resulting topology, producing better approximations. While in some cases, eKLD produces informative topologies that can better encode the dense information, in some cases it produces worse approximations and its computational cost is significantly higher than the rest of topology methods. While ODD outperforms MI in more populated topologies ($\gamma = 2.0$ and $\alpha = 0.85$), it produces similar or worse results in sparser cases. ODD is the fastest method even competing with CLT building since it is only evaluated once for all factor candidates.

Obviously, for a given population policy option, all four topology methods produce similarly sparse final graphs. Differences are due to concatenating sparsification processes. As we observed in the first results, the fill-in population policy produces more populated topologies in big Markov blankets. Then, to achieve similar approximations in big Markov blanket problems, a higher value of γ is needed for the tree-proportional policy leading to too much populated topologies in small problems. For this reason, the fill-in policy with $\alpha = 0.75$ achieves similar KLD and RMSE than tree-proportional with $\gamma = 2$ using less factors.

4.7 Discussion and final remarks

Marginalization and sparsification are valuable tools to reduce the size of SLAM problems without undermining the accuracy and efficiency of the NLS solvers.

Deciding the topology of the new factors to approximate the original distribution is a trade off between accuracy and computational cost. For the simplest topologies, e.g., spanning tree, factor recovery has a closed form solution but the approximations are the less accurate. On the contrary, more populated topologies prevent the applicability of a closed form solution, thus iterative optimization is required to solve factor recovery; this is more computationally costly. Furthermore, the more populated the topology is, the more computationally expensive the factor recovery becomes, regardless of which iterative factor recovery method is used. In exchange, the more populated the topology is, the better the original graph information can be encoded, reaching better approximations.

In this chapter Factor Descent (FD) and Non-Cyclic Factor Descent (ncFD) are presented, two optimization methods for the sparsification of populated topologies in large-scale graph-based SLAM. Our results show that both methods compete with the most popular state-of-art method (interior point) both in accuracy and computational time and even outperform it in most cases. At the same time, the simplicity of the algorithm makes FD and its non-cyclic version ncFD appealing approaches when compared with the interior point method, since they do not require delicate tuning of parameters, which in IP we have found to be problem-dependent. We demonstrated convergence improvements of the non-cyclic factor descent variant, especially in highly connected problems.

Moreover, the multi-node scheme for periodic marginalization and sparsification has been explored. This proved to be always a better choice in terms of approximation accuracy, and it is more efficient in moderately connected problems.

Furthermore, three new methods for building populated pose-graph topologies and a new policy to determine the topology population are proposed, outperforming the state-of-art approaches.

All the work presented in this chapter has been published in [Vallvé et al., 2017, 2018a,b].

5 Information Metrics for Active Pose-Graph SLAM

5.1 Introduction

As mentioned in the introduction chapter, we define exploration as the problem of driving the robot with the aim of creating a map of the environment. In exploration, perfect localization is assumed relying in the localization (or SLAM) algorithm. In other words, paths are planned only to pursue full coverage. However, the localization performance highly depends on the robot trajectory and decoupling localization from exploration may lead to a poor quality map.

Consequently, we define active SLAM as the problem of driving the robot with the aim of autonomously creating a map of the environment while maintaining also good localization. In active SLAM two driving forces come into play: exploration and re-localization. Both can be posed as map and localization uncertainty reductions, respectively. As presented before, entropy measures uncertainty. Therefore, active SLAM can be posed as the problem of driving the robot to reduce both map and localization entropies.

In fact, improving the localization accuracy also benefits the accuracy of the resulting map. In other words, an environment representation built from highly uncertain locations is very inaccurate. Therefore, while pursuing map uncertainty reduction may be understood as discovering unknown regions, reducing the robot trajectory estimation uncertainty could be understood as improving the quality of the map so far built.

5.1.1 Related work

Most exploration techniques [Yamauchi, 1997; Shade and Newman, 2011] do not take into account the uncertainties in robot motion and sensing and limit themselves to pursue full coverage.

Exploration strategies driven by map uncertainty reduction date back to the seminal work of Whaite and Ferrie [1997] for the acquisition of 3D models of objects from range data. Within the context of SLAM, it is the work of Feder et al. [1999], who first proposed a metric to evaluate uncertainty reduction as the sum of the independent robot and landmark entropies with an exploration horizon of one step to autonomously produce occupancy maps. Bourgault et al. [2002] alternatively proposed an utility function for exploration that trades off the potential reduction of vehicle localization uncertainty, measured as entropy over a feature-based map, and the information gained over an occupancy grid. In contrast to these approaches, which consider independently the reduction of vehicle and map entropies, Vidal-Calleja et al. [2010] tackled the issue of joint robot and map entropy reduction, taking into account robot and map cross correlations for the Visual SLAM EKF case. Torabi et al. [2007] jointly computed the entropy reduction directly in configuration space but for a limited number of configurations.

Stachniss et al. [2005] tackled the active SLAM problem as a one step look ahead entropy minimization problem using a Rao-Blackwellized particle filter. The technique extends the classical frontier-based exploration method [Yamauchi, 1997] to the full SLAM case. In [Valencia et al., 2012] a similar approach is proposed using a pose-graph SLAM instead. Both methods [Stachniss et al., 2005; Valencia et al., 2012] evaluate the joint map and trajectory entropy change for a few frontier-based exploratory and loop closure candidate trajectories. In [Valencia et al., 2012], exploratory actions considered omnidirectional sensing and evaluated paths toward positions near frontiers, disregarding orientation. In a more general setting, a sensor, such as a laser range finder or a camera, would have a narrow field of view, and hence, we need to deal with full poses not just positions.

Alternatively, exploration methods have been proposed based on the adaptation of different motion planning algorithms such as potential fields or probabilistic planning. Potential field methods have been previously suited to exploration by Prestes e Silva et al. [2002] and Shade and Newman [2011]. However, they directly evaluate boundary conditions on deterministic maps of obstacles and frontiers, without taking uncertainty into account. The adaptation of probabilistic planning methods to the exploration problem was introduced by Oriolo et al. [2004]. In their approach called SRT, the robot executed motion commands to a random pose sampled inside the newly observed area that was not observed in previous nodes. Posteriorly, Freda et al. presented a frontier-based SRT [Freda and Oriolo, 2005] which randomly selected the frontier centroids of the last observation as goals. However, both approaches directly executed the first random sample without any optimality evaluation.

5.2 Active pose-graph SLAM

One main objective of mobile robot exploration is obtaining a dense representation of the environment. In order to allow safe path planning, not only obstacles should be represented but also free space.

A widely used approach to do this is combining pose-graph SLAM with occupancy grids. Pose-graph SLAM only estimates the robot trajectory, which consists of nodes only corresponding to robot poses and relative motion factors. Motion factors mainly derive from odometry or motion commands (for consecutive nodes) or registration of extereoceptive sensory data (for closing loops). At any time, a dense map can be rendered using the trajectory estimate and the extereoceptive sensory data.

In our work, we focused on a 2D application using lidar. For closing loops, the iterative closest point (ICP) method provides the transformation between two robot poses by finding the best alignment of the laser scans taken at the two instances. A 2D occupancy grid is rendered periodically placing the corresponding scans accordingly to the trajectory estimate as described hereafter.

5.2.1 Log odds occupancy grid map

In our approach, the pose-graph SLAM estimate and raw sensor data are used to synthesize an occupancy map. This map will be used to decide the next trajectory to be executed in order to reduce the uncertainty in both the map and the trajectory. The mapping of unknown cells near obstacles in the presence of uncertainty might drive the robot to areas near collision, a situation we need to avoid. Moreover, there is a compromise between tractability and accuracy in choosing the resolution at which the occupancy cells are discretized.

To provide an accurate computation of the occupancy map, which is necessary for the proper active pose-graph SLAM operation, we render the map from all poses in the estimated trajectory, and not only a limited number of them. Moreover, the resolution at which the occupancy grid map is computed is very fine in order to effectively compute the most uncertainty reduction trajectory. Instead of repeating the ray-casting operation at each iteration, we store local occupancy maps at each robot pose, and aggregate them efficiently for the computation of a global occupancy map.

At each robot pose k, the raw sensor data is ray-casted once to accumulate the probability $p_k(c)$ of each cell c to be occupied. This is encoded in a log odds occupancy grid in local coordinates

$$m_k^L(c) = \log \frac{p_k(c)}{1 - p_k(c)} \ . \tag{5.1}$$

The use of log odds allows us to accumulate different observations of the cell occupancy by simple addition.

These local log odds occupancy maps are shown in Figure 5.1 for a small number of robot poses. Negative values (black) mean free space, and positive values (white) mean obstacles. A value of zero (gray) means unexplored. During open loop, each local map is aggregated into the global log odds occupancy map m. To relate them in a common reference frame,

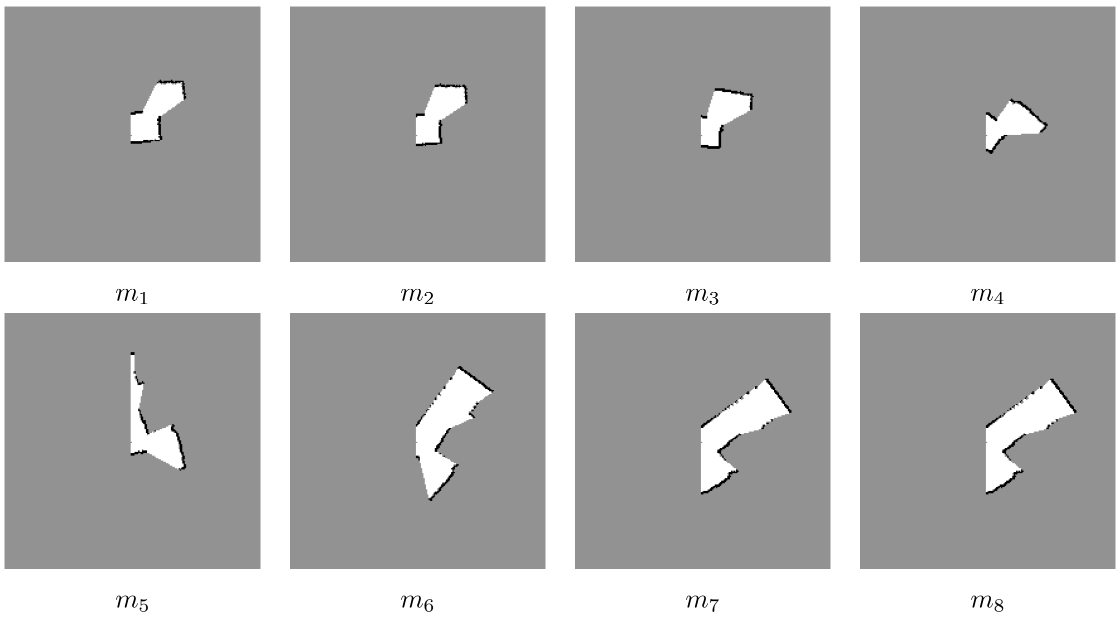


Figure 5.1: A number of log odds occupancy maps in local coordinates.

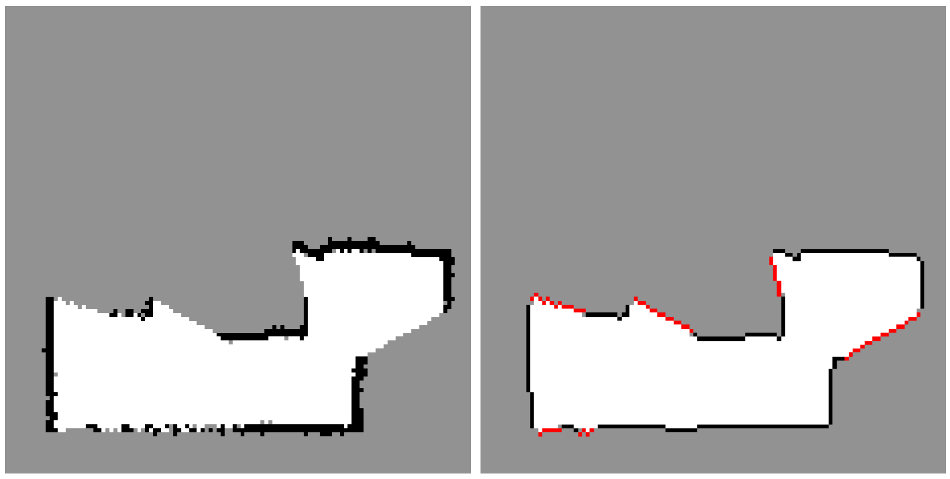


Figure 5.2: Left: Aggregated log odds occupancy map m. Right: Classified frontier cells (red), obstacle cells (black), free cells (white) and unobserved cells (grey).

each local map is rotated and translated using efficient image processing routines. Only at loop closure, i.e. big trajectory estimation change, the occupancy map is recomputed from scratch using all previously stored local log odds maps but oriented and translated according to the newly estimated robot poses. The result is shown in the left plot of Figure 5.2.

Generalizing (5.1) to the global coordinate case, we can solve directly for aggregated cell occupancy probabilities,

$$p(c) = \frac{\exp\big(m(c)\big)}{1 + \exp\big(m(c)\big)} \; . \tag{5.2}$$

Two occupancy probability thresholds define free cells ($p(c) < p_{free}$), and occupied cells ($p(c) > p_{obs}$). The rest of cells are labeled as unknown. Those free cells that is close to an unknown cell are defined as frontier cells.

Note that this map aggregation is computed only at the mean pose estimates. To smooth out misclassified cells and perform the classification into free, obstacle, unknown and frontier cells, morphological opening and closing operations on the global map are used. The resulting detection of frontiers, obstacles and free cells is exemplified in the right image of Figure 5.2.

5.2.2 Entropy minimization

The aim in active SLAM is to plan a trajectory that maps the environment while maintaining an accurate localization. We can rephrase this in terms of uncertainty. Maximizing the localization accuracy is equivalent to reducing the localization estimation uncertainty. Also, mapping the environment (i.e. discovering it) can be understood as reducing the uncertainty of its representation. In case of pose-graph SLAM using an occupancy grid as the map, and since this map is rendered from the trajectory estimate, the more accurate the trajectory, the more accurate will the map be. Considering all this, our aim is to minimize the joint map and trajectory uncertainty, thus its total entropy.

A pose-graph SLAM estimates the robot trajectory as a multivariate Gaussian distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and an occupancy map models the environment as a set of binary distributions organized in a spatial grid. The rendered map $\mathbf{m}$ depends on the trajectory used to render it. The joint map and path entropy is defined in Stachniss et al. [2005] as

$$H(\mathbf{x}, \mathbf{m}) = H(\mathbf{x}) + \int p(x) H(\mathbf{m}|\mathbf{x} = x) dx, \tag{5.3}$$

where $H(\mathbf{x})$ is the trajectory entropy, and $\int p(x)H(\mathbf{m}|\mathbf{x} = x)dx$ is the entropy average of all possible maps generated with all possible trajectories.

However, averaging over the entropy of all the (infinite) resulting maps corresponding to all the (infinite) trajectory samples in the probability distribution is unfeasible. An approximation should be adopted instead.

Such approximation must consider the following. In case of uncertain trajectory estimation, $p(x)$ is sparse and the entropy of most of the maps associated to each of the trajectory samples would be very high. This would lead to a higher averaged map entropy term in (5.3). Conversely, a very accurate trajectory estimation would lead to an average of very similar and accurate maps due to the concentration of the probability distribution. According to this, we propose the following approximation

$$H(\mathbf{x}, \mathbf{m}) \approx H(\mathbf{x}) + \alpha\big(p(\mathbf{x})\big) H(\mathbf{m}|\mathbf{x} = \boldsymbol{\mu}). \tag{5.4}$$

Here, the factor $\alpha(p(\mathbf{x}))$ aims at approximating the effect of averaging the map entropy over the whole trajectory distribution. Our approach to simulate this effect is to set the factor to the inverse of the determinant of the marginal covariance of the current robot pose estimate, $\alpha(p(\mathbf{x})) = |\boldsymbol{\Sigma}_{tt}|^{-1}$, where $|\boldsymbol{\Sigma}_{tt}|$ may be seen as a scalar measure of the localization uncertainty. Although this does not measure the whole trajectory uncertainty but only its last robot pose, its computational cost is very reduced. It also provided good results in our experiments.

For better readability, from now on we define $\mathbf{m}$ as the map rendered considering the current trajectory mean $\boldsymbol{\mu}$, thus $H(\mathbf{m}) = H(\mathbf{m}|\mathbf{x} = \boldsymbol{\mu})$.

Our approximation differs from the used in [Valencia et al., 2012], where $\alpha(p(\mathbf{x}))$ was not considered. Using our approximation, we experimented interesting behaviours in methods derived from minimizing the joint map and trajectory entropy. The exploratory and re-localization forces are balanced and modulated according to the current localization uncertainty, reducing the sensibility of the algorithm to environment specificities.

Since the trajectory estimation follows a multi-variate Gaussian distribution, the trajectory entropy $H(\mathbf{x})$ as introduced in Chapter 3 can be defined in terms of either covariance or information matrix as

$$H(\mathbf{x}) = \log\left((2\pi e)^{n/2}|\mathbf{\Sigma}|\right) = \log\frac{(2\pi e)^{n/2}}{|\mathbf{\Lambda}|}, \tag{5.5}$$

where n is the dimension of the Gaussian distribution, i.e., 3 times the number of poses in the trajectory in the 2D case.

The map entropy, as defined in [Stachniss et al., 2005], is the summation of the entropy of all binary random variables of each cell. However, in order to make it independent from the grid resolution, it can be weighted due to its size as proposed in the same paper and posteriorly used by Valencia et al. [2012]

$$H(\mathbf{m}) = l^2 \sum_{c\in\mathbf{m}} \Big(p(c)\log p(c) + \big(1-p(c)\big)\log\big(1-p(c)\big)\Big), \tag{5.6}$$

where l is the grid resolution and c are the grid cells.

Ultimately, active pose-graph SLAM becomes a search for the sequence of robot motions that produces the largest joint map and trajectory entropy reduction. To tackle it, we divide our work into two different research lines: search in configuration space and search in action space exposed in the following sections.

The **search in configuration space** relies on the computation of a dense estimation of the joint entropy decrease for all valid robot configurations. Since the estimation is dense, it performs an exhaustive search. In exchange, it entails ignoring the entropy change along the path to reach each configuration.

On the other hand, a **search in the action space** allows taking into account the entropy change along the path. However, it undermines the search completeness that can be achieved in a reasonable amount of time. Optimal path planning algorithms are suited to tackle it.

5.3 Search in configuration space

Some exploration and active SLAM approaches assume omnidirectional sensors. In a more general setting, sensors such as certain laser range finders or cameras have a narrow field of view, and hence, we need to deal with full poses, not just positions. Hence the search in the whole configuration space (C-space). As previously stated, our work is focused in the 2D case. Therefore, the C-space is a 3D space corresponding to robot position x, y and orientation θ. To tackle this problem numerically, the C-space is discretized in a 3D grid. Each cell of this grid corresponds to a configuration.

5.3.1 Dense entropy decrease estimation

The entropy decrease after going from the current pose to any given C-space configuration depends of the path taken to arrive to such pose. Different routes induce different decrease values of trajectory and map entropies. Take for instance two different routes to the same pose, one that goes close to previously visited locations and one that discovers unexplored areas. In the first, the robot would be able to close loops, and thus maintain bounded localization uncertainty. In the second, an exploratory route would reduce the map entropy instead.

There are infinite paths from the current position to a specific configuration, and each one induces a different effect on the entropy. This would lead to an exhaustive search of the optimal path to each configuration, which would be intractable. Instead, we consider the joint entropy decrease after appearing, or “turning up”, in each configuration. That is, not considering the effect of the path.

We are not interested in the joint entropy at one instant, but only on its change (its decrease) with respect to the current entropy, after “turning up” in each configuration. Figure 5.3 shows an example of the dense entropy decrease estimation. Red cells depict higher expected entropy decrease values after appearing at such configurations. On the contrary, blue cells depict configurations in which the joint entropy decrease is expected to be negligible. As shown in the figure, entropy decrease can be due to map discovery (map entropy decrease) or re-localization (trajectory entropy decrease) or a combination of both.

For the estimation of the overall joint entropy decrease we need only to evaluate separately the decrease of the two terms in (5.4), i.e., trajectory and map entropies, for each configuration in C-space, and balance the second using $\alpha(p(\mathbf{x}))$ as described. The following is a detailed explanation of how a dense estimation of each term is computed.

Trajectory entropy decrease estimation

Considering the “turning up” assumption, the jump from the current pose to each free configuration in the C-space will produce the same marginal posterior, i.e. the same trajectory entropy change, except when a loop closure can be detected at that configuration. Thus, we set the trajectory entropy decrease estimation of each 3D cell distinguishing between two

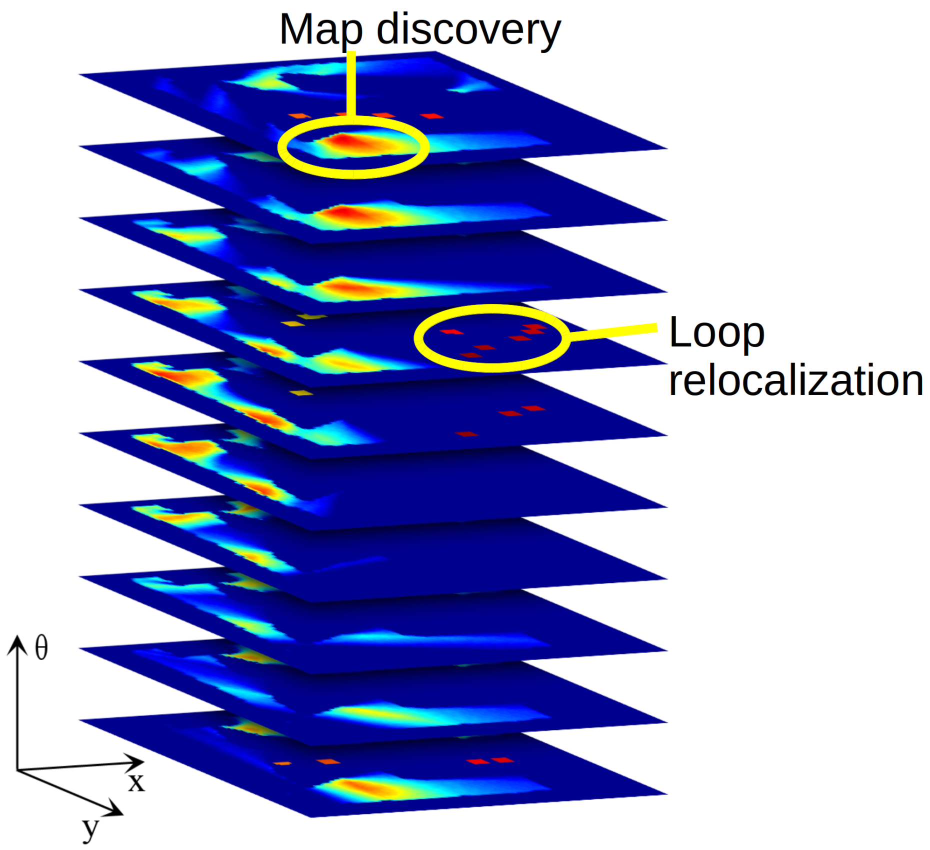


Figure 5.3: Example of a dense entropy decrease estimate in C-space. Each layer represents a slice of the C-space at a different orientation value. Red areas indicate candidate configurations with maximum entropy reduction values.

cases:

$$\Delta H(\mathbf{x}) = \begin{cases} \Delta H_{LC}(X) & \text{if a loop with the corresponding configuration can be closed,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

Therefore, it is necessary to identify all configurations in which a loop can be established with any trajectory node of the pose-graph. To do so, we define the match area of the sensor as the intervals in x, y and θ where loops can be closed. Thus, a loop can be closed in each configuration in the C-space inside the match area of any previous pose of the trajectory. Then, for each trajectory pose in the pose-graph, we annotate the C-space cells inside their match area with the corresponding entropy decrease.

This decrease is computed as follows. Similarly to the information gain used by Ila et al. [2010], the entropy change after a loop closure in pose-graph SLAM is

$$\Delta H_{LC}(\mathbf{x}) = -\log |\mathbf{I} + \mathbf{J}_k \mathbf{\Sigma} \mathbf{J}_k^\top \mathbf{\Omega}_k|, \tag{5.8}$$

being $\mathbf{\Omega}_k$ the information matrix of the loop closure measurement noise (see Appendix A, Prop. 6). Note that the computation of the trajectory entropy decrease at loop closure requires the covariance matrix $\mathbf{\Sigma}$ which is not directly available using pose-graph SLAM. Let us recall the Jacobians sparsity which only contains two non-zero blocks corresponding to the two observed pose-graph nodes (say i and j), $\mathbf{J}_k = [0 \cdots J_{ki} \cdots J_{kj} \cdots 0]$. Considering this, in order to compute 5.8, the covariance matrix is not required entirely. Only the the diagonal blocks $\mathbf{\Sigma}_{ii}$, $\mathbf{\Sigma}_{jj}$ and the off-diagonal block $\mathbf{\Sigma}_{ij}$ corresponding to the loop closing nodes are required:

$$\mathbf{J}_k \mathbf{\Sigma} \mathbf{J}_k^\top = \begin{bmatrix} J_{ki} & J_{kj} \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_{ii} & \mathbf{\Sigma}_{ij} \\ \mathbf{\Sigma}_{ji} & \mathbf{\Sigma}_{jj} \end{bmatrix} \begin{bmatrix} J_{ki}^\top \\ J_{kj}^\top \end{bmatrix}.$$

Following the “turning up” assumption, the covariance used to simulate the loop closure after appearing in each configuration is taken equal to the current configuration. Hence, to evaluate (5.8) for all trajectory nodes, all the diagonal blocks and the right-most block column of $\mathbf{\Sigma}$ must be recovered. This can be efficiently performed as in [Ila et al., 2010, 2015; Kaess and Dellaert, 2009].

To summarize, the dense trajectory entropy decrease estimation is built with a few steps. First, the block diagonal and right-most block column of the covariance matrix is recovered. Afterwards, for each node of the pose-graph trajectory, the entropy decrease of a loop closure (5.8) is evaluated. Then, the resulting value is annotated in the cells that are around the trajectory node within the match area. The rest of the cells remain with null trajectory entropy change.

More sophisticated approaches for trajectory entropy change in open loop could be devised. For instance, in order to approximate the noise propagation, the trajectory entropy change could be set as a growing value along the distance from the current robot configuration. However, the trajectory entropy does not necessarily increase in open loop. Depending on the relative uncertainty of the current robot pose estimation w.r.t. the average trajectory, the determinant in (5.5) can produce a decreasing entropy when, in fact, new nodes are injecting noise to the state. In our opinion, this entropy decrease happens since entropy is measuring the “average uncertainty” of a trajectory that is increasing its number of nodes. Comparing entropies of trajectories of different number of nodes is not straightforward since it means comparing the entropy of multi-variate Gaussian distributions of different dimensions. We are content with evaluating the most significant trajectory entropy changes, which happen in loop closures, avoiding the comparison of entropies in open loop.

Map entropy decrease estimation

According to the map entropy (5.6), the change in map entropy is attained after moving to a new configuration and changing the cells probability $p(c)$. Entropy will be decreased after classifying unknown cells ($p(c) \approx 0.5$), either to obstacle ($p(c) \approx 1$) or free ($p(c) \approx 0$).

Frontiers are widely used in exploration literature [Yamauchi, 1997]. They are defined as the boundary between the unknown and free areas. Trivially, unknown cells can only be classified thru frontiers.

Estimating the amount of cells that will be discovered behind a frontier is challenging, but it is heavily linked to the size of the frontier visible to the sensor. We are content with approximating entropy reduction as the map entropy reduction after classifying the frontier cells visible to the sensor from each robot configuration.

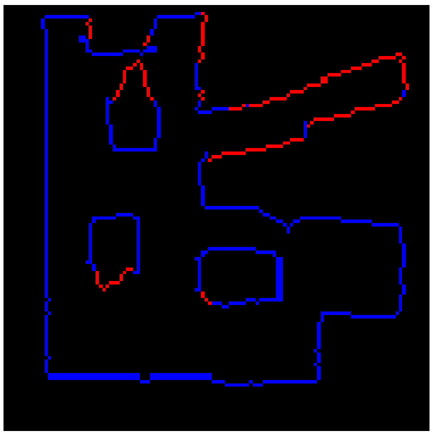
Some considerations can be made in order to improve the efficiency and the accurateness of this computation.

- The same robot position with different orientations share several beam directions of both scans.
- Close frontier cells can be discovered by different beams of the same scan and should be computed once.
- The use of image convolutions is an efficient tool to take benefit from.

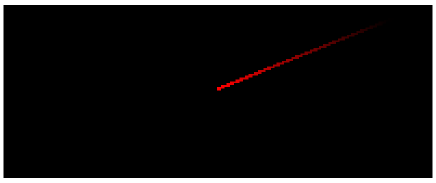
Taking these considerations into account, we designed a method to efficiently build a dense map entropy decrease over the discretized C-space (3D grid). First, a ray casting 2D grid is computed for each of several discretized laser beam directions. Afterwards, for each robot orientation layer, the corresponding laser beam directions are summed. The result is a 3D grid corresponding to the C-space that contains the amount of frontier cells that can be observed from each configuration.

This method is detailed below with figures to exemplify the process in a specific case of frontiers and obstacles and a beam direction of 202°.

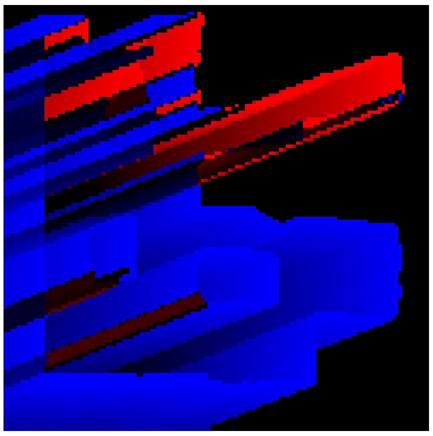
1. *Ray casting grid*: First, we want to compute the ray casting for all positions and all laser beam directions. That is, a 3D grid which dimensions are x, y, and the direction of each laser beam. Each layer of this grid corresponds to one discretized beam direction of the whole 360°. For each beam direction layer, we want to annotate for each position (x, y) if the nearest non-free cell along the beam direction is a frontier or not (an obstacle). For each beam direction layer, this can be computed using a one-dimensional convolution motion kernel as explained below.



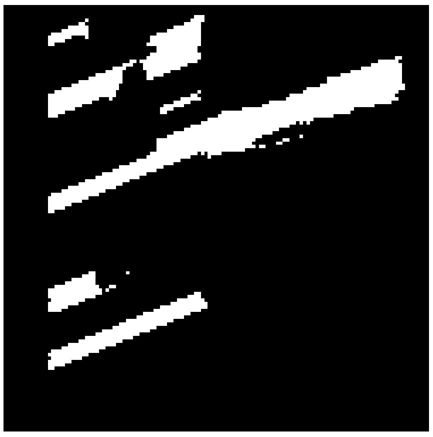
We start the process from an image containing ones (red) in frontier cells and a minus ones (blue) in obstacles. The rest of the cells contain zeros (black).



We use a kernel that contains zeros everywhere except for the cells in the corresponding beam direction which have positive exponentially decreasing values.



The result of a convolution of this kernel over the previous image contains positive values (red) for those cells such that their closest non-free cell in the beam direction is a frontier.
Otherwise, those cells such that the closest non-free cell in the beam direction is an obstacle contain negative values (blue).

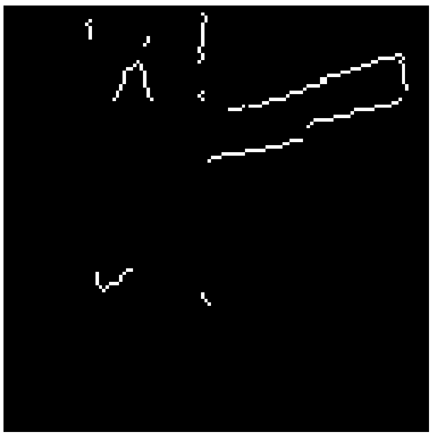


After binarizing the positive values, we obtain the cells from which a frontier can be observed in the beam direction. The values in unknown cells are set to zero since we only consider computing the map entropy change for free configurations.

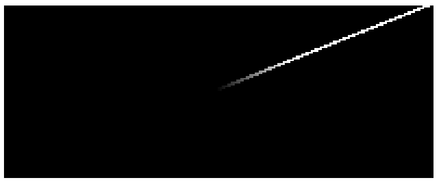
2. *Multiple beams compensation*: As previously stated, each frontier cell will be observed by a different amount of beams depending on how far is the sensor. Knowing the cell size l and the angle between two consecutive laser beams β, the amount of beams b that cast at the same cell is inveresly proportional to the distance r from the robot to that cell

$$b = \frac{l}{r \tan \beta} .$$

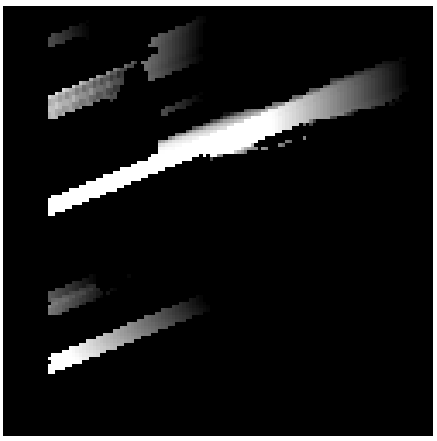
Therefore, before proceeding to sum different beam directions corresponding to laser angle of view at each robot orientation, the previous ray casting should be compensated in order not to overestimate the number of frontier cells being observed.



For each beam direction layer, a convolution is performed over an image with ones (white) at frontier cells and zeros (black) at the rest.

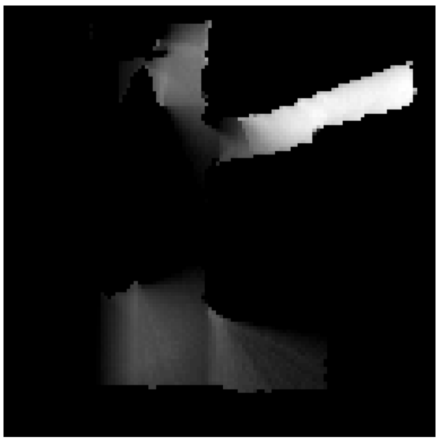


Again, the kernel is full of zeros except for the cells in the corresponding beam direction. In this case, these cells contain the compensation: $min(1, r \tan \beta / l)$, being r the distance to the kernel's center.



The result of this convolution is multiplied by the corresponding binary ray casting layer to discard occlusions. Finally, we obtained a compensated ray casting layer.

3. *Map entropy decrease grid.* Finally, we are able to compute the dense map entropy decrease estimation. That is, a 3D grid corresponding to the discretized C-space containing the map expected entropy decrease after appearing into each configuration. This is performed layer by layer, i.e. for each robot orientation.



Each orientation layer is obtained by adding the compensated ray casting beam directions corresponding to that orientation and the sensor angular aperture.
Since the result of the summation is the amount of frontiers cells discovered from each configuration, the entire 3D grid is weighted by the entropy change of discovering a single cell from unknown (to either free or obstacle), following (5.6).

Finally, to compute the joint entropy decrease, before summing the map entropy decrease estimation grid and the trajectory entropy decrease estimation grid, the former is weighted by the inverse to the determinant of the marginal covariance at the current configuration $\alpha(p(\mathbf{x})) = |\mathbf{\Sigma}_t|^{-1}$ according to (5.4).

Frontier-based strategies assume that the most map entropy decreasing configurations are just in front of a frontier. However these poses are not necessarily close to frontiers. Actually, the most map entropy decrease locations are normally far from frontiers (see Fig 5.4).

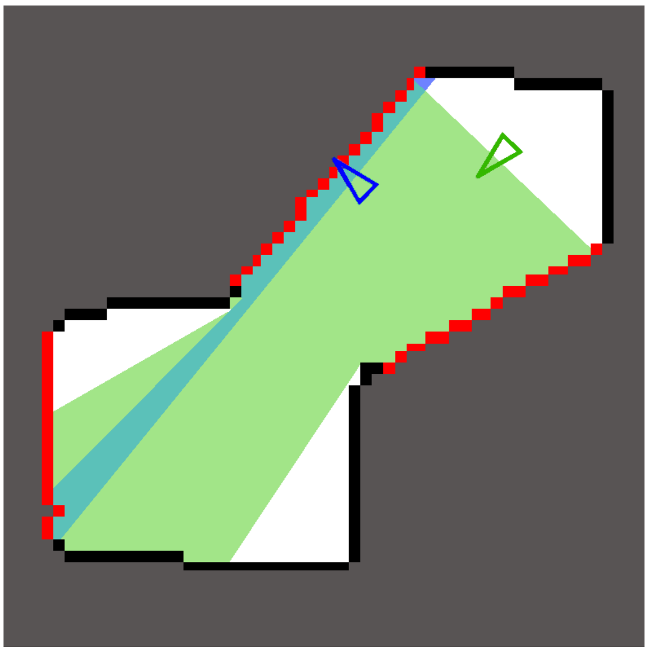


Figure 5.4: Exploration goal at a frontier (blue) and optimal map entropy reduction goal in C-space (green). Frontier cells are depicted in red, obstacles in black, and the laser scan ray-tracing of both configurations in their corresponding faded color.

5.3.2 Planning for joint entropy minimization

We propose two different ways of using the dense entropy decrease estimation in active pose-graph SLAM. Both strategies drive the robot with the objective of minimizing both map and trajectory entropies, i.e., maximizing coverage while maintaining the robot well localized. The first strategy builds an entropy-decrease potential field and plans a trajectory towards a minima using a gradient descent method. The second approach directly chooses the most entropy decrease configuration as the goal and plans a trajectory through the free space to reach it.

Gradient descent on an entropy decrease field

In gradient descent motion planning methods, the objective is to find a scalar function ϕ defined over all C-space cells such that its gradient $\nabla\phi$ will drive the robot to a minimum. In our case, the gradient of ϕ will drive the robot to the configuration corresponding to a joint trajectory and map entropy minimum.

Shade and Newman [2011] defined a potential scalar function using attraction and repulsion fields on frontiers and the current robot pose, with some boundary conditions on obstacles. Choosing frontiers as attractors poses some challenges. Frontier cells have a significant probability of being yet unseen obstacles. The use of potential fields to reach frontiers produces perpendicular robot configurations at the arriving locations, thus making the robot potentially face these new obstacles directly, with the consequent unavoidable collision. Moreover, as previously stated, normally the most entropy decrease configuration does not remain in the frontiers (see Fig 5.4).

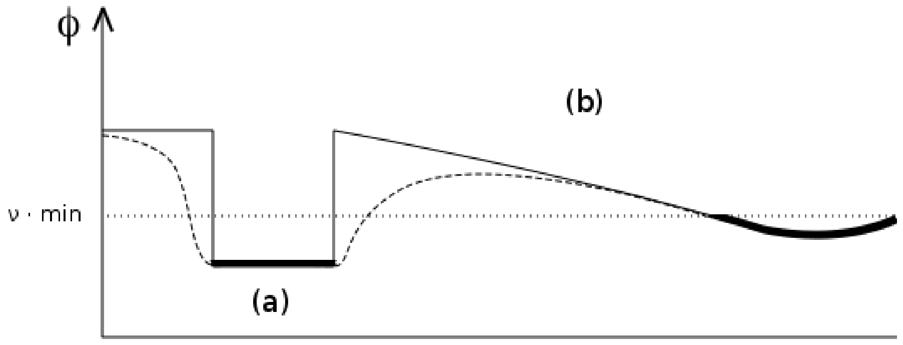


Figure 5.5: The entropy decrease grid is cropped at a desired value v and smoothed with a harmonic function to produce the desired information potential field. Zone (a) represents a region with steep entropy reduction due to a possible loop closure. Zone (b) represents an exploratory area.

In contrast with this approach, we propose to build a potential scalar function from the joint entropy decrease estimation. For it to become the scalar function ϕ, one more step is required. The dense entropy decrease estimation grid is turned into a potential field by performing some smoothing convolutions using an harmonic function of the form

$$\phi_{x,y,\theta} = \frac{1}{6}(\phi_{x-1,y,\theta} + \phi_{x+1,y,\theta} + \phi_{x,y-1,\theta} + \phi_{x,y+1,\theta} + \phi_{x,y,\theta-1} + \phi_{x,y,\theta+1}), \tag{5.9}$$

which basically performs a mean of all the neighbor cells in the C-space grid. To prevent the smoothing from removing small but high entropy decrease configurations, attraction areas are defined as those under the threshold of 60% of the minimum entropy value. The joint entropy decrease values of these cells are reset after each smoothing convolution (see Fig 5.5).

In our computation of the entropy grid we have considered obstacles to adequately propagate entropy change along sensor rays taking into account occlusions, but we have still not penalized configurations that get close to them. To this end, we resort to the use of boundary conditions as in [Shade and Newman, 2011], with the difference that instead of using Neumann boundary conditions to guarantee flow parallel to obstacles, we still want some repulsive perpendicular effect from them. This effect can be achieved by mirroring weighted inner cell values near obstacles. A unitari weight means parallel traverse along the obstacle boundary, and larger values induce repulsion. In the method reported here we start each planning step with low repulsion to reduce bottlenecks at local minima and increasing it and re-planning in case a collision is detected. The final path is obtained traversing the gradient descent steps.

The most interesting feature using gradient descent planning is the resulting path rather than the final configuration. While the convergence to the largest joint entropy reduction configuration cannot be assured, the path taken goes thru the potential field valleys, i.e. thru the most entropy decreasing configurations. For instance, it can produce a path that closes some loops before reaching an exploratory goal. Or, conversely, a path to a loop closures that explores some new areas along the travel.

Nevertheless, gradient descent methods perform poorly in large environments with many obstacles, requiring huge smoothing iterations and suffering severely of bottleneck effect and local minima (doors or thin passages). Also, holonomic traverse is assumed in this planner, providing paths that are not realizable by most of mobile robots.

RRT* to the configuration with largest entropy decrease

A second option explored in our work is to use a fast and asymptotically optimal planner like RRT* [Karaman and Frazzoli, 2010] to drive the robot to the largest joint entropy decrease configuration. This method renounces to the most entropy decreasing path and chooses instead the shortest path in the free C-space meeting non-holonomic restrictions if needed.

The RRT* planner does not suffer from local minima and it does not require the above-mentioned smoothing iterations. Then, it is significantly faster than gradient descent method.

RRT* is probabilistically complete, meaning that the probability of producing a solution (if it exists) tends to 1 as time tends to infinity. Also, it is asymptotically optimal, i.e. the solution converges to the optimal solution. However, the time to find a solution and to optimize in terms of cost (distance in our case) is unknown and depends on the environment complexity and size.

In our application, we prioritize having a quite good solution with bounded computational cost rather than the best solution. Consequently, in our implementation, if RRT* fails to get a path to the most entropy decrease configuration after a number of iterations, the path of the tree that leads to the largest joint entropy decrease configuration is chosen.

5.3.3 Results

Several simulations were performed to compare the two proposed active SLAM methods based on the entropy decrease estimation (EDE) against a frontier-based exploration [Yamauchi, 1997]. The frontier-based method drives always the robot to the closest frontier larger than a threshold, without considering the localization and map uncertainties. In our simulation the frontier size threshold was fixed at 5 cells. When there are no frontiers of this size, this threshold is reduced to 1. After the frontier-based method produces a goal, the RRT* planner is used to compute the path.

Simulations were executed in two commonly used environments of varying size and complexity, the Cave and Freiburg maps [Howard and Roy]. In all cases, robot motion was estimated with an odometric sensor with noise covariance $\mathbf{\Sigma}_u = \text{diag}(0.1\,\text{m}, 0.1\,\text{m}, 0.0026\,\text{rad})^2$. The robot is fitted with a laser range finder. Laser scans were simulated by ray casting over the ground truth grid map of the environment using the ground truth robot location. Loop closure constraints were computed using the iterative closest point algorithm fixing the noise covariance at $\mathbf{\Sigma}_y = \text{diag}(0.05\,\text{m}, 0.05\,\text{m}, 0.0017\,\text{rad})^2$. The loop closure match area was fixed at $\pm 1\,\text{m}$ in x and y, and $\pm 0.35\,\text{rad}$ in orientation. That is, the configuration area for which we can guarantee that a loop closure between two poses can be established.

All experiments were carried out with a Quad core Intel Xeon system at 3.10GHz and with 8GB of memory.

	Frontier-based RRT*	**EDE gradient descent**	**EDE RRT***
Computation time:	2814.14 s	601.02 s	487.15 s
Loops closed:	15.4	22	17

Table 5.4: Average computational time and amount of loops closed for all compared exploration strategies in the cave map.

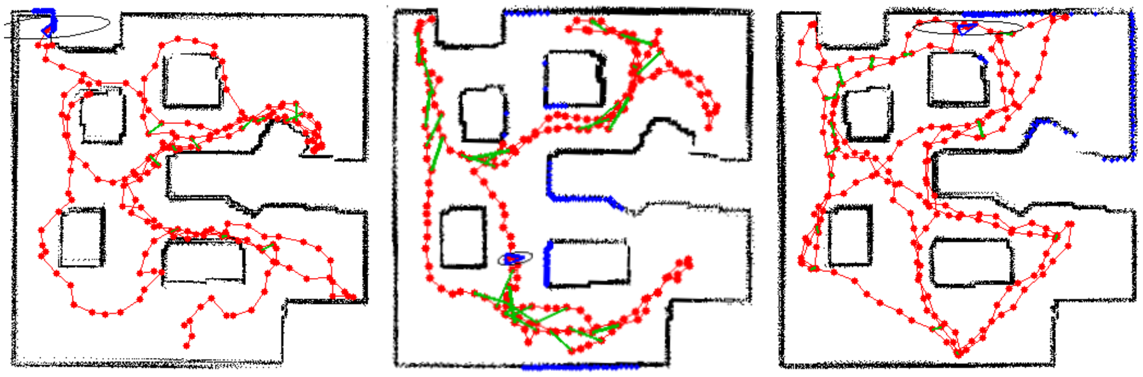

(a) Frontier-based with RRT* **(b)** EDE with gradient descent **(c)** EDE with RRT*

Figure 5.6: Final trajectories after one 200 steps simulation of each exploration strategy in the cave map. In red the robot trajectory, in green the loop closure links, and in black the occupancy map rendered at the last trajectory estimate.

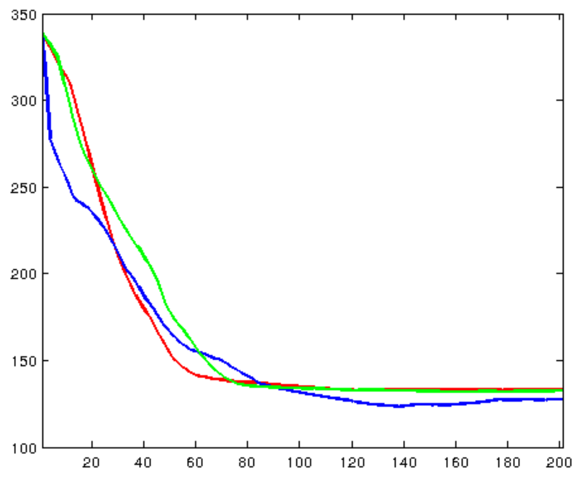
Exploration in the cave map

The cave map is a simple scenario consisting of a single room resembling an industrial plant or a house room. In our simulations, the map was scaled to a resolution of 20 m × 20 m. To account for random effects of the motion and sensor noises as well as of the RRT* tree growth, each simulation was executed 5 times for each exploration strategy and limited to 200 SLAM simulation steps and 100 planning steps.

Table 5.4 contains, for each compared method, the average values of the computational time of the whole experiment and the amount of loops closed. Since frontier-based strategies do not consider the trajectory entropy, the accumulation of localization error produces erroneous maps mostly around obstacles and frontiers. Planning over these maps complicates the finding of free trajectories to the goals, resulting in large computation times for the frontier-based strategy.

The plots in Figure 5.6 show one realization of the experiment for each strategy. Red dots and lines indicate the executed trajectories. Green lines indicate loop closures and the black dots are the accumulated laser scans according to the trajectory estimate.

It can be observed how the frontier-based strategy results in many collisions since frontiers are mostly misclassified due to the larger trajectory uncertainty. In contrast, EDE with gradient descent tends to produce valleys of high confidence. It is due to both obstacle repulsion and attractive trajectory entropy decrease configurations. Instead, the resulting paths of EDE with RRT* do not consider the effect on the entropy along the path. Alternation of exploratory and revisiting goals is clearly depicted.


(a) Map entropy.

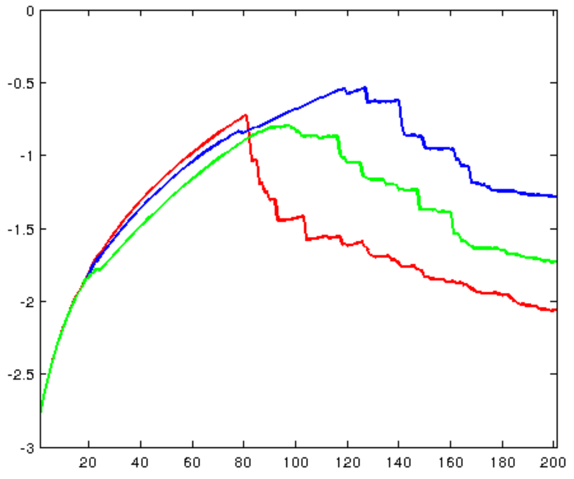

(b) Trajectory entropy.

Figure 5.7: Average entropies of 5 simulation runs in the Cave map. (blue) Frontier-based with RRT*. (green) Gradient descent on the EDE field. (red) EDE with RRT*.

Figure 5.7 shows the average map and trajectory entropy evolution for the three strategies. Similar map entropy (full coverage) is reached by all three methods while the entropy-based proposed methods significantly reduce the trajectory entropy compared to the frontier-based strategy. Surprisingly, all methods end up closing similar number of loops on average, the different trajectory entropy values indicate that the quality of those loops closed is significantly better for the EDE strategies. This happens because the frontier-based strategy closes loops only by chance resulting in a final map of worse quality.

Exploration in the Freiburg map

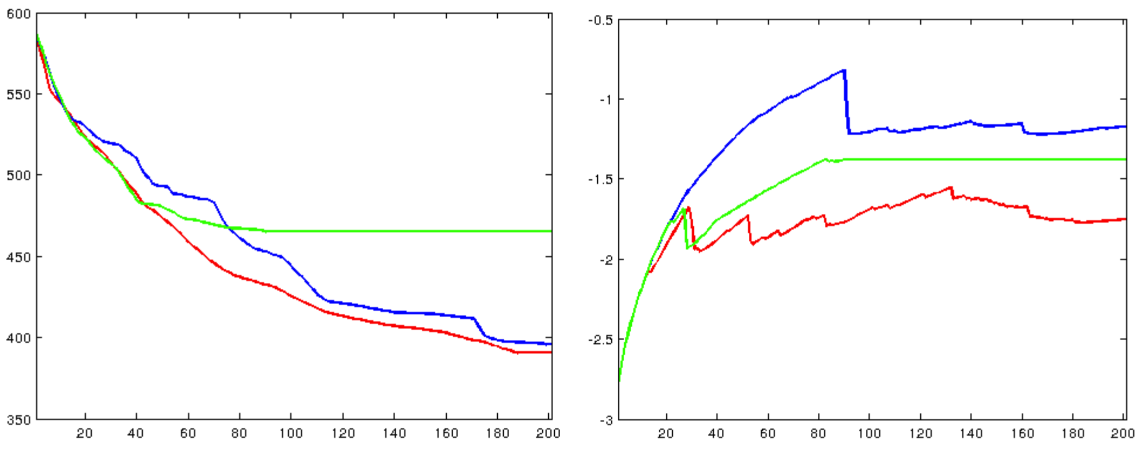
The second environment analyzed is the Freiburg indoor building 079, of a significant larger complexity and size. Three simulations were launched fixing the SLAM steps limit at 200 and the planning steps at 100.

Table 5.5 contains the average computation time as well as the amount of loops closed for each method. The plots in Figure 5.8 show average map and trajectory entropy evolution, respectively, of the 3 simulation runs for the three exploration strategies.

Given the highly complex nature of this environment, EDE with gradient descent was the worst performing method in this case, rapidly getting trapped in local minima (thin doors). This is depicted by the flat green line in frame (a).

	Frontier-based RRT*	**EDE Potential fields**	**EDE RRT***
Computation time	2505.44 s	2586.84 s	11951.18 s
Loops closed:	12.5	3	12.5

Table 5.5: Average computational time and amount of loops closed for all compared exploration strategies in the Freiburg building.



(a) Map entropy. **(b)** Trajectory entropy.

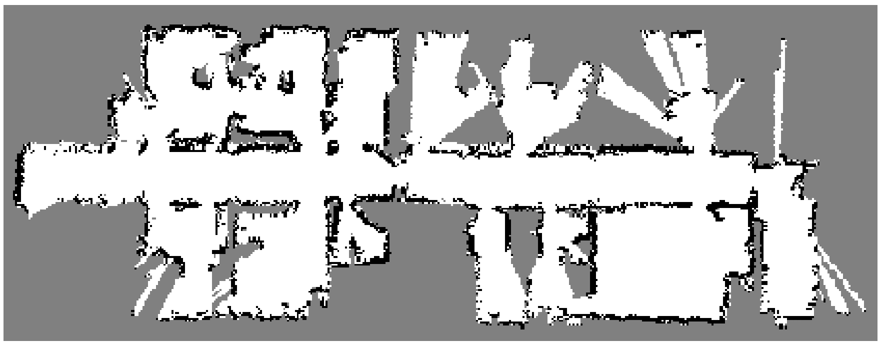
Figure 5.8: Average entropies for the 3 simulation runs in the Freiburg map. In blue, the frontier-based with RRT*; in green, EDE with potential gradient descent; in red, EDE with RRT*.

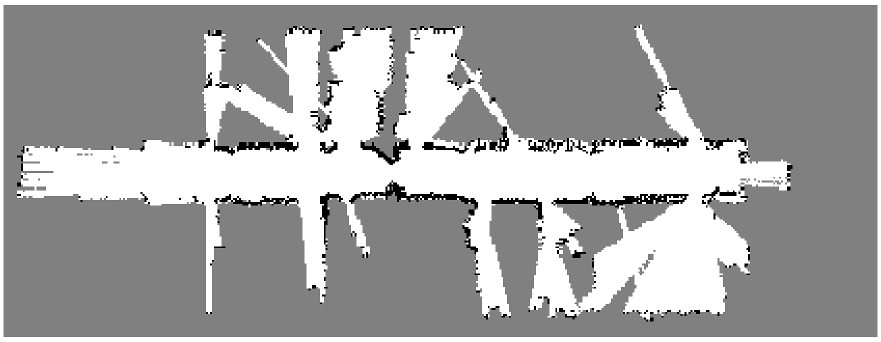
The frontier-based method is outperformed by EDE with RRT* both in map and trajectory entropies since it does not plan revisiting actions. In exchange, performing of the entropy decrease estimation for a bigger environment has significant computational cost.

Figure 5.9 shows one realization of the rendered occupancy map for each of the three strategies. The figure clearly depicts the various conclusions that had been already mentioned. Frontier-based exploration does not consider loop closing and thus produces maps with larger uncertainty, i.e., thicker grayish walls. Gradient descent on the EDE field ends up trapped in local minima. And, EDE with RRT* produces larger and more accurate maps.

Figures 5.10 and 5.11 show intermediate steps in the computation of the EDE-RRT* map. The first figure shows one instance of the computed map and the pose-graph SLAM trajectory estimation. Note how the pose graph effectively covers the whole explored area with only a few informative links between poses (green lines). The second figure shows an instance of the RRT* in green covering all free space and the path to the most entropy decrease configuration in blue. Long paths are normally obtained using this method.

EDE with RRT* does not consider visiting non optimal configurations that are closer and chooses always going to the one that produces the most entropy decrease. The distance of the path to each configuration, thus time spent on reaching it is not considered. Penalizing the path distance may lead to even better performance due to adding the temporal factor in the optimization.


(a) Frontier-based with RRT*


(b) EDE with gradient descent

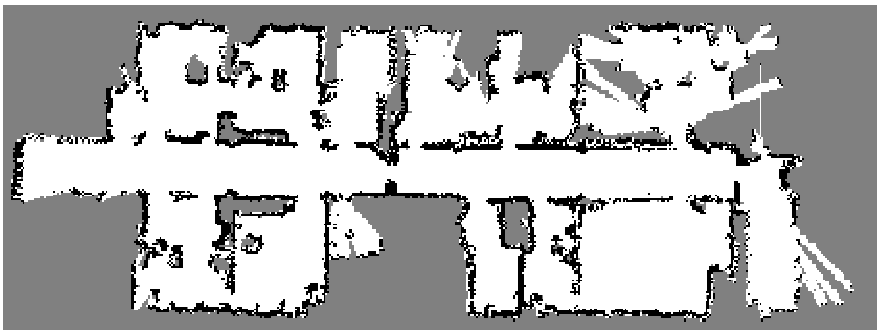

(c) EDE with RRT*

Figure 5.9: Occupancy grid map built after a 200 steps simulation in the Freiburg environment.

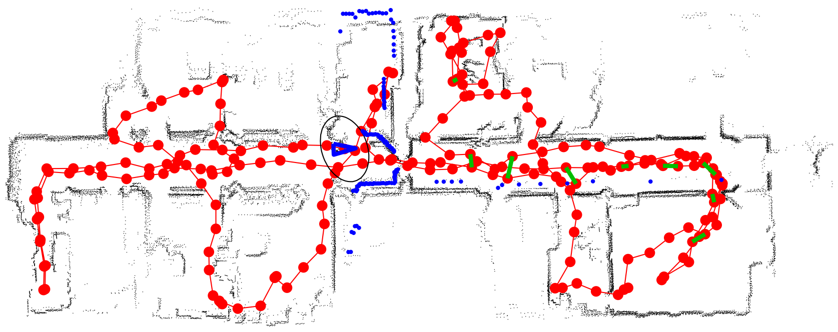


Figure 5.10: EDE-RRT* pose-graph SLAM. The red dots and lines indicate the robot trajectory. The blue dots represent the current sensor reading, and the blue triangle and the hyper-ellipsoid indicate the current robot pose estimate.

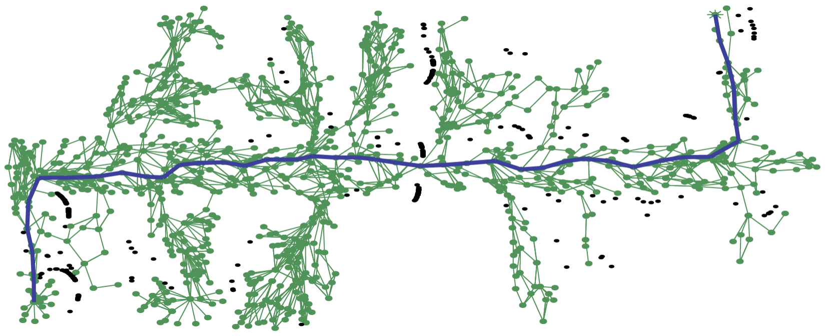


Figure 5.11: This is an instance of the RRT* tree and the computed path to the goal (blue) in the Freiburg map. The black dots indicate failed leaf extensions due to collision.

5.4 Search in action space

As explained in the previous section, searching in the C-space avoids taking into account the entropy changes along the planned path. It forces the strong assumption of only considering the entropy change after appearing at each evaluated configuration.

Therefore, if the effect of the path (actually, the paths) to reach a configuration want to be taken into account, we should switch to searching in action space (A-space). Action selection in Active SLAM has been approached in the past as the analysis of the effect on the joint entropy of a small heuristically chosen set of path candidates [Stachniss et al., 2005; Valencia et al., 2012]. In both cases, the very small sets of action candidates were built computing paths to revisitng configurations and frontier-based poses. This entails two main inconvenients.

First, since it is goal-based, the action set is built under the idea of a single purpose: exploration or re-localization. Considering that searching in the A-space allows taking into account the effect of the path, it is somehow renouncing to this possibility from the very begining. In our work, we consider paths that may accomplish both purposes by closing a loop in the way of an exploratory configuration or vice-versa.

Second, the optimal path in terms of joint entropy reduction it is not likely to be in a very small set of heuristically decided candidates. Our approach is based on creating a large action candidate set without assumptions about where the optimal configurations are. For instance, as seen in the previous section, optimal exploratory configurations are very usually far from frontiers.

Karaman and Frazzoli [2010] presented the asymptotically optimal planner RRT* designed for path planning given a goal and using cost functions that satisfy the triangular inequality. The expansive nature of the RRT* algorithm suggests that it can be useful in the action set generation for Active Pose SLAM. We propose to benefit from the RRT* tree expansion strategies and explore the use of an entropy-based cost function for tree expansion.

5.4.1 Entropy decrease estimation for several action candidates

We propose a method to estimate the joint entropy decrease of several action candidates. Also, we want to evaluate each action candidate taking into account the effect of the whole path. Consequently, the entropy decrease estimation will be performed considerable times, thus it demands further efficiency in its computation.

At the same time, the map and trajectory entropy decrease estimations used in the previous section did not take into account accumulation along the path. That map entropy decrease estimation compensates multiple observation of the same frontier cell by different laser scan beams of the same robot configuration but not from different planned path configurations. Analogously, trajectory entropy did not consider how the loop closure entropy change is reduced when closing other loops with other nodes in the vicinity.

In conclusion, the map and trajectory entropy decrease estimation methods for C-space methods described in sec. 5.3 should be redesigned for A-space search application.

We refer to specific parts of the state $\mathbf{x}$ using subindexes. So for instance, after some experiment steps, the robot performed a trajectory $\mathbf{x}_{1:t}$, which is estimated by the pose-graph SLAM. At this point we can render an occupancy map $\mathbf{m}_t$ from its means $\boldsymbol{\mu}_{1:t}$. From this moment, an action (or path) candidate $\mathbf{a}$ is defined as a sequence of relative motions $\mathbf{a} = \mathbf{u}_{t+1:T}$, which would produce a sequence of new robot configurations $\mathbf{x}_{t+1:T}$, from where tentative measurements $\mathbf{z}_{t+1:T}$ would be made, and an expected map $\mathbf{m}_T$ would be rendered.

In the previous section, the entropy decrease was estimated for all C-space configurations but ignoring the entropy change along the path to reach them ($T = t + 1$). In this case, the evaluation of candidates (paths) of different sizes will take place. Consequently, we propose the minimization of the joint entropy change divided by the distance of the path candidate

$$\mathbf{a}^* = \operatorname{argmin} \frac{H(\mathbf{x}_{1:T}, \mathbf{m}_T) - H(\mathbf{x}_{1:t}, \mathbf{m}_t)}{dist(\mathbf{u}_{t+1:T})}. \tag{5.10}$$

In other words, we will take the candidate that maximizes the information gain per meter traveled. Additionally, adding the distance (thus, time) into the cost, the consideration of non-optimal but close configurations may take place in contrast with the methods proposed in the previous section.

Then, at the planning step t the current joint entropy term of (5.10) is computed as in the previous section (5.4)

$$H(\mathbf{x}_{1:t}, \mathbf{m}_t) \approx H(\mathbf{x}_{1:t}) + |\boldsymbol{\Sigma}_{tt}|^{-1} H(\mathbf{m}_t). \tag{5.11}$$

It can be evaluated directly using the current pose marginal covariance $\boldsymbol{\Sigma}_{tt}$ as well as evaluating the map entropy of the current map (5.6) and the trajectory entropy (5.13) using the current trajectory estimation marginals.

To compute the expected joint entropy after executing an action candidate –the second numerator term in (5.10)–,

$$H(\mathbf{x}_{1:T}, \mathbf{m}_T) \approx H(\mathbf{x}_{1:T}) + |\boldsymbol{\Sigma}_{TT}|^{-1} H(\mathbf{m}_T), . \tag{5.12}$$

three components are required: the trajectory entropy $H(\mathbf{x}_{1:T})$, the map entropy $H(\mathbf{m}_T)$ and the marginal covariance of the last node $\boldsymbol{\Sigma}_{TT}$, necessary to compute the factor $\alpha(p(\mathbf{x}))$.

It is important to consider that a part of a path candidate $\mathbf{a}_k = \mathbf{u}_t{+}1 : k$ can be considered a path candidate as well. Also, the set of candidates will be generated by the RRT* algorithm, then the evaluation should exploit the tree morphology of the action set. Therefore, our approach is to compute the three required components $(H(\mathbf{m}_k), H(\mathbf{x}_{1:k}), \boldsymbol{\Sigma}_{kk})$ incrementally from the corresponding values of the previous path candidate step $(H(\mathbf{m}_{k-1}), H(\mathbf{x}_{1:k-1}), \boldsymbol{\Sigma}_{k-1k-1})$. The following is an exhaustive description of how we efficiently compute these three components.

5.4.2 Trajectory entropy decrease and last node's marginal covariance estimation

The trajectory entropy $H(\mathbf{x}_{1:k})$ could be computed knowing the entropy of multivariate Gaussian (3.2). However, this implies the computation of the determinant of a large matrix. Also, we might end up dealing with numerical and ill-defined matrix issues as explained in [Stachniss et al., 2005]. Hence, we opt for the same approximation also used in [Valencia et al., 2012], which averages over all individual pose marginals

$$H(\mathbf{x}_{1:k}) \approx \frac{1}{k} \sum_{i=1}^{k} \ln \left((2\pi e)^{\frac{n}{2}} |\mathbf{\Sigma}_{ii}| \right), \tag{5.13}$$

being n the dimension of the individual pose vector, $n = 3$ in our case.

Alternatively, a Pose-graph SLAM could be computed to evaluate (5.13) for each path candidate. But since we want to evaluate several path candidates, this method would not be efficient at all. Instead, an iterative estimation of the trajectory entropy (5.13) and the last node's marginal covariance is computed by distinguishing between two cases: open loop and loop closure.

Open loop

In open loop, a new node is added to the graph according to the motion commands or odometry measurement. The covariance matrix grows but the block corresponding to the old nodes remains the same. Then, the trajectory entropy at the k-th time step from (5.13) becomes

$$H(\mathbf{x}_{1:k}) \approx \frac{k-1}{k} H(\mathbf{x}_{1:k-1}) + \frac{1}{k} \ln \left((2\pi e)^{\frac{n}{2}} |\mathbf{\Sigma}_{kk}| \right). \tag{5.14}$$

Thus, the trajectory entropy in open loop can be incrementally computed using the resulting marginal covariance $\mathbf{\Sigma}_{kk}$ which can be effectively computed by linearly propagating it (2.26).

Loop closure

A loop closure is expected to happen when another trajectory pose happen to be within the match area of the current configuration. In other words, when a candidate robot configuration $\mathbf{x}_k$ falls inside the match area of any pose within the pose-graph SLAM trajectory estimate $\mathbf{x}_l$, $l \in [1, t]$, the expected observation $\mathbf{z}_k$ will produce a loop closure.

Encoding a pose-graph SLAM for each action candidate has been discarded because of its computational resources demand. We could instead use the trajectory entropy change from closing such loop as in the previous section (5.8). However, in one hand it would imply mixing different entropy definitions (5.13) with (3.2). On the other hand, evaluating the information gain (5.8) requires the covariance blocks corresponding to the observed nodes: $\mathbf{\Sigma}_{ll}$,$\mathbf{\Sigma}_{lk}$ and $\mathbf{\Sigma}_{kk}$. In the C-space search, they could be approximated as $\mathbf{\Sigma}_{ll}$, $\mathbf{\Sigma}_{lt}$ and $\mathbf{\Sigma}_{tt}$, respectively since all candidates had size one, i.e. $k = t + 1$. However, in longer paths, considering possible previous loop closures entails the need of maintaining the whole covariance matrix in each candidate. It would be equivalent to encoding a pose-graph SLAM or an EKF in all candidates which is unfeasible.

Instead, we propose an efficient way to approximate the trajectory entropy change and the resulting marginal covariance after a loop closure. When a loop is closed, all nodes' marginal covariances $\mathbf{\Sigma}_{ii} \forall i \in (1,k]$ change to new values $\mathbf{\Sigma}'_{ii}$. Then, the trajectory entropy change (5.13) can be expressed in terms of the change in the marginal covariances

$$\begin{aligned}
\Delta H_{LC}(x_{1:k}) &\approx \frac{1}{k}\sum_{i=1}^{k} \ln\left((2\pi e)^{\frac{n}{2}}|\Sigma'_{ii}|\right) - \frac{1}{k}\sum_{i=1}^{k} \ln\left((2\pi e)^{\frac{n}{2}}|\Sigma_{ii}|\right) \\
&= \frac{1}{k}\ln\prod_{i=1}^{k}\frac{|\Sigma'_{ii}|}{|\Sigma_{ii}|}. \\
&= \frac{1}{k}\ln\prod_{i=1}^{k}\rho_i. \qquad (5.15)
\end{aligned}$$

The marginal covariance determinant ratios ρ_i are approximated taking three assumptions or approximations. First, we assume "clean" loops, meaning that no node in the loop has more than two connections. Assuming this, a loop closure to the l-th node does not produce any change on the previous marginals ($\rho_i = 1,\ \forall i \leq l$). Secondly, the new marginal covariance $\mathbf{\Sigma}'_{kk}$ is approximated as a covariance propagation from $\mathbf{\Sigma}_{ll}$ resulting in the k-th determinant change ratio ρ_k. Thirdly, we assume the rest of the determinant change ratios to be lineraly distributed:

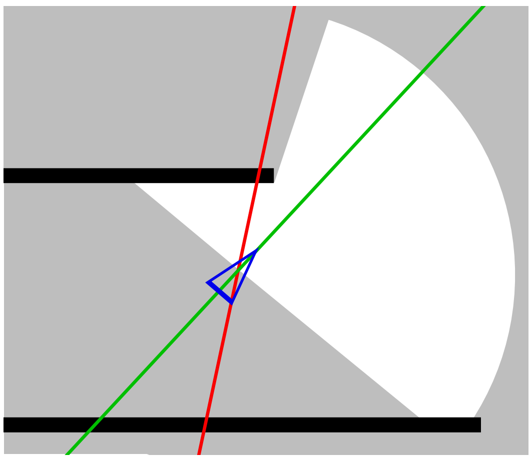
$$\begin{aligned}
\Delta H_{LC}(\mathbf{x}_{1:k}) &\approx \frac{1}{k}\ln\prod_{i=l+1}^{k}\rho_i \\
&\approx \frac{1}{k}\ln\prod_{j=1}^{k-l}\left(\rho_l + j\frac{\rho_k-\rho_l}{k-l}\right). \qquad (5.16)
\end{aligned}$$

Note that loop closure entropy change $\Delta H_{LC}(\mathbf{x}_{1:k})$ is applied to the open loop entropy $H(\mathbf{x}_{1:k})$ that was previously computed (5.14).

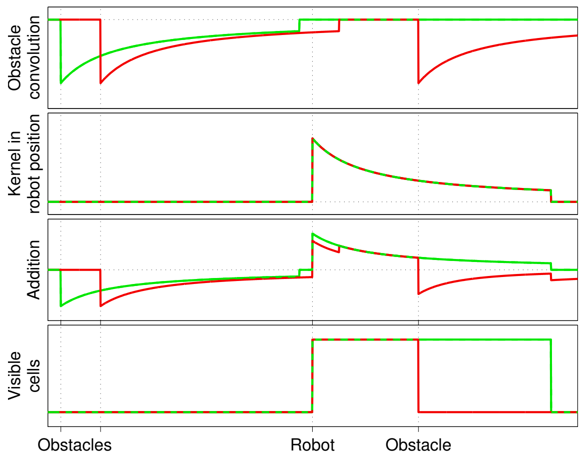
5.4.3 Map entropy decrease estimation

As previously introduced, the reduction in map entropy (5.6) is attained after moving to new locations and sensing new data and depends on the number of cells that will change its classification probability, i.e. either obstacle or free cells will be discovered.

Each intermediate pose $\mathbf{x}_k$ of each action candidate will produce a different observation of the environment $\mathbf{z}_k$, so different cells will be classified producing different map entropy changes. For each action candidate $\mathbf{a}_k$, the map entropy change will depend on the number of cells discovered in all observations made during the action. Therefore, it is important not to overcount cells discovered by different poses of the same path. As contrast with the search in C-space methods, we need a method for computing which (instead of just how many) cells will be discovered from any robot pose $\mathbf{x}_k$. Then, for a sequence of poses of a candidate path $\mathbf{a}_k$, we will be able to estimate the accumulated number of discovered cells.



(a) For a robot configuration example, the expected visible cells (white) taking into account the known obstacles (black).



(b) Example of convolution results in the two specific ray directions depicted in the left frame in their respective colors.

Figure 5.12: Example of ray casting for a particular robot configuration.

Extending the work presented in the previous section, an efficient method to compute ray casting from several robot configurations over the known environment is devised. It makes use of convolutions and pre-computes most of the process as well.

Given a robot configuration candidate $\mathbf{x}_k$, for each ray direction within the sensor spread we compute which cells are between the sensor and the nearest obstacle (if there is any) and not further than the maximum simulated sensor range (see Figure 5.12.a). To do this, We use a convolution using a kernel of zeros except for the cells in a specific direction which have exponentially decreasing values. For each ray direction, convolving this kernel with an image of ones at the obstacles, and substracting it from the same kernel centered at the robot position, the discovered cells will be those with positive values (see Figure 5.12.b).

This operation must be computed for each ray direction included in the sensor spread and for all robot configurations of all action candidates. However, we can pre-compute most of it. In one hand, the kernel for each of all 360° discretized directions is computed once at the beginning of the algorithm. On the other hand, at each planning step, the convolution over the obstacles can be precomputed for all discretized directions. Then, for a robot configuration candidate x_k, we only need to translate the kernels corresponding to all directions to this configuration, substract the convoluted obstacles and binarize (see Figure 5.12.b).

Then we accumulate the discovered cells from the directions within the sensor spread corresponding to the robot orientation (see Figure 5.12.a). Finally, accumulating the discovered cells in all robot configurations $\mathbf{x}_{t+1:k}$, we can estimate the map entropy change of the candidate $\mathbf{a}_k$ using (5.6).

Tuning as a parameter the sensor maximum range, i.e. the kernel size, we can be more conservative or optimistic in the estimation of cells discovered through the frontiers. Using the same value as the real sensor, we will assume that all the unknown cells are free whereas when using a lower value, we will assume the existence of obstacles. This value depends on the environment morphology and is fixed experimentally.

5.4.4 RRT*: Generating and evaluating the action set

The RRT* algorithm presented by Karaman and Frazzoli [2010] is an asymptotically optimal version of the rapidly-exploring random trees (RRT) by LaValle [1998]. It was designed to find an optimal path from the current robot pose to a given goal in a known environment.

The RRT has a simple *extend* routine: a new node in the RRT is added in the direction of a random sample from the nearest node with a distance ν. The RRT* *extend* routine is more complex. After creating a new node as in the RRT, it is not automatically connected to the tree from its predecessor. Instead, it is connected to one of its neighbors according to a given cost function. Afterwards, it is checked if the new node can yield a lower cost to any of each near neighbors and it is rewired if it is the case. These two actions allow the RRT* to be asymptotically optimal.

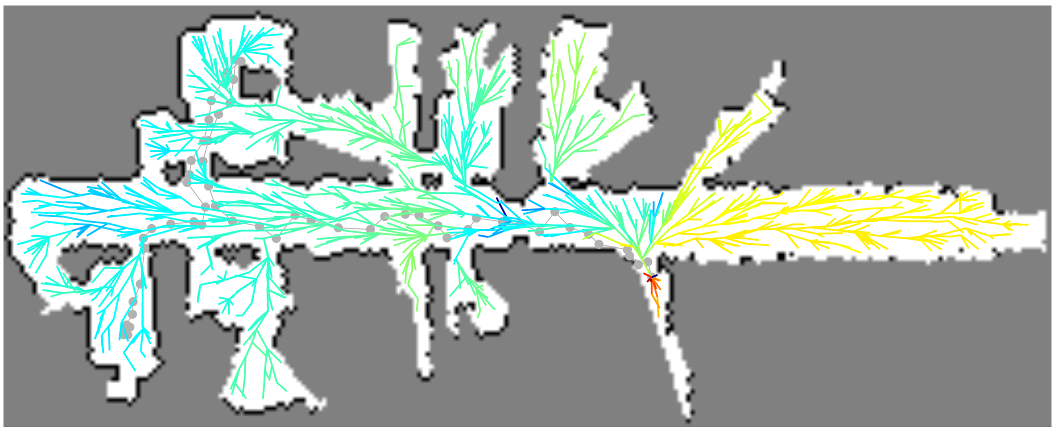
The expansive nature of the RRT and RRT* algorithms suggests that they can be useful to generate a large action set for active pose-graph SLAM since they provide several collision free paths from the current robot pose to several free configurations. Knowing this, we propose to grow an RRT* in the known and free environment and then take each one of the RRT* nodes and the path to reach it as an action candidate $\mathbf{a}_k$. After the tree is grown, the best action candidate $\mathbf{a}_k^*$ is chosen from all the paths to each node according to the evaluation introduced in Section (5.10).

The cost function used in the RRT* *extend* routine will affect the resulting paths of the tree. Using the distance as cost function, as usual, the algorithm will provide the shortest paths to several free configurations. However, using another cost function the tree will grow and rewire differently and the resulting RRT* paths will not be optimal in distance traveled but in that cost function. Considering this, we propose two action set generation alternatives using the RRT*. The first is using the distance as the cost function (dRRT*) as typically, and the second one is using the entropy change divided by the distance (eRRT*), as defined in (5.10).

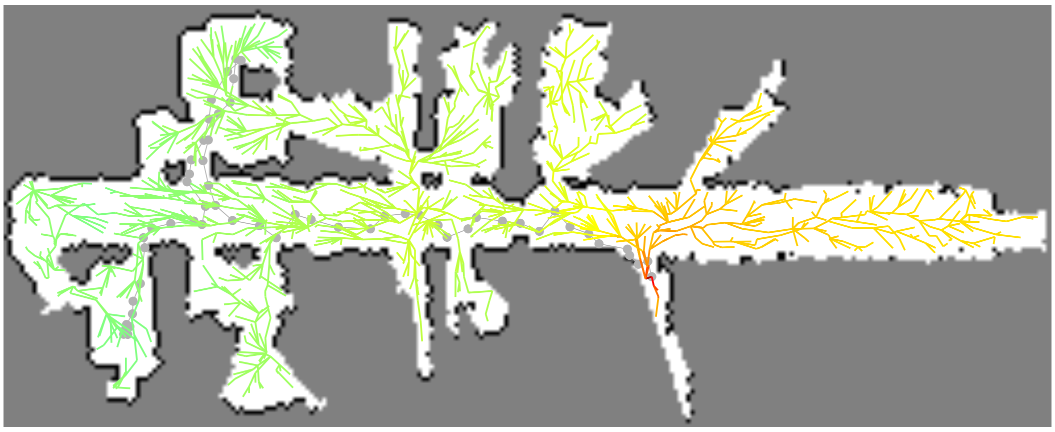
This cost function is not a distance since it is not always positive nor it meets the triangular inequality. This is because the map and trajectory entropy computation depends on previous configurations. Therefore, in one hand when a rewire is preformed, the cost of all node successors should be re-evaluated. On the other hand, the RRT* algorithm is not guaranteed to provide asymptotically optimal paths according to the entropy-based cost function used.

In RRT*, once a solution path is found, the extension process is continued for a time in order to keep improving this solution. Some heuristics can be set to end the extension once the solution has been sufficiently improved. In our case, we do not have a specific goal to be reached by the tree. Instead, our objective is to have several paths to several free configurations spread over the free discovered environment to take it as action set. Therefore, we stop extending after certain amount of nodes per free square meter is reached. This value is fixed experimentally since it depends on the morphology of the environment.

During the extension of eRRT*, the evaluation of the cost function (5.10) is computed (and recomputed after a rewiring). So when the tree extension is stopped, all candidates have



(a) dRRT*.



(b) eRRT*.

Figure 5.13: Two examples of the resulting path candidates using the two different cost functions proposed. Color of branches represent the value of the information gain per meter travelled (5.10) of the path that ends in the corresponding node. Red paths indicate better candidates. In gray, the pose-graph SLAM nodes.

its corresponding expected joint entropy change per meter. However, in the dRRT* case, it can be computed once the extension processes is stopped.

Figure 5.13 shows the two resulting action sets using dRRT* and eRRT*. The color corresponds to the evaluation of the cost function of each path. Both frames share the same color scale, the more red, the more entropy decrease path. It can be seen how eRRT* branches are better than the dRRT* in terms of joint entropy decrease per meter traveled. Indeed, although not optimality can be assured since (5.10) is not a distance, the rewiring improves the resulting candidates due to the cost function used.

5.4.5 Results

Several simulations were performed to compare the performance of the two variants of the presented Active Pose SLAM method (dRRT* and eRRT*) against three other exploration approaches. The first method is the typical frontier-based exploration [Yamauchi, 1997]. This method drives the robot to the closest frontier larger than a threshold (90 cm in our case), without considering the localization and map uncertainties. When there are no frontiers of

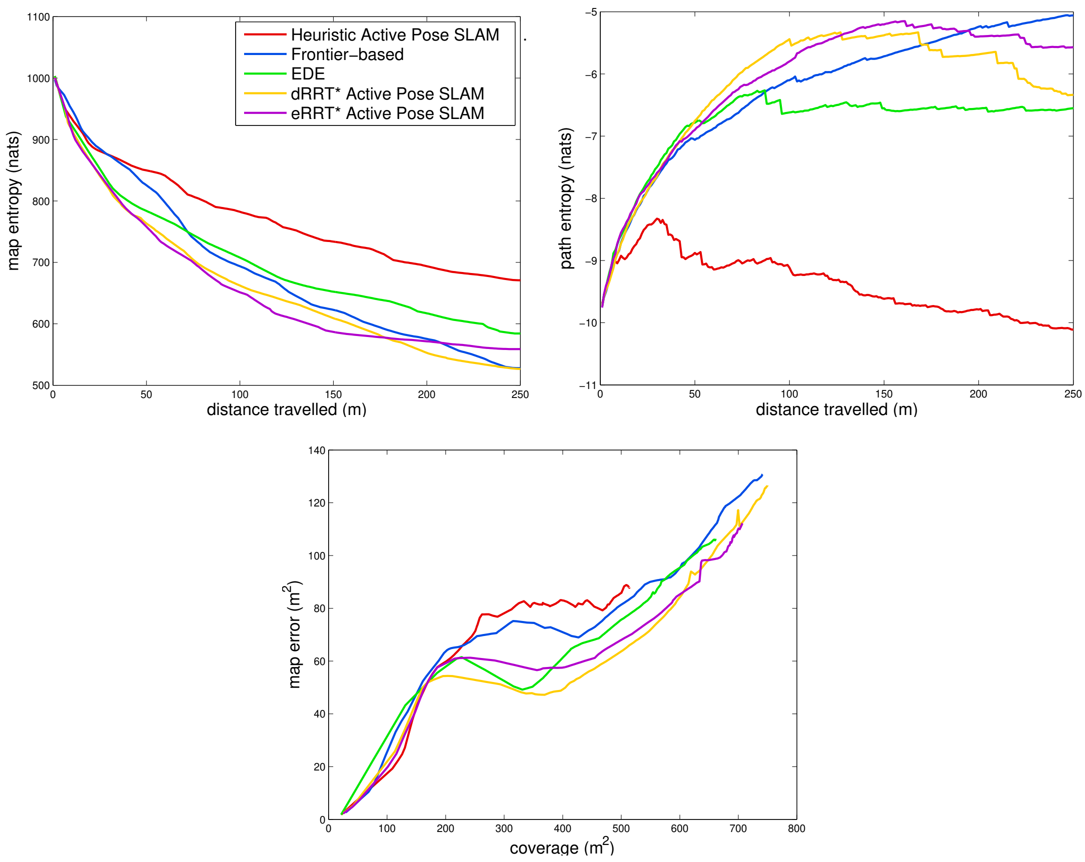


Figure 5.14: Average results for the five simulation runs in Freiburg 079.

this size, this threshold is reduced progressively. The path to the selected frontier centroid is planned using the RRT* planner using distance as cost function.

The second method to which we compare is Active Pose SLAM [Valencia et al., 2012], it evaluates the joint entropy decrease of three heuristically generated action candidates including one revisiting path and the two closest frontiers. And finally, the third method evaluated is the entropy decrease estimation (EDE) using RRT* described in the previous Section 5.3.

Five simulations are performed in the commonly used Freiburg 079 map [Howard and Roy] limiting the experiments to a 250m final trajectory. In all of them, robot motion was estimated with an odometric sensor with a noise covariance factor of 15%. The robot is fitted with a laser range finder sensor with a match area of $\pm 1\,\mathrm{m}$ in x and y, and $\pm 0.35\,\mathrm{rad}$ in orientation. This is the maximum range in C-space for which we can guarantee that a link between two poses can be established. Relative motion constraints were measured using the iterative closest point algorithm with noise covariance fixed at $\mathbf{\Sigma}_y = \mathrm{diag}(0.05\,\mathrm{m},\ 0.05\,\mathrm{m},\ 0.0017\,\mathrm{rad})^2$. Laser scans were simulated by ray casting over a ground truth grid map of the environment using the true robot location, and corrupted with similar values of Gaussian measurement noise.

A number of different metrics were used to compare the performance of the five methods with respect to the distance traveled. We stored average values for the 5 runs of trajectory and map entropy for each of the methods; the average map coverage, measured as the number

	Frontier based	**Heuristic Active Pose SLAM**	**EDE**	***d*RRT***	***e*RRT***
Final map entropy (nats)	527.92	670.93	584.23	526.58	558.80
Final trajectory entropy (nats)	-5.06	-10.11	-6.55	-6.34	-5.57
Total time (seconds)	1089.15	78997.9	1083.70	8752.13	59785.42
Loops closed	22.6	43.2	25.4	18.8	34.4
Coverage (m^2)	741.32	514.10	660.86	749.96	707.10
Map error (m^2)	130.80	87.84	106.34	126.56	112.22

Table 5.6: Average comparison of the performance of several exploration methods in the Freiburg 079 map.

of cells labeled in the occupancy map; and the average map error, measured as the number of cells in the occupancy map which were inconsistent with at least one rendered sensor data point measured at the respective mean of the estimated trajectory pose. Two other measures of performance were total execution time, including all the different processes of each method except for the map rendering, and the total number of loop closures computed by each of the methods. Table 5.6 shows the final average values of each metric for each method. The average evolution of the map and trajectory entropies along the traveled distance for all methods can be observed in the top frames of Figure 5.14. The map error with respect to coverage is plotted in the bottom frame of the same figure.

The computational cost of *e*RRT* is significantly larger than *d*RRT* because the cost function evaluation is computed several times for each node in the *e*RRT* extension process due to the rewiring, whereas in the *d*RRT* it is only computed once for each node after the tree is finished. This is a time consuming step because the cost function requires also the estimation of the joint trajectory and map entropy. Nonetheless, both approaches are less computationally expensive than the use of the Heuristic Active Pose SLAM method which simulates a pose-graph SLAM for each action candidate.

Frontier-based and EDE methods are much faster but they do not evaluate the effect of the paths with regards to entropy reduction, but only seek to minimize distance to pre-computed goals. The frontier-based method, reaches a low level of map entropy but with a high trajectory entropy value, and thus large map error, since it only pursues maximizing coverage. Heuristic active Pose SLAM performance is characterized by a very conservative behavior with regards to localization uncertainty because of the absence of the factor $\alpha(p(\mathbf{x}))$ in its entropy approximation function.

While the EDE method ends the simulations with lower trajectory entropy values, the two A-space search based proposed methods, *d*RRT* and *e*RRT*, are better in terms of map entropy on average. At the final part of the simulations, both methods have lower levels of map error for the same coverage levels than the rest (Figure 5.14 bottom). Moreover, we see how the *d*RRT* reaches higher coverage in the same distance traveled while the *e*RRT* improves over map error. The performance of *e*RRT* is slightly better than *d*RRT* with regards to map error, and significantly better than the heuristic Active Pose SLAM with regards to both coverage and error. However, entropy evaluation is time consuming in RRT* extension, thus an action set generation that only takes into account distance traveled as in dRRT* is an adequate compromise.

Next, we analyze the effect of loop closures in the final exploration results. For instance, the frontier-based method finalizes with higher trajectory entropy values on average than EDE and dRRT*, even after closing a similar amount of loops. This is because the loops closed by frontier-based were not optimally chosen to reduce uncertainty, but rather closed by chance. Obviously, the conservative behavior of heuristic Active Pose SLAM results in a large amount of loop closures on average. The eRRT* also closed a large amount of loops on average because using the entropy based cost function, the RRT* rewiring generates paths that include such loop closure poses.

Figure 5.15 shows single simulation runs for the five methods. In the first frame, we can observe the final localization error of the frontier-based exploration with the last sensor observation in blue. The heuristic Active Pose SLAM final graph is largely connected and all the trajectories remained near the initial robot pose, leaving the rest of the scene largely unexplored, as can be observed in Figure 5.15 b. The EDE and dRRT* final graphs (frames c and d) contain straight paths due to the distance cost function used in the RRT*. Conversely, the eRRT trajectory presents neither straight paths to the goal, nor strong loop closing trajectories, but rather a combination of the two for which the cost function is minimal.

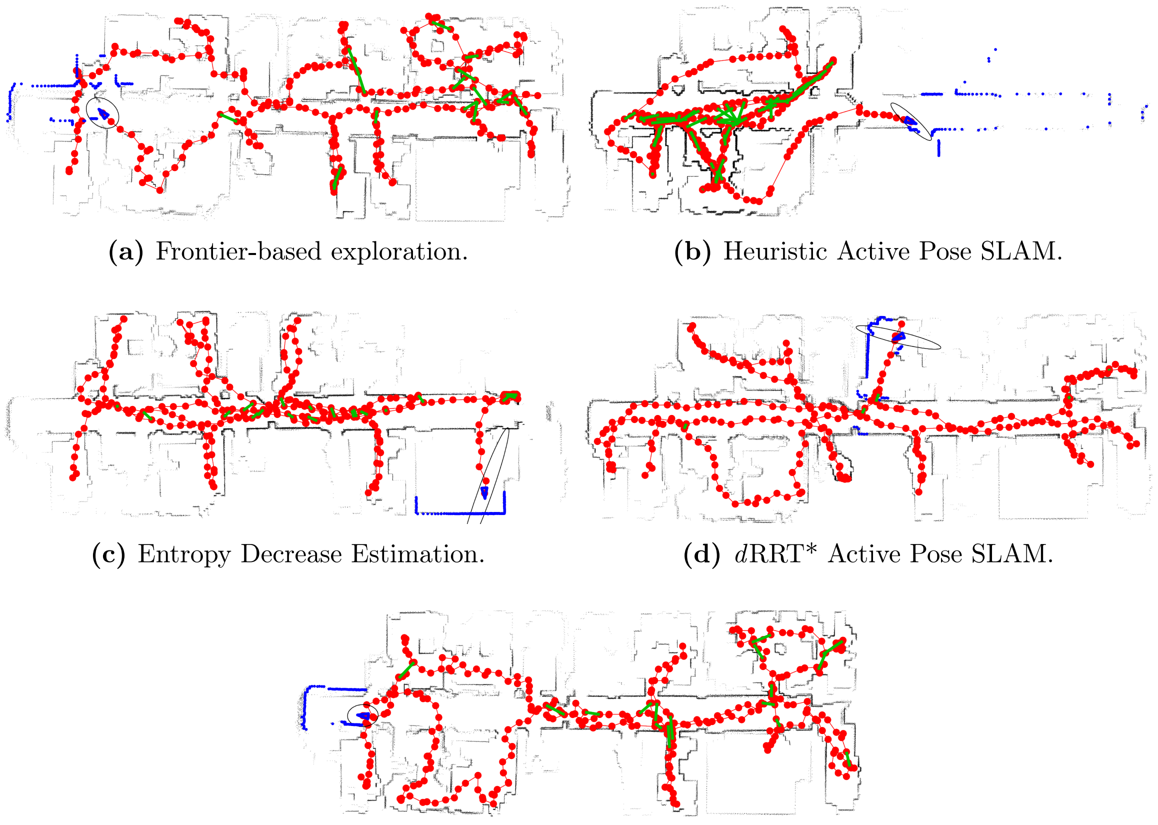

(a) Frontier-based exploration.
(b) Heuristic Active Pose SLAM.
(c) Entropy Decrease Estimation.
(d) *d*RRT* Active Pose SLAM.
(e) *e*RRT* Active Pose SLAM.

Figure 5.15: Final trajectories after a 250m exploration simulation of the Freiburg 079 map for all methods compared. In red the trajectory estimate, in green loop closure links, in black the whole raw sensor data rendered at the trajectory estimate, and in blue the marginal robot pose estimate for the current state (mean and variance) along with the sensed data at that location.

5.5 Discussion and final remarks

In this chapter, the application of entropy is proposed to formally measure the uncertainty reduction in the context of active SLAM. With this purpose, the state-of-art in entropy-based exploration was revised and a new joint entropy decrease formulation was proposed. An advantage of our joint estimation of map and trajectory entropy reduction is that depending on the localization uncertainty, it nicely rebalances between exploratory and re-localization trajectories. When the localization uncertainty rises, the entropy term corresponding to all loop closure configurations grows biasing the robot towards re-localization configurations.

Despite this is a pattern also experienced in previous formulations [Valencia et al., 2012], in our work, this behaviour is enhanced. The introduction of the factor $\alpha(p(\mathbf{x}))$ weights down the map entropy decrease contribution due to the localization uncertainty. As observed in the validation simulations, it makes the algorithm less sensitive to the environment size and complexity. Furthermore, as discovered in our experiments, even when there is a bias towards one behaviour, the other one is still contributing. For instance, when loop closure candidates are heavily weighted, the one with largest exploratory interest will most probably be chosen.

Two different perspectives were presented to approach the problem: searching in the configuration space and in the action space. While the first approach allows to compute a dense entropy decrease estimation over all possible configurations, this estimation does not take into account the effect of the path to reach such configuration. Conversely, searching on the A-space takes into account the entropy change produced during the path but the computational cost is directly related with the amount of candidates evaluated.

The methods have been validated and compared with a popular frontier-based exploration strategy and a similar state-of-art active SLAM method [Valencia et al., 2012]. Several simulations in two different scenarios were launched and the performance was measured in terms of computational cost, map and trajectory entropy as well as coverage and map error.

The work presented in this chapter has been partially published in [Vallvé and Andrade-Cetto, 2013, 2015a, 2014, 2015b].

6 Final remarks

6.1 Conclusions

In this thesis we explored the use of information metrics in mobile robotics localization and mapping applications.

An extensive introduction to SLAM is provided in Chapter 2. The problem formulation of the state-of-art NLS approach has been derived both for batch and incremental methods. Additionally, a complete SLAM literature review is given establishing connections with equivalent problems in other disciplines. The links with other popular SLAM approaches such as Bayesian filtering or relaxation methods were identified and their drawbacks have been stated. The Levenberg-Marquardt and Dogleg methods have been presented as a suitable alternative for SLAM setups with poor initialization.

Secondly, a brief overview of information theory is presented in Chapter 3 introducing some relevant metrics exploited in the subsequent chapters.

A core contribution of this thesis, the reduction of the SLAM problem size is addressed in Chapter 4 as the concatenation of node(s) marginalization and the posterior sparsificaiton of its dense and non-relinearizable result. Sparsification consists on computing an approximation of this dense distribution with a new set of factors allowing relinearization and maintaining the sparsity. The approximation goodness equally relies on two processes in which is formed of: topology building and factor recovery. The topology building establishes the amount of new factors used, as well as their measurement model and the arrangement of all new factors with the affected variables. Factor recovery computes the best approximation given the topology. Three new methods to build the topology are presented making use of information-based metrics for the pose-graph SLAM case. Also, we propose a more intuitive alternative policy to decide the topology population, i.e. the amount of new factors that constitute the sparse approximation. Factor recovery is posed as a Kullback-Liebler divergence minimization from the dense marginalization result to the sparse approximation, since it measures the similarity of two distributions. Two new methods have been proposed to solve the factor recovery: Factor Descent and its non-cyclic variant. Both are proved to compete with state-of-art methods both in computational time and approximation accuracy. More importantly, in contrast with state-of-art methods this proposed methods do not require parameter tuning and its implementation is very simple without need of optimization libraries.

In a second core contribution of this thesis, active SLAM is approached in Chapter 5 for the 2D pose-graph SLAM case using lidar. Entropy measures are exploited to frame the problem as a minimization of both map and trajectory estimation uncertainty. Then, the joint map and trajectory entropy for 2D pose-graph active SLAM is formulated. Four different active pose-graph SLAM methods are proposed following two different perspectives: searching in the configuration space and in the action space. While the former approach allows to compute an estimation of the entropy reduction densely all over the configuration space, it implies not considering the effect on entropy of the path to reach each configuration. Conversely, the action search uses a sparser action candidate set that takes into account the entropy effect of the whole candidate path. All proposed methods are demonstrated to outperform the state-of-art active SLAM methods with several simulations in synthetic and real world environments.

6.2 List of publications

On the course of the development of the present thesis, the author participated in nine scholar publications that are listed below.

Journals

- J. Vallvé, J. Solà and J. Andrade-Cetto. Pose-graph SLAM sparsification using factor descent. *Robotics and Autononous Systems*, conditionally accepted.

- J. Vallvé, J. Solà and J. Andrade-Cetto. Graph SLAM sparsification with populated topologies using factor descent optimization. *IEEE Robotics and Automation Letters*, vol. 3, pp. 1322-1329, 2018.

- J. Vallvé and J. Andrade-Cetto. Potential information fields for mobile robot exploration. *Robotics and Autonomous Systems*, vol. 69, pp. 68-79, 2015.

Conferences

- J. Vallvé, J. Solà and J. Andrade-Cetto. Factor Descent optimization for sparsification in graph SLAM. *8th European Confenrence on Mobile Robots (ECMR)*, Paris, Sep. 2017, pp. 95-100.

- A. Corominas Murtra, J. Vallvé, J. Solà, I. Flores and J. Andrade-Cetto. Observability analysis and optimal sensor placement in stereo radar odometry. *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, May 2016, pp. 3161-3166.

- J. Vallvé and J. Andrade-Cetto. Active Pose SLAM with RRT*. *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, May 2015, pp. 2167-2173.

- J. Vallvé and J. Andrade-Cetto. Dense entropy decrease estimation for mobile robot exploration. *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, May 2014, pp. 6083-6089.

- R. Valencia, J. Saarinen, H. Andreasson, J. Vallvé, J. Andrade-Cetto and A. Llilienthal. Localization in highly dynamic environments using dual-timescale NDT-MCL. *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, May 2014, pp. 3956-3962.

- J. Vallvé and J. Andrade-Cetto. Mobile robot exploration with potential information fields. *6th European Conference on Mobile Robots (ECMR)*, Barcelona, Sep. 2013, pp. 222-227.

6.3 Future work

Finally, we enumerate some possible future lines of research that emerged during the development of the work exposed in this thesis.

Regarding the sparsification problem, a new topology population policy can be devised by fixing the desired expected information loss. It would be challenging since it requires a way to estimate the KLD a priori, without knowing the specific topology nor the factor recovery solution given that topology.

Also, we consider the application of sparsification to highly dense problems with lower rank factors and heterogeneity of measurement models, such as in the visual inertial case [Hsiung et al., 2018]. Some work presented in this thesis has to be reformulated for these cases. While the factor recovery formulation and factor descent methods presented already consider these cases, new topology methods should be devised. The approach of using new synthetic factors with the same measurement model as real sensor measurement has to be revisited. New alternatives can be explored, from building topologies with visual and inertial measurements to designing new ad hoc measurement models. The selection of the node to be removed has to be also studied for this special case considering the implications in future SLAM iterations into the state estimation.

Actually, the node removal selection is still an open issue that has not been addressed in the present work. Different approaches can be devised depending on specific final applications. Making use of information metrics, the nodes to be removed can be selected by estimating the expected information loss or by anticipating the sparsification computational cost. In both cases, the Markov blanket size (i.e. the node connectivity) may be a good indicator. Alternatively, considering the effect on the state estimation and the real application, the temporal aspects could be taken into account: Recent nodes (whether trajectory poses or landmarks) are more likely to be engaged in new factors, and their estimation is also more likely to be refined. This suggests that a good approach would be biasing the node removal to the past, while maintaining a more populated graph for the present. Intuitively, this is what humans do when navigating: we forget the temporary map of the street (shops, trees, cars and so on) after turning a corner. Information metrics could be explored to tackle node selection in order to emulate this behaviour as well.

Referring to active SLAM, future work includes the extension to the 3D case and the use of new sensor setups and environment representations. Extending the presented methods to the 3D case would require new methods to create the expected joint entropy decrease estimation while the formulation is still applicable. In this case, the C-space dimension grows considerably to 6D. This would make the C-space search approaches almost unfeasible in terms of computational resources. Even though A-space search would become sparser in this higher dimensional C-space, it is easily scalable since computational cost and memory requirements is directly related to the RRT* size in both proposed methods *d*RRT* and *e*RRT*. Then, the approach remains still valid but new methods to estimate the entropy decrease should be developed. Specifically, the map entropy decrease would require an non-trivial extension to 3D occupancy grids.

Finally, the use of other sensors and/or other map representations also requires new map entropy decrease estimation methods. Dense occupancy-based representations of the environment should be considered in all cases instead of landmark-based representations, since only the first ones are capable of encoding free-space. Free space classification is critical to allow collision-free motion paths. Furthermore, the discovery of free space is the principal aim of exploration methods and one of the objectives of active SLAM.

A Propositions

Proposition 1 *The squared measurements Jacobian is equal to the information matrix of* $P(\mathbf{x}|Z)$*,* $\mathbf{A}^\top\mathbf{A} = \boldsymbol{\Lambda}$

Proof: The information form of the conditional probability of the state given all the measurement is

$$P(\mathbf{x}|Z) = \frac{P(Z|\mathbf{x})P(\mathbf{x})}{P(Z)}$$

since $P(\mathbf{x})$ is uniform and all measurements $\mathbf{z}_k$ are independent,

$$\propto \prod_k P(\mathbf{z}_k|\mathbf{x})$$

considering linearized factors $h_k(\mathbf{x}) = h_k(\boldsymbol{\mu}) + \mathbf{J}_k(\mathbf{x} - \boldsymbol{\mu}) + \mathbf{v}_k$, with $\mathbf{v}_k \sim \mathcal{N}(0, \boldsymbol{\Omega}_k^{-1})$

$$\begin{aligned}
&\propto \prod_k \exp\Big(-\tfrac{1}{2}\big(\underbrace{\mathbf{z}_k - h_k(\boldsymbol{\mu})}_{\mathbf{e}_k} - \mathbf{J}_k\mathbf{x} + \mathbf{J}_k\boldsymbol{\mu})\big)^\top \boldsymbol{\Omega}_k \big(\underbrace{\mathbf{z}_k - h_k(\boldsymbol{\mu})}_{\mathbf{e}_k} - \mathbf{J}_k\mathbf{x} + \mathbf{J}_k\boldsymbol{\mu})\big)\Big) \\
&= \exp\Big(-\tfrac{1}{2}\sum_k \mathbf{x}^\top\mathbf{J}_k^\top\boldsymbol{\Omega}_k\mathbf{J}_k\mathbf{x} - 2(\mathbf{e}_k + \mathbf{J}_k\boldsymbol{\mu})^\top\boldsymbol{\Omega}_k\mathbf{J}_k\mathbf{x} + (\mathbf{e}_k + \mathbf{J}_k\boldsymbol{\mu})^\top\boldsymbol{\Omega}_k(\mathbf{e}_k + \mathbf{J}_k\boldsymbol{\mu})\Big) \\
&\propto \exp\Bigg(-\tfrac{1}{2}\mathbf{x}^\top \underbrace{\Big(\sum_k \mathbf{J}_k^\top\boldsymbol{\Omega}_k\mathbf{J}_k\Big)}_{\boldsymbol{\Lambda}}\mathbf{x} + \underbrace{\Big(\sum_k (\mathbf{e}_k + \mathbf{J}_k\boldsymbol{\mu})^\top\boldsymbol{\Omega}_k\mathbf{J}_k\Big)}_{\boldsymbol{\eta}}\mathbf{x}\Bigg).
\end{aligned}$$

In the other hand, the squared measurements Jacobian is

$$\mathbf{A}^\top\mathbf{A} = \begin{bmatrix}\cdots & \mathbf{A}_k^\top & \cdots\end{bmatrix}\begin{bmatrix}\vdots \\ \mathbf{A}_k \\ \vdots\end{bmatrix} = \begin{bmatrix}\cdots & \mathbf{J}_k^\top\mathbf{\Omega}_k^{\top/2} & \cdots\end{bmatrix}\begin{bmatrix}\vdots \\ \mathbf{\Omega}_k^{1/2}\mathbf{J}_k \\ \vdots\end{bmatrix} = \sum_k \mathbf{J}_k^\top\mathbf{\Omega}_k\mathbf{J}_k. \quad \text{(A.1)}$$

Proposition 2 *The EIF information matrix update is obtained from applying the Woodbury matrix identity to the EKF covariance update.*

Proof: For clearity we simplify the notation ($\mathbf{J}_k = \mathbf{J}, \mathbf{\Sigma}_{\mathbf{v}_k} = \mathbf{\Sigma}_{\mathbf{v}}, \mathbf{K}_k = \mathbf{K}$) Condensing the EKF covariance update in a single expression

$$\begin{aligned}\mathbf{\Sigma}_{k|k} &= (\mathbf{I} - \mathbf{K}\mathbf{J})\mathbf{\Sigma}_{k|k-1} \\ &= \mathbf{\Sigma}_{k|k-1} - \mathbf{\Sigma}_{k|k-1}\mathbf{J}^\top(\mathbf{J}\mathbf{\Sigma}_{k|k-1}\mathbf{J}^\top + \mathbf{\Sigma}_{\mathbf{v}})^{-1}\mathbf{J}\mathbf{\Sigma}_{k|k-1},\end{aligned}$$

the Woodbury identity can be applied backwards leading to

$$\begin{aligned}&= (\mathbf{\Sigma}_{k|k-1}^{-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J})^{-1} \\ &= (\mathbf{\Lambda}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J})^{-1}.\end{aligned}$$

Then, trivially

$$\mathbf{\Lambda}_{k|k} = \mathbf{\Sigma}_{k|k}^{-1} = \mathbf{\Lambda}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J}.$$

For the information vector update, a previous identity need to be presented departing from Woodbury identity:

$$\begin{aligned}\mathbf{\Sigma}_{k|k-1} - \mathbf{\Sigma}_{k|k-1}\mathbf{J}^\top(\mathbf{J}\mathbf{\Sigma}_{k|k-1}\mathbf{J}^\top + \mathbf{\Sigma}_{\mathbf{v}})^{-1}\mathbf{J}\mathbf{\Sigma}_{k|k-1} &= (\mathbf{\Lambda}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J})^{-1} \\ \mathbf{\Sigma}_{k|k-1}\mathbf{J}^\top(\mathbf{J}\mathbf{\Sigma}_{k|k-1}\mathbf{J}^\top + \mathbf{\Sigma}_{\mathbf{v}})^{-1} &= \big(\mathbf{\Sigma}_{k|k-1} - (\mathbf{\Lambda}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J})^{-1}\big)\mathbf{\Sigma}_{k|k-1}^{-1}\mathbf{J}^\top(\mathbf{J}\mathbf{J}^\top)^{-1} \\ \mathbf{K} &= \big(\mathbf{I} - (\mathbf{\Lambda}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J})^{-1}\mathbf{\Lambda}_{k|k-1}\big)\mathbf{J}^\top(\mathbf{J}\mathbf{J}^\top)^{-1}\end{aligned} \tag{A.2}$$

Then, departing from the relation between the canonical form and the information form, we substitute the corresponding updates formulas

$$\begin{aligned}\boldsymbol{\eta}_{k|k} =& \mathbf{\Lambda}_{k|k}\boldsymbol{\mu}_{k|k} \\ =& \big(\mathbf{\Lambda}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J}\big)\Big(\boldsymbol{\mu}_{k|k-1} + \mathbf{K}\big(\mathbf{z}_k - h(\boldsymbol{\mu}_{k|k-1})\big)\Big)\end{aligned}$$

and according to the previously presented identity (A.2),

$$\begin{aligned}=&\boldsymbol{\eta}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J}\boldsymbol{\mu}_{k|k-1} + \\ &\big(\mathbf{\Lambda}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J}\big)\Big(\mathbf{I} - \big(\mathbf{\Lambda}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J}\big)^{-1}\mathbf{\Lambda}_{k|k-1}\Big)\mathbf{J}^\top\big(\mathbf{J}\mathbf{J}^\top\big)^{-1}\big(\mathbf{z}_k - h(\boldsymbol{\mu}_{k|k-1})\big) \\ =&\boldsymbol{\eta}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J}\boldsymbol{\mu}_{k|k-1} + \big(\mathbf{\Lambda}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J} - \mathbf{\Lambda}_{k|k-1}\big)\mathbf{J}^\top\big(\mathbf{J}\mathbf{J}^\top\big)^{-1}\big(\mathbf{z}_k - h(\boldsymbol{\mu}_{k|k-1})\big) \\ =&\boldsymbol{\eta}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\mathbf{J}\boldsymbol{\mu}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\big(\mathbf{z}_k - h(\boldsymbol{\mu}_{k|k-1})\big) \\ =&\boldsymbol{\eta}_{k|k-1} + \mathbf{J}^\top\mathbf{\Sigma}_{\mathbf{v}}^{-1}\big(\mathbf{z}_k - h(\boldsymbol{\mu}_{k|k-1}) + \mathbf{J}\boldsymbol{\mu}_{k|k-1}\big)\end{aligned}$$

Proposition 3 *If $\breve{\mathbf{\Lambda}}$ is invertible and $\breve{\mathbf{J}}_k$ is full rank, the derivative (4.11) becomes null in*

$$\breve{\mathbf{\Omega}}_k = (\breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1} - L^{-\top} \mathbf{Q}_L \big(\breve{\mathbf{\Upsilon}}_k - \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top (\mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top)^{-1} \mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \big) \mathbf{Q}_L^\top L^{-1} \tag{A.3}$$

being the LQ-decomposition of $\breve{\mathbf{J}}_k = \mathbf{LQ} = \begin{bmatrix} L & 0 \end{bmatrix} \begin{bmatrix} \mathbf{Q}_L \\ \mathbf{Q}_0 \end{bmatrix}$.

Proof: The derivative of (4.9) w.r.t $\breve{\mathbf{\Omega}}_k$ is

$$\frac{\partial D_{KL}}{\partial \breve{\mathbf{\Omega}}_k} = \breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top - \breve{\mathbf{J}}_k (\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top \breve{\mathbf{\Omega}}_k \breve{\mathbf{J}}_k)^{-1} \breve{\mathbf{J}}_k^\top \ . \tag{A.4}$$

Applying the decomposition into the second term:

$$\begin{aligned}
\breve{\mathbf{J}}_k (\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top \breve{\mathbf{\Omega}}_k \breve{\mathbf{J}}_k)^{-1} \breve{\mathbf{J}}_k^\top &= \mathbf{LQ} (\breve{\mathbf{\Upsilon}}_k + \mathbf{Q}^\top \mathbf{L}^\top \breve{\mathbf{\Omega}}_k \mathbf{LQ})^{-1} \mathbf{Q}^\top \mathbf{L}^\top \\
&= \mathbf{L} (\mathbf{Q} \breve{\mathbf{\Upsilon}}_k \mathbf{Q}^\top + \mathbf{L}^\top \breve{\mathbf{\Omega}}_k \mathbf{L})^{-1} \mathbf{L}^\top \\
&= \begin{bmatrix} L & 0 \end{bmatrix} \begin{bmatrix} L^\top \breve{\mathbf{\Omega}}_k L + \mathbf{Q}_L \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_L^\top & \mathbf{Q}_L \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top \\ \mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_L^\top & \mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top \end{bmatrix}^{-1} \begin{bmatrix} L^\top \\ 0 \end{bmatrix} \\
&= L \big(L^\top \breve{\mathbf{\Omega}}_k L + \mathbf{Q}_L \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_L^\top - \mathbf{Q}_L \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top (\mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top)^{-1} \mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_L^\top \big)^{-1} L^\top \\
&= \big(\breve{\mathbf{\Omega}}_k + L^{-\top} (\mathbf{Q}_L \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_L^\top - \mathbf{Q}_L \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top (\mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top)^{-1} \mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_L^\top) L^{-1} \big)^{-1}.
\end{aligned}$$

Substituting in (A.4) and imposing null derivative leads to (A.3). Since $\mathbf{Q}$ is orthogonal and $\mathbf{Q}_0 \breve{\mathbf{\Lambda}} \mathbf{Q}_0^\top = \mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top$, then $\exists \breve{\mathbf{\Lambda}}^{-1} \Rightarrow \exists (\mathbf{Q}_0 \breve{\mathbf{\Lambda}} \mathbf{Q}_0^\top)^{-1} \Rightarrow \exists (\mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top)^{-1}$.

Proposition 4 *If $\breve{\mathbf{\Upsilon}}_k$ is invertible and $\breve{\mathbf{J}}_k$ is full rank, the derivative (4.11) becomes null in*

$$\breve{\mathbf{\Omega}}_k = (\breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1} - (\breve{\mathbf{J}}_k \breve{\mathbf{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top)^{-1}. \tag{A.5}$$

Proof: If $\breve{\mathbf{\Upsilon}}_k$ is invertible, applying the Woodbury matrix identity forwards and backwards to the second term of (A.4)

$$\begin{aligned}
\breve{\mathbf{J}}_k (\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top \breve{\mathbf{\Omega}}_k \breve{\mathbf{J}}_k)^{-1} \breve{\mathbf{J}}_k^\top &= \breve{\mathbf{J}}_k (\breve{\mathbf{\Upsilon}}_k^{-1} - \breve{\mathbf{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top (\breve{\mathbf{\Omega}}_k^{-1} + \breve{\mathbf{J}}_k \breve{\mathbf{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top)^{-1} \breve{\mathbf{J}}_k \breve{\mathbf{\Upsilon}}_k^{-1}) \breve{\mathbf{J}}_k^\top \\
&= \breve{\mathbf{J}}_k \breve{\mathbf{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top - \breve{\mathbf{J}}_k \breve{\mathbf{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top (\breve{\mathbf{\Omega}}_k^{-1} + \breve{\mathbf{J}}_k \breve{\mathbf{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top)^{-1} \breve{\mathbf{J}}_k \breve{\mathbf{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top \\
&= ((\breve{\mathbf{J}}_k \breve{\mathbf{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top)^{-1} + \breve{\mathbf{\Omega}}_k)^{-1}.
\end{aligned}$$

Substituting in (A.4) and imposing null derivative leads to (A.5).

Proposition 5 *If $\breve{\mathbf{\Lambda}}$ is invertible, $\breve{\mathbf{J}}_k$ is full rank and $nul(\breve{\mathbf{\Upsilon}}_k) = rk(\breve{\mathbf{J}}_k)$, the derivative (4.11) becomes null in (4.4)*

$$\breve{\mathbf{\Omega}}_k = (\breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1}.$$

Proof: Consider $\breve{\mathbf{J}}_k \in \mathbb{R}^{m \times n}, \breve{\mathbf{\Upsilon}} \in \mathbb{R}^{n \times n}, n > m$. Since $\breve{\mathbf{J}}_k$ is full rank, $rk(\breve{\mathbf{J}}_k) = m$. According to Prop. 3, $\exists (\mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top)^{-1}$ and $rk(\mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top) = n - m$. Since $\mathbf{Q}$ is orthogonal, $rk(\mathbf{Q} \breve{\mathbf{\Upsilon}}_k \mathbf{Q}^\top) = rk(\breve{\mathbf{\Upsilon}}_k) = n - nul(\breve{\mathbf{\Upsilon}}_k) = n - rk(\breve{\mathbf{J}}_k) = n - m$. According to the Schur complement rank additivity formula

$$rk(\mathbf{Q} \breve{\mathbf{\Upsilon}}_k \mathbf{Q}^\top) = rk(\mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top) + rk(\mathbf{Q}_L \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_L^\top - \mathbf{Q}_L \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top (\mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top)^{-1} \mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_L^\top),$$

then $\mathbf{Q}_L \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_L^\top - \mathbf{Q}_L \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top (\mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_0^\top)^{-1} \mathbf{Q}_0 \breve{\mathbf{\Upsilon}}_k \mathbf{Q}_L^\top = 0$ since its rank is null. Then, (A.3) becomes (4.4).

Proposition 6 *Let $\mathbf{z}_k$ be a measure with corresponding Jacobian $\mathbf{J}_k$ and measurement noise information matrix $\mathbf{\Omega}_k$. The entropy change in the pose-graph estimation $X \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$ (or equivalently $X \sim \mathcal{N}^{-1}(\boldsymbol{\eta}, \mathbf{\Lambda})$) after adding the measure is*

$$\begin{aligned}
\Delta H(X) &= \log \frac{(2\pi e)^{n/2}}{|\mathbf{\Lambda} + \Delta\mathbf{\Lambda}|} - \log \frac{(2\pi e)^{n/2}}{|\mathbf{\Lambda}|} \\
&= \log \frac{\cancel{(2\pi e)^{n/2}}|\mathbf{\Lambda}|}{\cancel{(2\pi e)^{n/2}}|\mathbf{\Lambda} + \Delta\mathbf{\Lambda}|} \\
&= \log \frac{|\mathbf{\Lambda}|}{|\mathbf{\Lambda} + \mathbf{J}_k^\top \mathbf{\Omega}_k \mathbf{J}_k|} \\
&= \log \frac{|\mathbf{\Lambda}|}{|\mathbf{\Omega}_k^{-1} + \mathbf{J}_k^\top \mathbf{\Lambda}^{-1} \mathbf{J}_k||\mathbf{\Lambda}||\mathbf{\Omega}_k|} \\
&= -\log\left(|\mathbf{\Omega}_k^{-1} + \mathbf{J}_k \mathbf{\Sigma} \mathbf{J}_k^\top||\mathbf{\Omega}_k|\right) \\
&= -\log\left(|\mathbf{I} + \mathbf{J}_k \mathbf{\Sigma} \mathbf{J}_k^\top \mathbf{\Omega}_k|\right)
\end{aligned} \tag{A.6}$$

Bibliography

P. Agarwal and E. Olson. Variable reordering strategies for SLAM. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3844–3850, Vilamoura, Portugal, oct 2012.

S. Agarwal, K. Mierle, and Others. Ceres solver. URL http://ceres-solver.org.

S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 29–42, Crete, Greece, sep 2010. Springer, Berlin, Heidelberg.

N. A. Ahmed and D. V. Gokhale. Entropy expressions and their estimators for multivariate distributions. *IEEE Transactions on Information Theory*, 35(3):688–692, 1989.

P. R. Amestoy, T. A. Davis, and I. S. Duff. Algorithm 837: AMD, an approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software*, 30(3):381–388, 2004.

L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

N. Ayache and O. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6):804–819, 1989.

C. Benoït. Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues (procédé du commandant Cholesky). *Bulletin géodésique*, 2(1):67–77, 1924.

N. Börlin and P. Grussenmeyer. Bundle adjustment with and without damping. *The Photogrammetric Record*, 28(144):396–415, 2013.

F. Bourgault, A. A. Makarenko, S. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 540–545, Lausanne, Switzerland, dec 2002.

D. C. Brown. A solution to the general problem of multiple station analytical stereo triangulation. Technical report, 1958.

C. G. Broyden. The convergence of a class of double-rank minimization algorithms 2. The new algorithm. *IMA Journal of Applied Mathematics*, 6(3):222–231, 1970.

M. Byröd and K. Åström. Conjugate gradient bundle adjustment. In *European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 114–127, Crete, Greece, sep 2010. Springer, Berlin, Heidelberg.

N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice. Generic node removal for factor-graph SLAM. *IEEE Transactions on Robotics*, 30(6):1371–1385, 2014.

L. Carlone. Luca Carlone datasets. URL http://www.lucacarlone.com/index.php/resources/datasets.

Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software*, 35(3):22:1–22:14, 2008.

S. Choudhary, V. Indelman, H. I. Christensen, and F. Dellaert. Information-based reduced landmark SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2015-June, pages 4620–4627, Seattle, WA, USA, may 2015.

A. Corominas-Murtra, J. M. Mirats Tur, and A. Sanfeliu. Action evaluation for mobile robot global localization in cooperative environments. *Robotics and Autonomous Systems*, 56 (10):807–818, 2008.

A. Corominas-Murtra, J. Vallvé, J. Sola, I. Flores, and J. Andrade-Cetto. Observability analysis and optimal sensor placement in stereo radar odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3161–3166, Stockholm, Sweden, may 2016.

W. C. Davidon. Variable metric method for minimization. Technical report, 1959.

T. A. Davis. SuiteSparse: A suite of sparse matrix software (version 5.3). URL http://faculty.cse.tamu.edu/davis/suitesparse.html.

T. A. Davis. *Direct methods for sparse linear systems.* SIAM, 2006.

T. A. Davis, J. Gilbert, S. Larimore, and E. Ng. A column approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software*, 30(3):353–376, 2004.

F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25 (12):1181–1203, 2006.

F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1322–1328, Detroit, MI, USA, may 1999.

F. Dellaert, J. Carlson, V. Ila, K. Ni, and C. E. Thorpe. Subgraph-preconditioned conjugate gradients for large scale SLAM. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2566–2571, Taipei, Taiwan, oct 2010.

A. Doucet, N. de Freitas, K. P. Murphy, and S. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Conference on Uncertainty in Artificial Intelligence*, pages 176–183, Catalina Island, CA, USA, jul 2000.

T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3841–3846, San Francisco, CA, USA, apr 2000.

T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300, 2002.

H. F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, 4(1):23–31, 1988.

H. F. Durrant-Whyte, D. Rye, and E. Nebot. Localization of autonomous guided vehicles. In *Robotics Research*, pages 613–625. Springer, London, 1996.

K. Eckenhoff, L. Paull, and G. Huang. Decoupled, consistent node removal and edge sparsification for graph-based SLAM. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3275–3282, Daejeon, South Korea, oct 2016.

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22 (6):46–57, 1989.

J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1449–1456, Sydney, Australia, dec 2013.

R. M. Eustice, H. Singh, and J. J. Leonard. Exactly sparse delayed-state filters. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2428–2435, Barcelona, Spain, apr 2005.

H. J. S. Feder, J. J. Leonard, and C. M. Smith. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18(7):650–668, 1999.

R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3): 317–322, 1970.

R. Fletcher and M. J. D. Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163–168, 1963.

J. Folkesson and H. I. Christensen. Graphical SLAM - a self-correcting map. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 383–390, New Orleans, LA, USA, apr 2004.

J. Folkesson, P. Jensfelt, and H. I. Christensen. Graphical SLAM using vision and the measurement subspace. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 325–330, Edmonton, Alberta, Canada, aug 2005.

D. Fox, W. Burgard, and S. Thrun. Active Markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3):195–207, 1998.

L. Freda and G. Oriolo. Frontier-based probabilistic strategies for sensor-based exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3881–3887, Barcelona, Spain, apr 2005.

U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):196–207, 2005.

W. Givens. Computation of plain unitary rotations transforming a general matrix to triangular form. *Journal of the Society for Industrial and Applied Mathematics*, 6(1):26–50, 1958.

D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

G. H. Golub and R. J. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra and its Applications*, 34:3–28, 1980.

J. P. Gram. Ueber die entwickelung reeller functionen in reihen mittelst der methode der kleinsten quadrate. *Journal für die reine und angewandte Mathematik*, 94:41–73, 1883.

G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based SLAM with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2432–2437, Barcelona, Spain, apr 2005.

G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

M. R. Hestenes and E. Stiefel. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC, 1952.

N. J. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118, 1988.

A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM (JACM)*, 5(4):339–342, 1958.

A. Howard and N. Roy. The robotics data set repository (Radish). URL http://radish.sourceforge.net.

A. Howard, M. J. Mataric, and G. S. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1055–1060, Maui, HI, USA, oct 2001.

J. Hsiung, M. Hsiao, E. Westman, R. Valencia, and M. Kaess. Information Sparsification in Visual-Inertial Odometry. In *IEEE International Conference on Intelligent Robots and Systems (*, page to appear, Madrid, Spain, oct 2018.

V. Ila, J. M. Porta, and J. Andrade-Cetto. Information-based compact pose SLAM. *IEEE Transactions on Robotics*, 26(1):78–93, 2010.

V. Ila, L. Polok, M. Solony, P. Smrz, and P. Zemcik. Fast covariance recovery in incremental nonlinear least square solvers. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4636–4643, Seattle, WA, USA, may 2015.

V. Ila, L. Polok, M. Solony, and P. Svoboda. SLAM++ A highly efficient and temporally scalable incremental SLAM framework. *The International Journal of Robotics Research*, 36(2):210–230, 2017.

Y. D. Jian and F. Dellaert. iSPCG: Incremental subgraph-preconditioned conjugate gradient method for online SLAM with many loop-closures. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2647–2653, Chicago, USA, sep 2014.

Y. D. Jian, D. C. Balcan, and F. Dellaert. Generalized subgraph preconditioners for large-scale bundle adjustment. In *Outdoor and Large-Scale Real-World Scene Analysis*, Lecture Notes in Computer Science, pages 131–150. Springer, Berlin, Heidelberg, 2012.

Y. D. Jian, D. C. Balcan, I. Panageas, P. Tetali, and F. Dellaert. Support-theoretic subgraph preconditioners for large-scale SLAM. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 9–16, Tokyo, Japan, nov 2013.

H. Johannsson, M. Kaess, M. F. Fallon, and J. J. Leonard. Temporally scalable visual SLAM using a reduced pose graph. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 54–61, Karlsruhe, Germany, may 2013.

S. J. Julier and J. K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 4238–4243, Seoul, South Korea, may 2001.

M. Kaess and F. Dellaert. Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Systems*, 57(12):1198–1210, 2009.

M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering ASME*, pages 35–45, 1960.

S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104:2, 2010.

G. Karypis and V. Kumar. METIS – Unstructured graph partitioning and sparse matrix ordering system, version 2.0. Technical report, 1995. URL http://dm.kaist.ac.kr/kse625/resources/metis.pdf.

L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration for fast path planning. In *IEEE International Conference on Robotics and Automation*, pages 2138–2145, San Diego, CA, USA, may 1994.

K. Khosoussi, G. S. Sukhatme, S. Huang, and G. Dissanayake. Designing sparse reliable pose-graph SLAM: A graph-theoretic approach. *arXiv preprint arXiv:1611.00889*, 2016.

K. Konolige. Large-scale map-making. In *AAAI Conference on Artificial Intelligence*, pages 457–463, San Jose, California, jul 2004.

K. Konolige. Sparse sparse bundle adjustment. In *British Machine Vision Conference*, pages 102.1–102.11, Aberystwyth, UK, apr 2010.

K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent. Efficient sparse pose adjustment for 2D mapping. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 22–29, Taipei, Taiwan, oct 2010.

H. Kretzschmar and C. Stachniss. Information-theoretic compression of pose graphs for laser-based SLAM. *The International Journal of Robotics Research*, 31(11):1219–1230, 2012.

F. R. Kschischang, B. J. Frey, and H. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

S. Kullback. *Information theory and statistics.* Courier Corporation, 1959.

S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, Shanghai, China, may 2011.

S. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Comp. Sc. Dept., Iowa St. Univ., 1998.

J. J. Leonard, H. F. Durrant-Whyte, and I. J. Cox. Dynamic map building for an autonomous mobile robot. *The International Journal of Robotics Research*, 11(4):286–298, 1992.

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

M. Li and A. I. Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

M. L. A. Lourakis and A. A. Argyros. Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1526–1531, Beijing, China, oct 2005.

F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

M. Mazuran and G. D. Tipaldi. Nonlinear graph sparsification for SLAM. In *Robotics: Science and Systems*, pages 1–8, 2014.

M. Mazuran, W. Burgard, and G. D. Tipaldi. Nonlinear factor recovery for long-term SLAM. *The International Journal of Robotics Research*, 35(1-3):50–72, 2016.

J. A. Meijerink, V. D. Vorst, and H. A. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation*, 31 (137):148–162, 1977.

E. M. Mikhail and F. E. Ackermann. *Observations and least squares.* Harper and Row, New York, 1976.

M. Montemerlo. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Eighteenth National Conference on Artificial Intelligence*, pages 593–598, Edmonton, Alberta, Canada, jul 2002.

M. Montemerlo and S. Thrun. *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, volume 27 of *Springer Tracts in Advanced Robotics*. Springer, 2007.

H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A cersatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

K. P. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems*, pages 1015–1021, Denver, USA, 2000.

R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *International Conference on Computer Vision*, pages 2320–2327, Barcelona, Spain, 2011.

J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

E. Olson, J. J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2262–2269, Orlando, FL, USA, may 2006.

G. Oriolo, M. Vendittelli, L. Freda, and G. Troso. The SRT Method: Randomized strategies for exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4688–4694, New Orleans, LA, USA, apr 2004.

M. A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *International Joint Conference on Artificial Intelligence*, pages 1157–1164, Acapulco, Mexico, aug 2003.

L. Polok, V. Ila, M. Solony, P. Smrz, and P. Zemcik. Incremental block cholesky factorization for nonlinear least squares in robotics. *Robotics: Science and Systems*, 2013.

M. J. D. Powell. A new algorithm for unconstrained optimization. In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65. Academic Press, 1970.

E. Prestes e Silva, P. M. Engel, M. Trevisan, and M. A. P. Idiart. Exploration method using harmonic functions. *Robotics and Autonomous Systems*, 40(1):25–42, 2002.

E. Schmidt. Zur theorie der linearen und nichtlinearen integralgleichungen. *Mathematische Annalen*, 63(4):433–476, 1907.

M. Schmidt, E. Berg, M. Friedlander, and K. P. Murphy. Matlab PQN toolbox. URL https://www.cs.ubc.ca/{~}schmidtm/Software/PQN.html.

M. Schmidt, E. Berg, M. Friedlander, and K. P. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. In *Artificial Intelligence and Statistics*, pages 456–463, Clearwater Beach, Florida USA, apr 2009.

R. Shade and P. Newman. Choosing where to go: Complete 3D exploration with stereo. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2811, Shanghai, China, may 2011.

D. F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656, 1970.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, jul 1948.

R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.

R. C. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *International Symposium on Robotics Research*, pages 467–474, Santa Cruz, CA, USA, aug 1988. MIT Press.

R. C. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, pages 167–193. 1990.

C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. *Robotics: Science and Systems*, pages 65–72, 2005.

S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *The International Journal of Robotics Research*, 19(11):972–999, 2000.

S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. F. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics ResearchInternational Journal Robotics Research*, 23(7-8): 693–716, 2004.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics.* MIT Press, Cambridge, 2005.

W. F. Tinney and J. W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. *Proceedings of the IEEE*, 55(11):1801–1809, 1967.

L. Torabi, M. Kazemi, and K. Gupta. Configuration space based efficient view planning and exploration with occupancy grids. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2827–2832, San Diego, CA, USA, oct 2007.

R. Valencia, J. V. Miró, G. Dissanayake, and J. Andrade-Cetto. Active pose SLAM. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1885–1891, Vilamoura, Portugal, oct 2012.

R. Valencia, J. Saarinen, H. Andreasson, J. Vallvé, J. Andrade-Cetto, and A. J. Lilienthal. Localization in highly dynamic environments using dual-timescale NDT-MCL. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3956–3962, Hong Kong, China, may 2014.

J. Vallvé and J. Andrade-Cetto. Mobile robot exploration with potential information fields. In *IEEE European Conference on Mobile Robots (ECMR)*, pages 222–227, Barcelona, Spain, sep 2013.

J. Vallvé and J. Andrade-Cetto. Dense entropy decrease estimation for mobile robot exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6083–6089, Hong Kong, China, may 2014.

J. Vallvé and J. Andrade-Cetto. Potential information fields for mobile robot exploration. *Robotics and Autonomous Systems*, 69(1):68–79, 2015a.

J. Vallvé and J. Andrade-Cetto. Active pose SLAM with RRT*. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2167–2173, Seattle, WA, USA, may 2015b.

J. Vallvé, J. Solà, and J. Andrade-Cetto. Factor descent optimization for sparsification in graph SLAM. In *IEEE European Conference on Mobile Robots (ECMR)*, pages 95–100, Paris, France, sep 2017.

J. Vallvé, J. Solà, and J. Andrade-Cetto. Graph SLAM sparsification with populated topologies using factor descent optimization. *IEEE Robotics and Automation Letters*, 3(2):1322–1329, 2018a.

J. Vallvé, J. Solà, and J. Andrade-Cetto. Pose-graph SLAM sparsification using factor descent. *Robotics and Autonomous Systems*, page conditionally accepted, 2018b.

J. Vial, H. F. Durrant-Whyte, and T. Bailey. Conservative sparsification for efficient and consistent approximate estimation. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 886–893, San Francisco, CA, USA, sep 2011.

T. Vidal-Calleja, A. Sanfeliu, and J. Andrade-Cetto. Action selection for single camera SLAM. *IEEE Transactions on Systems, Man and Cybernetics*, 40(6):1567–1581, 2010.

P. Whaite and F. P. Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Anal. Mach. Intell.*, 19(3):193–205, 1997.

P. Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

P. Wolfe. Convergence conditions for ascent methods. II: Some corrections. *SIAM Review*, 13(2):185–188, 1971.

B. Yamauchi. A frontier-based approach for autonomous exploration. In *IEEE International Symposium Computational Intelligence in Robotics and Automation*, pages 146–151, Monterey, CA, USA, jul 1997.

M. Yannakakis. Computing the minimum fill-In is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981.