Factor Graphs for Robot Perception

Frank Dellaert
Georgia Institute of Technology
dellaert@cc.gatech.edu

Michael Kaess
Carnegie Mellon University
kaess@cmu.edu

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Robotics

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

F. Dellaert and M. Kaess. *Factor Graphs for Robot Perception*. Foundations and Trends® in Robotics, vol. 6, no. 1-2, pp. 1–139, 2017.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-326-3

Full text available at: http://dx.doi.org/10.1561/1400000039

Foundations and Trends® in Robotics
Volume 6, Issue 1-2, 2017

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Robotics publishes survey and tutorial articles in the following topics:

- Mathematical modelling
- Kinematics
- Dynamics
- Estimation methods
- Artificial intelligence in robotics
- Software systems and architectures
- Sensors and estimation
- Planning and control
- Human-robot interaction
- Industrial robotics
- Service robotics

Information for Librarians

Foundations and Trends® in Robotics, 2017, Volume 6, 4 issues. ISSN paper version 1935-8253. ISSN online version 1935-8261. Also available as a combined paper and online subscription.

Foundations and Trends® in Robotics
Vol. 6, No. 1-2 (2017) 1–139

DOI: 10.1561/2300000043

now
the essence of knowledge

Factor Graphs for Robot Perception

Frank Dellaert
Georgia Institute of Technology
dellaert@cc.gatech.edu

Michael Kaess
Carnegie Mellon University
kaess@cmu.edu

Contents

Abstract

We review the use of factor graphs for the modeling and solving of large-scale inference problems in robotics. Factor graphs are a family of probabilistic graphical models, other examples of which are Bayesian networks and Markov random fields, well known from the statistical modeling and machine learning literature. They provide a powerful abstraction that gives insight into particular inference problems, making it easier to think about and design solutions, and write modular software to perform the actual inference. We illustrate their use in the simultaneous localization and mapping problem and other important problems associated with deploying robots in the real world. We introduce factor graphs as an economical representation within which to formulate the different inference problems, setting the stage for the subsequent sections on practical methods to solve them. We explain the nonlinear optimization techniques for solving arbitrary nonlinear factor graphs, which requires repeatedly solving large sparse linear systems.

The sparse structure of the factor graph is the key to understanding this more general algorithm, and hence also understanding (and improving) sparse factorization methods. We provide insight into the graphs underlying robotics inference, and how their sparsity is affected by the implementation choices we make, crucial for achieving highly performant algorithms. As many inference problems in robotics are incremental, we also discuss the iSAM class of algorithms that can reuse previous computations, re-interpreting incremental matrix factorization methods as operations on graphical models, introducing the Bayes tree in the process. Because in most practical situations we will have to deal with 3D rotations and other nonlinear manifolds**,** we also introduce the more sophisticated machinery to perform optimization on nonlinear manifolds. Finally, we provide an overview of applications of factor graphs for robot perception, showing the broad impact factor graphs had in robot perception.

F. Dellaert and M. Kaess. *Factor Graphs for Robot Perception*. Foundations and Trends® in Robotics, vol. 6, no. 1-2, pp. 1–139, 2017.

DOI: 10.1561/2300000043.

1

Introduction

This article reviews the use of factor graphs for the modeling and solving of large-scale inference problems in robotics, including the simultaneous localization and mapping (SLAM) problem. Factor graphs are a family of probabilistic graphical models, other examples of which are Bayesian networks and Markov random fields, which are well known from the statistical modeling and machine learning literature. They provide a powerful abstraction to give insight into particular inference problems, making it easier to think about and design solutions, and write modular, flexible software to perform the actual inference. Below we illustrate their use in SLAM, one of the key problems in mobile robotics. Other important problems associated with deploying robots in the real world are localization, tracking, and calibration, all of which can be phrased in terms of factor graphs, as well.

In this first section we introduce Bayesian networks and factor graphs in the context of robotics problems. We start with Bayesian networks as they are probably the most familiar to the reader, and show how they are useful to *model* problems in robotics. However, since sensor data is typically given to us, we introduce factor graphs as a more relevant and economical representation. We show Bayesian

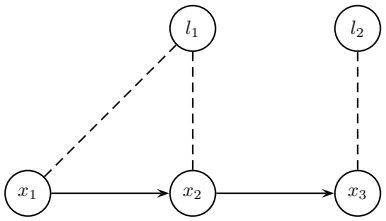

Figure 1.1: A toy SLAM (simultaneous localization and mapping) example with three robot poses and two landmarks. Above we schematically indicate the robot motion with arrows, while the dotted lines indicate bearing measurements.

networks can be effortlessly converted to factor graphs by conditioning on the sensor data. We then formulate the different inference problems as optimization problems on factor graphs, setting the stage for the subsequent sections on practical methods to solve them.

1.1 Inference Problems in Robotics

To act sensibly in the world, robots need to infer knowledge about the world from their sensors, while drawing on a priori knowledge. There are many different such inference problems in robotics, but none of them have received as much attention as simultaneous localization and mapping (SLAM). We discuss SLAM in detail and use it as a motivating example below. Other inference problems include localization in a *known* environment, tracking other actors in the environment, and multi-robot versions of all of the above. More specialized problems are also of interest, e.g., calibration or long-term inertial navigation.

In the SLAM problem the goal is to localize a robot using the information coming from the robot's sensors. In a simple case this could be a set of bearing measurements to a set of landmarks. If the landmarks' positions are known, this comes down to a triangulation problem reminiscent of how ships navigate at sea. However, the additional wrinkle in SLAM is that we do *not* know the landmark map a priori, and hence we have to infer the unknown map simultaneously with localization with respect to the evolving map.

Figure 1.1 shows a simple toy example illustrating the structure of the problem graphically. A robot located at three successive poses x_1, x_2, and x_3 makes bearing observations on two landmarks l_1 and l_2. To anchor the solution in space, let us also assume there is an absolute position/orientation measurement on the first pose x_1. Without this there would be no information about absolute position, as bearing measurements are all relative.

1.2 Probabilistic Modeling

Because of measurement uncertainty, we cannot hope to recover the true state of the world, but we can obtain a probabilistic description of what can be inferred from the measurements. In the Bayesian probability framework, we use the language of probability theory to assign a subjective degree of belief to uncertain events. Many excellent texts are available and listed at the end of this section that treat this subject in depth, which we do not have space for here.

In robotics we typically need to model a belief over continuous, multivariate random variables $x \in \mathbb{R}^n$. We do this using **probability density functions** (PDFs) $p(x)$ over the variables x, satisfying

$$\int p(x)dx = 1. \tag{1.1}$$

In terms of notation, we use lowercase letters for random variables, and uppercase letters to denote sets of them.

In SLAM we want to characterize our knowledge about the unknowns X, in this case robot poses and the unknown landmark positions, when given a set of *observed* measurements Z. Using the language of Bayesian probability, this is simply the conditional density

$$p(X|Z), \tag{1.2}$$

and obtaining a description like this is called **probabilistic inference**. A prerequisite is to first specify a probabilistic model for the variables of interest and how they give rise to (uncertain) measurements. This is where probabilistic graphical models enter the picture.

Probabilistic graphical models provide a mechanism to compactly describe complex probability densities by exploiting the struc-

ture in them [121]. In particular, high-dimensional probability densities can often be factorized as a product of many factors, each of which is a probability density over a much smaller domain. This will be explicitly modeled when we introduce factor graphs, later in this section. However, below we first introduce a different and perhaps more familiar graphical model, Bayesian networks, as they provide a gentler introduction into generative modeling.

1.3 Bayesian Networks for Generative Modeling

Bayesian networks are an expedient graphical language for modeling inference problems in robotics. This is because it is often easy to think about how measurements are generated by sensors. For example, if someone tells us the exact location of a landmark and the pose of a robot, as well as the geometry of its sensor configuration, it is not hard to *predict* what the measurement should be. And we can either assume or learn a *noise model* for a particular sensor. Measurement predictions and noise models are the core elements of a generative model, which is well matched with the Bayesian network framework.

Formally, a Bayesian network [163] or **Bayes net** is a directed graphical model where the nodes represent variables θ_j. We denote the entire set of random variables of interest as $\Theta = \{\theta_1 \dots \theta_n\}$. A Bayes net then defines a joint probability density $p(\Theta)$ over all variables Θ as the product of conditional densities associated with each of the nodes:

$$p(\Theta) \triangleq \prod_j p(\theta_j|\pi_j). \tag{1.3}$$

In the equation above $p(\theta_j|\pi_j)$ is the conditional density associated with node θ_j, and π_j is an assignment of values to the *parents* of θ_j. Hence, in a Bayes net, the factorization of the joint density is dictated by its graph structure, in particular the node-parent relationships.

As an example, let us consider the Bayes net associated with the toy SLAM example from Figure 1.1. In this case the random variables of interest are $\Theta = \{X, Z\}$, i.e., the unknown poses and landmarks X, *as well as* the measurements Z. The corresponding Bayes net for this toy example is shown in Figure 1.2, with the measurements shown in

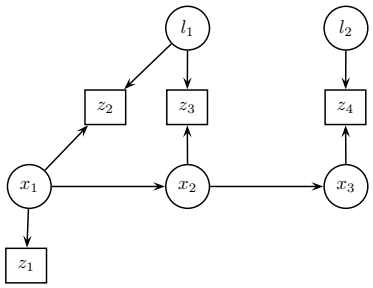

Figure 1.2: Bayes net for the toy SLAM example from Figure 1.1. Above we showed measurements with square nodes, as these variables are typically observed.

boxes as they are *observed*. Per the general definition of Bayes nets, the joint density $p(X, Z) = p(x_1, x_2, x_3, l_1, l_2, z_1, z_2, z_3, z_4)$ is obtained as a product of conditional densities:

$$\begin{aligned} p(X, Z) &= p(x_1)p(x_2|x_1)p(x_3|x_2) && (1.4)\\ &\times p(l_1)p(l_2) && (1.5)\\ &\times p(z_1|x_1) && (1.6)\\ &\times p(z_2|x_1, l_1)p(z_3|x_2, l_1)p(z_4|x_3, l_2). && (1.7) \end{aligned}$$

One can see that the joint density in this case consists of four qualitatively different sets of factors:

- A "Markov chain" $p(x_1)p(x_2|x_1)p(x_3|x_2)$ on the poses x_1, x_2, and x_3 [Eq. 1.4]. The conditional densities $p(x_{t+1}|x_t)$ might represent prior knowledge or can be derived from known control inputs.
- "Prior densities" $p(l_1)$ and $p(l_2)$ on the landmarks l_1 and l_2 (often omitted in SLAM settings when there is no prior map) [Eq. 1.5].
- A conditional density $p(z_1|x_1)$ corresponding to the absolute pose measurement on the first pose x_1 [Eq. 1.6].
- Last but not least, a product of three conditional densities, $p(z_2|x_1, l_1)p(z_3|x_2, l_1)p(z_4|x_3, l_2)$, corresponding to the three bearing measurements on the landmarks l_1 and l_2 from the poses x_1, x_2, and x_3 [Eq. 1.7].

Note that the graph structure makes an explicit statement about data association, i.e., for every measurement z_k we know which landmark it is a measurement of. While it is possible to model unknown data association in a graphical model context, in this text we assume that data association is given to us as the result of a pre-processing step.

1.4 Specifying Probability Densities

The exact form of the densities above depends very much on the application and the sensors used. The most often-used densities involve the **multivariate Gaussian distribution**, with probability density

$$\mathcal{N}(\theta; \mu, \Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left\{-\frac{1}{2}\left\|\theta - \mu\right\|_{\Sigma}^{2}\right\}, \tag{1.8}$$

where $\mu \in \mathbb{R}^n$ is the mean, Σ is an $n \times n$ covariance matrix, and

$$\left\|\theta - \mu\right\|_{\Sigma}^{2} \triangleq (\theta - \mu)^{\top} \Sigma^{-1} (\theta - \mu) \tag{1.9}$$

denotes the squared Mahalanobis distance. For example, priors on unknown quantities are often specified using a Gaussian density.

In many cases it is both justified and convenient to model measurements as corrupted by zero-mean Gaussian noise. For example, a bearing measurement from a given pose x to a given landmark l would be modeled as

$$z = h(x, l) + \eta, \tag{1.10}$$

where $h(.)$ is a **measurement prediction function**, and the noise η is drawn from a zero-mean Gaussian density with measurement covariance R. This yields the following conditional density $p(z|x, l)$ on the measurement z:

$$p(z|x, l) = \mathcal{N}(z; h(x, l), R) = \frac{1}{\sqrt{|2\pi R|}} \exp\left\{-\frac{1}{2}\left\|h(x, l) - z\right\|_{R}^{2}\right\}. \tag{1.11}$$

The measurement functions $h(.)$ are often nonlinear in practical robotics applications. Still, while they depend on the actual sensor used, they are typically not difficult to reason about or write down. The measurement function for a 2D bearing measurement is simply

$$h(x, l) = \text{atan2}(l_y - x_y, l_x - x_x), \tag{1.12}$$

where atan2 is the well-known two-argument arctangent variant. Hence, the final **probabilistic measurement model** $p(z|x,l)$ is obtained as

$$p(z|x,l) = \frac{1}{\sqrt{|2\pi R|}} \exp\left\{-\frac{1}{2}\left\|\text{atan2}(l_y - x_y, l_x - x_x) - z\right\|_R^2\right\}. \quad (1.13)$$

Note that we will not *always* assume Gaussian measurement noise: to cope with the occasional data association mistake, for example, many authors have proposed the use of robust measurement densities, with heavier tails than a Gaussian density.

Not all probability densities involved are derived from measurements. For example, in the toy SLAM problem we have densities of the form $p(x_{t+1}|x_t)$, specifying a **probabilistic motion model** which the robot is assumed to obey. This *could* be derived from odometry measurements, in which case we would proceed exactly as described above. Alternatively, such a motion model could arise from known control inputs u_t. In practice, we often use a conditional Gaussian assumption,

$$p(x_{t+1}|x_t,u_t) = \frac{1}{\sqrt{|2\pi Q|}} \exp\left\{-\frac{1}{2}\left\|g(x_t,u_t) - x_{t+1}\right\|_Q^2\right\}, \quad (1.14)$$

where $g(.)$ is a motion model, and Q a covariance matrix of the appropriate dimensionality, e.g., 3×3 in the case of robots operating in the plane. Note that for robots operating in three-dimensional space, we will need slightly more sophisticated machinery to specify densities on nonlinear manifolds such as $SE(3)$, as discussed in Section 6.

1.5 Simulating from a Bayes Net Model

As an aside, once a probability model is specified as a Bayes net, it is easy to simulate from it. This is the reason why Bayes nets are the language of choice for generative modeling, and we mention it here because it is often beneficial to think about this when building models.

In particular, to simulate from $P(\Theta) \stackrel{\Delta}{=} \prod_j P(\theta_j|\pi_j)$, one simply has to topologically sort the nodes in the graph and sample in such a way that all parent values π_j are generated before sampling θ_j from the conditional $P(\theta_j|\pi_j)$, which can always be done. This technique is called *ancestral sampling* [16].

As an example, let us again consider the SLAM toy problem. Even in this tiny problem it is easy to see how the factorization of the joint density affords us to think *locally* rather than having to think globally. Indeed, we can use the Bayes net from Figure 1.2 as a guide to simulate from the joint density $p(x_1, x_2, x_3, l_1, l_2, z_1, z_2, z_3, z_4)$ by respectively

1. sampling the poses x_1, x_2, and x_3 from $p(x_1)p(x_2|x_1)p(x_3|x_2)$, i.e., simulate a robot trajectory;

2. sampling l_1 and l_2 from $p(l_1)$ and $p(l_2)$, i.e., generate some plausible landmarks;

3. sampling the measurements from the conditional densities $p(z_1|x_1)$, $p(z_2|x_1, l_1)$, $p(z_3|x_2, l_1)$, and $p(z_4|x_3, l_2)$, i.e., simulate the robot's sensors.

Many other topological orderings are possible. For example, steps 1 and 2 above can be switched without consequence. Also, we can generate the pose measurement z_1 at any time after x_1 is generated, etc.

1.6 Maximum a Posteriori Inference

Now that we have the means to model the world, we can infer knowledge about the world when given information about it. Above we saw how to fully specify a joint density $P(\Theta)$ in terms of a Bayes net: its factorization is given by its graphical structure, and its exact computational form by specifying the associated priors and conditional densities.

In robotics we are typically interested in the **unknown state variables** X, such as poses and/or landmarks, *given* the measurements Z. The most often used *estimator* for these unknown state variables X is the maximum a posteriori or **MAP estimate**, so named because it maximizes the posterior density $p(X|Z)$ of the states X given the measurements Z:

$$X^{MAP} = \operatorname*{argmax}_X p(X|Z) \tag{1.15}$$

$$= \operatorname*{argmax}_X \frac{p(Z|X)p(X)}{p(Z)}. \tag{1.16}$$

The second equation above is Bayes' law, and expresses the posterior as the product of the measurement density $p(Z|X)$ and the prior $p(X)$ over the states, appropriately normalized by the factor $p(Z)$.

However, a different expression of Bayes law is the key to understanding the true computation underlying MAP inference. Indeed, all of the quantities in Bayes' law as stated in (1.16) can in theory be computed from the Bayes net. However, as the measurements Z are *given*, the normalization factor $p(Z)$ is irrelevant to the maximization and can be dropped. In addition, while the conditional density $p(Z|X)$ is a properly normalized Gaussian density in Z, we are only concerned with it as a function in the unknown states X. Hence the second and more important form of Bayes' law:

$$X^{MAP} = \underset{X}{\operatorname{argmax}}\, l(X; Z)p(X). \tag{1.17}$$

Here $l(X; Z)$ is the **likelihood of the states X given the measurements Z**, and is defined as any function proportional to $p(Z|X)$:

$$l(X; Z) \propto p(Z|X). \tag{1.18}$$

The notation $l(X; Z)$ emphasizes the fact that the likelihood is a function of X and *not* Z, which acts merely as a parameter in this context.

It is important to realize that conditioning on the measurements yields likelihood functions that *do not look like Gaussian densities*, in general. To see this, consider again the 2D bearing measurement density in Equation 1.13. When written as a likelihood function we obtain

$$l(x, l; z) \propto \exp\left\{-\frac{1}{2}\left\|\text{atan2}(l_y - x_y, l_x - x_x) - z\right\|_R^2\right\}, \tag{1.19}$$

which is Gaussian in z (after normalization), but decidedly not so in any other variable. Even in the case of a *linear* measurement function, the measurement z is often of lower dimensionality than the unknown variables it depends on. Hence, conditioning on it results in a degenerate Gaussian density on the unknowns, at best; it is only when we fuse the information from several measurements that the density on the unknowns becomes a proper probability density. In the case that not enough measurements are available to fully constrain all variables,

MAP inference will fail, because a unique maximizer of the posterior (1.17) is not available.

All of the above motivates the introduction of factor graphs in the next section. The reasons for introducing a new graphical modeling language are (a) the distinct division between states X and measurements Z, and (b) the fact that we are more interested in the non-Gaussian likelihood functions, which are not proper probability densities. Hence, the Bayes net language is rather mismatched with the actual optimization problem that we are concerned with. Finally, we will see in Section 3 that the structure of factor graphs is intimately connected with the computational strategies to solve large-scale inference problems.

1.7 Factor Graphs for Inference

While Bayes nets are a great language for modeling, factor graphs are better suited to perform inference. Like Bayes nets, factor graphs allow us to specify a joint density as a product of factors. However, they are more general in that they can be used to specify *any* factored function $\phi(X)$ over a set of variables X, not just probability densities.

To motivate this, consider performing MAP inference for the toy SLAM example. After conditioning on the observed measurements Z, the posterior $p(X|Z)$ can be re-written using Bayes' law (1.16) as

$$\begin{aligned} p(X|Z) &\propto p(x_1)p(x_2|x_1)p(x_3|x_2) && (1.20)\\ &\quad \times p(l_1)p(l_2) && (1.21)\\ &\quad \times l(x_1; z_1) && (1.22)\\ &\quad \times l(x_1, l_1; z_2)l(x_2, l_1; z_3)l(x_3, l_2; z_4). && (1.23) \end{aligned}$$

It is clear that the above represents a factored probability density on the unknowns only, albeit unnormalized.

To make this factorization explicit, we use a **factor graph**. Figure 1.3 introduces the corresponding factor graph by example: all unknown states X, both poses and landmarks, have a node associated with them, as in the Bayes net. However, unlike the Bayes net case, measurements are *not* represented explicitly as they are given, and hence not of interest. Rather than associating each node with a conditional density, in

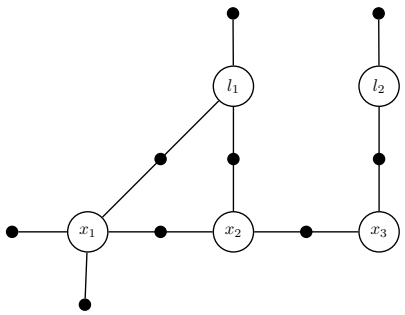

Figure 1.3: Factor graph resulting from the Bayes net in Figure 1.2 on page 6 after conditioning on the measurements Z.

factor graphs we explicitly introduce an additional node type to represent every *factor* in the posterior $p(X|Z)$. In the figure, each small black node represents a factor, and—importantly—is connected to only those state variables it is a function of. For example, the likelihood factor $l(x_3, l_2; z_4)$ is connected only to the variable nodes x_3 and l_2. Using this as a guide, it should be easy to associate each of the 9 factor nodes in the graph with the 9 factors in the posterior $p(X|Z)$.

Formally a factor graph is a bipartite graph $F = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ with two types of nodes: **factors** $\phi_i \in \mathcal{U}$ and **variables** $x_j \in \mathcal{V}$. Edges $e_{ij} \in \mathcal{E}$ are always between factor nodes and variables nodes. The set of variable nodes adjacent to a factor ϕ_i is written as $\mathcal{N}(\phi_i)$, and we write X_i for an assignment to this set. With these definitions, a factor graph F defines the factorization of a global function $\phi(X)$ as

$$\phi(X) = \prod_i \phi_i(X_i). \tag{1.24}$$

In other words, the independence relationships are encoded by the edges e_{ij} of the factor graph, with each factor ϕ_i a function of *only* the variables X_i in its adjacency set $\mathcal{N}(\phi_i)$.

Every Bayes net can be trivially converted to a factor graph. Recall that every node in a Bayes net denotes a conditional density on the corresponding variable and its parent nodes. Hence, the conversion is quite simple: every Bayes net node splits in *both* a variable node and a factor node in the corresponding factor graph. The factor is connected

to the variable node, as well as the variable nodes corresponding to the parent nodes in the Bayes net. If some nodes in the Bayes net are evidence nodes, i.e., they are given as known variables, we omit the corresponding variable nodes: the known variable simply becomes a fixed parameter in the corresponding factor.

Following this recipe, in the simple SLAM example we obtain the following factor graph factorization,

$$\begin{aligned}\phi(l_1, l_2, x_1, x_2, x_3) &= \phi_1(x_1)\phi_2(x_2, x_1)\phi_3(x_3, x_2) && (1.25)\\ &\times \phi_4(l_1)\phi_5(l_2) && (1.26)\\ &\times \phi_6(x_1) && (1.27)\\ &\times \phi_7(x_1, l_1)\phi_8(x_2, l_1)\phi_9(x_3, l_2), && (1.28)\end{aligned}$$

where the correspondence between the factors and the original probability densities and/or likelihood factors in Equations 1.20-1.23 should be obvious, e.g., $\phi_7(x_1, l_1) = l(x_1, l_1; z_2) \propto p(z_2|x_1, l_1)$.

1.8 Computations Supported by Factor Graphs

While in the remainder of this document we concentrate on fast optimization methods for SLAM, it is of interest to ask what types of computations are supported by factor graphs *in general*. Converting a Bayes net $p(X, Z)$ to a factor graph (by conditioning on the evidence Z) yields a representation of the posterior $\phi(X) \propto p(X|Z)$, and it is natural to ask what we can do with this. While in SLAM we will be able to fully exploit the specific form of the factors to perform very fast inference, some domain-agnostic operations that are supported are *evaluation*, several *optimization* methods, and *sampling*.

Given any factor graph defining an unnormalized density $\phi(X)$, we can easily **evaluate** it for any given value, by simply evaluating every factor and multiplying the results. Often it is easier to work in log or negative log-space because of the small numbers involved, in which case we have to sum as many numbers as there are factors. Evaluation opens up the way to **optimization**, and nearly all gradient-agnostic optimization methods can be applied. If the factors are differentiable functions in continuous variables, gradient-based methods can quickly

find local maxima of the posterior. In the case of discrete variables, graph search methods can be applied, but they can often be quite costly. The hardest problems involve both discrete and continuous variables.

While local or global maxima of the posterior are often of most interest, **sampling** from a probability density can be used to visualize, explore, and compute statistics and expected values associated with the posterior. However, the ancestral sampling method from Section 1.5 only applies to directed acyclic graphs. The general sampling algorithms that are most useful for factor graphs are Markov chain Monte Carlo (MCMC) methods. One such method is Gibbs sampling, which proceeds by sampling one variable at a time from its conditional density given all other variables it is connected to via factors. This assumes that this conditional density can be easily obtained, however, which is true for discrete variables but far from obvious in the general case.

Below we use factor graphs as the organizing principle for all sections on specific inference algorithms. They aptly describe the independence assumptions and sparse nature of the large nonlinear least-squares problems arising in robotics, and that is where we start in the next section. But their usefulness extends far beyond that: they are at the core of the sparse linear solvers we use as building blocks, they clearly show the nature of filtering and incremental inference, and lead naturally to distributed and/or parallel versions of robotics. Before we dive in, we first lay out the roadmap for the remainder of the document.

1.9 Roadmap

In the next section, Section 2, we discuss **nonlinear optimization** techniques for solving the map inference problem in SLAM. Doing so requires repeatedly solving large sparse linear systems, but we do not go into detail on how this is done. The resulting graph-based optimization methods are now the most popular methods for the SLAM problem, at least when solved offline or in batch.

In Section 3 we make the connection between factor graphs and **sparse linear algebra** more explicit. While there exist efficient software libraries to solve sparse linear systems, these are but instantiations of a much more general algorithm: the elimination algorithm.

In Section 4 we discuss elimination **ordering** strategies and their effect on performance. This will also allow us to understand, in Section 5, the effects of marginalizing out variables, and its possibly deleterious effect on sparsity, especially in the SLAM case. Other inference problems in robotics do benefit from only keeping track of the most recent state estimate, which leads to filtering and/or fixed-lag smoothing algorithms.

In Section 5 we discuss **incremental factorization** and reinterpret it in terms of graphical models. We introduce the Bayes tree to establish a connection between sparse matrix factorization and graphical models, based on which incremental smoothing and mapping algorithms are developed.

While in many robotics problems we can get away with vector-valued unknowns, 3D rotations and other nonlinear **manifolds** need slightly more sophisticated machinery. Hence, in Section 6 we discuss optimization on manifolds.

1.10 Bibliographic Remarks

The SLAM problem [174, 129, 186] has received considerable attention in mobile robotics as it is one way to enable a robot to explore and navigate previously unknown environments. In addition, in many applications the map of the environment itself is the artifact of interest, e.g., in urban reconstruction, search-and-rescue operations, and battlefield reconnaissance. As such, it is one of the core competencies of autonomous robots [187]. A comprehensive review was done by Durrant-Whyte and Bailey in 2006 [59, 6] and more recently by Cadena et al. [19], but the field is still generating a steady stream of contributions at the top-tier robotics conferences.

The foundational book by Pearl [163] is still one of the best places to read about Bayesian probability and Bayesian networks, as is the tome by Koller and Friedman [121], and the book by Darwiche [38]. Although in these works the emphasis is (mostly) on problems with discrete-valued unknowns, they can just as easily be applied to continuous estimation problems like SLAM.

Because of their ability to represent the unnormalized posterior for MAP inference problems, factor graphs are an ideal graphical model for probabilistic robotics. However, factor graphs are also used extensively in a variety of other computer science fields, including Boolean satisfiability, constraint satisfaction, and machine learning. Excellent overviews of factor graphs and their applications are given by Kschischang et al. [125], and Loeliger [139].

Markov chain Monte Carlo (MCMC) and Gibbs sampling provide a way to sample over high-dimensional state-spaces as described by factor graphs, and are discussed in [151, 82, 55].

[8] Beeri, C., Fagin, R., Maier, D., Mendelzon, A., Ullman, J., and Yannakakis, M. (1981). Properties of acyclic database schemes. In *ACM Symp. on Theory of Computing (STOC)*, pages 355–362, New York, NY, USA. ACM Press.

[9] Bellman, R. and Dreyfus, S. (1962). *Applied Dynamic Programming*. Princeton University Press.

[10] Bertele, U. and Brioschi, F. (1972a). *Nonserial Dynamic Programming*. Academic Press.

[11] Bertele, U. and Brioschi, F. (1972b). On the theory of the elimination process. *J. Math. Anal. Appl.*, 35(1):48–57.

[12] Bertele, U. and Brioschi, F. (1973). On nonserial dynamic programming. *J. Combinatorial Theory*, 14:137–148.

[13] Bichucher, V., Walls, J., Ozog, P., Skinner, K., and Eustice, R. (2015). Bathymetric factor graph SLAM with sparse point cloud alignment. In *OCEANS, 2015. MTS/IEEE Conference and Exhibition*.

[14] Bierman, G. (1977). *Factorization methods for discrete sequential estimation*, volume 128 of *Mathematics in Science and Engineering*. Academic Press, New York.

[15] Bierman, G. (1978). An application of the square-root information filter to large scale linear interconnected systems. *IEEE Trans. Automat. Contr.*, 23(1):91–93.

[16] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag, Secaucus, NJ, USA.

[17] Blair, J. and Peyton, B. (1993). An introduction to chordal graphs and clique trees. In [80], pages 1–27.

[18] Brown, D. C. (1976). The bundle adjustment - progress and prospects. *Int. Archives Photogrammetry*, 21(3).

[19] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robotics*, 32(6):1309–1332.

[20] Cannings, C., Thompson, E., and Skolnick, M. (1978). Probability functions on complex pedigrees. *Advances in Applied Probability*, 10:26–61.

[21] Carlevaris-Bianco, N. and Eustice, R. M. (2013a). Generic factor-based node marginalization and edge sparsification for pose-graph SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 5728–5735.

[22] Carlevaris-Bianco, N. and Eustice, R. M. (2013b). Long-term simultaneous localization and mapping with generic linear constraint node removal. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan.

[23] Carlevaris-Bianco, N., Kaess, M., and Eustice, R. (2014). Generic factor-based node removal: Enabling long-term SLAM. *IEEE Trans. Robotics*, 30(6):1371–1385.

[24] Carlone, L., Dong, J., Fenu, S., Rains, G., and Dellaert, F. (2015). Towards 4D crop analysis in precision agriculture: Estimating plant height and crown radius over time via expectation-maximization. In *ICRA Workshop on Robotics in Agriculture*.

[25] Carré, B. A. (1971). An algebra for network routing problems. *J. Inst. Math. Appl.*, 7:273–294.

[26] Castellanos, J., Montiel, J., Neira, J., and Tardós, J. (1999). The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Trans. Robot. Automat.*, 15(5):948–953.

[27] Chatila, R. and Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 138–145.

[28] Chen, Y., Davis, T., Hager, W., and Rajamanickam, S. (2008). Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3):22:1–22:14.

[29] Chiu, H., S., Dellaert, F., Samarasekera, S., and Kumar, R. (2013). Robust vision-aided navigation using sliding-window factor graphs. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe; Germany.

[30] Chiu, H., Zhou, X., Carlone, L., Dellaert, F., Samarasekera, S., and Kumar, R. (2014). Constrained optimal selection for multi-sensor robot navigation using plug-and-play factor graphs. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Hong Kong.

[31] Cooper, M. and Robson, S. (1996). Theory of close range photogrammetry. In Atkinson, K., editor, *Close range photogrammetry and machine vision*, chapter 1, pages 9–51. Whittles Publishing.

[32] Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer-Verlag.

[33] Cunningham, A., Indelman, V., and Dellaert, F. (2013). DDF-SAM 2.0: Consistent distributed smoothing and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany.

[34] Cunningham, A., Paluri, M., and Dellaert, F. (2010). DDF-SAM: Fully distributed SLAM using constrained factor graphs. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[35] Cunningham, A., Wurm, K., Burgard, W., and Dellaert, F. (2012). Fully distributed scalable smoothing and mapping with robust multi-robot data association. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, St. Paul, MN.

[36] Cuthill, E. and McKee, J. (1969). Reducing the bandwidth of sparse symmetric matrices. In *Proc. of the 1969 24th ACM national conference*, pages 157–172, New York, NY, USA. ACM Press.

[37] D'Ambrosio, B. (1994). Symbolic probabilistic inference in large BN2O networks. In *Proc.* 10^{th} *Conf. on Uncertainty in AI (UAI)*, pages 128–135, Seattle, WA.

[38] Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

[39] Davis, T. (2011). Algorithm 915: SuiteSparseQR, a multifrontal multithreaded sparse QR factorization package. *ACM Trans. Math. Softw.*, 38(1):8:1–8:22.

[40] Davis, T., Gilbert, J., Larimore, S., and Ng, E. (2004). A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):353–376.

[41] Davis, T. and Hager, W. (1996). Modifying a sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications*, 20(3):606–627.

[42] Dechter, R. (1996). Bucket elimination: A unifying framework for several probabilistic inference algorithms. In *Proc.* 12^{th} *Conf. on Uncertainty in AI (UAI)*, Portland, OR.

[43] Dechter, R. (1998). Bucket Elimination: A unifying framework for reasoning. In Jordan, M., editor, *Learning in Graphical Models*, pages 75–104. Kluwer Academic Press. Also published by MIT Press, 1999.

[44] Dechter, R. (1999). Bucket Elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85.

[45] Dechter, R. and Pearl, J. (1987). Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34(1):1–38.

[46] Dellaert, F. (2005). Square Root SAM: Simultaneous location and mapping via square root information smoothing. In *Robotics: Science and Systems (RSS)*.

[47] Dellaert, F. (2012). Factor graphs and GTSAM: A hands-on introduction. Technical Report GT-RIM-CP&R-2012-002, Georgia Institute of Technology.

[48] Dellaert, F. and Kaess, M. (2006). Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–1203.

[49] Dellaert, F., Kipp, A., and Krauthausen, P. (2005). A multifrontal QR factorization approach to distributed inference applied to multi-robot localization and mapping. In *Proc. 22nd AAAI National Conference on AI*, pages 1261–1266, Pittsburgh, PA.

[50] Dennis, J. and Schnabel, R. (1983). *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall.

[51] Dhiman, V., Kundu, A., Dellaert, F., and Corso, J. (2014). Modern map inference methods for accurate and faster occupancy grid mapping on higher order factor graphs. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

[52] Dissanayake, M., Newman, P., Durrant-Whyte, H., Clark, S., and Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. Robot. Automat.*, 17(3):229–241.

[53] Dong, J., Burnham, J., Boots, B., Rains, G., and Dellaert, F. (2017). 4D crop monitoring: Spatio-temporal reconstruction for agriculture. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

[54] Dong, J., Nelson, E., Indelman, V., Michael, N., and Dellaert, F. (2015). Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

[55] Doucet, A., de Freitas, N., and Gordon, N., editors (2001). *Sequential Monte Carlo Methods In Practice*. Springer-Verlag, New York.

[56] Duckett, T., Marsland, S., and Shapiro, J. (2002). Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300.

[57] Duff, I. S. and Reid, J. K. (1983). The multifrontal solution of indefinite sparse symmetric linear systems. *ACM Trans. Math. Softw.*, 9(3):302–325.

[58] Durrant-Whyte, H. (1988). Uncertain geometry in robotics. *IEEE Trans. Robot. Automat.*, 4(1):23–31.

[59] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics & Automation Magazine*.

[60] Estrada, C., Neira, J., and Tardós, J. (2005). Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Trans. Robotics*, 21(4):588–596.

[61] Eustice, R., Singh, H., and Leonard, J. (2005). Exactly sparse delayed-state filters. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2417–2424.

[62] Fagin, R., Mendelzon, A., and Ullman, J. (1982). A simplied universal relation assumption and its properties. *ACM Trans. Database Syst.*, 7(3):343–360.

[63] Farrell, J. (2008). *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill.

[64] Faugeras, O. (1993). *Three-dimensional computer vision: A geometric viewpoint*. The MIT press, Cambridge, MA.

[65] Fiduccia, C. and Mattheyses, R. (1982). A linear time heuristic for improving network partitions. In *Proc. 19th IEEE Design Automation and Conference*, pages 175–181.

[66] Forney, Jr, G. (2001). Codes on graphs: Normal realizations. *IEEE Trans. Inform. Theory*, 47(2):520–548.

[67] Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2015). IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems (RSS)*.

[68] Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2016). On-manifold preintegration for real-time visual-inertial odometry. *IEEE Trans. Robotics*.

[69] Fourie, D., Leonard, J., and Kaess, M. (2016). A nonparametric belief solution to the Bayes tree. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Daejeon, Korea.

[70] Frese, U. (2005). Treemap: An $O(\log n)$ algorithm for simultaneous localization and mapping. In *Spatial Cognition IV*, pages 455–476. Springer Verlag.

[71] Frese, U., Larsson, P., and Duckett, T. (2005). A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Trans. Robotics*, 21(2):196–207.

[72] Frese, U. and Schröder, L. (2006). Closing a million-landmarks loop. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5032–5039.

[73] Freuder, E. C. (1982). A sufficient condition for backtrack-free search. *J. ACM*, 29(1):24–32.

[74] Frey, B., Kschischang, F., Loeliger, H.-A., and Wiberg, N. (1997). Factor graphs and algorithms. In *Proc. 35th Allerton Conf. Communications, Control, and Computing*, pages 666–680.

[75] Gauss, C. (1809). *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Mabientium [Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections]*. Perthes and Besser, Hamburg, Germany. English translation available at http://name.umdl.umich.edu/AGG8895.0001.001.

[76] Gauss, C. (1810). Disquisitio de elementis ellipticis Palladis [Disquisition on the elliptical elements of Pallas]. *Göttingische gelehrte Anzeigen*.

[77] Gentleman, W. (1973). Least squares computations by Givens transformations without square roots. *IMA J. of Appl. Math.*, 12:329–336.

[78] George, A. (1973). Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363.

[79] George, A., Liu, J., and E., N. (1984). Row-ordering schemes for sparse Givens transformations. I. Bipartite graph model. *Linear Algebra Appl*, 61:55–81.

[80] George, J., Gilbert, J., and Liu, J.-H., editors (1993). *Graph Theory and Sparse Matrix Computations*, volume 56 of *IMA Volumes in Mathematics and its Applications*. Springer-Verlag, New York.

[81] Gilbert, J. and Ng, E. (1993). Predicting structure in nonsymmetric sparse matrix factorizations. In [80].

[82] Gilks, W., Richardson, S., and Spiegelhalter, D., editors (1996). *Markov chain Monte Carlo in practice*. Chapman and Hall.

[83] Gill, P., Golub, G., Murray, W., and Saunders, M. (1974). Methods for modifying matrix factorizations. *Mathematics and Computation*, 28(126):505–535.

[84] Golub, G. and Loan, C. V. (1996). *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition.

[85] Golub, G. and Plemmons, R. (1980). Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra and Its Applications*, 34:3–28.

[86] Goodman, N. and Shmueli, O. (1982). Tree queries: a simple class of relational queries. *ACM Trans. Database Syst.*, 7(4):653–677.

[87] Granshaw, S. (1980). Bundle adjustment methods in engineering photogrammetry. *Photogrammetric Record*, 10(56):181–207.

[88] Griffith, S. and Pradalier, C. (2017). Survey registration for long-term natural environment monitoring. *J. of Field Robotics*, 34(1):188–208.

[89] Grisetti, G., Kuemmerle, R., and Ni, K. (2012). Robust optimization of factor graphs by using condensed measurements. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[90] Grisetti, G., Kuemmerle, R., Stachniss, C., Frese, U., and Hertzberg, C. (2010). Hierarchical optimization on manifolds for online 2D and 3D mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Anchorage, Alaska.

[91] Guivant, J. and Nebot, E. (2001). Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Trans. Robot. Automat.*, 17(3):242–257.

[92] Guivant, J., Nebot, E., Nieto, J., and Masson, F. (2004). Navigation and mapping in large unstructured environments. *Intl. J. of Robotics Research*, 23:449–472.

[93] Gupta, A., Karypis, G., and Kumar, V. (1997). Highly scalable parallel algorithms for sparse matrix factorization. *IEEE Trans. Parallel and Distributed Systems*, 8(5):502–520.

[94] Gutmann, J.-S. and Konolige, K. (2000). Incremental mapping of large cyclic environments. In *IEEE Intl. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325.

[95] Hartley, R. and Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*. Cambridge University Press.

[96] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

[97] Heggernes, P. and Matstoms, P. (1996). Finding good column orderings for sparse QR factorization. In *Second SIAM Conference on Sparse Matrices*.

[98] Hover, F., Eustice, R., Kim, A., Englot, B., Johannsson, H., Kaess, M., and Leonard, J. (2012). Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *Intl. J. of Robotics Research*, 31(12):1445–1464.

[99] Hsiao, M., Westman, E., Zhang, G., and Kaess, M. (2017). Keyframe-based dense planar SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Singapore.

[100] Huang, T. and Kaess, M. (2015). Towards acoustic structure from motion for imaging sonar. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 758–765, Hamburg, Germany.

[101] Indelman, V., Nelson, E., Dong, J., Michael, N., and Dellaert, F. (2016). Incremental distributed inference from arbitrary poses and unknown data association: Using collaborating robots to establish a common reference. *IEEE Control Systems Magazine (CSM), Special Issue on Distributed Control and Estimation for Robotic Vehicle Networks*, 36(2):41–74.

[102] Indelman, V., Wiliams, S., Kaess, M., and Dellaert, F. (2012). Factor graph based incremental smoothing in inertial navigation systems. In *Intl. Conf. on Information Fusion, FUSION*.

[103] Indelman, V., Wiliams, S., Kaess, M., and Dellaert, F. (2013). Information fusion in navigation systems via factor graph based incremental smoothing. *Robotics and Autonomous Systems*, 61(8):721–738.

[104] Johannsson, H., Kaess, M., Fallon, M., and Leonard, J. (2013). Temporally scalable visual SLAM using a reduced pose graph. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 54–61, Karlsruhe, Germany.

[105] Julier, S. and Uhlmann, J. (2001). A counter example to the theory of simultaneous localization and map building. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 4, pages 4238–4243.

[106] Kaess, M. (2015). Simultaneous localization and mapping with infinite planes. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4605–4611, Seattle, WA.

[107] Kaess, M., Ila, V., Roberts, R., and Dellaert, F. (2010). The Bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Intl. Workshop on the Algorithmic Foundations of Robotics*.

[108] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., and Dellaert, F. (2011). iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China.

[109] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., and Dellaert, F. (2012a). iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31:217–236.

[110] Kaess, M., Ranganathan, A., and Dellaert, F. (2007). Fast incremental square root information smoothing. In *Intl. Joint Conf. on AI (IJCAI)*, pages 2129–2134, Hyderabad, India.

[111] Kaess, M., Ranganathan, A., and Dellaert, F. (2008). iSAM: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378.

[112] Kaess, M., Wiliams, S., Indelman, V., Roberts, R., Leonard, J., and Dellaert, F. (2012b). Concurrent filtering and smoothing. In *Intl. Conf. on Information Fusion, FUSION*.

[113] Karczmarczuk, J. (1998). Functional differentiation of computer programs. In *Intl. Conf. on Functional Programming (ICFP)*, pages 195–203.

[114] Karypis, G. and Kumar, V. (1998). Multilevel algorithms for multi-constraint graph partitioning. In *Supercomputing '98: Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–13, Washington, DC, USA. IEEE Computer Society.

[115] Kelly, III, C. and Barclay, S. (1973). A general Bayesian model for hierarchical inference. *Organizational Behavior and Human Performance*, 10:388–403.

[116] Kernighan, B. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307.

[117] Khan, Z., Balch, T., and Dellaert, F. (2006). MCMC data association and sparse factorization updating for real time multitarget tracking with merged and multiple measurements. *IEEE Trans. Pattern Anal. Machine Intell.*, 28(12):1960–1972.

[118] Kim, A. and Eustice, R. M. (2015). Active visual SLAM for robotic area coverage. *Intl. J. of Robotics Research*, 34(4-5):457–475.

[119] Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., and Teller, S. (2010). Multiple relative pose graphs for robust cooperative mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3185–3192, Anchorage, Alaska.

[120] Knight, J., Davison, A., and Reid, I. (2001). Towards constant time SLAM using postponement. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 405–413.

[121] Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

[122] Konolige, K. (2004). Large-scale map-making. In *Proc.* 21^{th} *AAAI National Conference on AI*, San Jose, CA.

[123] Krauthausen, P., Dellaert, F., and Kipp, A. (2006). Exploiting locality by nested dissection for square root smoothing and mapping. In *Robotics: Science and Systems (RSS)*.

[124] Kschischang, F. (2003). Codes defined on graphs. *IEEE Signal Proc. Mag.*, 41:118–125.

[125] Kschischang, F., Frey, B., and Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2).

[126] Kümmerle, R., Grisetti, G., and Burgard, W. (2012). Simultaneous calibration, localization, and mapping. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3716–3721.

[127] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China.

[128] Latif, Y., Cadena, C., and Neira, J. (2013). Robust loop closing over time for pose graph SLAM. *Intl. J. of Robotics Research*, 32(14):1611–1626.

[129] Leonard, J., Durrant-Whyte, H., and Cox, I. (1992). Dynamic map building for an autonomous mobile robot. *Intl. J. of Robotics Research*, 11(4):286–289.

[130] Leonard, J. and Feder, H. (2001). Decoupled stochastic mapping. *IEEE Journal of Oceanic Engineering*, pages 561–571.

[131] Leutenegger, S., Furgale, P., Rabaud, V., Chli, M., Konolige, K., and Siegwart, R. (2013). Keyframe-based visual-inertial SLAM using nonlinear optimization. In *Robotics: Science and Systems (RSS)*.

[132] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *Intl. J. of Robotics Research*, 34(3):314–334.

[133] Levenberg, K. (1944). A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math*, 2(2):164–168.

[134] Ling, F. (1991). Givens rotation based least squares lattice related algorithms. *IEEE Trans. Signal Processing*, 39(7):1541–1551.

[135] Lipton, R., Rose, D., and Tarjan, R. (1979). Generalized nested dissection. *SIAM Journal on Applied Mathematics*, 16(2):346–358.

[136] Lipton, R. and Tarjan, R. (1979). A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189.

[137] Liu, J. (1985). Modification of the minimum-degree algorithm by multiple elimination. *ACM Trans. Math. Softw.*, 11(2):141–153.

[138] Liu, J. W. H. (1989). A graph partitioning algorithm by node separators. *ACM Trans. Math. Softw.*, 15(3):198–219.

[139] Loeliger, H.-A. (2004). An introduction to factor graphs. *IEEE Signal Processing Magazine*, pages 28–41.

[140] Lourakis, M. and Argyros, A. A. (2005). Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *Intl. Conf. on Computer Vision (ICCV)*, volume 2, pages 1526–1531. IEEE.

[141] Lu, F. and Milios, E. (1997a). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, pages 333–349.

[142] Lu, F. and Milios, E. (1997b). Robot pose estimation in unknown environments by matching 2D range scans. *J. of Intelligent and Robotic Systems*, page 249:275.

[143] Lupton, T. and Sukkarieh, S. (2012). Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Trans. Robotics*, 28(1):61–76.

[144] Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11(2):431–441.

[145] Maybeck, P. (1979). *Stochastic Models, Estimation and Control*, volume 1. Academic Press, New York.

[146] Mazuran, M., Burgard, W., and Tipaldi, G. (2016). Nonlinear factor recovery for long-term SLAM. *Intl. J. of Robotics Research*, 35(1-3):50–72.

[147] McDonald, J., Kaess, M., Cadena, C., Neira, J., and Leonard, J. (2013). Real-time 6-DOF multi-session visual SLAM over large scale environments. *Robotics and Autonomous Systems*, 61(10):1144–1158.

[148] Montanari, U. (1974). Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132.

[149] Mur-Artal, R. and Tardos, J. D. (2016). Visual-inertial monocular SLAM with map reuse. Technical report, Universidad de Zaragoza. arXiv:1610.05949.

[150] Murray, R., Li, Z., and Sastry, S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press.

[151] Neal, R. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto.

[152] Ni, K. and Dellaert, F. (2010). Multi-level submap based SLAM using nested dissection. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[153] Ni, K. and Dellaert, F. (2012). HyperSfM. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 144–151. IEEE.

[154] Ni, K., Steedly, D., and Dellaert, F. (2007). Tectonic SAM: Exact; out-of-core; submap-based SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Rome; Italy.

[155] Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag.

[156] Olson, E. and Agarwal, P. (2013). Inference on networks of mixtures for robust robot mapping. *Intl. J. of Robotics Research*, 32(7):826–840.

[157] Olson, E., Leonard, J., and Teller, S. (2006). Fast iterative alignment of pose graphs with poor initial estimates. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2262–2269.

[158] Ozog, P., Troni, G., Kaess, M., Eustice, R., and Johnson-Roberson, M. (2015). Building 3D mosaics from an autonomous underwater vehicle and 2D imaging sonar. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1137–1143, Seattle, WA.

[159] Parter, S. (1961). The use of linear graphs in Gauss elimination. *SIAM Rev.*, 3(2):119–130.

[160] Paskin, M. (2003). Thin junction tree filters for simultaneous localization and mapping. In *Intl. Joint Conf. on AI (IJCAI)*.

[161] Paz, L. M., Pinies, P., Tardós, J. D., and Neira, J. (2008). Large scale 6DOF SLAM with stereo-in-hand. *IEEE Transactions on Robotics*, 24(5):946–957.

[162] Pearl, J. (1982). Reverend Bayes on inference engines: a distributed hierarchical approach. In *Proc. First AAAI National Conference on AI*, pages 133–136, Carnegie Mellon University, Pittsburgh, PA.

[163] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[164] Pinies, P., Paz, P., Haner, S., and Heyden, A. (2012). Decomposable bundle adjustment using a junction tree. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1246–1253. IEEE.

[165] Pothen, A. and Sun, C. (1992). Distributed multifrontal factorization using clique trees. In *Proc. of the Fifth SIAM Conf. on Parallel Processing for Scientific Computing*, pages 34–40. Society for Industrial and Applied Mathematics.

[166] Pothen, A., Simon, H., and Wang, L. (1992). Spectral nested dissection. Technical Report CS-92-01, Penn. State.

[167] Powell, M. (1970). A new algorithm for unconstrained optimization. In Rosen, J., Mangasarian, O., and Ritter, K., editors, *Nonlinear Programming*, pages 31–65. Academic Press.

[168] Rosen, D., Kaess, M., and Leonard, J. (2014). RISE: An incremental trust-region method for robust online sparse least-squares estimation. *IEEE Trans. Robotics*, 30(5):1091–1108.

[169] Rosen, R. (1968). Matrix bandwidth minimization. In *Proceedings of the 1968 23rd ACM national conference*, pages 585–595, New York, NY, USA. ACM Press.

[170] Salas-Moreno, R., Newcombe, R., Strasdat, H., Kelly, P., and Davison, A. (2013). SLAM++: Simultaneous localisation and mapping at the level of objects. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1352–1359.

[171] Segal, A. and Reid, I. (2014). Hybrid inference optimization for robust pose graph estimation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2675–2682.

[172] Seidel, R. (1981). A new method for solving constraint satisfaction problems. In *Intl. Joint Conf. on AI (IJCAI)*, pages 338–342.

[173] Slama, C., editor (1980). *Manual of Photogrammetry*. American Society of Photogrammetry and Remote Sensing, Falls Church, VA.

[174] Smith, R. and Cheeseman, P. (1987). On the representation and estimation of spatial uncertainty. *Intl. J. of Robotics Research*, 5(4):56–68.

[175] Smith, R., Self, M., and Cheeseman, P. (1988). A stochastic map for uncertain spatial relationships. In *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, pages 467–474.

[176] Smith, R., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In Cox, I. and Wilfong, G., editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag.

[177] Spivak, M. (1965). *Calculus on manifolds*. Benjamin New York;.

[178] Stewart, G. (2000). The decompositional approach to matrix computation. *Computing in Science & Engineering*, 2(1):50–59. Special issue on the Top 10 Algorithms in Science & Engineering.

[179] Sünderhauf, N. and Protzel, P. (2012a). Switchable constraints for robust pose graph SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[180] Sünderhauf, N. and Protzel, P. (2012b). Towards a robust back-end for pose graph SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1254–1261. IEEE.

[181] Szeliski, R. and Kang, S. (1993). Recovering 3D shape and motion from image streams using non-linear least squares. Technical Report CRL 93/3, DEC Cambridge Research Lab.

[182] Szeliski, R. and Kang, S. (1994). Recovering 3D shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 5(1).

[183] Tanner, R. (1981). A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, 27(5):533–547.

[184] Tardós, J., Neira, J., Newman, P., and Leonard, J. (2002). Robust mapping and localization in indoor environments using sonar data. *Intl. J. of Robotics Research*, 21(4):311–330.

[185] Teixeira, P., Kaess, M., Hover, F., and Leonard, J. (2016). Underwater inspection using sonar-based volumetric submaps. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4288–4295, Daejeon, Korea.

[186] Thrun, S. (2003). Robotic mapping: a survey. In *Exploring artificial intelligence in the new millennium*, pages 1–35. Morgan Kaufmann, Inc.

[187] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT press, Cambridge, MA.

[188] Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., and Durrant-Whyte, H. (2004). Simultaneous localization and mapping with sparse extended information filters. *Intl. J. of Robotics Research*, 23(7-8):693–716.

[189] Toohey, L., Pizarro, O., and Williams, S. (2014). Multi-vehicle localisation with additive compressed factor graphs. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4584–4590.

[190] Trevor, A. J. B., Rogers III, J. G., and Christensen, H. I. (2012). Planar surface SLAM with 3D and 2D sensors. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, St. Paul, MN. IEEE.

[191] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment – a modern synthesis. In Triggs, W., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *LNCS*, pages 298–372. Springer Verlag.

[192] Tweddle, B., Saenz-Otero, A., Leonard, J., and Miller, D. (2015). Factor graph modeling of rigid-body dynamics for localization, mapping, and parameter estimation of a spinning object in space. *J. of Field Robotics*, 32(6):897–933.

[193] Usenko, V., Engel, J., Stückler, J., and Cremers, D. (2016). Direct visual-inertial odometry with stereo cameras. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1885–1892, Stockholm, Sweden.

[194] VanMiddlesworth, M., Kaess, M., Hover, F., and Leonard, J. (2013). Mapping 3D underwater environments with smoothed submaps. In *Field and Service Robotics (FSR)*, pages 17–30, Brisbane, Australia.

[195] Walter, M., Hemachandra, S., Homberg, B., Tellex, S., and Teller, S. (2014). A framework for learning semantic maps from grounded natural language descriptions. *Intl. J. of Robotics Research*, 33(9):1167–1190.

[196] Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J., and McDonald, J. (2015). Real-time large scale dense RGB-D SLAM with volumetric fusion. *Intl. J. of Robotics Research*, 34(4-5):598–626.

[197] Whelan, T., Kaess, M., Leonard, J. J., and McDonald, J. (2013). Deformation-based loop closure for large scale dense RGB-D SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo,Japan.

[198] Wiberg, N. (1996). *Codes and Decoding on General Graphs*. PhD thesis, Linkoping University, Sweden.

[199] Wiberg, N., Loeliger, H.-A., and Kötter, R. (1995). Codes and iterative decoding on general graphs. In *Euro. Trans. Telecomm.*, volume 6, pages 513–525.

[200] Williams, S., Indelman, V., Kaess, M., Roberts, R., Leonard, J. J., and Dellaert, F. (2014). Concurrent filtering and smoothing: A parallel architecture for real-time navigation and full smoothing. *Intl. J. of Robotics Research*.

[201] Winkler, G. (1995). *Image analysis, random fields and dynamic Monte Carlo methods*. Springer Verlag.

[202] Yan, X., Indelman, V., and Boots, B. (2017). Incremental sparse GP regression for continuous-time trajectory estimation and mapping. *Robotics and Autonomous Systems*, 87(1):120–132.

[203] Yannakakis, M. (1981). Computing the minimum fill-in is NP-complete. *SIAM J. Algebraic Discrete Methods*, 2.

[204] Yedidia, J., Freeman, W., and Y.Weiss (2000). Generalized belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 689–695.

[205] Zhang, J., Kaess, M., and Singh, S. (2014). Real-time depth enhanced monocular odometry. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4973–4980, Chicago, IL.

[206] Zhang, J., Kaess, M., and Singh, S. (2017). A real-time method for depth enhanced monocular odometry. 41(1):31–43.

[207] Zhang, N. and Poole, D. (1994). A simple approach to Bayesian network computations. In *Proc. of the 10th Canadian Conf. on AI*, Banff, Alberta, Canada.

Appendices

A

Multifrontal Cholesky Factorization

We recover sparse **multifrontal Cholesky factorization** if we instead use partial Cholesky factorization when eliminating a single variable. To enable this, when eliminating the variable x_j, the product factor $\psi(x_j, S_j)$ is handled in a slightly different way. In particular, we define the augmented Jacobian matrix $\widehat{A}_j \triangleq [\bar{A}_j | \bar{b}_j]$ associated with the product factor $\psi(x_j, S_j)$, and the corresponding augmented state $\widehat{x} \triangleq [x_j; S_j; 1]$. We then have

$$\left\| \bar{A}_j [x_j; S_j] - \bar{b}_j \right\|_2^2 = \widehat{x}^\top (\widehat{A}_j^\top \widehat{A}_j) \widehat{x}, \tag{A.1}$$

where $\widehat{\Lambda}_j \triangleq \widehat{A}_j^\top \widehat{A}_j$ is the **augmented Hessian matrix** associated with the product factor $\psi(x_j, S_j)$. As an example, eliminating l_2 in the toy example yields the product factor

$$\widehat{\Lambda}_2 = \begin{bmatrix} A_{52}^\top A_{52} + A_{92}^\top A_{92} & A_{92}^\top A_{95} & A_{52}^\top b_5 + A_{92}^\top b_9 \\ - & A_{95}^\top A_{95} & A_{95}^\top b_9 \\ - & - & b_5^\top b_5 + b_9^\top b_9 \end{bmatrix}, \tag{A.2}$$

which one can see to be the sum of two outer products, corresponding to the factors ϕ_5 and ϕ_9.

We partition $\widehat{\Lambda}_j$ into 4 blocks, isolating the blocks associated with the variable x_j, and perform the following partial Cholesky factorization:

$$\widehat{\Lambda}_j = \begin{bmatrix} \widehat{\Lambda}_{11} & \widehat{\Lambda}_{12} \\ \widehat{\Lambda}_{21} & \widehat{\Lambda}_{22} \end{bmatrix} = \begin{bmatrix} R_j^\top & \\ S^\top & L^\top \end{bmatrix} \begin{bmatrix} R_j & S \\ & L \end{bmatrix}. \tag{A.3}$$

The upper triangular matrix R_j, satisfying $R_j^\top R_j = \widehat{\Lambda}_{11}$, will be identical to the one obtained by QR factorization up to possibly sign flips on the diagonal. The remaining blocks S and L can be computed by

$$\begin{aligned} S &= R_j^{-\top} \widehat{\Lambda}_{12} && \text{(A.4)} \\ L^\top L &= S^\top S && \text{(A.5)} \\ &= \widehat{\Lambda}_{22} - \widehat{\Lambda}_{12}^\top \widehat{\Lambda}_{11}^{-1} \widehat{\Lambda}_{12}. && \text{(A.6)} \end{aligned}$$

The latter computation, known as the Schur complement, has a nice information-theoretic interpretation: we *downdate* the information $\widehat{\Lambda}_{22}$ on the separator S_j with the information we "consume" in order to determine the eliminated variable x_j. The more information $\widehat{\Lambda}_{11}$ we had on x_j, the more information remains on the separator S_j.

After the partial Cholesky step, the algorithm proceeds by creating a conditional density from R and S, given by

$$p(x_j|S_j) \propto \exp\left\{-\frac{1}{2}\left\|R_j x_j + T_j S_j - d_j\right\|_2^2\right\} \tag{A.7}$$

with $[T_j|d_j] = S$. This conditional is exactly the same as the one we recover via the QR path. Adding the new factor on the separator S_j corresponding to $L^\top L$ needs some care: we can indeed create a new factor, but with the corresponding error

$$\tau(S_j) = \exp\left\{-\frac{1}{2}\widehat{S_j}^\top (L^\top L)\widehat{S_j}\right\} \tag{A.8}$$

rather than the Jacobian form as used in Equation 3.20 on page 39.

B

Lie Groups and other Manifolds

Many of the unknown variables in robotics live in well-known continuous transformation groups known as **Lie groups**. A rigorous definition will take us too far afield, but roughly speaking a Lie group is simply a manifold with a smooth group operation defined on it. The most important examples are reviewed below.

B.1 2D Rotations

One of the simplest Lie groups is the space of 2D rotations with composition as the group operator, also known as the **Circle Group**. The easiest way to define it is as the subset of all 2×2 invertible matrices that are both orthogonal and have determinant one, i.e., 2×2 rotation matrices. Because of this definition, people often refer to this Lie group the as the **Special Orthogonal Group** in dimension 2, written as $SO(2)$. Here "special" refers to the unit determinant property.

The nonlinear orthogonality and unit determinant constraints define a nonlinear, one-dimensional manifold within the larger 4-dimensional space of 2×2 invertible matrices. In fact, the manifold has the topology of a circle, but it remains a group: matrix multiplica-

tion of two rotation matrices in $SO(2)$ is closed, the identity matrix I_2 is in $SO(2)$, and the inverse element of each rotation R is its transpose $R^\top$, which is also in $SO(2)$. Hence, $SO(2)$ is a subgroup of the **General Linear Group** $GL(2)$ of 2×2 invertible matrices.

What makes this Lie group stand out from all other groups we discuss below is that the group operation is **commutative**: $R_1 R_2 = R_2 R_1$ for all $R_1, R_2 \in SO(2)$. This explains why people often simply represent a planar rotation with an angle $\theta \in \mathbb{R}$, and use scalar addition as a proxy for the group operation. However, while matrix multiplication respects the circle topology, scalar addition does not.

An important representation that *does* respect the wrap-around property is the group of unit-norm complex numbers $\cos\theta + i \sin\theta \in \bar{\mathbb{C}}$ with complex multiplication, which is isomorphic to $SO(2)$.

In summary, these are the three most common representations used for rotations: angles, complex numbers, and 2×2 rotation matrices,

$$\mathbb{R} \to \bar{\mathbb{C}} \leftrightarrow SO(2) \tag{B.1}$$

$$\theta \to \cos\theta + i \sin\theta \leftrightarrow \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{B.2}$$

where the first arrow indicates an (undesirable) many to-one mapping.

B.2 2D Rigid Transformations

Equipped with $SO(2)$ we can model the orientation of robots moving in the plane. Just as it was convenient to embed the one-dimensional manifold $SO(2)$ in $GL(2)$, we likewise embed both orientation $R \in SO(2)$ and position $t \in \mathbb{R}^2$ in the space of 3×3 matrices, as follows:

$$T \triangleq \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}. \tag{B.3}$$

The above defines the **Special Euclidean Group** $SE(2)$. It is a subgroup of the general linear group $GL(3)$, with matrix multiplication as the group operation. The identity element is $I_3 \in GL(3)$, and we have

$$T^{-1} = \begin{bmatrix} R^\top & -R^\top t \\ 0 & 1 \end{bmatrix} \tag{B.4}$$

and

$$T_1 T_2 = \begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2 & t_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1 R_2 & R_1 t_2 + t_1 \\ 0 & 1 \end{bmatrix}. \tag{B.5}$$

Note that composition in $SE(2)$ is *not* commutative.

For planar robots, we can use elements of $SE(2)$ to represent the **2D pose** x of the robot, i.e., $x \in SE(2)$. We can interpret a pose $x_i = T_i \in SE(2)$ as the transformation that would take us from the origin to the coordinate frame associated with the robot's current pose.

Relative poses are also elements of $SE(2)$: suppose $x_i = T_i$ and $x_j = T_j$, then we have

$$x_j = T_j = T_i T_i^{-1} T_j = x_i (T_i^{-1} T_j) = x_i T_j^i \tag{B.6}$$

and hence $T_j^i \triangleq T_i^{-1} T_j$ is the transformation that takes x_i to x_j.

The natural **group action** associated with an element $T_i \in SE(2)$ transforms points $p^i \in \mathbb{R}^2$ in coordinate frame i to points $q^g \in \mathbb{R}^2$ in the global frame by embedding both in $\mathbb{P}^2$ using homogeneous coordinates:

$$\begin{bmatrix} q^g \\ 1 \end{bmatrix} = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p^i \\ 1 \end{bmatrix} = \begin{bmatrix} R_i p^i + t_i \\ 1 \end{bmatrix}. \tag{B.7}$$

We write $q^g = T_i \otimes p^i$, and the change from local to global coordinates is $q^g = R_i p^i + t_i$, i.e., the local point p^i is rotated and then translated.

To model measurements taken from a particular robot pose $x_i = T_i$, a more important question is: if we know the location of a landmark $l_j = q^g \in \mathbb{R}^2$ in the global coordinate frame, what are its coordinates p^i in the robot's frame? Since the inverse of R_i is $R_i^\top$, the inverse transformation follows easily from (B.7) as $p^i = R_i^\top (q^g - t_i)$.

B.3 3D Rotations

The Lie group $SO(3)$ of rotations in 3D (aka spatial rotations) is represented by the set of 3×3 matrices that are orthogonal and have determinant 1. 3D rotations are important in robotics but also in navigation and many other fields, and hence this Lie group is one of the most studied and well-known structures in applied math.

$SO(3)$ is a three-dimensional manifold embedded within a 9-dimensional ambient space, and forms a subgroup within $GL(3)$ in the same way $SO(2)$ is a subgroup of $GL(2)$. However, unlike planar rotations, *spatial rotations do not commute*. In other words,

$$R_1 R_2 \neq R_2 R_1 \tag{B.8}$$

for most $R_1, R_2 \in SO(3)$. Of course, since $SO(2)$ is a subgroup of $SO(3)$ (keep any axis fixed), it is clear that *some* combinations of rotation matrices do commute, just not all.

The subgroup relationship between $SO(2)$ and $SO(3)$ gives rise to the commonly used **axis-angle** representation for spatial rotations. It consists of the pair $(\bar{\omega}, \theta)$, where the axis $\bar{\omega} \in S^2$ is a unit vector on the sphere and $\theta \in \mathbb{R}$ is a rotation angle around this axis. Both can be combined in a single three-vector $\omega = \theta\bar{\omega}$. While convenient for some operations, composition of two rotations is cumbersome and is best achieved by converting back to rotation matrices. In addition, because of the dependence on a scalar angle θ, there is again an undesirable many-to-one mapping from axis-angle to $SO(3)$.

Another, very common way to represent 3D rotations is using **unit quaternions** $q \in \bar{\mathbb{Q}}$, analogous to the role unit complex numbers play for $SO(2)$. Quaternions, like complex numbers, have a real part and an imaginary part, but the imaginary part in quaternions is three-dimensional, with axes i, j, and k. The easiest way to introduce unit-quaternions as a way to represent rotations is by converting from the axis angle representation,

$$(\bar{\omega}, \theta) \rightarrow \cos\frac{\theta}{2} + (\bar{\omega}_x i + \bar{\omega}_y j + \bar{\omega}_z k)\sin\frac{\theta}{2}, \tag{B.9}$$

which highlights that the axis $\bar{\omega}$ is encoded in the imaginary part. Unit quaternions are more compact than 3×3 matrices and, equipped with quaternion multiplication, are *almost* isomorphic to $SO(3)$. Indeed, their only flaw is that there is a two-to-one mapping from $\bar{\mathbb{Q}}$ to $SO(3)$: q and $-q$ represent the same rotation. Despite this minor annoyance, they are a popular representation in robotics.

Finally, the most intuitive but often problematic representation for 3D rotations consists of using **Euler angles**. These are quite useful

from a readability point of view, because rotations around identity can be easily understood as a combination of **roll ϕ, pitch θ, and yaw ψ**—making the three degrees of freedom palatable where rotation matrices and unit quaternions obfuscate. However, far from identity, Euler angles exhibit singularities which complicate optimizing over them when used in those regimes.

In summary, these are the four most common representations used for spatial rotations: axis-angle, unit quaternions, and 3×3 rotation matrices, and Euler angles:

$$S^2 \times \mathbb{R} \leftrightarrow \bar{\mathbb{Q}} \rightrightarrows SO(3) \leftarrow \mathbb{R}^3 \tag{B.10}$$

$$(\bar{\omega}, \theta) \leftrightarrow \cos\frac{\theta}{2} + (\bar{\omega}_x i + \bar{\omega}_y j + \bar{\omega}_z k)\sin\frac{\theta}{2} \rightrightarrows R \leftarrow \phi, \theta, \psi, \tag{B.11}$$

where the double arrow represents the double covering property of unit quaternions, and the last arrow indicates the undesirable many to-one mapping from Euler angles to rotation matrices (even more so now, because of the inherent singularities).

B.4 3D Rigid Transformations

The full 6 DOF pose of a robot operating in free space or on undulating terrain can be represented using rigid 3D transformations. The situation is completely analogous to the 2D case in Section B.2: we embed a rotation matrix $R \in SO(3)$ and a translation vector $t \in \mathbb{R}^3$ in a 4×4 matrix

$$T \triangleq \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{B.12}$$

to define the **Special Euclidean Group** $SE(3)$ of rigid 3D transformations. Again, the group operation is matrix multiplication, and $SE(3)$ is a subgroup of the 4×4 invertible matrices $GL(4)$.

B.5 Directions in 3D

An important nonlinear manifold that is *not* a group is the set of all directions in 3D space. These are useful for reasoning about a robot's

orientation with respect to gravity, such as measured by an accelerometer for instance. Another use case is visual odometry using a monocular camera only, in which case absolute scale is unobservable between two frames, but translation direction is.

A direction in space is conveniently represented by a unit 3-vector, i.e., $p = \begin{bmatrix} x & y & z \end{bmatrix}^\top$ with the nonlinear constraint $x^2 + y^2 + z^2 = 1$. In other words, the manifold of directions in 3D space is the **Sphere in 3D**, typically denoted S^2. It is a *two-dimensional* manifold, as the nonlinear constraint takes away one degree of freedom, and indeed, the sphere is intuitively familiar to us as a two-dimensional surface.