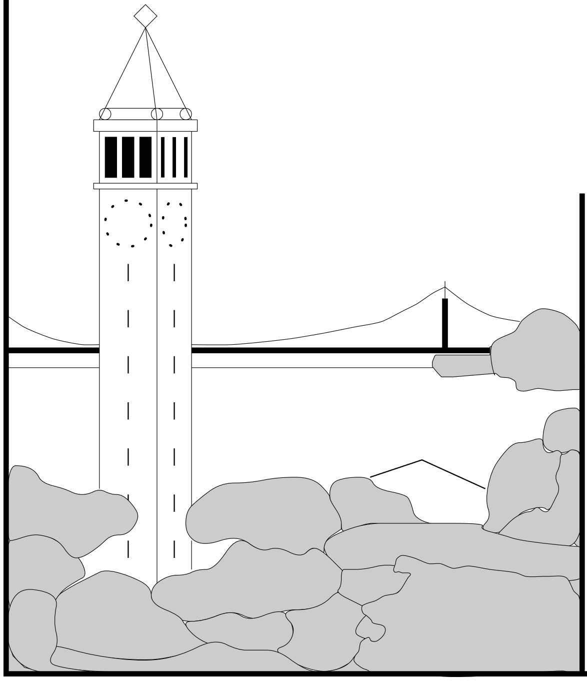
Thin Junction Tree Filters for Simultaneous Localization and Mapping

(Revised May 14, 2003)

Mark A. Paskin



Report No. UCB/CSD-02-1198

September 2002

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Thin Junction Tree Filters for Simultaneous Localization and Mapping

(Revised May 14, 2003)[*]

Mark A. Paskin
paskin@cs.berkeley.edu

September 2002

Abstract

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in mobile robotics: while a robot navigates in an unknown environment, it must incrementally build a map of its surroundings and localize itself within that map. Traditional approaches to the problem are based upon Kalman filters, but suffer from complexity issues: the size of the belief state and the time complexity of the filtering operation grow quadratically in the size of the map. This paper presents a filtering technique that maintains a tractable approximation of the filtered belief state as a thin junction tree. The junction tree grows under measurement and motion updates and is periodically "thinned" to remain tractable via efficient maximum likelihood projections. When applied to the SLAM problem, these thin junction tree filters have a linear-space belief state representation, and use a linear-time filtering operation. Further approximation can yield a constant-time filtering operation, at the expense of delaying the incorporation of observations into the majority of the map. Experiments on a suite of SLAM problems validate the approach.

[*] This is a reorganized and expanded version of the original report that includes some experimental results. Shortly it will be revised again to include the results of more experiments.

Contents

1 Introduction

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in mobile robotics: while a robot navigates in an unknown environment, it must incrementally build a map of its surroundings and localize itself within that map [Thrun, 2002]. The traditional approach treats SLAM as a Kalman filtering problem [Smith and Cheeseman, 1986; Smith *et al.*, 1990; Leonard *et al.*, 1992]. The system state at time t is

$$\boldsymbol{m}_t = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{l}_1 \\ \vdots \\ \boldsymbol{l}_{n_t} \end{bmatrix} \tag{1}$$

where $\boldsymbol{x}_t$ is the pose of the robot at time t, $\boldsymbol{l}_i$ is the state of landmark i (landmarks are assumed stationary) and n_t is the number of landmarks observed up to time t.[1] (Thus, the length of the state vector will increase over time, as more landmarks are observed.) The filtered belief state at time t, $p(\boldsymbol{m}_t \mid \text{observations to time } t)$, is a multivariate Gaussian distribution $\mathcal{N}(\mu_t, \Sigma_t)$; we can view its mean μ_t as an estimate of the map $\boldsymbol{m}_t$ and its covariance matrix Σ_t as a measure of its estimation confidence.

The Kalman filter solution is elegant, but it does not scale to large SLAM problems. Because it explicitly represents correlations between all pairs of landmark estimates (in Σ_t), the size of its belief state grows as $O(n_t^2)$; and because each of these correlations must be updated whenever a landmark is reobserved, the time complexity of its filter operation is also $O(n_t^2)$. This quadratic complexity renders the Kalman filter inapplicable to large SLAM problems and gives rise to the need for principled, efficient approximations.

The simplest approach to this problem is to discard correlations so each variable is estimated independently; however, this presents two problems. First, ignoring robot-landmark correlations leads to overconfidence and divergence; this happens because the filter fails to account for the fact that multiple observations of the same landmark may give the same or similar information about its location [Hébert *et al.*, 1996]. Second, maintaining inter-landmark correlations is required for quick convergence when the robot *closes a loop*, i.e., it reobserves known landmarks after an extended period of mapping unknown territory. Thus, the challenge of scalable SLAM filtering is to *estimate and reason with quadratically many correlations without quadratic time or space complexity.*

In this report we present a novel and general approximate filtering method satisfying this criterion. Our point of departure (in Section 2) is to view the belief state of the Kalman filter as a Gaussian graphical model that evolves over time; this allows us to express correlations—which the Kalman filter represents explicitly—in terms of direct and indirect dependencies. Analyzing the evolution of this graphical model motivates an approximation scheme in which weak or redundant dependencies are removed to improve the complexity of inference. Because some direct dependencies remain, indirect dependencies—and thus correlations—persist between each pair of variables.

To implement such a scheme, we represent the filter belief state using a junction tree. In Section 3 we present methods for updating this representation to reflect measurements of the system and its evolution. These updates can cause the width of the junction tree to grow, which makes inference more expensive; in Section 4 we present a novel "thinning" operation over junction trees called *variable contraction.*[2] We prove that each variable contraction is a maximum likelihood approximation that removes a *set* of edges from the corresponding graphical model. The approximation error introduced by a variable contraction can

[1] Boldface roman letters denote random variables and the corresponding plain letters denote specific values for these variables; parameters are denoted by Greek letters. Where it will not cause confusion, time subscripts are omitted.

[2] We use the word "thin" in the same sense as in [Bach and Jordan, 2002]: a *thin* junction tree is one with limited width (or maximum cluster size). Whereas Bach and Jordan present a method of density estimation that introduces new edges subject to the constraint that the junction tree is thin, we do exactly the opposite: we remove edges until the junction tree is thin.

be computed efficiently; this allows us to choose which edges to remove at each time step so as to minimize the error.

In Section 5 we apply these techniques to the SLAM problem and obtain a *Thin Junction Tree Filter* (TJTF) with a $O(n_t)$ space belief state representation and a $O(n_t)$ time filter operation. By delaying the incorporation of recent evidence into the majority of the map, we improve the filter's time complexity to $O(1)$. We also present a method of evaluating the significance evidence has on different portions of the map, which can be used to adaptively interpolate between constant and linear-time filter operations. Empirically, we find that these adaptive filters choose constant-time updates when mapping new territory, and when closing a loop, they use time linear in the length of the loop. This is perhaps the best time complexity one would hope for in the SLAM problem, since linearly-many estimates cannot be improved in constant time. Section 6 presents the results of simulation experiments that compare TJTF to other SLAM filters and Section 7 concludes by drawing connections between the present proposal and other approaches.

1.1 The Kalman filter approach to SLAM

The traditional approach to SLAM endows $\boldsymbol{m}$ with a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, and assumes known affine measurement and motion models so that the Kalman filter can be used to estimate the map given the robot's actions and observations. (See Appendix A for background on the multivariate Gaussian distribution, the Kalman filter, and its traditional application to SLAM.) The two operations the filter must support are measurement updates (in which the robot observes a landmark or receives an odometry measurement), and motion updates (in which the robot moves).

At time step t, the robot makes k_t landmark measurements, which we will denote $z_t^1, z_t^2, \ldots, z_t^{k_t}$. The correspondence between the measurements and the landmarks from which they originated is represented by a *data association variable* $\boldsymbol{\delta}_t$: landmark measurement z_t^i originates from landmark $\boldsymbol{\delta}_t^i$. Given $\boldsymbol{\delta}_t$, each landmark measurement is modeled as a noisy affine function of the state:

$$\boldsymbol{z}_t^i = E\boldsymbol{x}_t + F\boldsymbol{l}_{\boldsymbol{\delta}_t^i} + g + \boldsymbol{u} \tag{2}$$

where $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, S)$ is an independent white noise variable. (Of course, the parameters E, F, g and S may differ for different measurements.) Thus, all measurements are conditionally independent given $\boldsymbol{m}_t$ and $\boldsymbol{\delta}_t$.

When the data association $\boldsymbol{\delta}_t$ is unknown, it is popular to assume it is distributed uniformly and approximate it with its *a posteriori* maximum likelihood value under $p(\boldsymbol{\delta}_t \,|\, \boldsymbol{m}_t, z_t)$; i.e., we choose δ_t to be that value that associates each measurement with the landmark that was most likely to generate it. In some domains, data associations are constrained such that two measurements cannot be associated with the same landmark; in this case, we can evaluate the likelihoods $p(z_t^i \,|\, \boldsymbol{x}_t, \boldsymbol{l}_j, \boldsymbol{\delta}_t^i = j)$ for all measurements i and landmarks j, and performing a global matching (using, say, the Hungarian algorithm). Using this technique in very noisy environments can lead to incorrect associations, which are usually followed by catastrophic divergence of the filter; more sophisticated techniques maintain a set of hypothetical associations. In this work, we will assume that the data association is known.

New landmarks can be added to the map by a technique called *gating*, wherein measurements whose likelihoods under all data associations are below some threshold cause new landmarks to be added to the map (see Appendix B.3). In this case, the landmark state is added to the map with an uninformative (infinite variance, zero covariance) prior, and the measurement update yields an informed posterior estimate of its state (see Appendix B.2). In order to avoid incorrectly adding landmarks due to extremely noisy measurements, practical implementations usually require a hypothesis landmark to have absorbed a certain number of measurements before it is permanently added to the map.

After observing the landmarks, the robot moves based upon a noisy affine model:

$$\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + b + \boldsymbol{v} \tag{3}$$

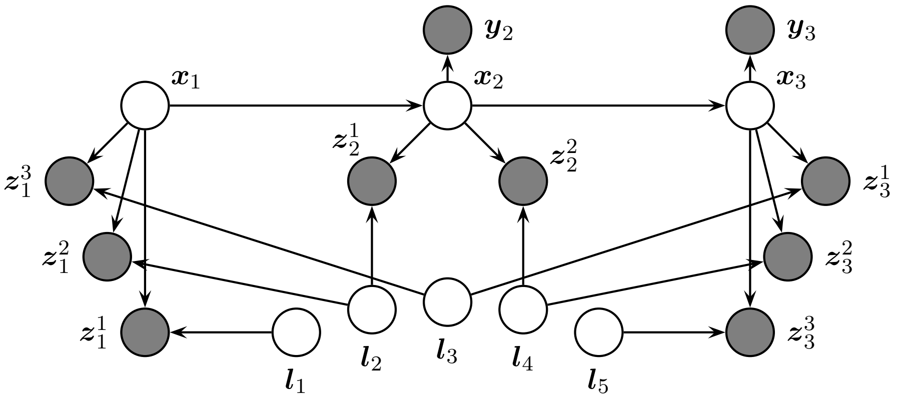


Figure 1: A DBN representing three time steps of an example SLAM problem. The edges from landmarks to measurements are determined by data association; e.g., $\delta_2^1 = 2$ in this example, and so there is an edge from $\boldsymbol{l}_2$ to $\boldsymbol{z}_2^1$.

where $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, Q)$ is an independent white noise variable. (The parameters of this model, A, b, and Q, may depend implicitly upon a control u_t applied at time t.) Then it obtains an odometry measurement y_{t+1}, which is a noisy affine function of its new state:

$$\boldsymbol{y}_{t+1} \quad = \quad C\boldsymbol{x}_{t+1} + d + \boldsymbol{w} \tag{4}$$

where $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, R)$ is an independent white noise variable.

In the common case where the measurement and motion functions are not affine with independent Gaussian noise, it is standard to approximate them as such using either the *Extended Kalman Filter* (EKF) or the *Unscented Kalman Filter* (UKF) [Wan and van der Merwe, 2000]. The UKF generally gives more accurate results because it uses second-order terms from the function's Taylor expansion, whereas the EKF is a first-order method. However, the UKF requires knowledge of both the mean and covariance of the function inputs, whereas the EKF requires only the mean. Because this linearization requires inference to determine around what input the nonlinear function should be approximated, fast inference is essential to SLAM filtering. (See Appendix B.4 for details on the EKF and UKF.)

Perhaps the best way to visualize the dependencies between the variables of a SLAM problem is with a dynamic Bayesian network (DBN).[3] Figure 1 depicts the DBN for three time steps of an example SLAM inference problem. For example, each measurement $\boldsymbol{z}_t^i$ depends only upon the state of the robot $\boldsymbol{x}_t$ and the state of the landmark that was observed $\boldsymbol{l}_{\delta_t^i}$ (assuming a known data association δ_t). Hence, in Figure 1, each observed measurement node takes only these nodes as parents. Also, the state of the robot at time $t+1$ depends only upon its previous state $\boldsymbol{x}_t$. Finally, the odometry measurement $\boldsymbol{y}_t$ at time t depends only upon the current state of the robot $\boldsymbol{x}_t$.

1.2 The complexity problem and previous approaches

Using the Kalman filter for SLAM quickly runs into problems. First, its representation of the belief state (or more specifically, the covariance matrix Σ) grows quadratically with the number of landmarks. Second, the time complexity of incorporating measurements via equation (71) also grows quadratically in the number of landmarks, regardless of how few variables are being measured.[4]

[3] See [Cowell *et al.*, 1999] for a rigorous introduction to graphical models, and [Russell and Norvig, 2002] for a gentle introduction to directed graphical models, including DBNs.

[4] Actually, this cost may be avoided when incorporating first-time landmark observations, which can be handled in linear-time as described in Appendix B.2.

The computational complexity of the Kalman filter make it inapplicable to large scale mapping problems, and give rise to the need for principled, efficient approximations. Significant interest in this problem has led to a number of approaches. We will examine three: Rao-Blackwellised particle-filters, sub-map approaches, and Sparse Extended Information Filters.

1.2.1 Rao-Blackwellised particle filters

One family of techniques involves using approximate inference. From the model description above, it can be shown that conditioned on the path of the robot, the states of all the landmarks are independent of each other. (For example, in Figure 1 $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ d-separate all the $\boldsymbol{l}_i$ from each other.) This motivates a Rao-Blackwellized particle filtering scheme: the estimate of the robot's path is represented by a set of particles (or samples); within each particle is a bank of tiny Kalman filters, one to represent the state estimate of each landmark, conditioned on the robot's path [Doucet *et al.*, 2000]. This representation, combined with a clever particle representation that avoids unnecessary copying of the landmark estimates during resampling, comprises the FastSLAM algorithm, which has an impressively fast $O(k \log n)$ time filter operation (where k is the number of particles) [Montemerlo *et al.*, 2002].

The primary advantages of the FastSLAM approach are threefold: it has excellent time complexity; it does not need to linearize the robot's motion or odometry models[5]; and, most importantly, it can represent multimodal distributions over the robot's state, which makes it particularly useful in tasks where global localization is also important [Thrun *et al.*, 2001].

However, there are also drawbacks. The first is that each particle has a different view of the map, including the estimates and even the number of landmarks; integrating these views to obtain a single map is nontrivial, and more importantly, data association must be performed for each particle independently. While this gives the representation a great deal of flexibility (in that the model need not commit to a particular maximum likelihood data association), it also introduces a significant computational burden (because data association can be expensive). Methods of integrating data association into FastSLAM are presented in [Montemerlo and Thrun, 2002].

The second problem is that FastSLAM is prone to diverge in regions where its measurements are not very informative, either due to high noise or a sparseness of landmarks. In particle filters like FastSLAM, good performance hinges on having a particle population that is large enough to "fill" the high-probability portion of the state space; for if there are too few particles, the filter is essentially ignoring likely hypotheses of the hidden state. In FastSLAM, the problem is aggravated by the curse of dimensionality: the state space of the particles is the set of all possible *paths* of the robot—not states—and so its dimension increases linearly with time. Thus, we have a situation where a constant number of particles is used to fill the high probability region of a state space whose volume is growing exponentially over time. The only regime in which we can hope to avoid the curse of dimensionality is when the volume of the high-probability region remains small, in spite of the increasing dimension of its embedding space. This volume increases when the robot moves due to motion noise, but it decreases when informative landmark and odometry measurements are made; thus, FastSLAM requires frequent, informative measurements to avoid divergence.[6] The successful application of FastSLAM to real-world SLAM problems indicates that this is possible [Montemerlo and Thrun, 2002].

1.2.2 Submap approaches

Rather than using approximate inference on the exact model, we can also use exact inference on tractable approximations of the true model. Another family of techniques involves a decomposition of the map

[5] The landmark measurement model must still be linearized because the landmark states are estimated using Kalman filters.

[6] One example of where this can become problematic is when the robot only observes new landmarks—which do not help the robot localize itself—for an extended period of time. It is possible to select control policies that prevent this from happening.

into a set of submaps, one or more of which maintain (independent) estimates of the robot's state. When the submaps are small enough, mapping each independently is tractable; the difficulty is "stitching" the submaps together.

There are now numerous approaches based upon this idea. The *Decoupled Stochastic Mapping* technique of [Leonard and Feder, 2000] uses a global coordinate system and splits the world into overlapping submaps. The *Atlas* framework of [Bosse *et al.*, 2002] uses submaps that have independent local coordinate systems and are related by a graph of coordinate transformations. The *Suboptimal* SLAM *filter* of [Guivant and Nebot, 2000] does not have an explicit notion of submap, but updates a small number of local landmarks at each time step. Each of these techniques obtains a constant-time update (because the size of the mapping problem addressed at each time step is independent of the total number of landmarks). However, techniques for passing information between the maps are limited, and thus these methods can converge very slowly.

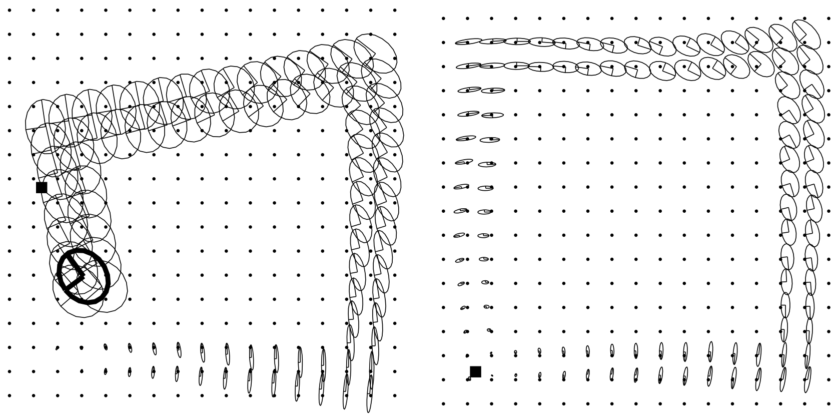
The *Compressed Filter* [Guivant and Nebot, 2000] and *Linear Sequence Mapping* [Tardós *et al.*, 2001] use the submap approach in a somewhat different way: the robot maps a local region until its map reaches a predefined size, and then the map is incorporated into a (large) global map in a single Kalman update. This does not avoid the quadratic complexity of the Kalman filter, but it can significantly reduce the constant factor. The map update is correct so long as the global and local maps are conditionally independent given the current state of the robot, which happens whenever they do not represent a landmark in common.

1.2.3 Sparse extended information filters

Sparse Extended Information Filters (SEIF) [Thrun *et al.*, 2002] go one step further, and employ approximate inference over an approximate model. As its name suggests, SEIF is based upon the information filter and uses the EKF to linearize nonlinear motion and measurement models. The fundamental observation of SEIF is that even if the covariance matrix Σ is dense, the precision matrix Λ may be sparse or many of its entries may be small; moreover, when Λ is sparse, measurement and motion updates can be made to happen in constant time. Thus, at each time step SEIF selectively zeros-out small entries of Λ, and thereby obtains a constant-time approximate filter update. (In Section 2.4 we develop a similar filter that takes makes no sparseness assumptions and thereby obtains a quadratic-time update.)

It is worth noting that it is nontrivial to zero-out arbitrary entries of a precision matrix, because the resulting matrix must still be positive semidefinite. The maximum likelihood approximation to Λ with an arbitrary sparsity pattern can be computed by the Iterative Proportional Fitting algorithm [Speed and Kiiveri, 1986]; however, as this algorithm is iterative (and slow), it is unsuitable for a projection operation that must be applied at every time step. SEIF avoids this cost by allowing new nonzero entries (which do not significantly affect the time complexity) to be introduced when others are removed.

In spite of its unbeatable time complexity, SEIF has a significant representational problem. Recovering the map μ or the covariance matrix Σ from the canonical parameterization used by SEIF requires inverting Λ, which takes time cubic in the number of landmarks. This is troublesome not only because the map is inaccessible; subvectors of μ are required to use the EKF to linearize nonlinear measurement and motion models, and subblocks of Σ are required for the UKF or for data association. The solution adopted by SEIF is to efficiently approximate the map μ by taking a constant number of hill-climbing steps in the probability density, and to approximate the marginal landmark covariances needed for data association using *conditional covariances* that can be computed in constant time. There is empirical evidence that these approximations can perform well; simulation studies have shown that the approximate map recovery is not a large source of error [Thrun *et al.*, 2002], and data association using conditional covariances seems to work well, in spite of the fact that it should be overconfident [Liu and Thrun, 2002].



(a) Before closing the loop.

(b) After closing the loop.

Figure 2: In this simulation, a full Kalman filter was used; the robot travels counter-clockwise on a noisy square path through a world randomly populated with point landmarks. The 95% confidence regions are shown (bold for the robot). The true position of the robot is shown by a square. (a) Before closing the loop, accumulated error has led to uncertain and drifted estimates for the positions of the robot and the recently-observed landmarks. (b) After closing the loop, the estimates are all improved and the confidence is greatly increased.

1.3 On constant-time algorithms and "closing the loop"

Some of the approximations described above achieve a constant-time filter update, which seems desirable. However, there are situations in SLAM problems where it seems that some measurements *should* cause a linear-time update.

Consider the case where the robot *closes the loop,* i.e., it reobserves a landmark after a long period of mapping unknown territory. In this scenario, reobserving landmarks whose positions are known with relative certainty helps the robot localize; its improved position estimate improves the estimates of recently observed landmarks via the robot-landmark correlations; finally, inter-landmark correlations improve the estimates of the other landmarks on the tour. (Robot-landmark correlations decay over time due to motion noise, so they are too weak to improve estimates of landmarks observed in the distant past.) Figure 2 demonstrates this phenomenon.

Thus, if the robot's path consists of one long loop, the robot's reobservation of its starting location can potentially improve *all* of its landmark position estimates. To do this, however, the measurement update would require time linear in the number of landmarks.[7] At best, a constant-time measurement update could improve the location estimates of constantly many landmarks, and achieve convergence by repeatedly retracing its steps; but this seems wasteful. Perhaps the best complexity one would want in a SLAM filter is one that can choose to perform constant time updates when mapping new territory (and the majority of the

[7] It may seem that FastSLAM can close the loop in constant time: when the robot integrates the landmark re-observation, particles whose paths are far from truth will have their weights reduced and will be resampled out, leaving only the particles whose paths are consistent with the reobservation; this has the effect of updating all the landmark estimates, which are marginals over the particle population. This presupposes, however, that a particle has survived that is "close enough" to the true path, which in general would require a number of particles that is exponential in the path length, or, put another way, exponential in the number of landmarks (assuming a constant number of new landmarks were observed on each step of the path).

map remains unchanged), and then switch to a linear-time update when the robot closes the loop.

2 A graphical model perspective

The approach presented here employs exact inference over an approximate model, and can be viewed as a generalization of both the submap approach and SEIF (as we discuss in Section 7). Importantly, it improves significantly over both techniques, culminating in an efficient, flexible, and principled approximation algorithm. The easiest way to motivate the present approach is in the graphical model framework. We begin by presenting our notation and vocabulary for graphical models.

2.1 Gaussian graphical models

A *potential function* is any function whose range is $\mathbb{R}^+$, the nonnegative real numbers. A *graphical model* $G = (V, \Psi)$ consists of a set of random variables V and a set of potential functions Ψ over subsets of V. The probability density represented by G is

$$p_G(V) = \frac{1}{Z} \prod_{\psi \in \Psi} \psi(V_\psi) \tag{5}$$

where $V_\psi \subseteq V$ is the domain of ψ and

$$Z = \int_V \prod_{\psi \in \Psi} \psi(V_\psi) \tag{6}$$

is a normalization constant to ensure p_G integrates to unity over V.

The *(Markov) graph* associated with G has vertex set V and an edge between two variables $\boldsymbol{u}$ and $\boldsymbol{v}$ iff $\boldsymbol{u}, \boldsymbol{v} \in V_\psi$ for some potential $\psi \in \Psi$. Given $p_G > 0$, the *Hammersley-Clifford Theorem* says (roughly) that the graph separation relation in the Markov graph of G is exactly the conditional independence relation over V, i.e., if A, B, and S are subsets of V such that S separates A and B in the Markov graph of G, then $A \perp\!\!\!\perp B \mid S$, i.e, A is conditionally independent of B given S.

Graphical models are useful to identify conditional independence structure in a multivariate Gaussian distribution. In this case, we typically use the canonical parameterization of the Gaussian (see Appendix A.1). The Gaussian (43) can be represented by a graphical model, since if we partition $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{i=1}^{n} \psi_i(\boldsymbol{x}_i) \prod_{j=i+1}^{n} \psi_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{7}$$

where

$$\begin{align} \psi_i(\boldsymbol{x}_i) &= \exp\left\{ \eta_i \boldsymbol{x}_i^T - \frac{1}{2} \boldsymbol{x}_i^T \Lambda_{ii} \boldsymbol{x}_i \right\} \tag{8} \\ \psi_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \exp\left\{ -\boldsymbol{x}_i^T \Lambda_{ij} \boldsymbol{x}_j \right\} \tag{9} \end{align}$$

and $Z = e^{-a}$ is the normalization constant. Thus, all the potential functions in a Gaussian graphical model are either unary (*node potentials*) or binary (*edge potentials*). We also have the important interpretation that if $\Lambda_{ij} = 0$, then $\psi_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is unity (and therefore superfluous), meaning there is no edge between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Thus, Λ can be viewed as an adjacency matrix for the graphical model.

When the belief state is represented as a graphical model, filtering can be viewed as a three-step procedure [Russell and Norvig, 2002]:

1. *prediction*, in which we augment the model with the state variables of the next time step;
2. *roll-up*, in which we marginalize out the state variables from the past time step; and
3. *estimation*, in which we incorporate the current time step's measurements.

In the next two sections, we examine the structural and parametric changes caused by these updates.

2.2 Structural updates during filtering

Representing the belief state using graphical models has some advantages over the traditional multivariate Gaussian representation. The first advantage is that the quadratic growth of the size of the belief state is easily understood by viewing filtering (which is nothing more than marginalizing out variables from past time steps) as an instance of the variable elimination algorithm [Russell and Norvig, 2002].

Let us begin with the first time slice of the example SLAM DBN in Figure 1. Figure 3 depicts the landmark measurement portion of the first time step. Eliminating each measurement node adds an elimination edge between the robot's state and those of the observed landmarks. (Intuitively, each measurement imposes an uncertain constraint between the state of the robot and the observed landmark, which is reflected by the undirected edge. We can imagine the robot thinking, "I'm not sure where I am or where that landmark is, but I'm pretty certain it is five meters in front of me; so if I learn where I am, I'll have a good idea of where it is, and vice-versa.") Using the terminology of [Thrun *et al.*, 2002], we will call the set of landmarks that are directly dependent upon $\boldsymbol{x}_t$ (i.e., no other set of variables can render them conditionally independent of $\boldsymbol{x}_t$) the *active landmarks*; thus, a landmark becomes active as soon as it is measured.

Thus far, our model is still tractable, since the graph is in the form of a tree. However, when we perform roll-up (depicted in Figure 4), the situation changes. Eliminating the robot's previous state variable creates an elimination clique over the robot's new state variable and *all* of the active landmarks. The intuition here is more subtle, but worth understanding: in the previous model, weak constraints existed between all pairs of active landmarks via their individual couplings to the previous state of the robot; when the robot's previous state variable is eliminated from the model, these indirect couplings must be represented by explicit edges. However, as Figure 5 demonstrates, these indirect couplings are weaker than the direct ones and can include a good deal of redundancy. These two facts suggest that eliminating some of these dependencies may not significantly affect the quality of the map estimate, an intuition that motivates both SEIF and the current proposal.

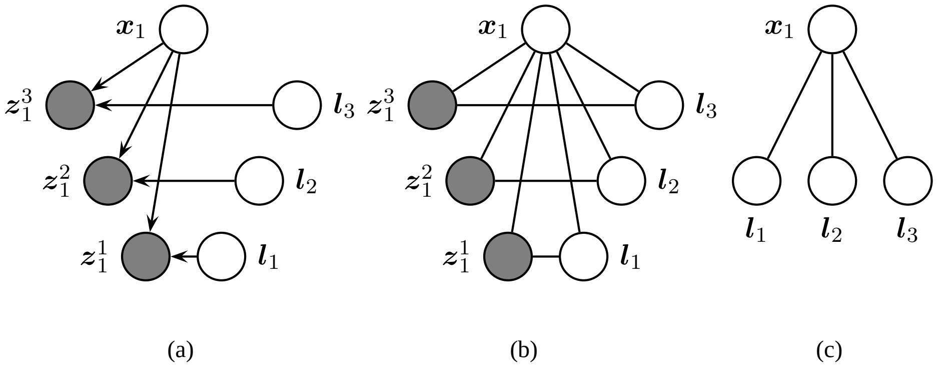
Once a landmark becomes active (by observation), it will never again become inactive. Thus, if k landmarks have been observed up to time t, then the filtered distribution after the time-t roll-up will have a cluster containing $k+1$ variables and it will take $O(k^2)$ space to represent the parameters of this cluster.

2.3 Parameter updates during filtering

While the graphical model we have given in Figure 1 is directed, we will focus more on undirected graphical models: they are the natural method of representing conditional independence statements about multivariate Gaussian distributions, and, the graphical models we are left with after estimation updates and roll-up are undirected (see Figures 3(c) and 4(d)).

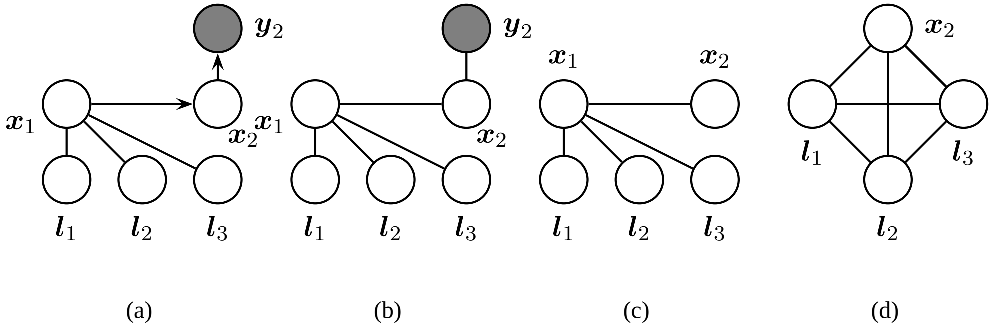
For conceptual clarity, we partition the map variable as follows. Let $\boldsymbol{m}_{(0)} = x$ and $\boldsymbol{m}_{(i)} = \boldsymbol{l}_i$, let $\eta_{(i)}$ be the subvector of η corresponding to $\boldsymbol{m}_{(i)}$, and let $\Lambda_{(ij)}$ be the submatrix of Λ corresponding to $\boldsymbol{m}_{(i)}$ and $\boldsymbol{m}_{(j)}$. Then we can write

$$p(\boldsymbol{m}) = \frac{1}{Z} \prod_{i=0}^{n} \psi_i(\boldsymbol{m}_{(i)}) \prod_{j=i+1}^{n} \psi_{ij}(\boldsymbol{m}_{(i)}, \boldsymbol{m}_{(j)}) \tag{10}$$



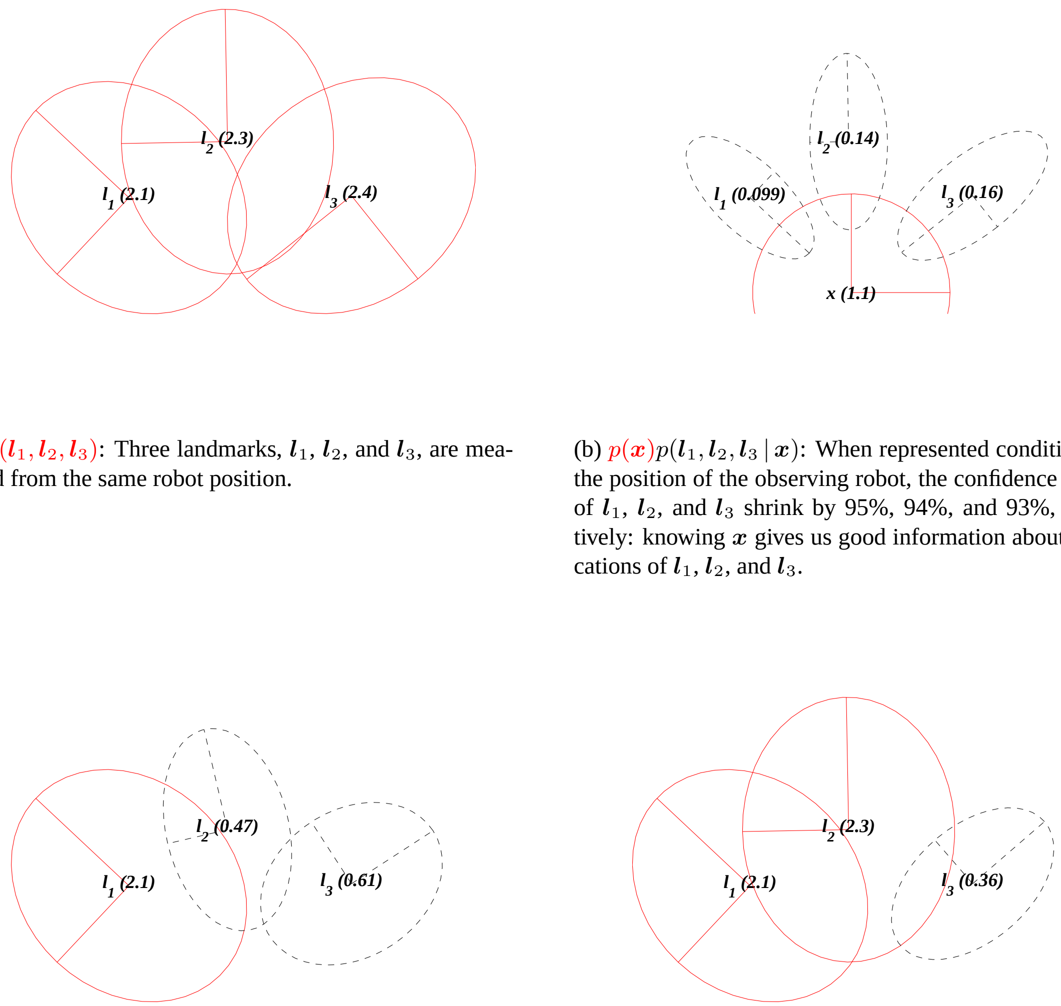
(a) (b) (c)

Figure 3: Incorporating landmark measurements. (a) At time one, three landmark measurements, $\boldsymbol{z}_1^1$, $\boldsymbol{z}_2^1$, and $\boldsymbol{z}_3^1$, are made of landmark states $\boldsymbol{l}_1$, $\boldsymbol{l}_2$, and $\boldsymbol{l}_3$. (b) Moralizing the directed graph produces the undirected graphical model in which the landmarks are coupled with the robot state. (c) Eliminating the observed measurement variables leaves a graph in which the robot state is directly dependent upon the states of all the measured landmarks.



(a) (b) (c) (d)

Figure 4: Incorporating robot motion. (a) The robot's state at the next time step and its odometry measurement $\boldsymbol{y}_2$ are added to the model. (b) Moralizing this graph adds no new edges. (c) Eliminating the odometry variables adds no new edges. (d) Eliminating the state at the previous time step adds an elimination clique that includes the current state $\boldsymbol{x}_2$ and all the active landmarks.



(a) $p(\boldsymbol{l}_1, \boldsymbol{l}_2, \boldsymbol{l}_3)$: Three landmarks, $\boldsymbol{l}_1$, $\boldsymbol{l}_2$, and $\boldsymbol{l}_3$, are measured from the same robot position.

(b) $p(\boldsymbol{x})p(\boldsymbol{l}_1, \boldsymbol{l}_2, \boldsymbol{l}_3 \mid \boldsymbol{x})$: When represented conditional on the position of the observing robot, the confidence regions of $\boldsymbol{l}_1$, $\boldsymbol{l}_2$, and $\boldsymbol{l}_3$ shrink by 95%, 94%, and 93%, respectively: knowing $\boldsymbol{x}$ gives us good information about the locations of $\boldsymbol{l}_1$, $\boldsymbol{l}_2$, and $\boldsymbol{l}_3$.

(c) $p(\boldsymbol{l}_1)p(\boldsymbol{l}_2, \boldsymbol{l}_3 \mid \boldsymbol{l}_1)$: When represented conditional on the state of $\boldsymbol{l}_1$, the confidence regions of $\boldsymbol{l}_2$ and $\boldsymbol{l}_3$ shrink by 80% and 75% (respectively) from their counterparts in (a): knowing $\boldsymbol{l}_1$ helps us localize $\boldsymbol{l}_2$ and $\boldsymbol{l}_3$. However, these reductions are not as significant as the reduction obtained by conditioning on $\boldsymbol{x}$, as in (b). Thus, the indirect landmark-landmark dependencies are weaker than the direct robot-landmark dependencies.

(d) $p(\boldsymbol{l}_1, \boldsymbol{l}_2)p(\boldsymbol{l}_3 \mid \boldsymbol{l}_1, \boldsymbol{l}_2)$: When represented conditional on the states of $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$, the confidence region of $\boldsymbol{l}_3$ shrinks by 85% (or only an additional 10% over the reduction obtained by conditioning on $\boldsymbol{l}_1$ alone), demonstrating redundancy in the indirect dependencies: the information $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ jointly give about $\boldsymbol{l}_3$ is not twice as much as the information given by $\boldsymbol{l}_1$ alone.

Figure 5: Illustration of direct versus indirect dependencies and redundancy. Each image visualizes a joint distribution. Variable distributions are shown by 95% confidence regions; marginal distributions are shown in red (solid) and conditional distributions are shown in black (dashed). The area of each confidence region (a measure of uncertainty) is shown in parentheses.

where

$$\psi_i(\boldsymbol{m}_{(i)}) = \exp\left\{\eta_{(i)}^T \boldsymbol{m}_{(i)} - \frac{1}{2}\boldsymbol{m}_{(i)}^T \Lambda_{(ii)} \boldsymbol{m}_{(i)}\right\} \tag{11}$$

and

$$\psi_{ij}(\boldsymbol{m}_{(i)}, \boldsymbol{m}_{(j)}) = \exp\left\{-\boldsymbol{m}_{(i)}^T \Lambda_{(ij)} \boldsymbol{m}_{(j)}\right\} \tag{12}$$

and $Z = e^{-a}$ is the normalization constant. We will alternatingly view the parameters of a Gaussian graphical model as a set of node and edge potentials (with functional forms (11) and (12)) or as the vector η and matrix Λ that underlie them. Thus, when we speak of "multiplying in" new unary and binary potentials, this corresponds to straightforward additive updates to η and Λ.

There are three types of changes made to the graphical model during filtering: estimation, prediction, and roll-up. Each of these updates can be implemented by multiplying small, simple potentials into the current graphical model. We begin with the following proposition, which summarizes some algebra:

Proposition 1. *Let $\boldsymbol{q}$ be a random n-vector distributed according to $p(\boldsymbol{q}, \boldsymbol{r})$ (for any arbitrary random variable $\boldsymbol{r}$) and let the random variable $\boldsymbol{s}$ be defined via the affine-Gaussian equation*

$$\boldsymbol{s} = s_0 + L\boldsymbol{q} + \boldsymbol{t} \tag{13}$$

where s_0 and L are constants and $\boldsymbol{t} \sim \mathcal{N}(0, U)$. Then

$$p(\boldsymbol{q}, \boldsymbol{r}, \boldsymbol{s}) \propto p(\boldsymbol{q}, \boldsymbol{r}) \cdot \phi_q(\boldsymbol{q}) \cdot \phi_s(\boldsymbol{s}) \cdot \phi_{qs}(\boldsymbol{q}, \boldsymbol{s}) \tag{14}$$

where

$$\begin{aligned} \phi_q(\boldsymbol{q}) &= \exp\left\{-s_0^T U^{-1} L\boldsymbol{q} - \frac{1}{2}\boldsymbol{q}^T L^T U^{-1} L\boldsymbol{q}\right\} && (15) \\ \phi_s(\boldsymbol{s}) &= \exp\left\{s_0^T U^{-1} \boldsymbol{s} - \frac{1}{2}\boldsymbol{s}^T U^{-1} \boldsymbol{s}\right\} && (16) \\ \phi_{qs}(\boldsymbol{q}, \boldsymbol{s}) &= \exp\left\{\boldsymbol{s}^T U^{-1} L\boldsymbol{q}\right\} && (17) \end{aligned}$$

Furthermore,

$$p(\boldsymbol{q}, \boldsymbol{r} \mid s) \propto p(\boldsymbol{q}, \boldsymbol{r}) \cdot \phi'_q(\boldsymbol{q}) \tag{18}$$

where

$$\phi'_q(\boldsymbol{q}) = \exp\left\{(s - s_0)^T U^{-1} L\boldsymbol{q} - \frac{1}{2}\boldsymbol{q}^T L^T U^{-1} L\boldsymbol{q}\right\} \tag{19}$$

(The proof of this and all other propositions is given in Appendix C.) The import of this proposition is that we can update the graphical model to reflect landmark measurements, odometry measurements, and robot motion by multiplying in small, simple potentials. Equivalently, since these potentials are all of the forms (11) or (12), these updates can be accomplished by additive updates to small portions of η and Λ. We describe this process in more detail below.

2.3.1 Estimation updates

Since all landmark measurements are assumed independent, we will deal with the case of a single measurement z originating from the landmark $\boldsymbol{l}$. By applying Proposition 1 (with $\boldsymbol{s} = \boldsymbol{z}$, $\boldsymbol{q} = (\boldsymbol{x}, \boldsymbol{l})$, $s_0 = g$, and $L = [\begin{array}{cc} E & F \end{array}]$), we find that we can condition on z by multiplying the binary potential

$$\phi'(\boldsymbol{x}, \boldsymbol{l}) = \exp\left\{(z-g)^T S^{-1}(E\boldsymbol{x} + F\boldsymbol{l}) - \frac{1}{2}(E\boldsymbol{x} + F\boldsymbol{l})^T S^{-1}(E\boldsymbol{x} + F\boldsymbol{l})\right\}$$

into the graphical model. Equivalently, we can update the parameters η and Λ as follows:

$$\begin{array}{rcl} \eta_x & \overset{+}{\leftarrow} & E^T S^{-1}(z-g) \\ \eta_l & \overset{+}{\leftarrow} & F^T S^{-1}(z-g) \\ \Lambda_{xx} & \overset{+}{\leftarrow} & E^T S^{-1} E \\ \Lambda_{ll} & \overset{+}{\leftarrow} & F^T S^{-1} F \\ \Lambda_{xl} & \overset{+}{\leftarrow} & E^T S^{-1} F \end{array} \tag{20}$$

where $a \overset{+}{\leftarrow} b$ means "increment a by the value b." We can now see algebraically how the new edges in Figure 3(c) arise, since Λ_{xl} becomes non-zero after this update.

A second application of the proposition gives that the odometry model $p(y_{t+1} \,|\, \boldsymbol{x}_{t+1})$ can be multiplied in via the following updates:

$$\begin{array}{rcl} \eta_{x_{t+1}} & \overset{+}{\leftarrow} & C^T R^{-1}(y-d) \\ \Lambda_{x_{t+1}x_{t+1}} & \overset{+}{\leftarrow} & C^T R^{-1} C \end{array} \tag{21}$$

2.3.2 Prediction update

The prediction update consists of adding a new node $\boldsymbol{x}_{t+1}$ and connecting it to $\boldsymbol{x}_t$ via the potential function $p(\boldsymbol{x}_{t+1} \,|\, \boldsymbol{x}_t)$. Using Proposition 1, we find that the motion model $p(\boldsymbol{x}_{t+1} \,|\, \boldsymbol{x}_t)$ can be multiplied into the graphical model via the following updates:

$$\begin{array}{rcl} \eta_{x_t} & \overset{+}{\leftarrow} & -b^T Q^{-1} A \\ \eta_{x_{t+1}} & \leftarrow & bQ^{-1} \\ \Lambda_{x_t x_t} & \overset{+}{\leftarrow} & A^T Q^{-1} A \\ \Lambda_{x_{t+1}x_{t+1}} & \leftarrow & Q^{-1} \\ \Lambda_{x_t x_{t+1}} & \leftarrow & A^T Q^{-1} \end{array} \tag{22}$$

where $a \leftarrow b$ means "assign a the value b."

2.3.3 Roll-up

To "roll up" the DBN, we must marginalize out the previous state variable $\boldsymbol{x}_t$. This too can be implemented by multiplying potentials onto the graphical model.

The rule for marginalizing out a variable $\boldsymbol{x}$ from a multivariate Gaussian distribution is given in Appendix A.2. However, by leveraging the structure of the graphical model, we can speed up the computation. Because zero entries of Λ correspond to missing edges in the graphical model, we can see that the only entries of η and Λ that change in (50) are those that correspond to variables in the *Markov blanket* of $\boldsymbol{x}_t$, i.e., the set of neighbors of $\boldsymbol{x}_t$. If we denote the Markov blanket of $\boldsymbol{x}$ by $\bar{\boldsymbol{x}}$, then the update is as follows:

$$\begin{array}{rcl} \eta_{\bar{x}} & \overset{+}{\leftarrow} & -\Lambda_{\bar{x}x}\Lambda_{xx}^{-1}\eta_x \\ \Lambda_{\bar{x}\bar{x}} & \overset{+}{\leftarrow} & -\Lambda_{\bar{x}x}\Lambda_{xx}^{-1}\Lambda_{x\bar{x}} \end{array} \tag{23}$$

After this update is completed, we can discard all potentials involving $\boldsymbol{x}$ and eliminate the node from our graphical model. Note that although we have written this update in terms of the parameters η and Λ, we could equally well express it in terms of multiplying in node and edge potentials over the Markov blanket of $\boldsymbol{x}_t$.

2.4 An information filter for SLAM

Our results thus far define a filter whose belief state is a graphical model. However, as we noted in Section 2.2, the graphical model is degenerate; it is complete after every roll-up step, which means that it is characterized by a single potential function. Because that potential function is in the canonical parameterization, we have effectively defined an *information filter* for the SLAM problem.

The belief state of the filter will be a single Gaussian $\mathcal{N}^{-1}(\eta, \Lambda)$. Our filter steps are:

1. *prediction*, where we add the state variables from the next time step. This can be accomplished in constant time using the update (22).

2. *estimation*, where we condition on the current time-slice's observations. Assuming constantly many landmarks are observed, this too can be accomplished in constant time. We apply (20) for each landmark measurement and (21) for the odometry update.

3. *roll-up*, where we marginalize out the state variables of the previous time slice. This can be accomplished using (23), which takes time quadratic in the number of landmarks.

The space complexity of this filter operation is linear in the length of the filtering problem (assuming a constant number of landmarks are observed per time step), because each observation can create one nonzero entry in Λ.

If the motion and measurement models are known in advance, then this completes the description of the algorithm. However, when we must linearize our motion and measurement models, we require subvectors of μ (for the EKF) and possibly subblocks of Σ (for the UKF) at each time step. We can compute these quantities exactly in cubic time by inverting Λ, or can approximate them in constant time using SEIF's techniques of approximate map recovery and using conditional landmark covariances rather than marginal ones [Thrun *et al.*, 2002].

3 Junction tree filtering

Exact inference in graphical models is frequently based upon message passing in a junction tree, which is a special data structure based upon the graphical model [Cowell *et al.*, 1999]. We adopt consistent junction trees as our representation of the belief state. This gives us efficient access the filtered marginals of any sets of variables contained in the same cluster; we just marginalize out the unwanted variables from the (consistent) cluster potential.[8]

In our context, the graphical model changes over time, and so we must consider methods of maintaining the junction tree incrementally over time. From the discussion above, we see that we require a junction tree

[8] This is sufficient for all inference problems required by SLAM, with one exception. In using the UKF to linearize the landmark measurement model, we require the joint distribution over $\boldsymbol{x}$ and $\boldsymbol{l}$, which may not reside in the same cluster. There are two solutions. The exact solution entails using the technique described in Section 3.2 to ensure $\boldsymbol{x}$ and $\boldsymbol{l}$ are in the same cluster; this incurs no added expense if the measurement is known to emanate from $\boldsymbol{l}$, since we will have to ensure this property anyway in order to multiply in the measurement potential $\psi(\boldsymbol{x}, \boldsymbol{l})$. However, if we are merely evaluating a possible data association, then a more efficient (maximum likelihood) approximate solution is to use the product of their marginal distributions. In practice, we find that the approximate solution introduces very little linearization error.

representation that allows us to (1) add new node and edge potentials (for observations and motion) and to (2) marginalize out nodes (specifically, the robot state at the previous time step). In this section, we describe how all of these updates can be applied to the junction tree incrementally. We begin with a brief review of junction trees to introduce notation.

3.1 Junction trees

A *junction tree* $\mathcal{T}$ for a graphical model $G = (V, \Psi)$ is an undirected graph such that each vertex (or *cluster*) C is a subset of the variables V and the following three properties hold:

1. *Singly connected property:* $\mathcal{T}$ is a tree.
2. *Potential property:* For every potential $\psi \in \Psi$ there is some cluster C such that $C \supseteq V_\psi$.
3. *Running intersection property:* If a variable is present in two clusters C_i and C_j of $\mathcal{T}$, it is also present in all clusters on the (unique) path between C_i and C_j in $\mathcal{T}$.

The set of $\mathcal{T}$'s clusters is denoted $\mathcal{C}$. A cluster C_i is *nonmaximal* if there is another cluster C_j such that $C_i \subseteq C_j$; typically we require $\mathcal{T}$ has no nonmaximal clusters. $\mathcal{T}$ also has a set of *separators* $\mathcal{S}$, one per edge $\{C_i, C_j\}$ such that $S_{ij} = C_i \cap C_j$.

The parameterization of $\mathcal{T}$ is a set of potential functions $\Phi = \{\phi_C : C \in \mathcal{C}\} \cup \{\phi_S : S \in \mathcal{S}\}$, one for each cluster and separator. The *charge on* $\mathcal{T}$ is defined to be

$$\phi_{\mathcal{T}}(V) = \frac{\prod_{C \in \mathcal{C}} \phi_C(C)}{\prod_{S \in \mathcal{S}} \phi_S(S)} \tag{24}$$

Φ is initialized by setting all cluster and separator potentials to unity, multiplying each potential $\psi \in \Psi$ into ϕ_C for some $C \supseteq V_\psi$ (which is possible by the potential property), and multiplying Z^{-1} into an arbitrary ϕ_C; then $\phi_{\mathcal{T}} = p_G$.

Inference in the graphical model G can be carried out by a message passing algorithm over $\mathcal{T}$. Passing a message from C_i to C_j consists of the following updates:

$$\phi^*_{S_{ij}} = \int_{C_i - S_{ij}} \phi_{C_i} \tag{25}$$

$$\phi^*_{C_j} = \phi_{C_j} \frac{\phi^*_{S_{ij}}}{\phi_{S_{ij}}} \tag{26}$$

Even though passing a message $C_i \rightarrow C_j$ changes the parameterization Φ, the charge $\phi_{\mathcal{T}}$ remains invariant. We *distribute evidence* from a cluster C by passing messages along edges in any preorder traversal from C. When messages are passed along every edge in both directions (in an appropriate schedule), the cluster and separator potentials are updated so that they are marginals of p_G over their respective variables. A junction tree in this state is called *consistent* and it can be used to obtain marginals over any set of variables that *cohabit*, i.e., that reside together in some cluster.

When $\mathcal{T}$ has no nonmaximal clusters, $|\mathcal{C}| \leq |V|$ so the number of messages required for inference is bounded by $2 \cdot |V|$. In the case of a Gaussian graphical model, the cluster and separator potentials are Gaussians. If they are represented in the canonical parameterization, the time complexity of passing a message is dominated by the cost of the marginalization in (25) which is implemented via (50); thus, it is at worst cubic in the size of the cluster. In sum, inference is linear in $|V|$ and cubic in the *width* of $\mathcal{T}$, traditionally defined as the size of the largest cluster minus one.

We will make use of two nonstandard operations on junction trees. To *clone* a cluster C_i, we create a copy C_j, attach C_j to C_i, and set $\phi_{S_{ij}}$ and ϕ_{C_j} equal to ϕ_{C_i}. When we *merge* a cluster C_j into one of its neighbors C_i, we: update C_i to $C_i \cup C_j$, update ϕ_{C_i} to $\phi_{C_i}\phi_{C_j}/\phi_{S_{ij}}$, swing all edges incident to C_j over to C_i, and remove C_j. It is easy to check that both of these operations preserve the structural constraints as well as the charge and consistency of a junction tree.

3.2 Adding potentials

Adding a node potential to the graphical model (as in the odometry update) requires no updates to the structure of the junction tree; we need only multiply the potential into one of the clusters containing the variable in order to update the charge, and then distribute evidence from that cluster to restore consistency.

In contrast, it is more complicated to update the junction tree to reflect the addition of an edge potential between variables $\boldsymbol{u}$ and $\boldsymbol{v}$ in the graphical model. If $\boldsymbol{u}$ and $\boldsymbol{v}$ do not appear in the same cluster, then the potential property will be violated. In order to restore it, we can add $\boldsymbol{v}$ to any cluster C containing $\boldsymbol{u}$. However, this update may violate the running intersection property, as C may not be a neighbor of another cluster containing $\boldsymbol{v}$. To restore this property, we can add an edge from C to any other cluster C' containing $\boldsymbol{v}$. However, this will violate the singly connected property.

There are several techniques for updating the structure of a junction tree to reflect an edge addition [Draper, 1995]. Here, we adopt the following simple method: after adding $\boldsymbol{v}$ to C, we find the cluster C' that contains $\boldsymbol{v}$ and is closest to C (in path length), and add $\boldsymbol{v}$ to every cluster along the path between C and C'. (The cluster and separator potentials can be updated by passing messages on the path from C_i to C_j; any clusters that become nonmaximal can be merged into their subsuming neighbors.) The resulting junction tree does not violate the singly connected property or the running intersection property, and therefore is correct for the graph with the new edge. (This update can be shown to be equivalent to loop-cutset conditioning on $\boldsymbol{v}$ [Shachter *et al.*, 1994]; it is also equivalent to repeated use of Lauritzen's PUSH operator [Lauritzen and Jensen, 2001].) To update the charge on the junction tree, we simply multiply the edge potential into the potential of C; then, to restore consistency, we distribute evidence from C. This is generalized to the case of potentials over arbitrarily many variables by Algorithm 1.

Algorithm 1 Multiply ψ into cluster C of $\mathcal{T}$

```
Require: $\mathcal{T}$ is valid, consistent, and has no nonmaximal clusters
Ensure: $\mathcal{T}$ is valid, consistent, has no nonmaximal clusters, and $\phi_{\mathcal{T}} \leftarrow \phi_{\mathcal{T}} \times \psi$
for all variables $\boldsymbol{u}$ bound by $\psi$ do {ensure the potential property}
  if $\boldsymbol{u} \notin C$ then
    if $\boldsymbol{u} \in \mathcal{T}$ then {must ensure the running intersection property}
      $C' \leftarrow$ cluster closest to $C$ containing $\boldsymbol{u}$
      $(C' = C_1, C_2, \ldots, C_{k-1}, C_k = C) \leftarrow$ path from $C'$ to $C$
      for $i = 2$ to $k$ do {restores the running intersection property}
        $C_i \leftarrow C_i \cup \{\boldsymbol{u}\}$
        $S_{i,i-1} \leftarrow S_{i,i-1} \cup \{\boldsymbol{u}\}$
        Pass a message from $C_{i-1}$ to $C_i$. {restore consistency}
        if $C_i$ subsumes one of its neighbors $C_k$ then {Ensure no non-maximal clusters}
          Swing all edges incident to $C_k$ over to $C_i$.
          Remove $C_k$ from $\mathcal{T}$.
    else {new variable}
      $C \leftarrow C \cup \{\boldsymbol{u}\}$
Multiply $\psi$ into $\phi_C$. {update charge}
Distribute evidence from $C$. {restore consistency}
```

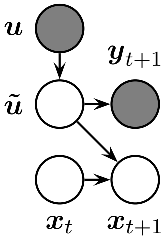


Figure 6: The graphical model representing a common type of dynamics/odometry model; $\boldsymbol{x}_t$ (and all active landmarks) are independent of $\boldsymbol{y}_{t+1}$ because $\boldsymbol{x}_{t+1}$ is unobserved.

3.3 Avoiding message passing

In three important cases that arise in SLAM, conditional independences render the evidence distribution step of Algorithm 1 superfluous; that is, it is not necessary to pass messages in order to restore consistency. This is a significant optimization, since the message passing is by far the most expensive operation.

For example, when adding $\boldsymbol{x}_{t+1}$ to the model and multiplying on the potential $p(\boldsymbol{x}_{t+1} \,|\, \boldsymbol{x}_t)$, we need not pass messages from the (unique) cluster containing $\boldsymbol{x}_{t+1}$ to any other clusters. This is because $\boldsymbol{x}_{t+1}$ is an unobserved (directed) leaf of the graphical model—a *barren node*—and therefore does not impact the marginal distributions of the other nodes [Geiger *et al.*, 1990].

Another case arises in a common type of odometry model $p(\boldsymbol{y} \,|\, \boldsymbol{x})$. Often we have the following type of situation: the state of the robot $\boldsymbol{x}$ consists of pose variables $\boldsymbol{p}$ (e.g., the coordinates of the robot and its heading) and some derivatives $\dot{\boldsymbol{p}}$ (e.g., the translational and rotational velocities); the dynamics model updates $\boldsymbol{p}$ by integrating in $\dot{\boldsymbol{p}}$ and then setting $\dot{\boldsymbol{p}}$ to $\tilde{\boldsymbol{u}}$, the result of corrupting a control u by noise; and, the odometry y is a noisy measurement of the noisy control $\tilde{\boldsymbol{u}}$. (For an example of this type of odometry model, see Section 6.1.) In this case, we can again appeal to conditional independencies (see Figure 6) to see that $\boldsymbol{x}_t$ and all landmark state variables are independent of y_{t+1}. Thus, in this setting, we can also avoid message passing in the odometry update.

The final case arises when observing a landmark for the first time. Intuitively, observing a landmark $\boldsymbol{l}$ whose absolute position is unknown cannot give you any information about your location. However, we cannot appeal to a graphical criterion to avoid message passing. Rather, the independence of $\boldsymbol{x}$ (and all other variables) from $\boldsymbol{l}$ is an artifact of the completely uninformative prior on $\boldsymbol{l}$ before the measurement z is made. In this case, "explaining away" leads to $\boldsymbol{l}$ assuming all responsibility for z, and therefore $\boldsymbol{x}$'s distribution remains unchanged. This is verified algebraically in the proof of

Proposition 2. *Observing a landmark for the first time does not change the map distribution.*

Therefore, to perform such an update, we simply multiply on the observation potential $p(z \,|\, \boldsymbol{x}, \boldsymbol{l})$ onto the appropriate cluster; no message passing is required.

In fact, the only type of update in the SLAM application that requires message passing (provided our odometry update is as described above) is when the robot *reobserves* a landmark. Later we will leverage this fact—by delaying our use of this additional information—to obtain a filter operation that is commonly constant time.

3.4 Marginalizing out nodes

As described in Section 2.3.3, marginalizing out a variable $\boldsymbol{x}$ creates a new potential over the Markov blanket of $\boldsymbol{x}$, $\text{MB}(\boldsymbol{x})$. Because the update potential factors into a product of unary and binary potentials, we could

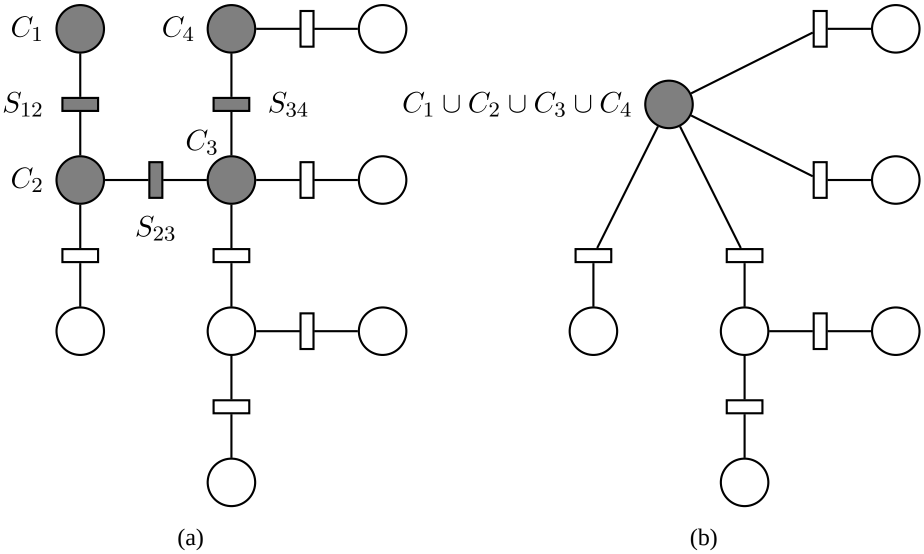


Figure 7: An example of cluster merging. (a) The shaded clusters and separators are those containing the variable $\boldsymbol{v}$. (b) In order to marginalize $\boldsymbol{v}$ out of the junction tree, we merge C_1, C_2, C_3 and C_4 and eliminate the separators between them.

add each of the edges of this cluster independently using the technique described above. However, there is a much simpler and more efficient method.

In order to preserve the potential property, the structure of the junction tree must be updated so that it has a cluster containing $\text{MB}(\boldsymbol{x})$. The Markov blanket of $\boldsymbol{x}$ may not be efficiently computable from the junction tree, but we are guaranteed that it is a subset of the union of all clusters containing $\boldsymbol{x}$ (since the junction tree has the potential property). Because the junction tree also has the running intersection property, all clusters containing $\boldsymbol{x}$ must constitute a subtree of the junction tree, which we denote $\mathcal{T}_{\boldsymbol{x}}$. We can therefore merge all of these clusters to obtain a new cluster that contains $\boldsymbol{x}$ and $\text{MB}(\boldsymbol{x})$; this new cluster will inherit all of the edges incident to all of the merged clusters. It is straightforward to verify that the resulting junction tree is valid for the original graphical model. An example of this transformation is given in Figure 7.

The charge on the junction tree can be preserved during this cluster merging by setting the potential on the new cluster to the product of the potentials of the merged clusters divided by the product of the potentials of the eliminated separators. Then, to marginalize $\boldsymbol{x}$ out of the junction tree, we simply marginalize $\boldsymbol{x}$ out of this new cluster, which is the only cluster that contains it. It is straightforward to verify that if the junction tree was previously consistent, then this operation preserves its consistency without message passing.

4 Thinning the junction tree

As discussed in Section 2.2, the size of the largest cluster of the junction tree will grow linearly in the number of landmarks. Thus, if we are to avoid cubic-time inference, we must resort to an approximation. In this section, we consider a method of thinning the junction tree so that the complexity of message passing remains manageable. The approach here is related to that of Kjærulff (1993); that work describes how removing an edge from a graphical model can sometimes result in a thinner junction tree. The approach presented here is simpler and more direct, as it operates on the junction tree without making reference to the underlying graphical model.

4.1 Variable contraction

The complexity of message passing is determined by the sizes of the clusters in the junction tree, and therefore our goal will be to define a thinning operation that reduces the size of clusters in the junction tree. We now describe such an operation, which we call *variable contraction*:

Definition 1. Let $\boldsymbol{x}$ be a variable that appears in more than one cluster of the junction tree $\mathcal{T}$, let C be a leaf of $\mathcal{T}_{\boldsymbol{x}}$ (the subtree of $\mathcal{T}$ induced by $\boldsymbol{x}$), and let S be the separator joining C to $\mathcal{T}_{\boldsymbol{x}}$. A *variable contraction of $\boldsymbol{x}$ from C* removes $\boldsymbol{x}$ from C and S and marginalizes $\boldsymbol{x}$ out of ϕ_C and ϕ_S; if C becomes non-maximal, it is merged into its subsuming neighbor.

As an illustration, consider the junction tree of Figure 7(a). Notice that the set of clusters containing $\boldsymbol{v}$ constitutes a subtree (because of the running intersection property). We can contract $\boldsymbol{v}$ from C_1 (and S_{12}) or C_4 (and S_{34}), because these are the leaves of the subtree. If we wish to contract $\boldsymbol{v}$ from C_3, an interior node, we can first contract it from C_4 (making C_3 a leaf of the subtree) and then contract it from C_3.

Variable contraction has some attractive properties:

Proposition 3. *Variable contractions preserve the singly connectedness property, the running intersection property, and consistency.*

Thus, the new junction tree is a valid junction tree for some graph, although perhaps not G: the potential property may be violated. Variable contraction is local and efficient: it requires marginalizing a variable out of one cluster potential and one separator potential, which in the Gaussian case can be accomplished in $O(|C|^2)$ time using (50).

Also, variable contraction is a general method of "thinning" a junction tree. Consider the problem of contracting an arbitrary variable $\boldsymbol{v}$ from some cluster C_i. If $\boldsymbol{v}$ satisfies the preconditions of Definition 1, then we can simply contract it. If not, there are two cases to consider. The first is that $\boldsymbol{v}$ appears only in C_i. In this case, we can

1. clone C_i to obtain C_j,
2. contract $\boldsymbol{v}$ from C_i to C_j, and then
3. contract some other variable $\boldsymbol{u}$ from C_j to C_i.

This operation is illustrated in Figure 8. The second case is when $\boldsymbol{v}$ appears elsewhere but $\boldsymbol{v}$ is not a leaf of $\mathcal{T}_{\boldsymbol{v}}$. In this case, we can contract $\boldsymbol{v}$ from other clusters until C_i is a leaf of $\mathcal{T}_{\boldsymbol{v}}$ and then contract $\boldsymbol{v}$ from C_i.

Finally, variable contraction is a maximum likelihood approximation:

Proposition 4. *Let $\hat{\mathcal{T}}$ be the junction tree obtained from the variable contraction of Definition 1. Then*

$$\phi_{\hat{\mathcal{T}}} = \underset{\{q\,:\,\boldsymbol{v} \perp\!\!\!\perp C-S \,|\, S\setminus\boldsymbol{v}\}}{\operatorname{argmax}} \int \phi_{\mathcal{T}} \log q \tag{27}$$

In other words, the probability distribution represented by $\hat{\mathcal{T}}$ has maximum likelihood (under the original junction tree's distribution) over all distributions in which $\boldsymbol{v}$ is conditionally independent of $C-S$ given $S \setminus \boldsymbol{v}$. It is as if we treated the distribution of $\mathcal{T}$ as a data set and then chose the maximum likelihood parameterization of a graphical model that is missing certain edges. Analyzing the conditional indepence properties of $\mathcal{T}$ and $\hat{\mathcal{T}}$ reveals that it is the edges between $\boldsymbol{v}$ and each $\boldsymbol{u} \in C-S$ that have been removed. Thus, we can consider each variable contraction to be a *maximum likelihood projection* that removes the edges between $\boldsymbol{v}$ and $C-S$.

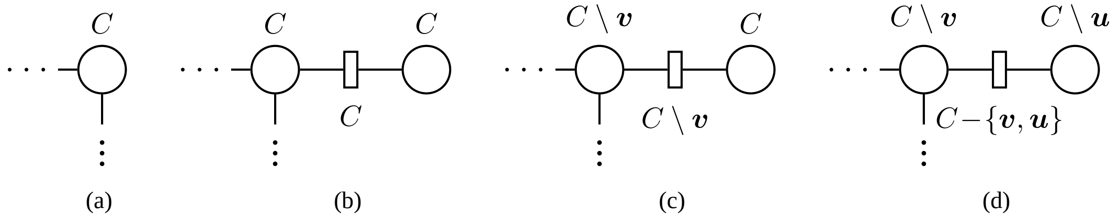

C
C
C
C
$C \setminus \boldsymbol{v}$
C
$C \setminus \boldsymbol{v}$
$C \setminus \boldsymbol{v}$
$C \setminus \boldsymbol{u}$
$C - \{\boldsymbol{v}, \boldsymbol{u}\}$
(a)
(b)
(c)
(d)

Figure 8: An example of a *cloning contraction*. (a) Assume $\boldsymbol{v}$ appears only in C. (b) We first clone C and attach the clone to C; thus, the separator must also contain the variables of C. Now the original cluster is a leaf of $\mathcal{T}_{\boldsymbol{v}}$. (c) We contract $\boldsymbol{v}$ from the original cluster to the clone. (d) We contract some other variable $\boldsymbol{u} \in C$ from the clone. Now the size of both clusters is one less than before.

While computing the maximum likelihood projection to a graphical model with an arbitrary set of edges may be computationally intensive (requiring Iterative Proportional Fitting), deleting particular sets of edges via variable contractions is efficient. This is because projection to *decomposable models* is an efficient operation, and junction trees represent decomposable models [Whittaker, 1989]. Thus, in settings such as SLAM where a graphical model becomes denser over time, periodically thinning the junction tree via variable contraction is an attractive means of ensuring the model remains tractable.

4.2 Thinning clusters with contractions

Consider the problem of reducing of a given junction tree cluster C by one. (A solution to this problem can be applied recursively to thin the junction tree by an arbitrary amount.) Let C_{shared} be the subset of variables in C that appear elsewhere in the junction tree. The question we must now ask is: which of the variables in C_{shared} should be contracted from C? (If we wish to contract any of the variables in C, then we can first clone C so that $C_{\text{shared}} = C$.) Clearly, we would like to contract that variable that minimizes the approximation error, which we take to be $D(\phi_{\mathcal{T}} \,||\, \phi_{\mathcal{T}'})$, the *(differential) relative entropy* (or *Kullback-Liebler divergence*) from the original distribution to the approximate one.

The following result, similar to Theorem 11 of [Kjærulff, 1993], proves that the error introduced by contracting $\boldsymbol{x}$ from a cluster C can be computed using only the marginal over C:

Proposition 5. *Let $\mathcal{T}'$ be the junction tree obtained from the variable contraction of Definition 1. Then $D(\phi_{\mathcal{T}} \,||\, \phi_{\mathcal{T}'}) = I(\boldsymbol{x}; C - S \,|\, S \setminus \boldsymbol{x})$.*

Thus, of all the variables $\boldsymbol{x} \in C_{\text{shared}}$ such that C is a leaf of $\mathcal{T}_{\boldsymbol{x}}$, we should contract the one that minimizes $I(\boldsymbol{x}; C - S \,|\, S \setminus \boldsymbol{x})$. While it is possible to *eventually* contract on all variables in C_{shared}, those variables for which C is not a leaf of $\mathcal{T}_{\boldsymbol{x}}$ will first have to be contracted from other clusters. The error introduced by a sequence of contractions is additive (cf. Theorem 12 of [Kjærulff, 1993]); thus, the error introduced by (eventually) contracting $\boldsymbol{x}$ from C can be computed by adding the errors of each of the successive contractions of $\boldsymbol{x}$.

The conditional mutual information $I(\boldsymbol{x}; C - S \,|\, S \setminus \boldsymbol{x})$ can be computed using $p(C)$. In a consistent junction tree, this marginal is simply ϕ_C, and therefore the approximation error of a variable contraction can be computed locally.

If these mutual information terms were expensive to compute (as they typically are), then this result would be of little value, since computing the best variable to contract would be costly. As it turns out, in the case where $p(C)$ is a Gaussian distribution, the approximation error $I(\boldsymbol{x}; C - S \mid S \setminus \boldsymbol{x})$ can be computed in a constant amount of time *independent of the sizes of C and S*.

Proposition 6. *Let C be a set of Gaussian random variables, let $S \subseteq C$, and let $\boldsymbol{x} \in S$. Then*

$$I(\boldsymbol{x}; C - S \mid S \setminus \boldsymbol{x}) \quad = \quad \frac{1}{2}\left(\log\det\Lambda^{C}_{xx} - \log\det\Lambda^{S}_{xx}\right) \tag{28}$$

where $\Lambda^C = Cov\,(C)^{-1}$ and $\Lambda^S = Cov\,(S)^{-1}$.

When our junction tree is consistent, Λ^C and Λ^S are simply the parameters of ϕ_C and ϕ_S; thus, we can extract the sub-blocks corresponding to $\boldsymbol{x}$ from each and compute the difference of their log determinants.

5 Thin junction tree filters for SLAM

We have now assembled most of the machinery we require to design a *thin junction tree filter* (TJTF) for the SLAM problem. All that remains is the logic that decides which clusters new potentals are multiplied into and also how variable contractions are employed to thin the junction tree.[9] We first present a simple linear-time filter, and then describe a further approximation that results in a constant-time filtering operation.

5.1 Linear-time approximate filtering

Recall from Section 3.1 that if the width of our junction tree is k, then it will require $O(|V|k^2)$ space and message passing will take $O(|V|k^3)$ time. In SLAM $|V| = n_t + 1$, so we can obtain a $O(n_t)$ space filter with a $O(n_t)$ time filter operation by periodically thinning the junction tree so that its width remains bounded by a constant k.

We initialize the filter with a junction tree that has a single cluster containing $\boldsymbol{x}$. We begin with motion update, which consists of the predition and odometry updates and then roll-up. Recall that when $\boldsymbol{x}_t$ is marginalized out in roll-up, we must merge all of the clusters in which it resides. In the worst case, $\boldsymbol{x}_t$ could reside in all of the junction tree's clusters, in which case our belief state would collapse to one large cluster. To prevent this from happening, we start the motion update by first contracting $\boldsymbol{x}_t$ until it resides in only one cluster; then, we perform the prediction and odometry updates and then marginalization. This update is summarized by Algorithm 2.

Algorithm 2 Prediction update and roll-up in the thin junction tree.

```
while $\boldsymbol{x}_t$ resides in more than one cluster do
    Contract $\boldsymbol{x}_t$ from a leaf of $\mathcal{T}_{\boldsymbol{x}}$ so as to minimize (28)
$C \leftarrow$ the only cluster containing $\boldsymbol{x}_t$
Multiply the evolution potential $\psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})$ into $\phi_C$.
Marginalize $\boldsymbol{x}_t$ out of $\phi_C$.
Multiply the odometry potential $\psi(\boldsymbol{x}_{t+1})$ into $\phi_C$.
Distribute evidence from $C$. {if necessary}
```

When updating with a measurement of landmark $\boldsymbol{l}$, the simple case is when $\boldsymbol{x}$ and $\boldsymbol{l}$ cohabit in some cluster C. In that case, we simply multiply the observation potential $\psi(\boldsymbol{x}, \boldsymbol{l})$ into ϕ_C and distribute evidence from C. There are two nontrivial cases:

[9] It is possible (and in fact, not difficult) to design these logics independently of the SLAM application, but doing so leads to algorithms that are inefficient and difficult to analyze in the SLAM domain.

1. If $\boldsymbol{l}$ has not yet been observed, then we multiply $\psi(\boldsymbol{x}, \boldsymbol{l})$ into the smallest cluster C that contains $\boldsymbol{x}$. If adding $\boldsymbol{l}$ to this cluster would cause it to violate the width limit, then we

   (a) clone C to obtain C',

   (b) attach C' to C,

   (c) contract $\boldsymbol{x}$ to C', and

   (d) thin C'.

   A *cluster overlap* parameter h governs by how much C' is thinned, and therefore how many variables reside in its separator. If it is thinned a lot, then it will admit more new landmarks before another cloning is required; the trade-off is that its separator will be small, and therefore the volume of information it can transmit is reduced.

   The motivation for contracting $\boldsymbol{x}$ to C' is two-fold. First, if C' is a perfect clone of C, then all contractions from C' have zero error; thus, in order to choose make good decisions when thinning C', it must have more variables than C. Second, since $\boldsymbol{x}$ is contracted to a single cluster during roll-up, it might as well be contracted to the cluster containing the most recently measured landmarks.

2. If $\boldsymbol{l}$ has been previously observed, then we multiply $\psi(\boldsymbol{x}, \boldsymbol{l})$ into the cluster C that contains $\boldsymbol{l}$ and is closest to another cluster containing $\boldsymbol{x}$. This may involve adding $\boldsymbol{x}$ to C and other clusters in order to restore the running intersection property.

This logic is summarized by Algorithm 3.

Algorithm 3 Multiply a measurement potential $\psi(\boldsymbol{x}, \boldsymbol{l})$ into the thin junction tree.

```
if $\boldsymbol{l}$ has not yet been observed then
  $C \leftarrow$ the smallest cluster containing $\boldsymbol{x}$
  if $|C| = k$ then
    Clone $C$ to obtain $C'$.
    Attach $C'$ to $C$
    Contract $\boldsymbol{x}$ to $C'$.
    while $|C'| > h$ do {thin the clone}
      Perform the variable contraction from $C'$ that minimizes (28).
  Multiply $\psi(\boldsymbol{x}, \boldsymbol{l})$ into $\phi_{C'}$. {updates charge}
else {$\boldsymbol{l}$ has been previously observed}
  $C \leftarrow$ the cluster closest to $\mathcal{T}_{\boldsymbol{x}}$ containing $\boldsymbol{l}$
  $p \leftarrow$ the path from $\mathcal{T}_{\boldsymbol{x}}$ to $C$
  Add $\boldsymbol{x}$ to all clusters and separators along $p$. {restores running intersection property}
  Pass messages along $p$. {restores consistency}
  Multiply $\psi(\boldsymbol{x}, \boldsymbol{l})$ into $\phi_C$. {updates charge}
  Distribute evidence from $C$. {restores consistency}
```

Figures 9, 10, and 11 demonstrate the thin junction tree filter for the example of Figure 1 in which we choose k (the width limit) to be three and h (the overlap) to be two.

Note that when reobservations are incorporated in Algorithm 3, it would also be valid to instead simply add $\boldsymbol{l}$ to the closest cluster containing $\boldsymbol{x}$ (and all clusters necessary to restore the running intersection property). However, this scheme has two disadvantages:

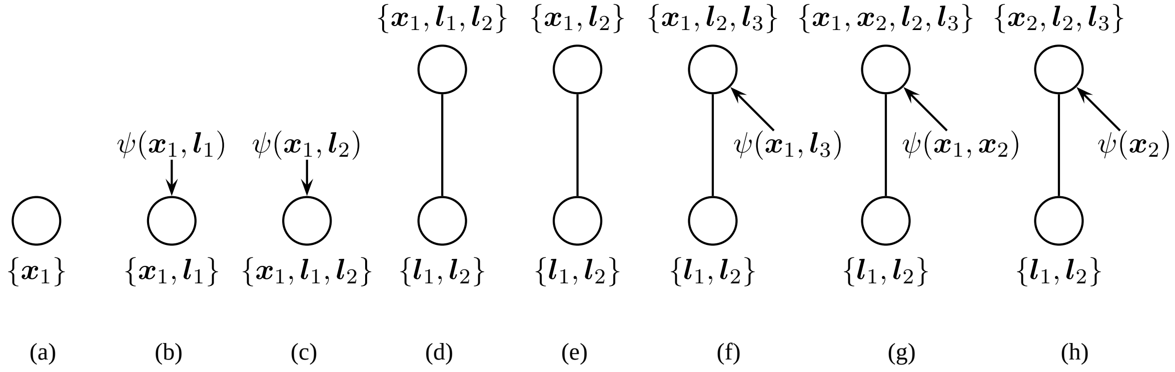

$\{\boldsymbol{x}_1\}$
$\psi(\boldsymbol{x}_1, \boldsymbol{l}_1)$
$\{\boldsymbol{x}_1, \boldsymbol{l}_1\}$
$\psi(\boldsymbol{x}_1, \boldsymbol{l}_2)$
$\{\boldsymbol{x}_1, \boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{x}_1, \boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{x}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{x}_1, \boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\psi(\boldsymbol{x}_1, \boldsymbol{l}_3)$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\psi(\boldsymbol{x}_1, \boldsymbol{x}_2)$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{x}_2, \boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\psi(\boldsymbol{x}_2)$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)

Figure 9: The updates at time $t = 1$, when landmarks one, two, and three are observed. (a) The initial junction tree. (b) $\boldsymbol{l}_1$ is observed and the measurement potential $\psi(\boldsymbol{x}_1, \boldsymbol{l}_1)$ is multiplied into the only cluster. (c) $\boldsymbol{l}_2$ is observed and $\psi(\boldsymbol{x}_1, \boldsymbol{l}_2)$ is multiplied into the only cluster. (d) $\boldsymbol{l}_3$ is observed, but $\psi(\boldsymbol{x}_1, \boldsymbol{l}_3)$ cannot be multiplied into the only cluster because its size is $k = 3$; the cluster is cloned and $\boldsymbol{x}_1$ is contracted to the clone. (e) The clone is thinned so that it contains $h = 2$ variables. Assume $I(\boldsymbol{l}_1; \boldsymbol{x}_1 \mid \boldsymbol{l}_2) < I(\boldsymbol{l}_2; \boldsymbol{x}_1 \mid \boldsymbol{l}_1)$, so contracting $\boldsymbol{l}_1$ yields the smallest approximation error. (f) $\psi(\boldsymbol{x}_1, \boldsymbol{l}_3)$ is multiplied into the top cluster's potential. (g) For the motion update, $\psi(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is multiplied into the top cluster's potential. (h) $\boldsymbol{x}_1$ is marginalized out of the top cluster and the odometry potential $\psi(\boldsymbol{x}_2)$ is multiplied in.

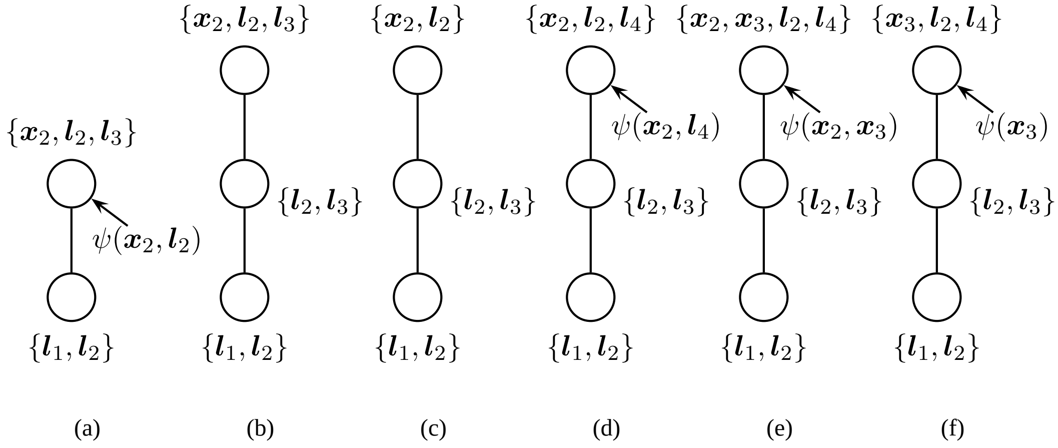

$\{\boldsymbol{x}_2, \boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\psi(\boldsymbol{x}_2, \boldsymbol{l}_2)$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{x}_2, \boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\{\boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{x}_2, \boldsymbol{l}_2\}$
$\{\boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{x}_2, \boldsymbol{l}_2, \boldsymbol{l}_4\}$
$\psi(\boldsymbol{x}_2, \boldsymbol{l}_4)$
$\{\boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{l}_2, \boldsymbol{l}_4\}$
$\psi(\boldsymbol{x}_2, \boldsymbol{x}_3)$
$\{\boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
$\{\boldsymbol{x}_3, \boldsymbol{l}_2, \boldsymbol{l}_4\}$
$\psi(\boldsymbol{x}_3)$
$\{\boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
(a)
(b)
(c)
(d)
(e)
(f)

Figure 10: The updates at time $t = 2$, when landmarks two and four are observed. (a) The measurement potential $\psi(\boldsymbol{x}_2, \boldsymbol{l}_2)$ is multiplied into the top cluster. (b) $\boldsymbol{l}_4$ is observed, but $\psi(\boldsymbol{x}_2, \boldsymbol{l}_4)$ cannot be multiplied into the top cluster because its size is $k = 3$; the top cluster is cloned, and $\boldsymbol{x}_2$ is contracted to the clone. (c) The clone is thinned so that it contains $h = 2$ variables. Assume $I(\boldsymbol{l}_3; \boldsymbol{x}_2 \mid \boldsymbol{l}_2) < I(\boldsymbol{l}_2; \boldsymbol{x}_2 \mid \boldsymbol{l}_3)$, so contracting $\boldsymbol{l}_3$ yields the smallest approximation error. (d) $\psi(\boldsymbol{x}_2, \boldsymbol{l}_4)$ is multiplied into the top cluster's potential. (e) $\psi(\boldsymbol{x}_2, \boldsymbol{x}_3)$ is multiplied into the top cluster's potential. (f) $\boldsymbol{x}_2$ is marginalized out, and $\psi(\boldsymbol{x}_3)$ is multiplied in.

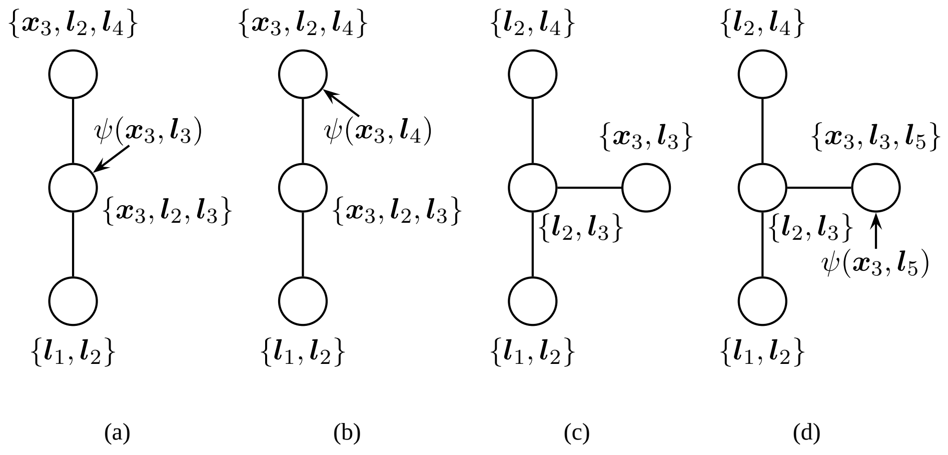

$\{\boldsymbol{x}_3, \boldsymbol{l}_2, \boldsymbol{l}_4\}$
$\psi(\boldsymbol{x}_3, \boldsymbol{l}_3)$
$\{\boldsymbol{x}_3, \boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
(a)
$\{\boldsymbol{x}_3, \boldsymbol{l}_2, \boldsymbol{l}_4\}$
$\psi(\boldsymbol{x}_3, \boldsymbol{l}_4)$
$\{\boldsymbol{x}_3, \boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
(b)
$\{\boldsymbol{l}_2, \boldsymbol{l}_4\}$
$\{\boldsymbol{x}_3, \boldsymbol{l}_3\}$
$\{\boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
(c)
$\{\boldsymbol{l}_2, \boldsymbol{l}_4\}$
$\{\boldsymbol{x}_3, \boldsymbol{l}_3, \boldsymbol{l}_5\}$
$\{\boldsymbol{l}_2, \boldsymbol{l}_3\}$
$\psi(\boldsymbol{x}_3, \boldsymbol{l}_5)$
$\{\boldsymbol{l}_1, \boldsymbol{l}_2\}$
(d)

Figure 11: The updates at time $t = 3$, when landmarks three, four, and five are observed. (a) $\psi(\boldsymbol{x}_3, \boldsymbol{l}_3)$ is multiplied into the middle cluster. (b) $\psi(\boldsymbol{x}_3, \boldsymbol{l}_4)$ is multiplied into the top cluster. (c) $\boldsymbol{l}_5$ is observed, but the smallest cluster containing $\boldsymbol{x}$ has reached the size limit k; the middle cluster is cloned, $\boldsymbol{x}$ is contracted to the clone, and then the clone is thinned so that it contains $h = 2$ variables. Assume $I(\boldsymbol{l}_2; \boldsymbol{x}_3 \mid \boldsymbol{l}_3) < I(\boldsymbol{l}_3; \boldsymbol{x}_3 \mid \boldsymbol{l}_2)$, so contracting $\boldsymbol{l}_2$ yields the smallest approximation error. (d) $\psi(\boldsymbol{x}_3, \boldsymbol{l}_5)$ is multiplied into the clone.

1. In Algorithm 3, restoring the running intersection property can add $\boldsymbol{x}$ to many clusters of the junction tree, but the subsequent execution of Algorithm 2 contracts it back to the active cluster; thus, the net change in the sizes of these clusters is zero. If during the measurement update we instead were to add $\boldsymbol{l}$ to the active cluster (and intermediate clusters), additional thinning operations would be required to ensure the intervening clusters satisfy the width limit.

2. Empirically, we find that variables that have not been observed for some time typically reside in clusters that are far from the active cluster. (This is intuitive, since the mutual information between $\boldsymbol{x}$ and a landmark variable decreases over time if the landmark is not reobserved.) Thus, closing the loop in the variant scheme would cause a mass migration of variables through the junction tree, requiring expensive thinning operations.

5.2 Constant-time approximate filtering

Because landmark observations cause the robot's state variable to move to clusters containing recently-observed landmarks, we expect that landmarks whose state variables are far from the robot's state variable (in the junction tree) are those that were last observed in the distant past. Thus, in the common case where the robot observes new landmarks and reobserves some recently-viewed landmarks, only a small portion of the junction tree's structure changes. Moreover, the linear-time filter presented above would take constant time, were it not for the fact that reobserving landmarks necessitates a round of message passing.

We can get a constant-time filter operation by employing a *lazy message passing* scheme, where we distribute evidence only to constantly many nearby clusters; the approximation is that the marginals of the remaining clusters will not be conditioned on the observation. This technique can be useful in domains where the robot has a limited number of floating-point operations it can perform in each time step.

This additional approximation has the disirable property that it is *temporary*: because we are still updating the charge correctly, at any later time we can use a complete round of message passing (taking linear

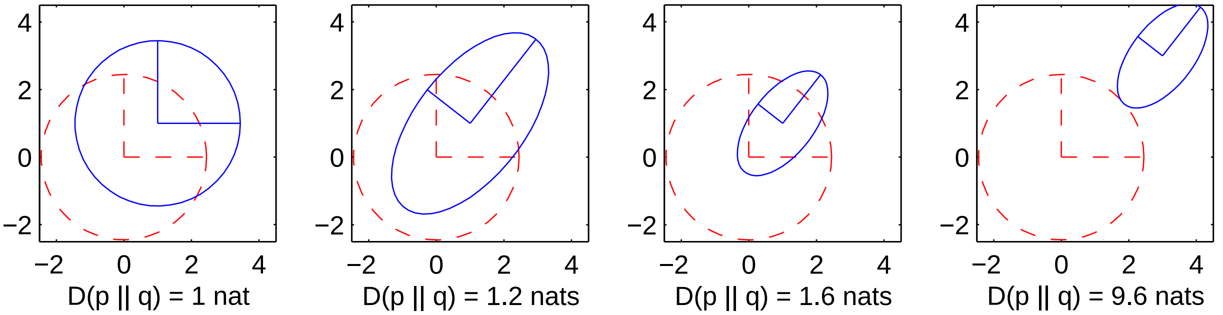


Figure 12: Four examples of differential relative entropy for two-dimensional Gaussians (such as planar location estimates). p is represented by the blue (solid) 95% confidence region, and q is represented by the red (dashed) 95% confidence region. As q becomes a poorer model for p, the relative entropy increases.

time) to update our map entirely; the filtered estimate becomes the same estimate that would have obtained from passing all messages at every time step. Our sacrifice has simply been a period during which the majority of the map was slightly out of date.

Alternatively, we can smoothly interpolate between the original linear-time filter and this constant-time filter by employing an *adaptive message passing* scheme, in which messages are propagated only as long as they induce significant changes in the belief state. If we define "significant" sensibly, this scheme will take constant time when mapping new territory and when closing loops, it will take time linear in the length of the loop.

To determine if a message $C_i \to C_j$ has induced a significant change, we compute the *(differential) relative entropy* (or *Kullback-Liebler divergence*) from the separator marginal before the message is passed to the separator marginal after the message is passed:

$$D(\phi^*_{S_{i,j}} \,||\, \phi_{S_{i,j}}) = \int \phi^*_{S_{i,j}} \log \frac{\phi^*_{S_{i,j}}}{\phi_{S_{i,j}}} \tag{29}$$

$$= \frac{1}{2}\left[\log \frac{\det \Sigma}{\det \Sigma^*} - n + \text{trace}(\Sigma^{-1}(\Sigma^* + (\mu^* - \mu)(\mu^* - \mu)^T)\right] \tag{30}$$

where $\phi_{S_{i,j}} = \mathcal{N}(\mu, \Sigma)$, $\phi^*_{S_{i,j}} = \mathcal{N}(\mu^*, \Sigma^*)$, and n is the sum dimension of the variables in $S_{i,j}$.

Our choice of the relative entropy can be justified in several ways. First, it is a well-known "dissimilarity" measure for probability distributions (i.e., it is non-negative and zero iff the two distributions are equal), and is able to quantify both changes in value and certainty; see Figure 12 for some examples.

Second, we can draw a valuable analogy between using relative entropy to determine when to pass messages in a junction tree and its use in Information Theory [Cover and Thomas, 1991]. In a communication context, the relative entropy from a distribution p to another distribution q can be thought of as the expected extra number of bits (actually, *nats*—natural logarithmic units—in the Gaussian case) necessary to communicate the value of a sample from the true distribution p given we have an optimal coding scheme based upon our approximation q. In our context, $D(\phi^*_{S_{i,j}} \,||\, \phi_{S_{i,j}})$ represents a measure of the overhead involved in communicating our updated knowledge $\phi^*_{S_{i,j}}$ using our outdated representation $\phi_{S_{i,j}}$; if the overhead is not too great, we will not bother to revise our beliefs.

Finally, it can be easily shown that the "significance" of evidence propagated from some cluster C to some other cluster C' (measured using relative entropy) decreases with the distance between C and C' in the junction tree [Kjærulff, 1993, Theorem 13]. Thus, we can be certain that if a message was not significant for a particular cluster, then that cluster need not continue the evidence propagation.

6 Experiments

We now present the results of experiments using simulated SLAM problems. In particular, we compare the computational cost and estimation accuracy of the Kalman filter, TJTF, and FastSLAM. It is appropriate to preface these results with a brief discussion of what conclusions one could reasonably draw from them.

In any realistic SLAM problem, *all three* of these algorithms are approximations (since the Kalman filter incurs linearization error). When one compares a set of approximation algorithms on particular applications, there is a danger that the particular problems considered may favor one approximation algorithm over another. This problem is especially troublesome when the problem class is diverse, as in SLAM: different state spaces, motion and sensor models, and control policies can dramatically affect how well a particular approximate algorithm will perform. For example:

1. If the motion or observation models are highly nonlinear, then techniques that must linearize them (the Kalman filter and TJTF must linearize both, and FastSLAM must linearize the measurement model) will perform poorly.

2. If the controller generates large loops, the world contains regions without enough landmarks, or there is a large amount of noise, FastSLAM will be more susceptible to divergence, for the reasons described in Section 1.2.1.

Thus, the strength of the conclusions that the reader can draw from the experiments presented below will depend greatly on the similarity of their application to the ones used below.

6.1 The SLAM simulations

We have chosen to use simulated SLAM problems rather than real ones in our experiments for two reasons: first, it is very time consuming to obtain ground-truth in real SLAM problems; and second, using simulations allows us to examine how these algorithms perform under varying conditions. The SLAM simulations we use are inspired by the familiar task of a mobile robot mapping a planar environment with feature-type landmarks.

In the family of SLAM problems we consider, the robot operates in a planar world of stationary point landmarks. The robot's state is described by a vector

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^x \\ \boldsymbol{x}^y \\ \boldsymbol{x}^h \\ \boldsymbol{x}^t \\ \boldsymbol{x}^r \end{bmatrix} \tag{31}$$

containing the (x, y) coordinates of the robot, its heading h (relative to the positive x-axis), its translational velocity t along that heading, and its rotational velocity r about the z-axis. The state of each landmark is represented by a vector

$$\boldsymbol{l} = \begin{bmatrix} \boldsymbol{l}^x \\ \boldsymbol{l}^y \end{bmatrix} \tag{32}$$

giving the (x, y) coordinates of the landmark's position.

We use a landmark observation model in which the robot makes noisy range and bearing measurements of every landmark that lies within a forward-looking "visual cone" of a prescribed angular width c_w and length c_l. Assume a landmark $\boldsymbol{l}$ falls within the visual cone and define

$$\delta = \begin{bmatrix} \boldsymbol{x}^x \\ \boldsymbol{x}^y \end{bmatrix} - \begin{bmatrix} \boldsymbol{l}^x \\ \boldsymbol{l}^y \end{bmatrix} \tag{33}$$

to be the vector from the position of the robot to that of the landmark. Then the robot receives a range-bearing measurement

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}^r \\ \boldsymbol{z}^b \end{bmatrix} = \begin{bmatrix} ||\delta||_2(1 + \boldsymbol{u}^r) + \boldsymbol{u}^a \\ \arctan(\delta) - \boldsymbol{x}^h + \boldsymbol{u}^b \end{bmatrix} \tag{34}$$

where $\boldsymbol{u}^a$, $\boldsymbol{u}^r$ and $\boldsymbol{u}^b$ are independent zero-mean Gaussian noise variables; thus, there is additive (or absolute) and multiplicative (or relative) Gaussian noise in the range measurements and absolute Gaussian noise in the bearing measurements. This has the effect of making the location of far-off landmarks harder to estimate than those close to the robot, while retaining some noise in observations of close landmarks. To avoid problems associated with linearizing functions with periodic outputs, the landmark measurement $\boldsymbol{z}$ is represented in Cartesian coordinates rather than the polar coordinates of (34).

At each time step the robot exerts a control

$$u \quad = \quad \begin{bmatrix} u^t \\ u^r \end{bmatrix} \tag{35}$$

consisting of a desired translational and rotational velocity. The robot translates along its current heading at its current translational velocity, rotates at its current rotational velocity, and then updates its velocities according to the control signal. The dynamics model is

$$\boldsymbol{x}_{t+1} = \begin{bmatrix} \boldsymbol{x}^x_{t+1} \\ \boldsymbol{x}^y_{t+1} \\ \boldsymbol{x}^h_{t+1} \\ \boldsymbol{x}^t_{t+1} \\ \boldsymbol{x}^r_{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^x_t + \boldsymbol{x}^t_t \cos \boldsymbol{x}^h_t \\ \boldsymbol{x}^y_t + \boldsymbol{x}^t_t \sin \boldsymbol{x}^h_t \\ \boldsymbol{x}^h_t + \boldsymbol{x}^r_t \\ u^t(1 + \boldsymbol{v}^t) + \boldsymbol{v}^a \\ u^r(1 + \boldsymbol{v}^r) + \boldsymbol{v}^b \end{bmatrix} \tag{36}$$

where $\boldsymbol{v}^t$, $\boldsymbol{v}^a$, $\boldsymbol{v}^r$ and $\boldsymbol{v}^b$ are independent zero-mean Gaussian noise variables; thus, there is both absolute and relative noise in the controls. After moving, the robot receives an odometry measurement of its translational and rotational velocity

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}^t \\ \boldsymbol{y}^r \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^t + \boldsymbol{w}^t \\ \boldsymbol{x}^r + \boldsymbol{w}^r \end{bmatrix} \tag{37}$$

where $\boldsymbol{w}^r$ and $\boldsymbol{w}^t$ are independent zero-mean Gaussian noise variables representing absolute noise in the differential odometry. (Note that this odometry model is of the type described in Section 3.3, and so odometry updates in TJTF will not require message passing.)

The motion and landmark observation models are nonlinear; we use the UKF for linearization in all cases. In the case of FastSLAM, this linearization was performed on a per-particle basis as required by the algorithm.

We use two types of SLAM problems (see Figure 13): a square loop (such as a robot would encounter when mapping the interior of a building) and a switchback pattern (which could be used to map an open area). In both cases, the robot maps a square region whose side length is L. In the case of switchbacks, the distance between the "passes" is fixed at half of the width of the visual cone; this ensures that in each pass the robot will reobserve approximately half of the landmarks it observed on the previous pass.

Simulation parameter		Default value
c_w	width of the visual cone	$180°$
c_l	length of the visual cone	$10m$
t_{max}	maximum translational velocity	$0.5m/s$
r_{max}	maximum rotational velocity	$30°/s$
$\text{STDEV}(\boldsymbol{v}^t)$	relative noise in translational velocity control (STDEV)	0.03
$\text{STDEV}(\boldsymbol{v}^a)$	absolute noise in translational velocity control (STDEV)	$0.02m/s$
$\text{STDEV}(\boldsymbol{v}^r)$	relative noise in rotational velocity control (STDEV)	0.05
$\text{STDEV}(\boldsymbol{v}^b)$	absolute noise in rotational velocity control (STDEV)	$1°/s$
$\text{STDEV}(\boldsymbol{w}^t)$	additive noise in translational velocity odometry (STDEV)	$0.02m/s$
$\text{STDEV}(\boldsymbol{w}^r)$	additive noise in rotational velocity odometry (STDEV)	$1°/s$
$\text{STDEV}(\boldsymbol{u}^b)$	additive noise in bearing measurements (STDEV)	$2°$
$\text{STDEV}(\boldsymbol{u}^r)$	relative noise in range measurements (STDEV)	0.1
$\text{STDEV}(\boldsymbol{u}^a)$	absolute noise in range measurements (STDEV)	$0.5m$
L	the length of the side of the square region being mapped	$100m$
N	number of landmarks	1000

Table 1: A table of the simulation parameters and their default values.

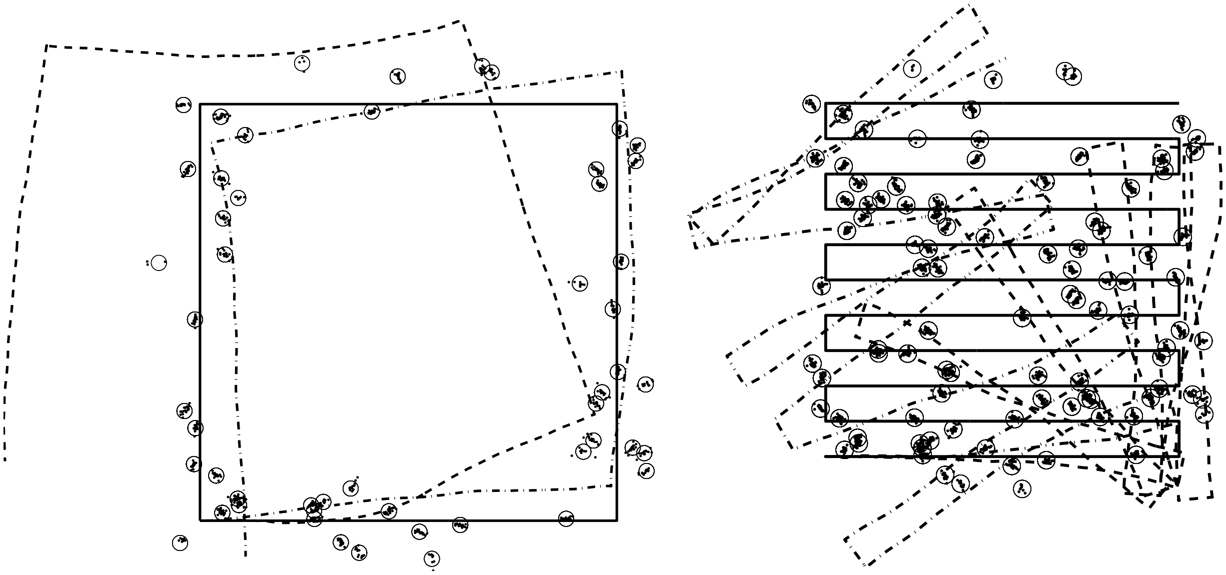
Choosing the controls to obtain these paths presents an interesting dilemma. We would like to fix the simulation in advance—which requires fixing the controls in advance—so that all algorithms receive the same input and our comparisons are fair. However, if we fix the control signal without taking motion noise into account, the actual robot path will differ markedly from the desired (square or switchbacks) trajectory (resulting, for example, in the robot not *physically* closing the loop of a square trajectory). Our solution is to fix the path of the robot and the motion noise in advance, and then *invert* the motion model to obtain an *omniscient controller*—i.e., one that chooses control sequence knowing the (in reality, unobservable) noise pattern in advance. (This explains the erratic path of the integrated control signal in Figure 13—to counteract the motion noise, the control signal must be noisy.) Using this technique, we compute the control law by having the robot follow the desired trajectory as quickly as possible, subject to the constraint that its translational and rotational velocity controls do not exceed t_{max} and r_{max}, respectively.

Finally, in each simulation the environment was populated with landmarks by sampling N landmark locations uniformly from the smallest rectangular region that could possible be observed by the robot, given its path and its visual field. Table 1 contains a summary of the parameters necessary to define a simulation and their default values. Figure 13 contains an example of each type of trajectory.

6.2 Metrics

In order to quantify the computational cost of each filter operation, we compute the number of floating-point operations used by each filter operation. The dominant cost in all three of these algorithms is floating point operations, and so this is a very accurate (and portable) method of quantifying not only the computational complexity of the algorithms (which we have already analyzed theoretically), but also their respective "constant factors".

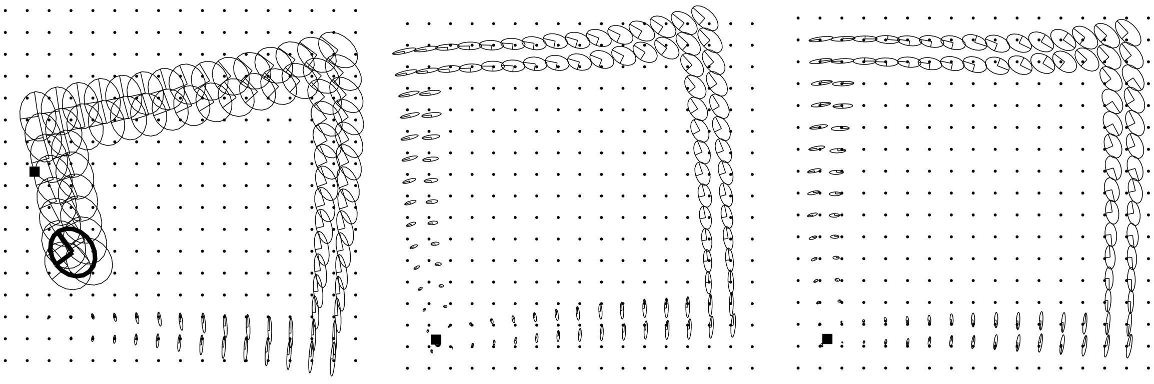
We divide the estimation error into *localization error* and *mapping error*. We could measure these errors using Euclidean distances between ground truth and the algorithm's mean estimate, but this would exhibit a bias towards the coordinate system used in the problem description. Even though we initialize every filter with the correct initial state of the robot, it is possible for the filter to settle into a coordinate system different



(a) A square trajectory.

(b) A switchback trajectory.

Figure 13: Two example simulations. The solid line is the actual robot path; the dashed line is the integrated odometry; the dash-dotted line is the integrated control signal; circles are landmarks; and dots are landmark observations (relative to the unknown actual viewpoint). For clarity, only a fraction of the 1000 landmarks are plotted.



(a) partial map

(b) final map

(c) aligned map

Figure 14: The Kalman filter is run on a square SLAM problem with the landmarks spaced on a grid. (a) Before closing the loop, accumulated error has caused the estimate of the robot's trajectory to be deformed. (b) After closing the loop, the trajectory estimate is improved (more square-like), but it is rotated slightly about the robot's starting point. (c) By choosing the optimal rigid transformation of the map that minimizes average landmark error, we see that the final map is quite good.

than that used by the problem description. This can happen, for example, when the filter makes an early mistake about the true heading or position of the robot and obtains a rotated or translated map; it can also occur when closing the loop causes all landmarks to shift. See Figure 14 for an example.

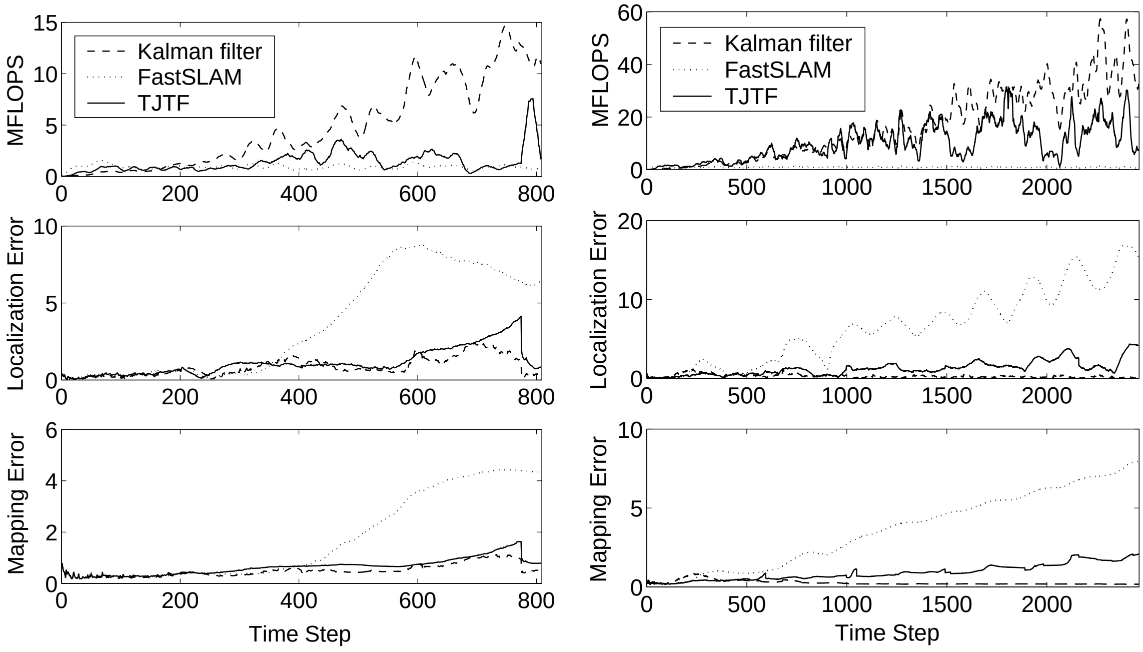
Because the value of a map is independent of its coordinate system, we calculate the localization and mapping errors as follows. We first compute the (unscaled) rigid transformation that minimizes the sum of squared Euclidean distances between landmark positions and their filter estimates (as described in [Umeyama, 1991]). Then, the localization error is computed as the Euclidean distance between robot's position and the transformed filter's estimate; the mapping error is computed as the average Euclidean distance between the landmarks' true positions and the transformed filter's estimates.

6.3 Comparisons to the Kalman filter and FastSLAM

Figure 15 shows how TJTF, the Kalman filter, and FastSLAM behave when run on the simulations shown in Figure 13. TJTF was run with $k = 16$, $h = 4$, and adaptive message passing with the significance threshold set at 0.1 natural logarithmic units; FastSLAM was run with 100 particles, as recommended in [Montemerlo *et al.*, 2002]. In these plots, the floating point counts are time-averaged for clarity.

We found that the estimation error of TJTF with maximum cluster sizes as small as 16 can be comparable with the Kalman filter, and that it gets smaller as k increases. This indicates that the edges removed by TJTF indeed carry little information; it also suggests that the estimation error of TJTF will be at least competitive with that of SEIF (an approximate form of the Kalman filter) and less than that of the submap approaches (which neglect long-distance correlations).

We also found that TJTF is good at closing loops; in Figure 15(a) we can see the localization and mapping error of TJTF suddenly drop at $t = 780$, when the robot first reobserves its starting point; also evident is a sudden increase in the computational cost: the filter is choosing to update the entire map in linear time rather than using cheaper constant-time updates. We found that FastSLAM had difficulty closing large loops (notice its divergence in Figure 15(a)) and that its estimation error in general was larger than that of TJTF.



(a) filter profile of 13(a)

(b) filter profile of 13(b)

Figure 15: Performance of TJTF ($k = 16$, $h = 4$, $s = 0.1$), the Kalman filter, and FastSLAM ($n = 100$) on the simulations depicted in Figure 13.

Using accurate counts of floating point operations, we found that TJTF can be as fast as FastSLAM, and that it becomes more efficient than the Kalman filter when the map contains a few hundred or more landmarks. Of course, the constant factor associated with TJTF varies significantly with the width limit k, the overlap h, and the significance s used in adaptive message passing, allowing the practitioner to trade-off computational cost for estimation accuracy.

7 Discussion

TJTF is a novel approach to maintaining tractable approximations to a belief state that can be represented by graphical models with changing dependencies. As such, it represents a principled, flexible, and efficient solution to the SLAM complexity problem. Having now worked through the details of the filter, we can now offer a detailed comparison of it with the other methods described in Section 1.2.

7.1 Comparison to previous approaches

In the SLAM domain, TJTF can be made to (frequently) operate in constant-time, making its time complexity competitive with that of the FastSLAM algorithm. TJTF also represent a single estimate of the map, in contrast to FastSLAM; while this improves the complexity of data association and extracting map marginals, it significantly curbs the representation's expressiveness. Finally, while FastSLAM relies on the diversity of the particle population to represent correlations between landmark estimates (and is therefore sensitive to noise, due to the curse of dimensionality), TJTF chooses the optimal correlations to represent directly, subject to a computational complexity constraint; thus, we expect it to be less susceptible to divergence.

Interestingly, the approach presented here has significant connections to both the submap approach and SEIF. First, the belief state of a TJTF has a natural interpretation as a coupled set of local maps, just as in the submap approaches. In particular, each cluster of the junction tree can be viewed as a submap, and the consistency of the junction tree can be viewed as a constraint that overlapping submaps must agree on their shared content. The TJTF formulation gives concrete semantics to the relationships between the maps, including how they must be updated, how consistency is maintained, and how the set of local maps can be determined online to minimize the approximation error subject to a complexity constraint.

Second, since thinning the junction tree is equivalent to removing sets of edges from the graphical model, which in turn is equivalent to zeroing out entries of Λ, TJTF can be viewed as a technique for making Λ sparse; thus, its goal is identical to that of SEIF. As stated above, the maximum likelihood approach to deleting single edges requires Iterative Proportional Fitting, and is computationally complex; SEIF employs an approximation that is constant time, but leads to representational problems. In contrast, the approach presented here is a maximum likelihood sparse approximation to Λ that can be computed analytically and efficiently; it permits constant-time access to marginal distributions, and it allows both constant-time and linear-time updates. As we have seen, these improvements arise because we restrict ourselves to a distinguished set of sparsity patterns—those that correspond to the set of decomposable models.

7.2 A more general understanding

TJTF falls into a broad class of approximate inference techniques for dynamic probabilistic inference called *Assumed Density Filtering* (ADF). An ADF algorithm performs standard filtering operations until the complexity of the belief state starts to become unmanageable; then it is projected to a more tractable family of distributions. (The standard application of ADF is to *Switching Kalman Filters*, where the belief state consists of a mixture of n Gaussians where n grows exponentially over time; ADF is used to periodically project the mixture to one with a constant number of mixing components.) TJTF follows the ADF model where the tractable family of distributions is the family of decomposable models characterized by thin junction trees; there is one caviat, however: the independence structure of the TJTF projection is not determined in advance, as in ADF; rather, the projection is determined online in order to minimize the approximation error. Thus, we can view TJTF as a type of *adaptive* ADF.

One of the more exciting developments in dynamic probabilistic inference is the analysis of ADF-type algorithms given in [Boyen and Koller, 1998], which proves that when the independence structure of the projection remains a fixed product of independent terms, the approximation error cannot increase without bound. This analysis was extended in [Boyen and Koller, 1999] to the case where the belief state is represented by an adaptively chosen junction tree, just as in TJTF. (In fact, TJTF can be viewed as perhaps the simplest method of implementing the architecture they describe.) They give theoretical results that bound the influence different clusters in the junction tree can have on one another (similar in flavor to the results in [Kjærulff, 1993]). These theoretical results give powerful intuitions into the design of good approximate filters.

Acknowledgments

I gratefully acknowledge Intel Corporation for supporting this research via an Intel Research Internship, as well as Sekhar Tatikonda, Francis Bach, and Andrew Ng for valuable discussions.

A The multivariate gaussian

This section presents the basic facts about Gaussian random variables that are used in this technical report.

A.1 Parameterizations

The multivariate Gaussian probability density function can be expressed in two forms. The first is called the *moment parameterization*:

$$p(x) = \frac{1}{\sqrt{(2\pi)^n \det \Sigma}} \exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\} \tag{38}$$

where μ (a vector) and Σ (a positive semidefinite matrix[10]) are the parameters (and n is the dimension of the vector-valued random variable $\boldsymbol{x}$). It is called the moment parameterization because $\mathbb{E}\left(\boldsymbol{x}\right) = \mu$ and $\mathrm{Cov}\left(\boldsymbol{x}\right) = \Sigma$. The quadratic form in the exponent, $(x-\mu)^T\Sigma^{-1}(x-\mu)$, is called the *Mahalanobis distance between x and μ under Σ*. It is a generalization of Euclidean distance that allows the coordinate system to be scaled and sheared; the squared Euclidean distance is obtained by taking $\Sigma = I$. If $\boldsymbol{x}$ is a random variable distributed according to a multivariate Gaussian distribution with moment parameters μ and Σ, then we write $\boldsymbol{x} \sim \mathcal{N}(\mu, \Sigma)$.

The second form is called the *canonical parameterization*, and is obtained from the moment parameterization by rearranging terms:

$$\begin{aligned}
p(x) &= \exp\left\{-\frac{1}{2}\log((2\pi)^n \det \Sigma) - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} && (39)\\
&= \exp\left\{-\frac{n}{2}\log(2\pi) + \frac{1}{2}\log\det\Sigma^{-1} - \frac{1}{2}(x^T\Sigma^{-1}x + \mu^T\Sigma^{-1}\mu - 2x^T\Sigma^{-1}\mu)\right\} && (40)\\
&= \exp\left\{-\frac{1}{2}(n\log(2\pi) - \log\det\Sigma^{-1} + \mu^T\Sigma^{-1}\mu) + x^T\Sigma^{-1}\mu - \frac{1}{2}x^T\Sigma^{-1}x\right\} && (41)\\
& && (42)
\end{aligned}$$

Defining $\Lambda = \Sigma^{-1}$ and $\eta = \Sigma^{-1}\mu$, we can write this as

$$p(x) = \exp\left\{a + \eta^T x - \frac{1}{2}x^T\Lambda x\right\} \tag{43}$$

where

$$a = -\frac{1}{2}(n\log(2\pi) - \log\det\Lambda + \eta^T\Lambda^{-1}\eta) \tag{44}$$

is the (log) normalization constant. The parameters η and Λ are called the *information vector* and the *information (or precision) matrix*, respectively. If $\boldsymbol{x}$ is a random variable distributed according to a multivariate Gaussian distribution with canonical parameters η and Λ, then we write $\boldsymbol{x} \sim \mathcal{N}^{-1}(\eta, \Lambda)$.

A.2 Marginalization and conditioning

In this section, we present the formulae for marginal and conditional Gaussian distributions for both the moment and canonical parameterizations; proofs of these formulae can be found in [Jordan, 2003]. Let $\boldsymbol{x}$ be a multivariate Gaussian random variable. Partition $\boldsymbol{x}$ as follows:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} \tag{45}$$

[10] We will regard all positive semidefinite matrices as symmetric, since we can always substitute $\frac{1}{2}(\Sigma + \Sigma^T)$.

Then $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are also multivariate Gaussian random variables. If $\boldsymbol{x} \sim \mathcal{N}(\mu, \Sigma)$ where

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \tag{46}$$

then $\boldsymbol{x}_1 \sim \mathcal{N}(\mu_1^{\mathrm{m}}, \Sigma_1^{\mathrm{m}})$ where

$$\mu_1^{\mathrm{m}} = \mu_1 \tag{47}$$

$$\Sigma_1^{\mathrm{m}} = \Sigma_{11} \tag{48}$$

Thus, marginalization in the moment parameterization is performed by simply selecting the relevant blocks of the parameters. If instead $\boldsymbol{x} \sim \mathcal{N}^{-1}(\eta, \Lambda)$ where

$$\eta = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \quad \Lambda = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix} \tag{49}$$

then $\boldsymbol{x}_1 \sim \mathcal{N}^{-1}(\eta_1^{\mathrm{m}}, \Lambda_1^{\mathrm{m}})$ where

$$\begin{aligned} \eta_1^{\mathrm{m}} &= \eta_1 - \Lambda_{12}\Lambda_{22}^{-1}\eta_1 \\ \Lambda_1^{\mathrm{m}} &= \Lambda_{11} - \Lambda_{12}\Lambda_{22}^{-1}\Lambda_{21} \end{aligned} \tag{50}$$

Thus, when the dimension of $\boldsymbol{x}_2$ is $O(n)$, computing the marginal of $\boldsymbol{x}_1$ using this technique can require $O(n^3)$ time.

The situation is reversed when we condition on $\boldsymbol{x}_2$ rather than marginalize it out: the computation is trivial in the canonical parameterization, and expensive in the moment parameterization. If $\boldsymbol{x} \sim \mathcal{N}^{-1}(\eta, \Lambda)$ then $p(\boldsymbol{x}_1 \,|\, x_2) = \mathcal{N}^{-1}(\eta_{1|2}^{\mathrm{c}}, \Lambda_{1|2}^{\mathrm{c}})$ where

$$\begin{aligned} \eta_{1|2}^{\mathrm{c}} &= \eta_1 - \Lambda_{12}x_2 \\ \Lambda_{1|2}^{\mathrm{c}} &= \Lambda_{11} \end{aligned} \tag{51}$$

If instead $\boldsymbol{x} \sim \mathcal{N}(\mu, \Sigma)$ then $p(\boldsymbol{x}_1 \,|\, x_2) = \mathcal{N}(\mu_{1|2}^{\mathrm{c}}, \Sigma_{1|2}^{\mathrm{c}})$ where

$$\begin{aligned} \mu_{1|2}^{\mathrm{c}} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \\ \Sigma_{1|2}^{\mathrm{c}} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \end{aligned} \tag{52}$$

A.3 Affine Gaussian functions

If $\boldsymbol{x} \sim \mathcal{N}(\mu_x, \Sigma_x)$ and $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, R)$ are independent and we define

$$\boldsymbol{y} = A\boldsymbol{x} + b + \boldsymbol{w} \tag{53}$$

then $\boldsymbol{y} \sim \mathcal{N}(\mu_y, \Sigma_{y,y})$ where

$$\mu_y = \mathbb{E}\left(b + A\boldsymbol{x} + \boldsymbol{w}\right) = b + A\mathbb{E}\left(\boldsymbol{x}\right) + \mathbb{E}\left(\boldsymbol{w}\right) = b + A\mu_x \tag{54}$$

$$\begin{aligned} \Sigma_{y,y} &= \mathbb{E}\left((\boldsymbol{y} - \mathbb{E}\left(\boldsymbol{y}\right))(\boldsymbol{y} - \mathbb{E}\left(\boldsymbol{y}\right))^T\right) \\ &= \mathbb{E}\left((A(\boldsymbol{x} - \mu_x) + \boldsymbol{w})(A(\boldsymbol{x} - \mu_x) + \boldsymbol{w})^T\right) \\ &= \mathbb{E}\left(A(\boldsymbol{x} - \mu_x)(\boldsymbol{x} - \mu_x)^T A^T + A(\boldsymbol{x} - \mu_x)\boldsymbol{w}^T + \boldsymbol{w}(\boldsymbol{x} - \mu_x)^T A^T + \boldsymbol{w}\boldsymbol{w}^T\right) \\ &= A\mathbb{E}\left((\boldsymbol{x} - \mu_x)(\boldsymbol{x} - \mu_x)^T\right) A^T + A\mathbb{E}\left((\boldsymbol{x} - \mu_x)\boldsymbol{w}^T\right) + \mathbb{E}\left(\boldsymbol{w}(\boldsymbol{x} - \mu_x)^T\right) A^T + \mathbb{E}\left(\boldsymbol{w}\boldsymbol{w}^T\right) \\ &= A\Sigma_x A^T + R \end{aligned} \tag{55}$$

Moreover, the cross-covariance $\Sigma_{x,y}$ is

$$\begin{aligned}
\Sigma_{x,y} &= \mathbb{E}\left((\boldsymbol{x} - \mathbb{E}(\boldsymbol{x}))(\boldsymbol{y} - \mathbb{E}(\boldsymbol{y}))^T\right) && (56)\\
&= \mathbb{E}\left((\boldsymbol{x} - \mu_x)(A(\boldsymbol{x} - \mu_x) + \boldsymbol{w})^T\right) && (57)\\
&= \mathbb{E}\left((\boldsymbol{x} - \mu_x)(\boldsymbol{x} - \mu_x)^T A^T + (\boldsymbol{x} - \mu_x)\boldsymbol{w}^T\right) && (58)\\
&= \mathbb{E}\left((\boldsymbol{x} - \mu_x)(\boldsymbol{x} - \mu_x)^T\right) A^T + \mathbb{E}\left((\boldsymbol{x} - \mu_x)\boldsymbol{w}^T\right) && (59)\\
&= \Sigma_x A^T && (60)
\end{aligned}$$

Therefore,

$$\left[\begin{array}{c} \boldsymbol{x} \\ \boldsymbol{y} \end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c} \mu_x \\ b + A\mu_x \end{array}\right], \left[\begin{array}{cc} \Sigma_x & \Sigma_x A^T \\ A\Sigma_x & A\Sigma_x A^T + R \end{array}\right]\right) \tag{61}$$

B The traditional approach to SLAM

In this section, we describe the traditional approach to SLAM as exemplified by [Smith and Cheeseman, 1986; Smith *et al.*, 1990; Leonard *et al.*, 1992]. We begin by deriving the Kalman filter and describing two important optimizations in its application to the SLAM problem. Then we describe *gating*, a technique used in maximum likelihood data association. Finally, we address the issue of dealing with non-linear models.

B.1 The Kalman filter

The Kalman Filter is an iterative algorithm for estimating the state of an unobserved system whose dynamics and observations are affine with Gaussian noise. The state variable evolves according to the affine-Gaussian relationship

$$\boldsymbol{s}_t = A\boldsymbol{s}_{t-1} + b + \boldsymbol{w} \tag{62}$$

where $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, R)$ is an independent white noise variable. At each time step, we receive an observation governed by another affine-Gaussian relationship

$$\boldsymbol{z}_t = C\boldsymbol{s}_t + d + \boldsymbol{u} \tag{63}$$

where $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, Q)$ is another independent white noise variable. This model is called the *state space model.* If the model parameters b, A, R, d, C and Q do not vary over time, the model is called *time invariant*.

Assume that $\boldsymbol{s}_{t-1} \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$. Then *time update* consists of computing the distribution of $\boldsymbol{s}_t$, which by (54) and (55), is $\mathcal{N}(\hat{\mu}_t, \hat{\Sigma}_t)$ where

$$\begin{aligned}
\hat{\mu}_t &= A\mu_{t-1} + b && (64)\\
\hat{\Sigma}_t &= A\Sigma_{t-1}A^T + R && (65)
\end{aligned}$$

In general, the time update takes time cubic in the dimension of the state space because of the matrix multiplications in (65).

The *measurement update* consists of computing the distribution $p(\boldsymbol{s}_t \mid z_t)$. To do this, we can form the joint distribution $p(\boldsymbol{s}_t, \boldsymbol{z}_t)$ using (61):

$$\left[\begin{array}{c} \boldsymbol{s}_t \\ \boldsymbol{z}_t \end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c} \hat{\mu}_t \\ C\hat{\mu}_t + d \end{array}\right], \left[\begin{array}{cc} \hat{\Sigma}_t & \hat{\Sigma}_t C^T \\ C\hat{\Sigma}_t & C\hat{\Sigma}_t C^T + Q \end{array}\right]\right) \tag{66}$$

Then we apply (52) to get $p(\boldsymbol{s}_t \mid z_t) = \mathcal{N}(\mu_t, \Sigma_t)$ where

$$\begin{aligned} \mu_t &= \hat{\mu}_t + \hat{\Sigma}_t C^T (C \hat{\Sigma}_t C^T + Q)^{-1} (z_t - d - C \hat{\mu}_t) && (67) \\ \Sigma_t &= \hat{\Sigma}_t - \hat{\Sigma}_t C^T (C \hat{\Sigma}_t C^T + Q)^{-1} C \hat{\Sigma}_t && (68) \end{aligned}$$

If we define

$$K_t = \hat{\Sigma}_t C^T (C \hat{\Sigma}_t C^T + Q)^{-1} \qquad (69)$$

to be the *Kalman gain matrix,* then the measurement update can be rewritten

$$\begin{aligned} \mu_t &= \hat{\mu}_t + K_t (z_t - d - C \hat{\mu}_t) && (70) \\ \Sigma_t &= (I - K_t C) \hat{\Sigma}_t && (71) \end{aligned}$$

Collectively, equations (64), (65), (69), (70), and (71) constitute the Kalman filter. If the dimension of the observations is small, then the time and space complexity of the measurement update is quadratic in the dimension of the state space because of the additive covariance update in (71).

As it became popular in control applications, practitioners discovered the Kalman filter suffered from numerical stability problems [Maybeck, 1979]. The preferred solution to these numerical issues is to use one of a family of *square-root filters*, which represent the belief state's covariance using the Cholesky decomposition of Σ_t (or its inverse). Motion and measurement updates can operate directly on these representations, reducing the susceptibility of the filter to round-off error. However, the measurement update of these filters requires time cubic in the dimension of the state variable. (Some special cases, like when R or Q is diagonal, this complexity can be reduced to quadratic.)

B.2 Application to SLAM

Two useful optimizations are possible when applying the Kalman filter to a SLAM problem. The first reduces the complexity of the time update from cubic to linear in the dimension of the state space, and the second reduces the complexity of the measurement update from quadratic to linear when a landmark is observed for the first time.

Because none of the landmark states change over time, the matrix A in (62) has the form of the identity matrix, except for the block corresponding to the robot state $\boldsymbol{x}_t$. This structure can be leveraged to yield a linear-time update:

Proposition 7. *Let $p(\boldsymbol{m}_t) = \mathcal{N}(\mu_t, \Sigma_t)$ where $\boldsymbol{\ell} = \{\boldsymbol{l}_1, \ldots, \boldsymbol{l}_n\}$ is the set of all landmarks and*

$$\mu_t = \begin{bmatrix} \mu_{x_t} \\ \mu_\ell \end{bmatrix} \text{ and } \Sigma_t = \begin{bmatrix} \Sigma_{x_t x_t} & \Sigma_{x_t \ell} \\ \Sigma_{x_t \ell}^T & \Sigma_{\ell\ell} \end{bmatrix} \qquad (72)$$

Let the robot's dynamics be governed by

$$\boldsymbol{x}_{t+1} = A \boldsymbol{x}_t + b + \boldsymbol{v} \qquad (73)$$

where $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, Q)$ is an independent white noise variable. Then $p(\boldsymbol{m}_{t+1}) = \mathcal{N}(\mu_{t+1}, \Sigma_{t+1})$ where

$$\mu_{t+1} = \begin{bmatrix} A \mu_{x_t} + b \\ \mu_\ell \end{bmatrix} \text{ and } \Sigma_{t+1} = \begin{bmatrix} A \Sigma_{x_t x_t} A^T + Q & A \Sigma_{x_t \ell} \\ \Sigma_{x_t \ell}^T A^T & \Sigma_{\ell\ell} \end{bmatrix} \qquad (74)$$

This update requires time linear in the number of landmarks, because only the elements of the covariance matrix that correspond to $\boldsymbol{x}_{t+1}$ must be computed.

Proof. Write

$$\boldsymbol{x}_{t+1} \;=\; \bar{A}\boldsymbol{m}_t + b + \boldsymbol{v} \tag{75}$$

where $\bar{A} = \left[\begin{array}{cc} A & 0 \end{array}\right]$ so that $\boldsymbol{x}_{t+1}$ is an affine Gaussian function of $\boldsymbol{m}_t$. By (61), the joint distribution of $\boldsymbol{m}_t$ and $\boldsymbol{x}_{t+1}$ is given by

$$p\left(\left[\begin{array}{c} \boldsymbol{m}_t \\ \boldsymbol{x}_{t+1} \end{array}\right]\right) \;=\; \mathcal{N}\left(\left[\begin{array}{c} \mu \\ \bar{A}\mu + b \end{array}\right], \left[\begin{array}{cc} \Sigma & \bar{A}\Sigma \\ \Sigma\bar{A}^T & \bar{A}\Sigma\bar{A}^T + Q \end{array}\right]\right) \tag{76}$$

Expanding $\boldsymbol{m}_t$, μ and Σ we get

$$p\left(\left[\begin{array}{c} \boldsymbol{x}_t \\ \boldsymbol{\ell} \\ \boldsymbol{x}_{t+1} \end{array}\right]\right) \;=\; \mathcal{N}\left(\left[\begin{array}{c} \mu_{x_t} \\ \mu_\ell \\ A\mu_{x_t} + b \end{array}\right], \left[\begin{array}{ccc} \Sigma_{x_t x_t} & \Sigma_{x_t \ell} & A\Sigma_{x_t x_t} \\ \Sigma_{x_t \ell}^T & \Sigma_{\ell\ell} & A\Sigma_{x_t \ell} \\ \Sigma_{x_t x_t}A^T & \Sigma_{x_t \ell}A^T & A\Sigma_{x_t x_t}A^T + Q \end{array}\right]\right) \tag{77}$$

Marginalizing $\boldsymbol{x}_t$ out of this distribution via (47) and (48) and reordering ℓ and $\boldsymbol{x}_{t+1}$ gives the result. □

The second optimization is that when a landmark is observed for the first time, we can avoid the quadratic time complexity of the measurement update.

Proposition 8. *Let $p(\boldsymbol{m}) = \mathcal{N}(\mu, \Sigma)$ where $\boldsymbol{\ell} = \{\boldsymbol{l}_1, \ldots, \boldsymbol{l}_n\}$ is the set of all landmarks and*

$$\mu = \left[\begin{array}{c} \mu_x \\ \mu_\ell \end{array}\right] \text{ and } \Sigma = \left[\begin{array}{cc} \Sigma_{xx} & \Sigma_{x\ell} \\ \Sigma_{x\ell}^T & \Sigma_{\ell\ell} \end{array}\right] \tag{78}$$

Let $\boldsymbol{l}$ be a landmark with a uniform prior (i.e., $p(\boldsymbol{l}) = 1$) which is observed for the first time via the measurement equation (2)*:*

$$\boldsymbol{z} \;=\; E\boldsymbol{x} + F\boldsymbol{l} + g + \boldsymbol{u} \tag{79}$$

where $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, S)$ and F is invertible. Then

$$p\left(\left[\begin{array}{c} \boldsymbol{m} \\ \boldsymbol{l} \end{array}\right] \middle| z\right) \;=\; \mathcal{N}\left(\left[\begin{array}{c} \mu_x \\ \mu_\ell \\ \mu_l \end{array}\right], \left[\begin{array}{ccc} \Sigma_{xx} & \Sigma_{x\ell} & \Sigma_{xl} \\ \Sigma_{x\ell}^T & \Sigma_{\ell\ell} & \Sigma_{\ell l} \\ \Sigma_{xl}^T & \Sigma_{\ell l}^T & \Sigma_{ll} \end{array}\right]\right) \tag{80}$$

where

$$\mu_l \;=\; F^{-1}(z - g - E\mu_x) \tag{81}$$

$$\Sigma_{ll} \;=\; F^{-1}(E\Sigma_{xx}E^T + S)F^{-T} \tag{82}$$

$$\Sigma_{xl} \;=\; -\Sigma_{xx}(F^{-1}E)^T \tag{83}$$

$$\Sigma_{\ell l} \;=\; -\Sigma_{\ell x}(F^{-1}E)^T \tag{84}$$

This update requires time linear in the number of landmarks, because only the elements of the covariance matrix that correspond to the new landmark change.

Proof. By Proposition 2, the marginal density of $\boldsymbol{m}$ does not change, and so μ_x, μ_ℓ, Σ_{xx}, $\Sigma_{x\ell}$, and $\Sigma_{\ell\ell}$ remain the same. Thus, we must only compute μ_l, Σ_{ll}, Σ_{xl}, and $\Sigma_{\ell l}$.

Let us first consider the case where $\boldsymbol{\ell} = \emptyset$, i.e., there are no landmarks in the map. Then the initial joint distribution over $\boldsymbol{x}$ and $\boldsymbol{l}$ (before incorporating the measurement) is

$$\left[\begin{array}{c} \boldsymbol{x} \\ \boldsymbol{l} \end{array}\right] \;\sim\; \mathcal{N}^{-1}\left(\left[\begin{array}{c} \Sigma_{xx}^{-1}\mu_x \\ 0 \end{array}\right], \left[\begin{array}{cc} \Sigma_{xx}^{-1} & 0 \\ 0 & 0 \end{array}\right]\right) \tag{85}$$

where the zero precision of $\boldsymbol{l}$ reflects our uninformative prior. Using (20), we know that incorporating the first measurement z of $\boldsymbol{l}$ results in the following distribution:

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{l} \end{bmatrix} \sim \mathcal{N}^{-1}\left(\underbrace{\begin{bmatrix} \Sigma_{xx}^{-1}\mu_x + E^T S^{-1}(z-g) \\ F^T S^{-1}(z-g) \end{bmatrix}}_{\eta}, \underbrace{\begin{bmatrix} \Sigma_{xx}^{-1} + E^T S^{-1} E & E^T S^{-1} F \\ F^T S^{-1} E & F^T S^{-1} F \end{bmatrix}}_{\Lambda}\right) \tag{86}$$

We now must perform the tedious conversion from the canonical parameterization back to the moment parameterization. Our first step is to invert Λ to obtain Σ:

$$\Sigma = \Lambda^{-1} = \begin{bmatrix} \Sigma_{xx}^{-1} + E^T S^{-1} E & E^T S^{-1} F \\ F^T S^{-1} E & F^T S^{-1} F \end{bmatrix}^{-1} = \begin{bmatrix} \Sigma_{xx} & -\Sigma_{xx} E^T F^{-T} \\ -F^{-1} E \Sigma_{xx} & F^{-1}(E\Sigma_{xx}E^T + S)F^{-T} \end{bmatrix} \tag{87}$$

This follows from the *partitioned inverse formula*; if

$$M = \begin{bmatrix} E & F \\ G & H \end{bmatrix} \tag{88}$$

then

$$M^{-1} = \begin{bmatrix} (M/H)^{-1} & -(M/H)^{-1} F H^{-1} \\ -H^{-1} G (M/H)^{-1} & H^{-1} + H^{-1} G (M/H)^{-1} F H^{-1} \end{bmatrix} \tag{89}$$

where $M/H = E - FH^{-1}G$ is the *Schur complement* of M with respect to H. We now have proved the correctness of (82), and (83).

We compute μ as

$$\begin{aligned}
\mu &= \Lambda^{-1}\eta = \Sigma\eta \\
&= \begin{bmatrix} \Sigma_{xx} & -\Sigma_{xx} E^T F^{-T} \\ -F^{-1} E \Sigma_{xx} & F^{-1}(E\Sigma_{xx}E^T + S)F^{-T} \end{bmatrix} \begin{bmatrix} \Sigma_{xx}^{-1}\mu_x + E^T S^{-1}(z-g) \\ F^T S^{-1}(z-g) \end{bmatrix} \\
&= \begin{bmatrix} \Sigma_{xx}\Sigma_{xx}^{-1}\mu_x + \Sigma_{xx}E^T S^{-1}(z-g) - \Sigma_{xx}E^T F^{-T} F^T S^{-1}(z-g) \\ -F^{-1}E\Sigma_{xx}\Sigma_{xx}^{-1}\mu_x - F^{-1}E\Sigma_{xx}E^T S^{-1}(z-g) + F^{-1}(E\Sigma_{xx}E^T + S)F^{-T}F^T S^{-1}(z-g) \end{bmatrix} \\
&= \begin{bmatrix} \mu_x \\ -F^{-1}E\mu_x - F^{-1}E\Sigma_{xx}E^T S^{-1}(z-g) + F^{-1}(E\Sigma_{xx}E^T + S)S^{-1}(z-g) \end{bmatrix} \\
&= \begin{bmatrix} \mu_x \\ F^{-1}(z - g - E\mu_x) \end{bmatrix}
\end{aligned}$$

Thus, we now have proved the correctness of (81).

To prove (84), we make use of the *iterated expectation theorem* and the *iterated covariance theorem*:

$$\mathbb{E}\left(\boldsymbol{x}\right) = \mathbb{E}\left(\mathbb{E}\left(\boldsymbol{x} \,|\, \boldsymbol{y}\right)\right) \tag{90}$$

$$\text{Cov}\left(\boldsymbol{x}, \boldsymbol{y}\right) = \text{Cov}\left(\mathbb{E}\left(\boldsymbol{x} \,|\, \boldsymbol{y}\right), \boldsymbol{y}\right) \tag{91}$$

Using these facts, we get

$$\begin{aligned}
\Sigma_{\ell l} = \text{Cov}\left(\boldsymbol{\ell}, \boldsymbol{l}\right) &= \text{Cov}\left(\mathbb{E}\left(\boldsymbol{\ell} \,|\, \boldsymbol{l}\right), \boldsymbol{l}\right) && (92) \\
&= \text{Cov}\left(\mathbb{E}\left(\mathbb{E}\left(\boldsymbol{\ell} \,|\, \boldsymbol{x}\right) \,|\, \boldsymbol{l}\right), \boldsymbol{l}\right) && (93) \\
&= \text{Cov}\left(\mathbb{E}\left(\mu_\ell + \Sigma_{\ell x}\Sigma_{xx}^{-1}(\boldsymbol{x} - \mu_x) \,|\, \boldsymbol{l}\right), \boldsymbol{l}\right) && (94) \\
&= \text{Cov}\left(\mu_\ell + \Sigma_{\ell x}\Sigma_{xx}^{-1}(\mu_x + \Sigma_{xl}\Sigma_{ll}^{-1}(\boldsymbol{l} - \mu_l) - \mu_x), \boldsymbol{l}\right) && (95) \\
&= \text{Cov}\left(\mu_\ell + \Sigma_{\ell x}\Sigma_{xx}^{-1}\Sigma_{xl}\Sigma_{ll}^{-1}(\boldsymbol{l} - \mu_l), \boldsymbol{l}\right) && (96) \\
&= \Sigma_{\ell x}\Sigma_{xx}^{-1}\Sigma_{xl}\Sigma_{ll}^{-1}\Sigma_{ll} && (97) \\
&= -\Sigma_{\ell x}(F^{-1}E)^T && (98)
\end{aligned}$$

□

Note that we can only get away with this "trick"—starting with an uninformative prior over the landmark state and then using the measurement to yield an informed posterior—when the measurements are informative enough. When the observation model is not invertible, then the problem is ill-posed; for example, a robot which receives only bearing (and not range) information cannot use this technique. In such cases, the robot must start with an informed prior distribution of the landmark state; however, a similar effect is obtained by choosing this prior to be very weak, e.g., choosing a prior centered at the origin with very large variance.

B.3 Gating

Gating is a statistical test to determine whether a measurement z issued from some landmark $\boldsymbol{l}$. One simple method of doing this is to use *confidence regions*. Let $\boldsymbol{l} \sim \mathcal{N}(\mu, \Sigma)$. If we choose some Mahalanobis threshold $k > 0$, this defines an ellipsoid

$$\mathcal{E} = \{x : (x - \mu)^T \Sigma^{-1} (x - \mu) \leq k\} \tag{99}$$

The points on the boundary of $\mathcal{E}$ all have equal likelihood, because they all have equal Mahalanobis distance.

Because $\boldsymbol{l}$ is distributed according to a multivariate Gaussian (say, of dimension d), the Mahalanobis distance is distributed according to a χ^2 distribution with d degrees of freedom. Thus, we can compute the probability mass of $\mathcal{E}$ by evaluating the cumulative distribution function of the χ^2 distribution (with d degrees of freedom) at our threshold k.

By inverting this computation, we can take a desired confidence value p and compute the Mahalanobis distance k defining the smallest ellipsoid containing p probability.[11] Therefore, for a fixed confidence p and a given measurement z, we can determine if the Mahalanobis distance of z is less than the threshold defined by p; if it is, we assume z originated from $\boldsymbol{l}$; otherwise, we assume it did not. Given our assumptions, this statistical test will fail with probability $1 - p$. For example, if we choose $p = 95\%$, then 5% of the time we will incorrectly assume z did not originate from $\boldsymbol{l}$ when it actually did. It is difficult to quantify how often the complementary sort of error will occur, i.e., how often we will assign z to $\boldsymbol{l}$ when it did not originate from $\boldsymbol{l}$; that would require a detailed model of noise, outliers, and the other landmarks in the region.

B.4 Dealing with nonlinear models

When the dynamic and measurement models are not affine with additive Gaussian noise, there are two standard techniques for approximating them with such functions (see Figure 16). The most commonly used is the *Extended Kalman Filter* (EKF), in which a nonlinear function is approximated by its first-order Taylor expansion about the expectation its input variables. This technique requires only the mean of the input distribution and a method for evaluating the Jacobian of the function at the mean. Unfortunately, the EKF can perform very poorly in cases where the function is not approximately linear in the likely region of its inputs.

The second technique, the *Unscented Kalman Filter* (UKF), trades the difficult problem of approximating a nonlinear function with the simpler problem of approximating the output distribution as a conditional Gaussian [Wan and van der Merwe, 2000]. Once this is accomplished, the conditional Gaussian distribution can be easily translated into a corresponding affine approximation of the nonlinear function. To approximate the output distribution, a small set of weighted points in the input space (called *sigma points*) are strategically selected to characterize the Gaussian input distribution (typically the mean and the endpoints of the axes of

[11] In Matlab, this can be accomplished using `chi2inv`.

the one standard deviation covariance ellipse); then the output distribution is approximated by computing the weighted mean and covariance of the images of the sigma points under the nonlinear function. The UKF is generally more precise than the EKF and has comparable computational complexity, but it requires knowledge of both the mean and covariance of the function input, whereas the EKF requires only the mean. Lerner has shown that the UKF is merely one choice from a family of *numerical integration* techniques that can smoothly trade computational effort for approximation accuracy [Lerner, 2002].

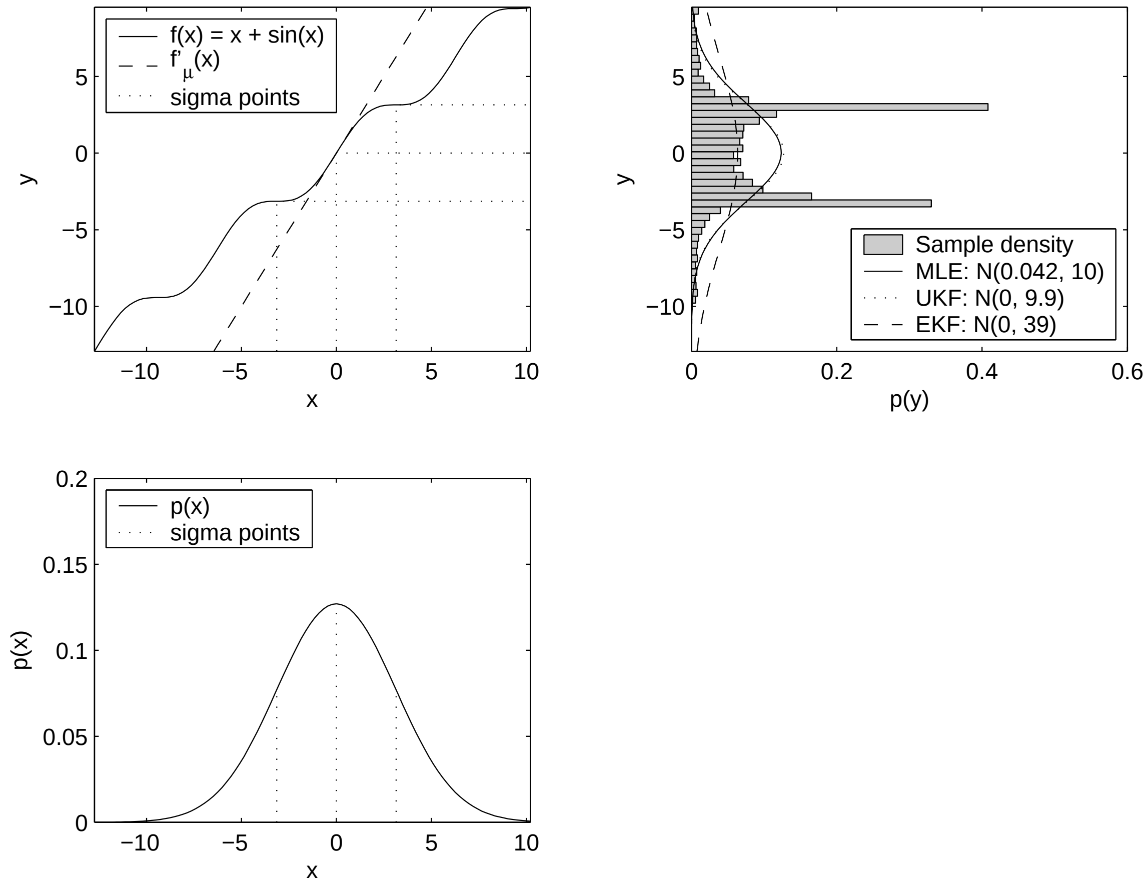


Figure 16: An illustration of the UKF and EKF. The bottom graph gives the density of $\boldsymbol{x} \sim \mathcal{N}(0, \pi^2)$ and shows the sigma points selected by the UKF; the upper left graph shows a nonlinearity $f(x) = x + \sin x$, along with the linearization used by the EKF and the images of sigma points chosen by the UKF. On the right is a histogram of 10,000 samples of $\boldsymbol{x}$ passed through $f(x)$ and the maximum likelihood, EKF, and UKF Gaussian approximations.

Linearization is more complicated in the case of observing a landmark for the first time. In this case, we have an observation z, a distribution over the robot state $p(\boldsymbol{x})$, a white noise distribution $p(\boldsymbol{u})$, and a nonlinear landmark observation function $z = f(x, l, u)$ that takes as input the robot state x, the landmark state l, and a white noise value u. However, because we do not have an informative prior over $\boldsymbol{l}$, the state of the landmark, we cannot apply the linearization techniques above.

We can solve this problem if the landmark measurement model is invertible. Assume that in addition to f, we also have its inverse with respect to l; i.e., we have the function $l = f^{-1}(x, z, u)$ which computes the state of the landmark given the robot state, the measurement, and the noise value. Because z is known, we can linearize f^{-1} using the techniques above to obtain an estimate of $p(\boldsymbol{l} \mid z)$. We can then use this estimate as our prior belief in $\boldsymbol{l}$ to linearize $f(x, l, u)$.

C Proofs

Proof of Proposition 1. (Sketch) $p(\boldsymbol{s} \mid \boldsymbol{q})$ is a conditional Gaussian distribution with mean $s_0 + L\boldsymbol{q}$ and covariance U. Thus, the canonical parameters of its conditional Gaussian distribution are $U^{-1}(s_0 + L\boldsymbol{q})$ and U^{-1}. Writing out the distribution using these canonical parameters and collecting terms reveals that $p(\boldsymbol{s} \mid \boldsymbol{q})$ is proportional to the product $\phi_q(\boldsymbol{q}) \cdot \phi_s(\boldsymbol{s}) \cdot \phi_{qs}(\boldsymbol{q}, \boldsymbol{s})$, which in combination with the chain rule yields (14). (18) is obtained by substituting $\boldsymbol{s} = s$ and eliminating constant terms. □

Proof of Proposition 2. Let $\boldsymbol{\ell} = (\boldsymbol{l}_1, \ldots, \boldsymbol{l}_n)$ be the currently known landmarks; when the new landmark $\boldsymbol{l}$ is first added to the map, the state variable and the parameters are augmented to

$$\overline{\boldsymbol{m}} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\ell} \\ \boldsymbol{l} \end{bmatrix}, \overline{\eta} = \begin{bmatrix} \eta_x \\ \eta_\ell \\ \mathbf{0} \end{bmatrix}, \text{ and } \overline{\Lambda} = \begin{bmatrix} \Lambda_{xx} & \Lambda_{x\ell} & \mathbf{0} \\ \Lambda_{\ell x} & \Lambda_{\ell\ell} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

where the zero precision of $\boldsymbol{l}$ reflects our uninformative prior. Using (20), we know that incorporating the first measurement z of $\boldsymbol{l}$ results in the following updates:

$$\overline{\eta}_z = \begin{bmatrix} \eta_x + E^T S^{-1}(z - g) \\ \eta_\ell \\ F^T S^{-1}(z - g) \end{bmatrix} \tag{100}$$

$$\overline{\Lambda}_z = \begin{bmatrix} \Lambda_{xx} + E^T S^{-1} E & \Lambda_{x\ell} & E^T S^{-1} F \\ \Lambda_{\ell x} & \Lambda_{\ell\ell} & \mathbf{0} \\ F^T S^{-1} E & \mathbf{0} & F^T S^{-1} F \end{bmatrix} \tag{101}$$

Applying (50) to marginalize out the new landmark $\boldsymbol{l}$, we find

$$\overline{\eta}_z^{\mathrm{m}} = \begin{bmatrix} \eta_x + E^T S^{-1}(z - g) - E^T S^{-1} F (F^T S^{-1} F)^{-1} F^T S^{-1}(z - g) \\ \eta_\ell \end{bmatrix} \tag{102}$$

$$= \begin{bmatrix} \eta_x + E^T S^{-1}(z - g) - E^T S^{-1} F (F^{-1} R F^{-T}) F^T S^{-1}(z - g) \\ \eta_\ell \end{bmatrix} \tag{103}$$

$$= \begin{bmatrix} \eta_x + E^T S^{-1}(z - g) - E^T S^{-1}(z - g) \\ \eta_\ell \end{bmatrix} = \begin{bmatrix} \eta_x \\ \eta_l \end{bmatrix} \tag{104}$$

and

$$\overline{\Lambda}_z^{\mathrm{m}} = \begin{bmatrix} \Lambda_{xx} + E^T S^{-1} E - E^T S^{-1} F (F^T S^{-1} F)^{-1} F^T S^{-1} E & \Lambda_{x\ell} \\ \Lambda_{\ell x} & \Lambda_{\ell\ell} \end{bmatrix} \tag{105}$$

$$= \begin{bmatrix} \Lambda_{xx} + E^T S^{-1} E - E^T S^{-1} F (F^{-1} R F^{-T}) F^T S^{-1} E & \Lambda_{x\ell} \\ \Lambda_{\ell x} & \Lambda_{\ell\ell} \end{bmatrix} \tag{106}$$

$$= \begin{bmatrix} \Lambda_{xx} + E^T S^{-1} E - E^T S^{-1} E & \Lambda_{x\ell} \\ \Lambda_{\ell x} & \Lambda_{\ell\ell} \end{bmatrix} = \begin{bmatrix} \Lambda_{xx} & \Lambda_{x\ell} \\ \Lambda_{\ell x} & \Lambda_{\ell\ell} \end{bmatrix} \tag{107}$$

which parameterized the distribution before the new measurement was incorporated; thus, the measurement update for the new landmark does not affect their marginal distribution. □

Proof of Proposition 3. (Sketch.) Variable contraction preserves the singly connected property, because it introduces no new edges into the graph. (While it may empty a separator, this does not remove the corresponding edge. Merging a nonmaximal cluster into its subsuming neighbor also does not change violate the singly connected property because they must be neighbors in the junction tree.) Variable contraction preserves the running intersection property by construction, since it contracts a variable $\boldsymbol{x}$ only from a leaf of its induced subtree; more concretely, any path between two clusters containing $\boldsymbol{x}$ (except C, of course) is preserved. Finally, marginalizing $\boldsymbol{x}$ out of ϕ_C and ϕ_S does not change the marginals of the other variables under ϕ_C and ϕ_S, and so consistency is preserved. □

Proof of Proposition 4. We must first prove that the distribution of $\mathcal{T}'$ has the conditional independence structure of $\mathcal{T}$, plus $\boldsymbol{x} \perp\!\!\!\perp C - S \,|\, S \setminus \boldsymbol{x}$. But this is a simple consequence of $C_i \perp\!\!\!\perp C_j \,|\, S_{ij}$, that two adjacent clusters are conditionally independent given their separator. Let C' be the cluster joined to C via S, and recall $\boldsymbol{x} \in C'$. In $\mathcal{T}$ we had $C' \perp\!\!\!\perp C \,|\, S$, or equivalently,

$$C' - C \perp\!\!\!\perp C - S \,|\, S \tag{108}$$

because $S = C' \cap C$. In $\mathcal{T}'$ this becomes $C' - (C \setminus \boldsymbol{x}) \perp\!\!\!\perp C - S \,|\, S \setminus \boldsymbol{x}$, because $\boldsymbol{x}$ has been removed from C and S. Comparing this to (108), we see there is now one additional conditional independence: $\boldsymbol{x} \perp\!\!\!\perp C - S \,|\, S \setminus \boldsymbol{x}$.

We now prove that of all distributions with the same conditional independence structure, the distribution $\phi_{\mathcal{T}'}$ has maximum likelihood (under $\phi_{\mathcal{T}}$). Because the probability model of $\mathcal{T}'$ is decomposable, we know it is parameterized in terms of the marginals of its clusters and separators [Whittaker, 1989]. It immediately follows that choosing $\phi_{\hat{C}}$ to be $\int \phi_C d\boldsymbol{x}$ and $\phi_{\hat{S}}$ to be $\int \phi_S d\boldsymbol{x}$ (and leaving the other cluster potentials alone) is a maximum likelihood projection, since those are the maximum-likelihood marginals as obtained from $\phi_{\mathcal{T}}$. □

Proof of Proposition 5.

$$\begin{aligned}
D(p \,||\, q) &= \int p(u) \log \frac{p(U)}{q(U)} du \\
&= \int p(u) \log \frac{\prod_{C' \in \mathcal{C}} p(C')/q(C')}{\prod_{S' \in \mathcal{S}} p(S')/q(S')} du \\
&= \int p(u) \log \frac{p(C)q(S)}{q(C)p(S)} du \\
&= \int p(C) \log \frac{p(C)p(S \setminus \boldsymbol{x})}{p(C \setminus \boldsymbol{x})p(S)} dC \\
&= \int p(C) \log \frac{p(C \,|\, S)}{p(C \setminus \boldsymbol{x} \,|\, S \setminus \boldsymbol{x})} dC \\
&= H(C \setminus \boldsymbol{x} \,|\, S \setminus \boldsymbol{x}) - H(C \,|\, S) \\
&= H(C - S \,|\, S \setminus \boldsymbol{x}) - H(C - S \,|\, S) \\
&= H(C - S \,|\, S \setminus \boldsymbol{x}) - H(C - S \,|\, (S \setminus \boldsymbol{x}) \cup \{\boldsymbol{x}\}) \\
&= I(\boldsymbol{x}; C - S \,|\, S \setminus \boldsymbol{x})
\end{aligned}$$

□

Proof of Proposition 6. Using the identity $I(\boldsymbol{u}; \boldsymbol{v}) = H(\boldsymbol{u}) - H(\boldsymbol{u} \,|\, \boldsymbol{v})$, we get that

$$I(\boldsymbol{x}; C - S \,|\, S \setminus \boldsymbol{x}) = H(\boldsymbol{x} \,|\, S \setminus \boldsymbol{x}) - H(\boldsymbol{x} \,|\, (C - S) \cup (S \setminus \boldsymbol{x})) \tag{109}$$

$$= H(\boldsymbol{x} \,|\, S \setminus \boldsymbol{x}) - H(\boldsymbol{x} \,|\, C \setminus \boldsymbol{x}) \tag{110}$$

Now, for a k-dimensional Gaussian random variable $\boldsymbol{u}$, the differential entropy is (see [Cover and Thomas, 1991, Theorem 9.4.1])

$$\begin{aligned} H(\boldsymbol{u}) &= \frac{1}{2}\log((2\pi e)^k \det \mathsf{Cov}\,(\boldsymbol{u})) \\ &= \frac{k}{2}\log(2\pi e) - \frac{1}{2}\log\det \mathsf{Cov}\,(\boldsymbol{u})^{-1} \end{aligned}$$

Thus, we need only the precision matrices of the conditional distributions $p(\boldsymbol{x} \,|\, S \setminus \boldsymbol{x})$ and $p(\boldsymbol{x} \,|\, C \setminus \boldsymbol{x})$. Using (110) and (51) we get

$$\begin{aligned} I(\boldsymbol{x}; C - S \,|\, S \setminus \boldsymbol{x}) &= -\frac{1}{2}\log\det \mathsf{Cov}\,(\boldsymbol{u})^{-1} + \frac{1}{2}\log\det \mathsf{Cov}\,(\boldsymbol{u})^{-1} & (111) \\ &= \frac{1}{2}\left(\log\det \Lambda^C_{xx} - \log\det \Lambda^S_{xx}\right) & (112) \end{aligned}$$

where $\Lambda^C = \mathsf{Cov}\,(C)^{-1}$ and $\Lambda^S = \mathsf{Cov}\,(S)^{-1}$. □

References

[Bach and Jordan, 2002] F. R. Bach and M. I. Jordan. Thin junction trees. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 569–576, Cambridge, MA, 2002. MIT Press.

[Bosse *et al.*, 2002] Michael Bosse, Paul M. Newman, John J. Leonard, and Seth Teller. An atlas framework for scalable mapping. MIT Marine Robotics Laboratory Technical memorandum 2002-04, Massachusetts Institute of Technology, 2002.

[Boyen and Koller, 1998] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proceedings of the 14th Annual Conference on Uncertainty in AI (UAI)*, pages 33–42, Madison, Wisconsin, July 1998.

[Boyen and Koller, 1999] Xavier Boyen and Daphne Koller. Exploiting the architecture of dynamic systems. In *AAAI/IAAI*, pages 313–320, 1999.

[Cover and Thomas, 1991] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, NY, USA, 1991.

[Cowell *et al.*, 1999] R. Cowell, P. Dawid, S. Lauritzen, and D. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, New York, NY, 1999.

[Doucet *et al.*, 2000] Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 176–183, 2000.

[Draper, 1995] Denise L. Draper. Clustering without (thinking about) triangulation. In *Uncertainty in Artificial Intelligence: Proceedings of the Eleventh Conference (UAI-1995)*, pages 125–133, San Francisco, CA, 1995. Morgan Kaufmann Publishers.

[Geiger *et al.*, 1990] D. Geiger, T. S. Verma, and J. Pearl. Identifying independence in bayesian networks. *Networks*, 20:507–534, 1990.

[Guivant and Nebot, 2000] J. E. Guivant and E. M. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotic and Automation*, 17(3):242–257, 2000.

[Hébert *et al.*, 1996] P. Hébert, S. Betgé-Brezetz, and R. Chatila. Probabilistic map learning: Necessity and difficulties. In Leo Dorst, Michiel van Lambalgen, and Frans Voorbraak, editors, *Reasoning with Uncertainty in Robotics*, volume 1093 of *Lecture Notes in Computer Science*. Springer, 1996.

[Jordan, 2003] Michael I. Jordan. *Graphical Models*. 2003. Forthcoming.

[Kjærulff, 1993] Uffe Kjærulff. Approximation of bayesian networks through edge removals. Dept. of Mathematics and Computer Science Research Report IR-93-2007, Aalborg University, 1993.

[Lauritzen and Jensen, 2001] S. Lauritzen and F. Jensen. Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, 11:191–203, 2001.

[Leonard and Feder, 2000] J. J. Leonard and H. J. S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *Proceedings of the Ninth International Symposium on Robotics Research*, pages 169–176. Springer-Verlag, 2000.

[Leonard *et al.*, 1992] John. J. Leonard, Ingemar. J. Cox, and Hugh F. Durrant-Whyte. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):286–298, August 1992.

[Lerner, 2002] Uri Lerner. *Hybrid Bayesian Networks for Reasoning About Complex Systems*. PhD thesis, Stanford University, October 2002.

[Liu and Thrun, 2002] Yufeng Liu and Sebastian Thrun. Results for outdoor-SLAM using sparse extended information filters. Submitted for publication, 2002.

[Maybeck, 1979] P.S̃. Maybeck. *Stochastic Models, Estimation and Control*, volume 1. Academic, New York, 1979.

[Montemerlo and Thrun, 2002] Mike Montemerlo and Sebastian Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. Submitted for publication, 2002.

[Montemerlo *et al.*, 2002] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 593–598. AAAI Press, 2002.

[Russell and Norvig, 2002] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2002.

[Shachter *et al.*, 1994] R. D. Shachter, S. K. Andersen, and P. Szolovits. Global conditioning for probabilistic inference in belief networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 514–522, San Francisco, 1994. Morgan Kaufmann.

[Smith and Cheeseman, 1986] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–58, 1986.

[Smith *et al.*, 1990] Randall C. Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990.

[Speed and Kiiveri, 1986] T. P. Speed and H. T. Kiiveri. Gaussian markov distributions over finite graphs. *Annals of Statistics*, 14(1):138–150, March 1986.

[Tardós *et al.*, 2001] Juan D. Tardós, José Neira, Paul M. Newman, and John J. Leonard. Robust mapping and localization in indoor environments using sonar data. MIT Marine Robotics Laboratory Technical Memorandum 2001-04, Massachusetts Institute of Technology, 2001.

[Thrun *et al.*, 2001] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1–2):99–141, 2001.

[Thrun *et al.*, 2002] Sebastian Thrun, Daphne Koller, Zoubin Ghahmarani, Hugh Durrant-Whyte, and Andrew Ng. Simultaneous localization and mapping with sparse extended information filters: Theory and initial results. Computer Science Technical Report CMU-CS-02-112, Carnegie Mellon University, Pittsburgh, PA 15213, September 2002.

[Thrun, 2002] Sebastian Thrun. Robotic mapping: A survey. Technical Report CMU-CS-02-111, Carnegie Mellon University, Pittsburgh, PA 15213, February 2002.

[Umeyama, 1991] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, April 1991.

[Wan and van der Merwe, 2000] E. Wan and R. van der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of IEEE Symposium 2000 (AS-SPCC)*, 2000.

[Whittaker, 1989] Joe Whittaker. *Graphical Models in Applied Mathematical Multivariate Statistics*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1989.