Square Root SAM

Simultaneous Localization and Mapping via Square Root Information Smoothing

Frank Dellaert and Michael Kaess
Center for Robotics and Intelligent Machines, College of Computing
Georgia Institute of Technology, Atlanta, GA 30332-0280

To appear in the Intl. Journal of Robotics Research

Abstract

Solving the SLAM problem is one way to enable a robot to explore, map, and navigate in a previously unknown environment. We investigate smoothing approaches as a viable alternative to extended Kalman filter-based solutions to the problem. In particular, we look at approaches that factorize either the associated information matrix or the measurement Jacobian into square root form. Such techniques have several significant advantages over the EKF: they are faster yet exact, they can be used in either batch or incremental mode, are better equipped to deal with non-linear process and measurement models, and yield the entire robot trajectory, at lower cost for a large class of SLAM problems. In addition, in an indirect but dramatic way, column ordering heuristics automatically exploit the locality inherent in the geographic nature of the SLAM problem. In this paper we present the theory underlying these methods, along with an interpretation of factorization in terms of the graphical model associated with the SLAM problem. We present both simulation results and actual SLAM experiments in large-scale environments that underscore the potential of these methods as an alternative to EKF-based approaches.

1 Introduction

The problem of simultaneous localization and mapping (SLAM) [69, 51, 76] has received considerable attention in mobile robotics as it is one way to enable a robot to explore and navigate previously unknown environments. In addition, in many applications the map of the environment itself is the artifact of interest, e.g. in urban reconstruction, search-and-rescue operations, battlefield reconnaissance etc. As such, it is one of the core competencies of autonomous robots [77].

We will primarily be concerned with landmark-based SLAM, for which the earliest and most popular methods are based on the extended Kalman filter (EKF) [70, 62, 61, 2, 71, 53, 51]. The EKF recursively estimates a Gaussian density over the current pose of the robot and the position of all landmarks (the map). However, it is well known that the computational complexity of the EKF becomes intractable fairly quickly, and hence a large number of efforts have focused on

modifying and extending the filtering approach to cope with larger-scale environments [64, 17, 21, 8, 45, 49, 38, 65, 50, 5, 78, 39, 75, 80, 67]. However, *filtering* itself has been shown to be inconsistent when applied to the inherently non-linear SLAM problem [44], i.e., even the average taken over a large number of experiments diverges from the true solution. Since this is mainly due to linearization choices that cannot be undone in a filtering framework, there has recently been considerable interest in the smoothing version of the SLAM problem.

Smoothing and Mapping or SAM

A *smoothing* approach to SLAM involves not just the most current robot location, but the entire robot trajectory up to the current time. A number of authors considered the problem of smoothing the robot trajectory only [9, 56, 57, 40, 46, 24], which is particularly suited to sensors such as laser-range finders that easily yield pairwise constraints between nearby robot poses. More generally, one can consider the *full SLAM problem* [77], i.e., the problem of optimally estimating the entire set of sensor poses along with the parameters of all features in the environment. In fact, this problem has a long history in surveying [36], photogrammetry [7, 37, 68, 10], where it is known as "bundle adjustment", and computer vision [25, 72, 73, 79, 41], where it is referred to as "structure from motion". Especially in the last five years there has been a flurry of work where these ideas were applied in the context of SLAM [16, 22, 23, 43, 30, 31, 26, 27, 28, 29, 77].

In this paper we show how smoothing can be a very fast alternative to filtering-based methods, and that in many cases keeping the trajectory around helps rather than hurts performance. In particular, the optimization problem associated with full SLAM can be concisely stated in terms of sparse linear algebra, which is traditionally concerned with the solution of large least-squares problems [35]. In this framework, we investigate factorizing either the information matrix $\mathcal{I}$ or the measurement Jacobian A into square root form, as applied to the problem of simultaneous *smoothing and mapping* (SAM). Because they are based on matrix square roots, we will refer to this family of approaches as *square root SAM*, or $\sqrt{\text{SAM}}$ for short, first introduced in [18]. We propose that $\sqrt{\text{SAM}}$ is a fundamentally better approach to the problem of SLAM than the EKF, based on the realization that,

- in contrast to the extended Kalman filter covariance or information matrix, which *both* become fully dense over time [65, 78], the information matrix $\mathcal{I}$ associated with smoothing is and stays sparse;
- in typical mapping scenarios (ie. not repeatedly traversing a small environment) this matrix $\mathcal{I}$ or, alternatively, the measurement Jacobian A, are much more compact representations of the map covariance structure
- $\mathcal{I}$ or A, both sparse, can be factorized efficiently using sparse Cholesky or QR factorization, respectively, yielding a square root information matrix R that can be used to immediately obtain the optimal robot trajectory and map;

Factoring the information matrix is known in the sequential estimation literature as square root information filtering (SRIF), and was developed in 1969 for use in JPL's Mariner 10 missions to Venus (as recounted by [3]). The use of square roots of either the covariance or information matrix results in more accurate and stable algorithms, and, quoting Maybeck [59] "a number of

practitioners have argued, with considerable logic, that square root filters should *always* be adopted in preference to the standard Kalman filter recursion". Maybeck briefly discusses the SRIF in a chapter on square root filtering, and it and other square root type algorithms are the subject of a book by Bierman [3]. However, as far as this can be judged by the small number of references in the literature, the SRIF and the square root information *smoother* (SRIS) are not often used.

Sparse Linear Algebra, Graph Theory, and Sparse $\sqrt{\text{SAM}}$

The key to performance is to capitalize on the large body of work in sparse linear algebra and fully exploit the sparseness of the matrices associated with the smoothing SLAM problem. The most dramatic improvement in performance comes from choosing a good *variable ordering* when factorizing a matrix. To understand this fact better, one needs to examine the close relationship between SLAM-like problems, linear algebra, and graph theory. Graph theory has been central to sparse linear algebra for more than 30 years [32]. For example, while most often used as a "black box" algorithm, QR factorization is in fact an elegant computation on a graph. This is a recurring theme: the state of the art in linear algebra is a blend of numerical methods and advanced graph theory, sharing many characteristics with inference algorithms typically found in the graphical models literature [11]. In particular, the graph-theoretic algorithm underlying the solution of sparse least-squares is *variable elimination*, in which each variable (such as a robot pose or a landmark position) is expressed in terms of other variables. The order in which the variables are eliminated has a large impact on the running time of matrix factorization algorithms such as QR and Cholesky factorization. Finding an optimal ordering is an NP-complete problem, but there are several ordering heuristics and approximate algorithms that perform well on general problems [1, 42] and are built into programs like MATLAB [58].

While general-purpose ordering methods drastically improve performance, yet another order-of-magnitude improvement can be obtained by exploiting the specific graphical structure of the SLAM problem when ordering the variables. Looking at SLAM in terms of graphs has a rich history in itself [6, 74, 33, 34] and has especially lead to several novel and exciting developments in the past few years [63, 60, 30, 65, 26, 31, 78, 27, 77]. Below we more closely examine the tight connection between the graphical model view and the sparse linear algebra formulation of SLAM. It is well known that the information matrix $\mathcal{I}$ is associated with the undirected graph connecting robot poses and landmarks (see e.g [78]). Less readily appreciated is the fact that the measurement Jacobian A is the matrix of the *factor graph* associated with SLAM. In addition, the square root information matrix R, the result of factorizing either $\mathcal{I}$ or A, is essentially in correspondence with a junction tree, known from inference in graphical models [11] and also recently applied in SLAM [65]. Exploiting domain knowledge to obtain good orderings is also a trend in linear algebra (e.g. [15]), and we believe that even more efficient algorithms can be developed by viewing the problem as one of computation on a graph.

2 SLAM and its Graphs

SLAM refers to the problem of localizing a robot while simultaneously mapping its environment, illustrated by the example of Figure 1. In this section we introduce the SLAM problem, the notation we use, and show how the three main graphical model representations known in the literature each

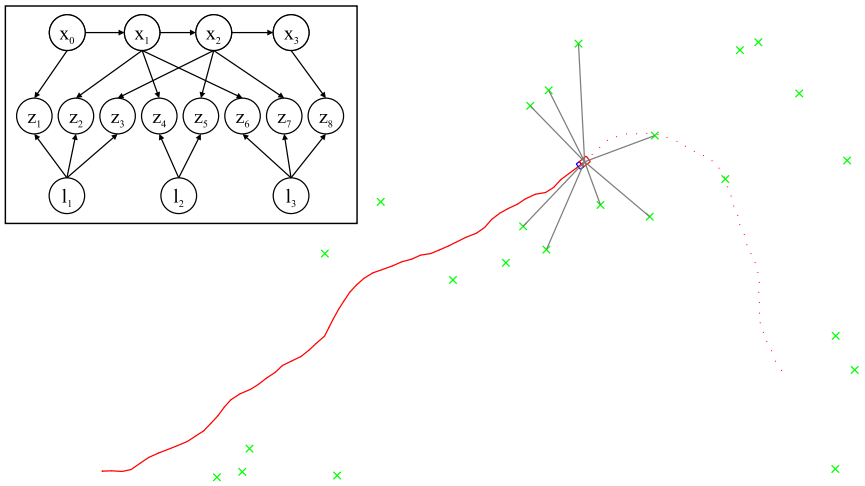


Figure 1: Bayesian belief network of a small SLAM problem (top left) and a more complex synthetic environment. The synthetic environment contains 24 landmarks with a simulated robot taking 422 bearing & range measurements along a trajectory of 95 poses. The objective of SLAM is to localize the robot while simultaneously building a map of the environment. In addition to this, *Full SLAM* seeks to recover the entire robot trajectory. In the Bayesian belief network representation, the state x of the robot is governed by a Markov chain, on top, while the environment of the robot is represented at the bottom by a set of landmarks l. The measurements z, in the middle layer, are governed both by the state of the robot and the parameters of the landmark measured.

yield a unique view on the SLAM problem that emphasizes a certain aspect of the problem. Later in the paper the connection is made between these graphs and their (sparse) matrix equivalents. Below we assume familiarity with EKF-based approaches to SLAM [71, 51, 8, 21]. We do not re-derive the extended Kalman filter. Rather, in Section 3 we immediately take a smoothing approach, in which both the map and the robot trajectory are recovered.

SLAM As a Belief Net

Following the trend set by FastSLAM and others [63, 60, 65, 30, 26, 77], we formulate the problem by referring to a *belief net* representation. A belief net is a directed acyclic graph that encodes the conditional independence structure of a set of variables, where each variable only directly depends on its predecessors in the graph. The model we adopt is shown in the top left of Figure 1. Here we denote the state of the robot at the i^{th} time step by x_i, with $i \in 0..M$, a landmark by l_j, with $j \in 1..N$, and a measurement by z_k, with $k \in 1..K$. The joint probability model corresponding to this network is

$$P(X, L, Z) = P(x_0) \prod_{i=1}^{M} P(x_i|x_{i-1}, u_i) \prod_{k=1}^{K} P(z_k|x_{i_k}, l_{j_k}) \tag{1}$$

where $P(x_0)$ is a prior on the initial state, $P(x_i|x_{i-1}, u_i)$ is the *motion model*, parameterized by a control input u_i, and $P(z|x, l)$ is the *landmark measurement model*. The above assumes a uniform prior over the landmarks l. Furthermore, it assumes that the data-association problem has been solved, i.e., that the indices i_k and j_k corresponding to each measurement z_k are known.

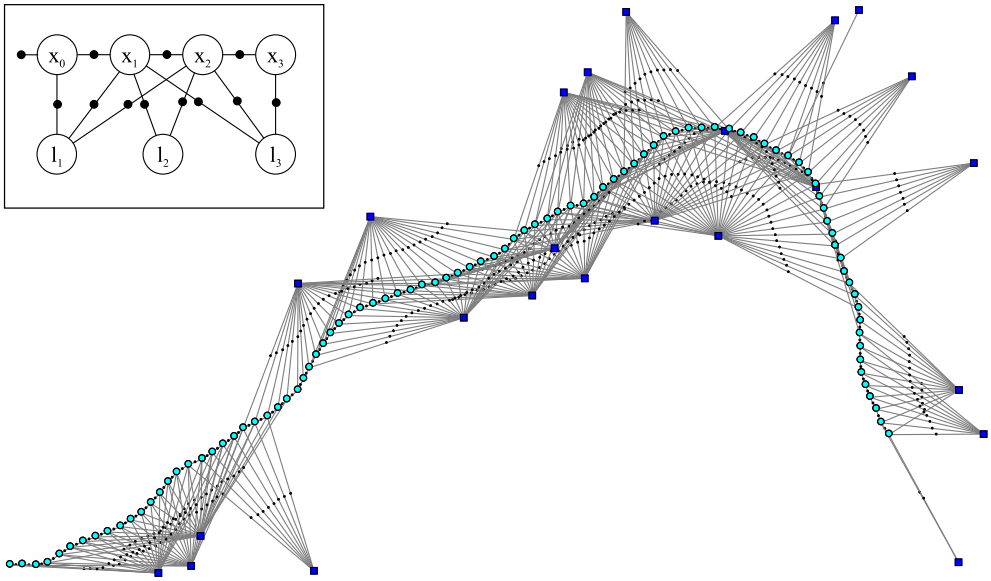


Figure 2: Factor graph representation of the Full SLAM problem for both the simple example and the synthetic environment in Figure 1. The unknown poses and landmarks correspond to the circular and square variable nodes, respectively, while each measurement corresponds to a factor node (filled black circles).

As is standard in the SLAM literature [71, 51, 8, 21], we assume Gaussian process and measurement models [59], defined by

$$x_i = f_i(x_{i-1}, u_i) + w_i \quad \Leftrightarrow \quad P(x_i|x_{i-1}, u_i) \propto \exp -\frac{1}{2}\|f_i(x_{i-1}, u_i) - x_i\|^2_{\Lambda_i} \tag{2}$$

where $f_i(.)$ is a process model, and w_i is normally distributed zero-mean process noise with covariance matrix Λ_i, and

$$z_k = h_k(x_{i_k}, l_{j_k}) + v_k \quad \Leftrightarrow \quad P(z_k|x_{i_k}, l_{j_k}) \propto \exp -\frac{1}{2}\|h_k(x_{i_k}, l_{j_k}) - z_k\|^2_{\Sigma_k} \tag{3}$$

where $h_k(.)$ is a measurement equation, and v_k is normally distributed zero-mean measurement noise with covariance Σ_k. Above $\|e\|^2_\Sigma \triangleq e^T\Sigma^{-1}e$ is defined as the squared Mahalanobis distance given a covariance matrix Σ. The equations above model the robot's behavior in response to control input, and its sensors, respectively.

SLAM as a Factor Graph

While belief nets are a very natural representation to think about the generative aspect of the SLAM problem, *factor graphs* have a much tighter connection with the underlying optimization problem. As the measurements z_k in Figure 1 are known (evidence, in graphical model parlance), we are free to eliminate them as variables. Instead we consider them as parameters of the joint probability factors over the *actual* unknowns. This naturally leads to the well known factor graph representation, a class of bipartite graphical models that can be used to represent such factored

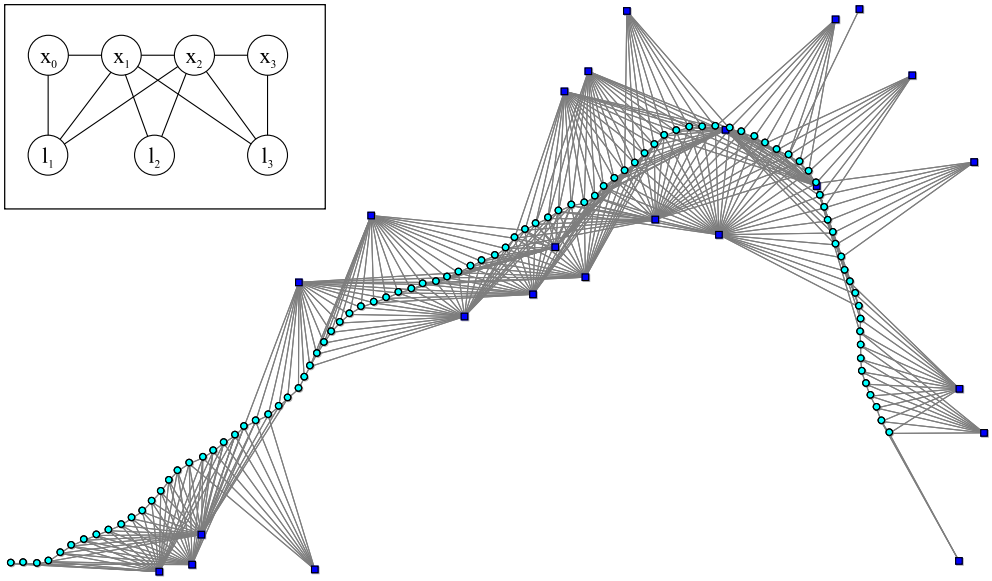


Figure 3: The undirected Markov random fields associated with the SLAM problems from Figure 1.

densities [48]. In a factor graph there are nodes for unknowns and nodes for the probability factors defined on them, and the graph structure expresses which unknowns are involved in each factor. The factor graphs for the examples from Figure 1 are shown in Figure 2. As can be seen, there are factor nodes for both landmark measurements z_k and odometry links u_i.

In the SLAM problem we (typically) consider only single and pairwise cliques, leading to the following factor graph expression of the joint density over a set of unknowns Θ:

$$P(\Theta) \propto \prod_i \phi_i(\theta_i) \prod_{\{i,j\},i<j} \psi_{ij}(\theta_i, \theta_j) \tag{4}$$

Typically the potentials $\phi_i(\theta_i)$ encode a prior or a single measurement constraint at an unknown $\theta_i \in \Theta$, whereas the pairwise potentials $\psi_{ij}(\theta_i, \theta_j)$ relate to measurements or constraints that involve the relationship between two unknowns θ_i and θ_j. Note that the second product is over pairwise cliques $\{i, j\}$, counted once. The equivalence between equations (1) and (4) can be readily established by taking

$$\phi_0(x_0) \propto P(x_0)$$
$$\psi_{(i-1)i}(x_{i-1}, x_i) \propto P(x_i|x_{i-1}, u_i)$$
$$\psi_{i_k j_k}(x_{i_k}, l_{j_k}) \propto P(z_k|x_{i_k}, l_{j_k})$$

SLAM as a Markov Random Field

Finally, a third way to express the SLAM problem in terms of graphical models is via *Markov random fields*, in which the factor nodes themselves are eliminated. The graph of an MRF is undirected and does not have factor nodes: its adjacency structure indicates which variables are linked by a common factor (measurement or constraint). At this level of abstraction, the form (4)

corresponds exactly to the expression for a pairwise Markov random field [82], hence MRFs and factor graphs are equivalent representations here. The MRFs for the examples from Figure 1 are shown in Figure 4. Note that it looks very similar to Figure 2, but an MRF is a fundamentally different structure from the factor graph (undirected vs. bipartite).

3 SAM as a Least Squares Problem

While the previous section was concerned with modeling, we now discuss inference, i.e., obtaining an optimal estimate for the set of unknowns given all measurements available to us. We are concerned with *smoothing* rather than filtering, i.e., we would like to recover the maximum a posteriori (MAP) estimate for the entire trajectory $X \triangleq \{x_i\}$ and the map $L \triangleq \{l_j\}$, given the measurements $Z \triangleq \{z_k\}$ and control inputs $U \triangleq \{u_i\}$. Let us collect all unknowns in X and L in the vector $\Theta \triangleq (X, L)$. Under the assumptions made above, we obtain the maximum a posteriori (MAP) estimate by maximizing the joint probability $P(X, L, Z)$ from Equation 1,

$$\begin{aligned} \Theta^* \triangleq \underset{\Theta}{\text{argmax}} \; P(X, L|Z) &= \underset{\Theta}{\text{argmax}} \; P(X, L, Z) \\ &= \underset{\Theta}{\text{argmin}} \; -\log P(X, L, Z) \end{aligned}$$

which leads us via (2) and (3) to the following non-linear least-squares problem:

$$\Theta^* \triangleq \underset{\Theta}{\text{argmin}} \left\{ \sum_{i=1}^{M} \|f_i(x_{i-1}, u_i) - x_i\|^2_{\Lambda_i} + \sum_{k=1}^{K} \|h_k(x_{i_k}, l_{j_k}) - z_k\|^2_{\Sigma_k} \right\} \tag{5}$$

Regarding the prior $P(x_0)$, we will assume that x_0 is given and hence it is treated as a constant below. This considerably simplifies the equations in the rest of this document. This is what is often done in practice: the origin of the coordinate system is arbitrary, and we can then just as well fix x_0 at the origin. The exposition is easily adapted to the case where this assumption is invalid.

In practice one always considers a linearized version of problem (5). If the process models f_i and measurement equations h_k are non-linear and a good linearization point is not available, non-linear optimization methods such as Gauss-Newton iterations or the Levenberg-Marquardt algorithm will solve a succession of linear approximations to (5) in order to approach the minimum [20]. This is similar to the extended Kalman filter approach to SLAM as pioneered by [70, 71, 52], but allows for iterating multiple times to convergence while controlling in which region one is willing to trust the linear assumption (hence, these methods are often called region-trust methods).

We now linearize all terms in the non-linear least-squares objective function (5). In what follows, we will assume that either a good linearization point is available or that we are working on one iteration of a non-linear optimization method. In either case, we can linearize the process terms in (5) as follows:

$$f_i(x_{i-1}, u_i) - x_i \approx \left\{ f_i(x^0_{i-1}, u_i) + F_i^{i-1} \delta x_{i-1} \right\} - \left\{ x_i^0 + \delta x_i \right\} = \left\{ F_i^{i-1} \delta x_{i-1} - \delta x_i \right\} - a_i \tag{6}$$

where F_i^{i-1} is the Jacobian of $f_i(.)$ at the linearization point x^0_{i-1}, defined by

$$F_i^{i-1} \triangleq \left. \frac{\partial f_i(x_{i-1}, u_i)}{\partial x_{i-1}} \right|_{x^0_{i-1}}$$

and $a_i \triangleq x_i^0 - f_i(x_{i-1}^0, u_i)$ is the odometry prediction error (note that u_i here is given and hence constant). The linearized measurement terms in (5) are obtained similarly,

$$h_k(x_{i_k}, l_{j_k}) - z_k \approx \left\{h_k(x_{i_k}^0, l_{j_k}^0) + H_k^{i_k}\delta x_{i_k} + J_k^{j_k}\delta l_{j_k}\right\} - z_k = \left\{H_k^{i_k}\delta x_{i_k} + J_k^{j_k}\delta l_{j_k}\right\} - c_k \quad (7)$$

where $H_k^{i_k}$ and $J_k^{j_k}$ are respectively the Jacobians of $h_k(.)$ with respect to a change in $x_{i_k}$and l_{j_k}, evaluated at the linearization point $(x_{i_k}^0, l_{j_k}^0)$:

$$H_k^{i_k} \triangleq \left.\frac{\partial h_k(x_{i_k}, l_{j_k})}{\partial x_{i_k}}\right|_{(x_{i_k}^0, l_{j_k}^0)} \qquad J_k^{j_k} \triangleq \left.\frac{\partial h_k(x_{i_k}, l_{j_k})}{\partial l_{j_k}}\right|_{(x_{i_k}^0, l_{j_k}^0)}$$

and $c_k \triangleq z_k - h_k(x_{i_k}^0, l_{j_k}^0)$ is the measurement prediction error.

Using the linearized process and measurement models (6) and (7), respectively, (5) begets

$$\delta^* = \underset{\delta}{\text{argmin}} \left\{\sum_{i=1}^{M} \|F_i^{i-1}\delta x_{i-1} + G_i^i \delta x_i - a_i\|_{\Lambda_i}^2 + \sum_{k=1}^{K} \|H_k^{i_k}\delta x_{i_k} + J_k^{j_k}\delta l_{j_k} - c_k\|_{\Sigma_k}^2\right\} \quad (8)$$

i.e., we obtain a *linear* least-squares problem in δ that needs to be solved efficiently. To avoid treating δx_i in a special way, we introduce the matrix $G_i^i = -I_{d\times d}$, with d the dimension of x_i.

By a simple change of variables we can drop the covariance matrices Λ_i and Σ_k from this point forward. With $\Sigma^{-1/2}$ the matrix square root of Σ we can rewrite the Mahalanobis norm as follows,

$$\|e\|_\Sigma^2 \triangleq e^T\Sigma^{-1}e = (\Sigma^{-T/2}e)^T(\Sigma^{-T/2}e) = \left\|\Sigma^{-T/2}e\right\|_2^2$$

i.e., we can always eliminate Λ_i from (8) by pre-multiplying F_i^{i-1}, G_i^i, and a_i in each term with $\Lambda_i^{-T/2}$, and similarly for the measurement covariance matrices Σ_k. *For scalar measurements this simply means dividing each term by the measurement standard deviation*. Below we assume that this has been done and drop the Mahalanobis notation.

Finally, after collecting the Jacobian matrices into a matrix A, and the vectors a_i and c_k into a right-hand side (RHS) vector b, we obtain the following standard least-squares problem,

$$\delta^* = \underset{\delta}{\text{argmin}} \ \|A\delta - b\|_2^2 \quad (9)$$

which is our starting point below. A can grow to be very large, but is quite sparse, as illustrated in Figure 4. If d_x, d_l, and d_z are the dimensions of the state, landmarks, and measurements, A's size is $(Nd_x + Kd_z) \times (Nd_x + Md_l)$. In addition, A has a typical block structure, e.g., with $M = 3$, $N = 2$, and $K = 4$:

$$A = \begin{bmatrix} G_1^1 & & & & & \\ F_2^1 & G_2^2 & & & & \\ & F_3^2 & G_3^3 & & & \\ H_1^1 & & & J_1^1 & & \\ H_2^1 & & & & J_2^2 & \\ & H_3^2 & & J_3^1 & & \\ & & H_4^3 & & J_4^2 & \end{bmatrix}, \quad b = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

Above the top half describes the robot motion, and the bottom half the measurements. A mixture of landmarks and/or measurements of different types (and dimensions) is easily accommodated.

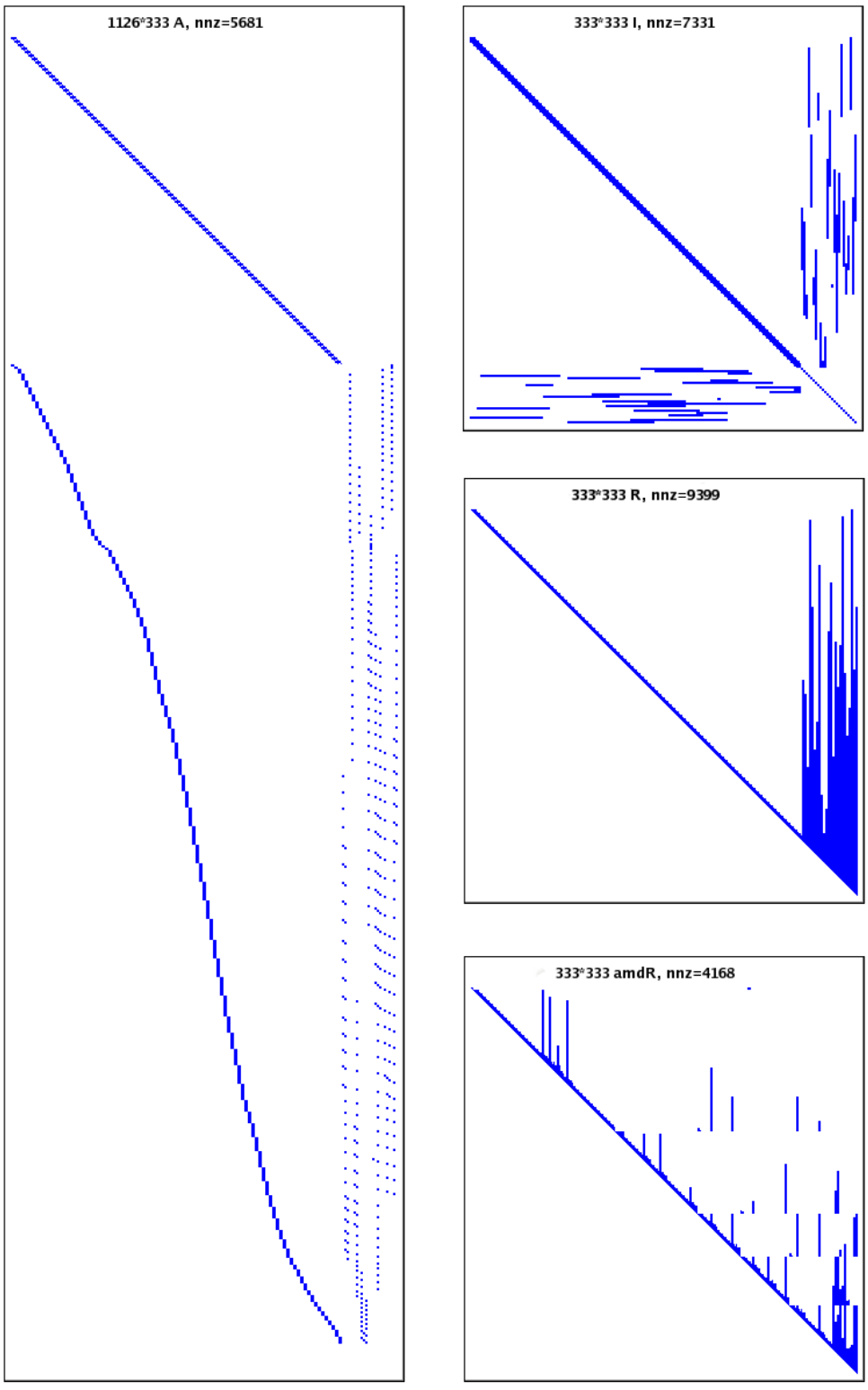


Figure 4: On the left, the measurement Jacobian A associated with the problem in Figure 1, which has $3\times95+2\times24 = 333$ unknowns. The number of rows, 1126, is equal to the number of (scalar) measurements. On the right: (top) the information matrix $\mathcal{I} \stackrel{\Delta}{=} A^T A$; (middle) its upper triangular Cholesky triangle R; (bottom) an alternative factor $amdR$ obtained with a better variable ordering (colamd from [14]).

4 A Linear Algebra Perspective

In this section we briefly review Cholesky and QR factorization and their application to the full rank linear least-squares (LS) problem in (9). This material is well known and is given primarily for review and to contrast the linear algebra algorithms with the graph-theoretic view of the next section. The exposition closely follows [35], which can be consulted for a more in-depth treatment.

For a full-rank $m \times n$ matrix A, with $m \geq n$, the unique LS solution to (9) can be found by solving the *normal equations*:

$$A^T A \delta^* = A^T b \tag{10}$$

This is normally done by Cholesky factorization of the *information matrix* $\mathcal{I}$, defined and factorized as follows:

$$\mathcal{I} \triangleq A^T A = R^T R \tag{11}$$

The *Cholesky triangle* R is an upper-triangular $n \times n$ matrix[1] and is computed using *Cholesky factorization*, a variant of LU factorization for symmetric positive definite matrices. For dense matrices Cholesky factorization requires $n^3/3$ flops. After this, δ^* can be found by solving

$$\text{first } R^T y = A^T b \text{ and then } R\delta^* = y$$

by back-substitution. The entire algorithm, including computing half of the symmetric $A^T A$, requires $(m + n/3)n^2$ flops.

For the example of Figure 1, both $\mathcal{I}$ and its Cholesky triangle R are shown alongside A in Figure 4. Note the very typical block structure of $\mathcal{I}$ when the columns of A are ordered in the standard way [79], e.g., trajectory X first and then map L (to which we refer below as the XL ordering):

$$\mathcal{I} = \begin{bmatrix} A_X^T A_X & \mathcal{I}_{XL} \\ \mathcal{I}_{XL}^T & A_L^T A_L \end{bmatrix}$$

Here $\mathcal{I}_{XL} \triangleq A_X^T A_L$ encodes the correlation between robot states X and map L, and the diagonal blocks are band-diagonal.

A variant of Cholesky factorization which avoids computing square roots is LDL factorization, which computes a lower triangular matrix L and a diagonal matrix D such that

$$\mathcal{I} = R^T R = L D L^T$$

An alternative to Cholesky factorization that is both more accurate and numerically stable is to proceed via QR-factorization *without* computing the information matrix $\mathcal{I}$. Instead, we compute the QR-factorization of A itself along with its corresponding RHS:

$$Q^T A = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad Q^T b = \begin{bmatrix} d \\ e \end{bmatrix}$$

Here Q is an $m \times m$ orthogonal matrix, and R is the upper-triangular Cholesky triangle. The preferred method for factorizing a dense matrix A is to compute R column by column, proceeding

[1] Some treatments, including [35], define the Cholesky triangle as the lower-triangular matrix $G = R^T$, but the other convention is more convenient here.

from left to right. For each column j, all non-zero elements below the diagonal are zeroed out by multiplying A on the left with a *Householder reflection matrix* H_j. After n iterations A is completely factorized:

$$H_n..H_2H_1A = Q^TA = \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{12}$$

The orthogonal matrix Q is not usually formed: instead, the transformed RHS Q^Tb is computed by appending b as an extra column to A. Because the Q factor is orthogonal, we have:

$$\|A\delta - b\|_2^2 = \left\|Q^TA\delta - Q^Tb\right\|_2^2 = \|R\delta - d\|_2^2 + \|e\|_2^2$$

Clearly, $\|e\|_2^2$ will be the least-squares residual, and the LS solution δ^* can be obtained by solving the square system

$$R\delta = d \tag{13}$$

via back-substitution. The cost of QR is dominated by the cost of the Householder reflections, which is $2(m - n/3)n^2$.

Comparing QR with Cholesky factorization, we see that both algorithms require $O(mn^2)$ operations when $m \gg n$, but that QR-factorization is a factor of 2 slower. While these numbers are valid for dense matrices only, we have seen that in practice LDL and Cholesky factorization *far* outperform QR factorization on sparse problems as well, and not just by a constant factor.

5 A Graphical Model Perspective

5.1 Matrices $\Leftrightarrow$ Graphs

From the exposition above it can now be readily appreciated that the measurement Jacobian A is the matrix of the *factor graph* associated with SLAM. We can understand this statement at two levels. First, every block of A corresponds to one term in the least-squares criterion (8), either a landmark measurement or an odometry constraint, and every block-row corresponds to one factor in the factor graph. Within each block-row, the sparsity pattern indicates which unknown poses and/or landmarks are connected to the factor. Hence, *the block-structure of A corresponds exactly to the adjacency matrix of the factor graph associated with SAM*.

Second, at the scalar level, every row A_i in A (see Figure 4) corresponds to a scalar term $\|A_i\delta - b_i\|_2^2$ in the sparse matrix least-squares criterion (9) , as

$$\|A\delta - b\|_2^2 = \sum_i \|A_i\delta - b_i\|_2^2$$

Hence, this defines a *finely structured factor graph*, via

$$P(\delta) \propto \exp -\frac{1}{2}\|A\delta - b\|_2^2 = \prod_i \exp -\frac{1}{2}\|A_i\delta - b_i\|_2^2$$

It is important to realize, that in this finer view, the block-structure of the SLAM problem is discarded, and that it is this graph that is examined by general purpose linear algebra methods. By working with the block-structure instead, we will be able to do better.

As noted before in [78] and by others, the information matrix $\mathcal{I} = A^T A$ is the matrix of the Markov random field associated with the SLAM problem. Again, *at the block-level the sparsity pattern of $A^T A$ is exactly the adjacency matrix of the associated MRF.* The objective function in Equation 5 corresponds to a pairwise Markov random field (MRF) [81, 82] through the Hammersley-Clifford theorem [81], and the nodes in the MRF correspond to the robot states and the landmarks. Links represent either odometry or landmark measurements.

In [65, 78] the MRF graph view is taken to expose the correlation structure inherent in the *filtering version* of SLAM. It is shown there that inevitably, when marginalizing out the past trajectory $X_{1:M-1}$, the information matrix becomes completely dense. Hence, the emphasis in these approaches is to selectively remove links to reduce the computational cost of the filter, with great success. **In contrast, in this paper we consider the MRF associated with the *smoothing* information matrix $\mathcal{I}$, which does *not* become dense, as past states are never marginalized out.**

5.2 Factorization $\Leftrightarrow$ Variable Elimination

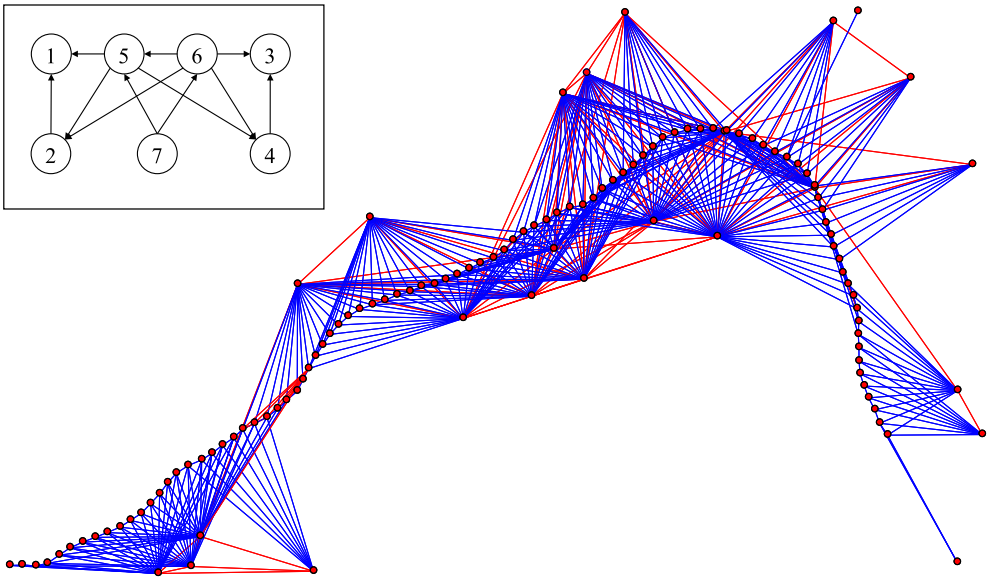


Figure 5: The triangulated graph for a good ordering (colamd, as in Figure 4). This is a *directed* graph, where each edge corresponds to a non-zero in the Cholesky triangle R. Note that we have dropped the arrows in the simulation example for simplicity.

The one question left is what graph the square root information matrix R corresponds to? Remember that R is the result of factorizing either $\mathcal{I}$ or A as in Section 4. Cholesky or QR factorization are most often used as "black box" algorithms, but in fact they are similar to much more recently developed methods for inference in graphical models [11]. It will be seen below that R is essentially in correspondence with a junction tree, known from inference in graphical models and also recently applied in SLAM [65].

Both factorization methods, QR and Cholesky (or LDL), are based on the variable elimination algorithm [4, 11]. The difference between these methods is that QR eliminates variable nodes from the factor graph and obtains $A = QR$, while Cholesky or LDL start from the MRF and hence obtain

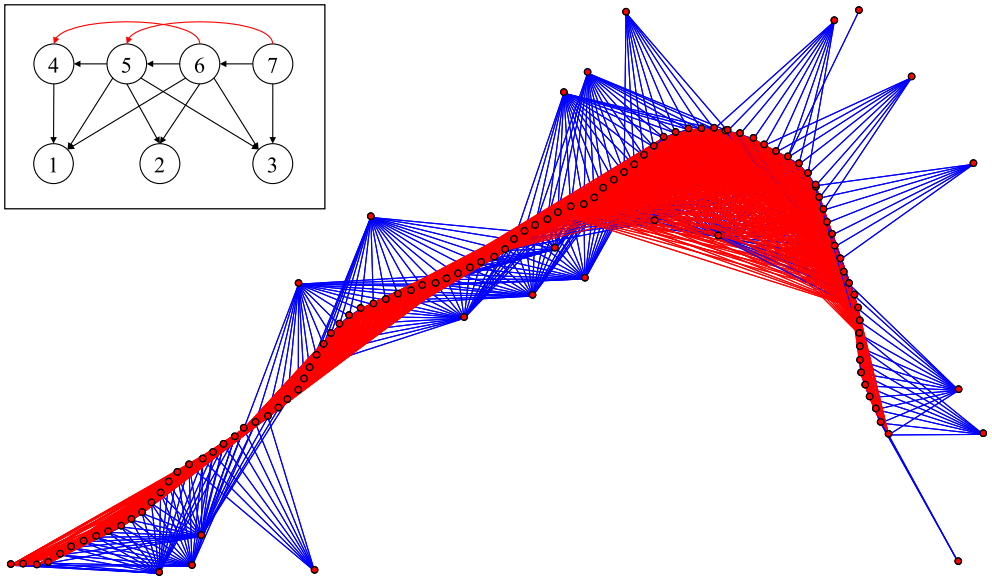


Figure 6: The triangulated graph corresponding to the well known heuristic of first eliminating landmarks, and then poses.

$A^TA = R^TR$. Both methods *eliminate* one variable at a time, starting with δ_1, corresponding to the leftmost column of either A or $\mathcal{I}$. The result of the elimination is that δ_1 is now expressed as a linear combination of all other unknowns $\delta_{j>1}$, with the coefficients residing in the corresponding row R_1 of R. In the process, however, new dependencies are introduced between all variables connected to δ_1, *which causes edges to be added to the graph*. The next variable is then treated in a similar way, until all variables have been eliminated. This is exactly the process of moralization and triangulation familiar from graphical model inference. The result of eliminating all variables is a directed, triangulated (chordal) graph, shown for our example in Figure 5.

Once the chordal graph (R!) is obtained, one can obtain the *elimination tree* of R, defined as a depth-first spanning tree of the chordal graph after elimination, and which is useful in illustrating the flow of computation during the back-substitution phase. To illustrate this, Figure 6 shows the chordal graph obtained by using the well-known heuristic of first eliminating landmarks [79], and then poses (which we will call the LX ordering). The corresponding elimination tree is shown in Figure 7. The root of the tree corresponds to the last variable δ_n to be eliminated, which is the first to be computed in back-substitution (Equation 13). Computation then proceeds down the tree, and while this is typically done in reverse column order, variables in disjoint subtrees may be computed in any order, see [19] for a more detailed discussion. In fact, if one is only interested in certain variables, there is no need to compute any of the subtrees that do not contain them.

However, the analysis does not stop there. The graph of R has a clique-structure which can be completely encapsulated in a rooted tree data-structure called a *clique tree* [66, 4], also known as the *junction tree* in the AI literature [11]. As an example, the clique-tree for the LX-ordering on the problem of Figure 1 is shown in Figure 8. The correspondence is almost one-to-one: every R corresponds to exactly one clique tree, and vice versa, modulo column re-orderings within cliques. The clique tree is also the basis for multifrontal QR methods [58], which we have also evaluated in our simulations below. In multifrontal QR factorization, computation progresses in this tree

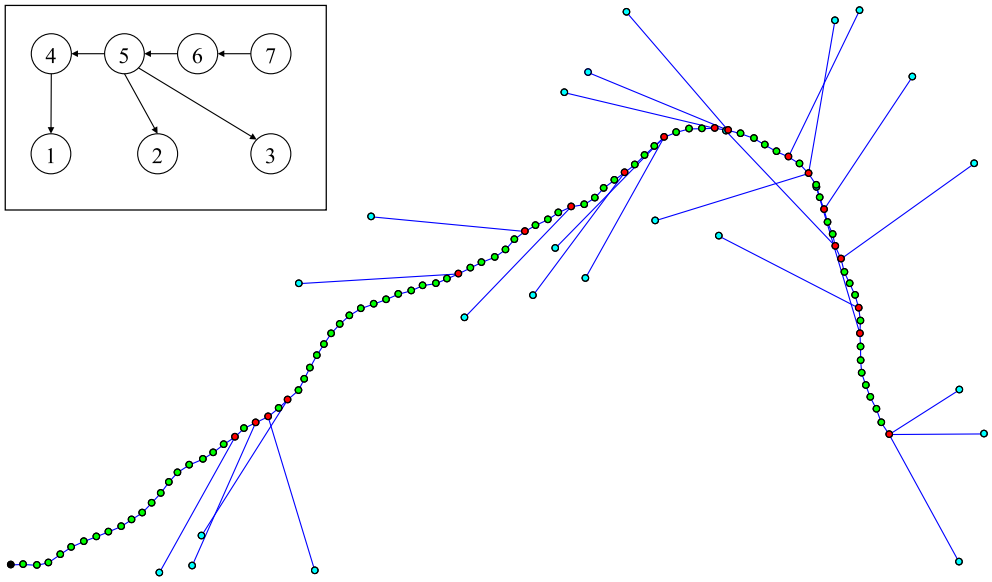


Figure 7: The elimination tree for the LX ordering, showing how the state and landmarks estimates will be computed via back-substitution: the root is computed first - here the first pose on the left - and proceeds down the Markov chain to the right. Landmarks can be computed as soon as the pose they are connected to has been computed.

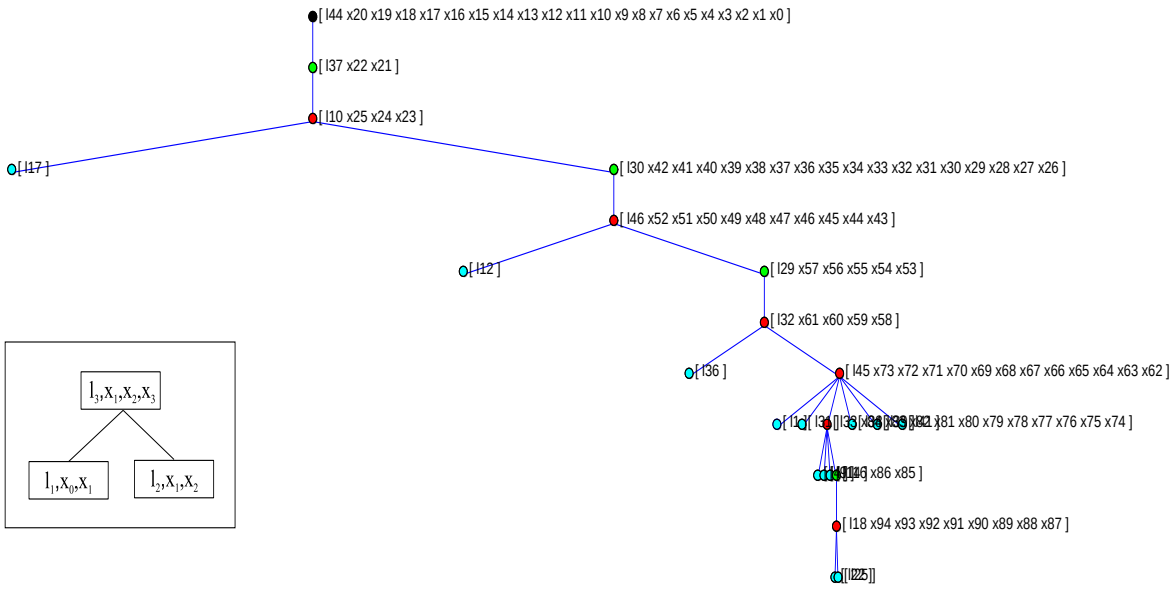


Figure 8: The clique-tree corresponding to the LX ordering, i.e., Figures 6 and 7, showing at each node the frontal variables of each clique (those that do not also appear in cliques below that node). It can be seen that the root consists of the small piece of trajectory from the root to where the first landmark "subtree" branches off.

from the leaves to the root to factorize a sparse matrix, and then from the root to the leaves to perform a backward substitution step. A complete treatment of the relationship between square root information matrices and clique trees is beyond the scope of the current paper, but in other work we have used the clique-tree structure in novel algorithms for distributed inference [19].

5.3 Improving Performance $\Leftrightarrow$ Reducing Fill-in

The single most important factor to good performance is the order in which variables are eliminated. Different variable orderings can yield dramatically more or less *fill-in*, defined as the amount of edges added into the graph during factorization. As each edge added corresponds to a non-zero in the Cholesky triangle R, both the cost of computing R and back-substitution is heavily dependent on how much fill-in occurs. Unfortunately, finding an optimal ordering is NP-complete. Discovering algorithms that approximate the optimal ordering is an active area of research in sparse linear algebra. A popular method for medium-sized problems is *colamd* [1], which works on the columns of A. Another popular method, based on graph theory and often used to speed up finite element methods, is generalized nested dissection [55, 54].

Given that an optimal ordering is out of reach in general, heuristics or domain-knowledge can do much better than general-purpose algorithms. A simple idea is to use a standard method such as *colamd*, but have it work on the sparsity pattern of the blocks instead of passing it the original measurement Jacobian A. In other words we treat a collection of scalar variables such as the x and y position and orientation α as a single variable and create a smaller graph which encapsulates the constraints between these blocks rather than the individual variables. Not only is it cheaper to call *colamd* on this smaller graph, it also leads to a substantially improved ordering. As we mentioned above, the block-structure is real knowledge about the SLAM problem and is not accessible to colamd or any other approximate ordering algorithm. While the effect on the performance of colamd is negligible, we have found that making it work on the SLAM MRF instead of on the sparse matrix $\mathcal{I}$ directly can yield improvements of 2 to sometimes a 100-fold, with 15 being a good rule of thumb.

Note that there are cases in which any ordering will result in the same large fill-in. The worst-case scenario is a fully connected bipartite[2] MRF: every landmark is seen from every location. In that case, eliminating any variable will completely connect all variables on the other side, and after that the structure of the clique tree is completely known: if a pose was chosen first, the root will be the entire map, and all poses will be computed once the map is known. Vice versa, if a landmark is chosen, the trajectory will be the clique tree root clique, and computation will proceed via an (expensive) trajectory optimization, followed by (very cheap) computation of landmarks. Interestingly, these two cases form the basis of the standard partitioned inverse, or "Schur complement", which is well-known in structure from motion applications [79, 41] and also used in GraphSLAM [77].

However, the worst-case scenario outlined above is an exceptional case in robotics: sensors have limited range and are occluded by walls, objects, buildings, etc. This is especially true in large-scale mapping applications, and it essentially means that the MRF will in general be sparsely connected, even though it is one large connected component.

[2]bipartite is here used to indicate the partition of variables into poses and landmarks, not in the factor graph sense.

6 Square Root SAM

In this section we take everything we know from above and state three simple $\sqrt{\text{SAM}}$ variants depending on whether they operate in batch or incremental mode, and on whether non-linearities are involved or not.

6.1 Batch $\sqrt{\text{SAM}}$

A batch-version of *square root information smoothing and mapping* is straightforward and a completely standard way of solving a large, sparse least-squares problem:

Algorithm 1 Batch $\sqrt{\text{SAM}}$

1. Build the measurement Jacobian A and the RHS b as explained in Section 3.
2. Find a good column ordering p, and reorder $A_p \overset{p}{\leftarrow} A$
3. Solve $\delta_p^* = \text{argmin}_{\ \delta} \|A_p \delta_p - b\|_2^2$ using either the Cholesky or QR factorization method from Section 4
4. Recover the optimal solution by $\delta \overset{r}{\leftarrow} \delta_p$, with $r = p^{-1}$

In tests we have obtained the best performance with sparse LDL factorization [13], which, as mentioned above, is a variant on Cholesky factorization that computes $\mathcal{I} = LDL^T$, with D a diagonal matrix and L a lower-triangular matrix with ones on the diagonal.

The same algorithm applies in the non-linear case, but is simply called multiple times by the non-linear optimizer. Note that the ordering has to be computed only once, however.

6.2 Linear Incremental $\sqrt{\text{SAM}}$

In a robotic mapping context, an incremental version of the above algorithm is of interest. The treatment below holds for either the linear case, or when a good linearization point is available.

It is well known that factorizations can be updated incrementally. One possibility is to use a rank 1 Cholesky update, a standard algorithm that computes the factor R' corresponding to a

$$\mathcal{I}' = \mathcal{I} + aa^T$$

where a^T is a new row of the measurement Jacobian A, corresponding to new measurements that come in at any given time step. However, these algorithms are typically implemented for dense matrices only, and it is imperative that we use a sparse storage scheme for optimal performance. While sparse Cholesky updates exist [12], they are relatively complicated to implement. A second possibility, easy to implement and suited for sparse matrices, is to use a series of Givens rotations (see [35]) to eliminate the non-zeros in the new measurement rows one by one.

A third possibility, which we have adopted for the simulations below, is to update the matrix $\mathcal{I}$ and simply use a full Cholesky (or LDL) factorization. While this seems expensive, we have found that with good orderings the dominant cost is no longer the factorization but rather the updating of

$\mathcal{I}$. For example, the experimental results in Section 8 show that the sparse multiplication to obtain the Hessian is about 5 times as expensive than the factorization at the end of the run. A more detailed discussion of asymptotic bounds can be found in [47]. A comparison with the Givens scheme above would be of interest, but we have not done so yet.

Importantly, because the entire measurement history is implicit in $\mathcal{I}$, one does not need to factorize at every time-step. In principle, we can wait until the very end and then compute the entire trajectory and map. At any time during an experiment, however, the map and/or trajectory can be computed by a simple factorization and back-substitution, e.g., for visualization and/or path planning purposes.

6.3 Non-linear Incremental $\sqrt{\text{SAM}}$

If in an incremental setting the SLAM problem is non-linear and a linearization point is available, then the linear incremental solution above applies. Whether this is the case very much depends on the sensors and prior knowledge available to the robot. For example, in indoor mobile robots there is often no good absolute orientation sensor, which means we have to linearize around the current guess for the orientation. This is exactly why EKF's have problems in larger-scale environments, as these potentially wrong linearization points are "baked into" the information matrix when past robot poses are eliminated from the MRF.

A significant advantage of the smoothing approach is the fact that we never commit to a given linearization, as no variables are ever eliminated from the graph. There are two different ways to re-linearize: In certain scenarios, like closing of a large loop along the complete trajectory, it is cheaper to re-linearize all variables. Essentially this means that we have to call the batch version above each time. On the upside, our experimental results will show that even this seemingly expensive algorithm is quite practical on large problems where an EKF approach is intractable. The alternative way is favorable when only a smaller number of variables is affected by the change in the linearization point. In this case downdating and updating techniques [79] can be applied to temporarily remove these variables from the factor, followed by adding them in again using the new linearization point. We plan to evaluate this approach in future work.

7 Simulation Results

7.1 Batch $\sqrt{\text{SAM}}$

We have experimented at length with different implementations of Cholesky, LDL, and QR factorization to establish which performed best. All simulations were done in MATLAB on a 2GHz Pentium 4 workstation running Linux. Experiments were run in synthetic environments like the one shown in Figure 9, with 180 to 2000 landmarks, for trajectories of length 200, 500, and 1000. Each experiment was run 10 times for 5 different methods:

- *none*: no factorization performed
- *ldl*: Davis' sparse LDL factorization [13]
- *chol*: MATLAB built-in Cholesky factorization

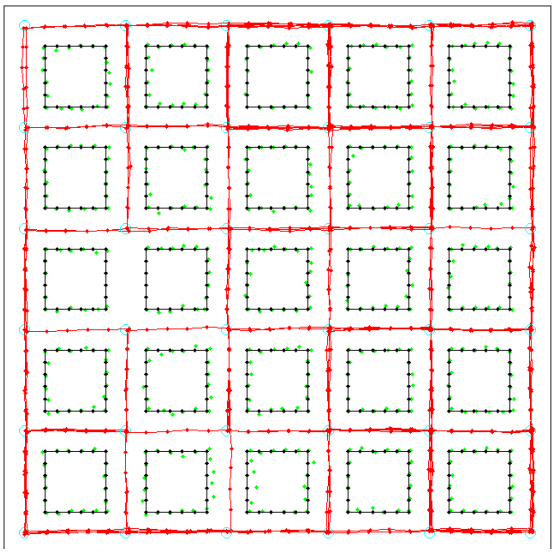


Figure 9: A synthetic Manhattan world with 500 landmarks (black dots on square city blocks) along with a 1000-step random street walk trajectory (red lines with a dot for each step), corresponding to 14000 measurements taken. The measurements (light green) do not line up exactly with the landmarks due to simulated noise.

M	N	none	ldl	chol	mfqr	qr
200	180	0.031	0.062	0.092	0.868	1.685
	500	0.034	0.062	0.094	1.19	1.256
	1280	0.036	0.068	0.102	1.502	1.21
	2000	0.037	0.07	0.104	1.543	1.329
500	180	0.055	0.176	0.247	2.785	11.92
	500	0.062	0.177	0.271	3.559	8.43
	1280	0.068	0.175	0.272	5.143	6.348
	2000	0.07	0.181	0.279	5.548	6.908
1000	180	0.104	0.401	0.523	10.297	42.986
	500	0.109	0.738	0.945	12.112	77.849
	1280	0.124	0.522	0.746	14.151	35.719
	2000	0.126	0.437	0.657	15.914	25.611

Figure 10: Averaged simulation results over 10 trials, in seconds, for environments with various number of landmarks N and simulations with trajectory lengths M. The methods are discussed in more detail in the text. The *none* method corresponds to doing no factorization and measures the overhead.

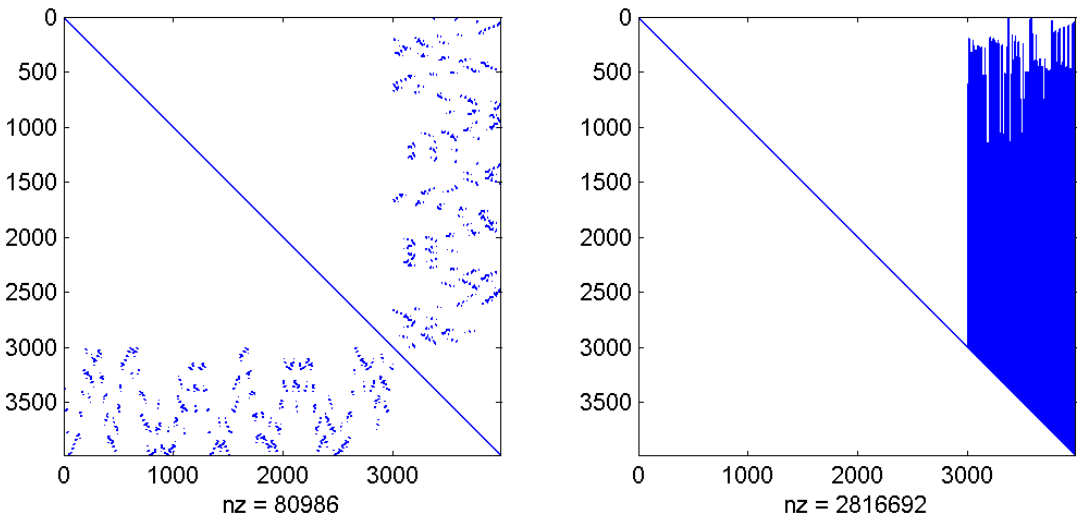


Figure 11: Original information matrix $\mathcal{I}$ and its Cholesky triangle. Note the dense fill-in on the right, linking the entire trajectory to all landmarks. The number of non-zeroes is indicated by nz.

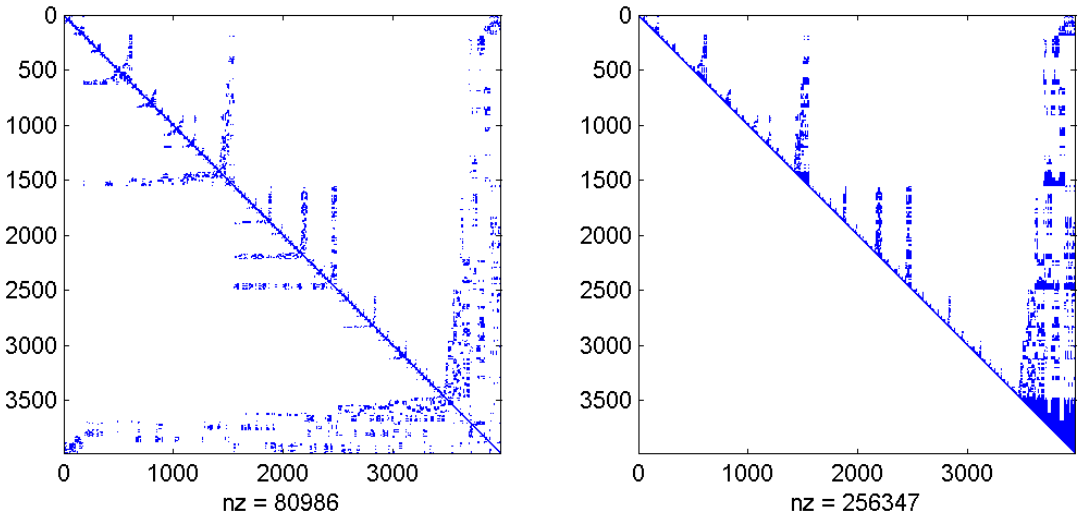


Figure 12: Information matrix $\mathcal{I}$ after reordering and its Cholesky triangle. Reordering of columns (unknowns) does not affect the sparseness of $\mathcal{I}$, but the number of non-zeroes in R has dropped from approximately 2.8 million to about 250 thousand.

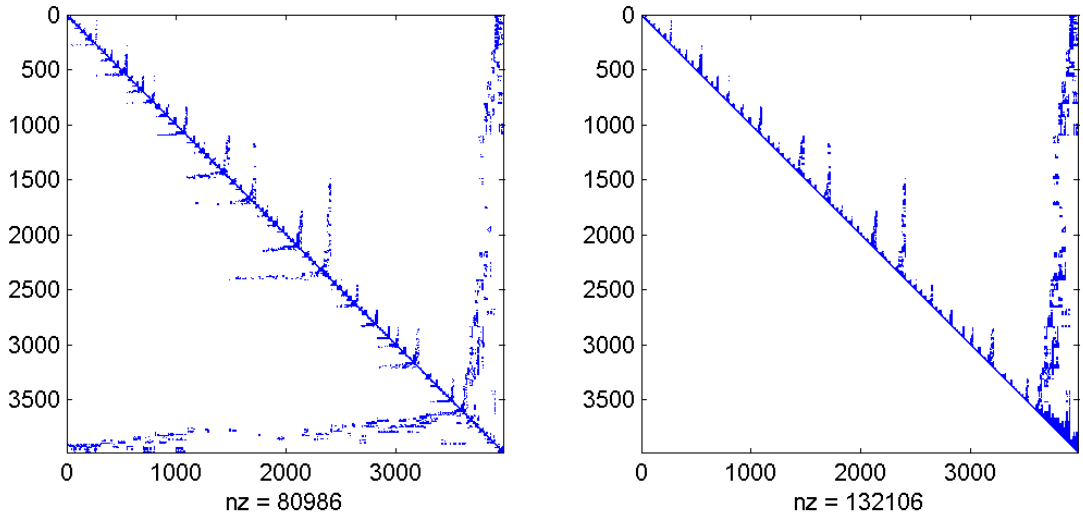


Figure 13: By doing the reordering while taking into account the special block-structure of the SLAM problem, the non-zero count can be reduced even further, to about 130K, a reduction by a factor of 20 with respect to the original R, and substantially less than the 500K entries in the filtering covariance matrix.

- *mfqr*: multifrontal QR factorization [58]
- *qr*: MATLAB built-in QR factorization

The results are summarized in Figure 10. We have found that, under those circumstances,

1. The freely available sparse LDL implementation by T. Davis [13] outperforms MATLAB's built-in Cholesky factorization by a factor of 30%.
2. In MATLAB, the built-in Cholesky outperforms QR factorization by a large factor.
3. Multifrontal QR factorization is better than MATLAB's QR, but still slower than either Cholesky or LDL.
4. While this is not apparent from the table, using a good column ordering is *much* more important than the choice of factorization algorithm: QR factorization with a good ordering will outperform LDL or Cholesky with a bad ordering.

The latter opens up a considerable opportunity for original research in the domain of SLAM, as we found that injecting even a small amount of domain knowledge into that process yields immediate benefits. To illustrate this, we show simulation results for a length 1000 random walk in a 500-landmark environment, corresponding to Figure 9. Both $\mathcal{I}$ and R are shown in Figure 11 for the standard (and detrimental) XL ordering with states and landmarks ordered consecutively. The dramatic reduction in fill-in that occurs when using a good re-ordering is illustrated by Figure 12, where we used *colamd* [1]. Finally, when we use the block-oriented ordering heuristic discussed in Section 5.3, the fill-in drops by another factor of 2, significantly increasing computational efficiency.

7.2 Incremental $\sqrt{\text{SAM}}$

We also compared the performance of an incremental version of $\sqrt{\text{SAM}}$, described in Section 6, with a standard EKF implementation by simulating 500 time steps in a synthetic environment with 2000 landmarks. The results are shown in Figure 14. The factorization of $\mathcal{I}$ was done using sparse LDL [13], while for the column ordering we used *symamd* [1], a version of *colamd* for symmetric positive definite matrices.

Smoothing *every time step* becomes cheaper than the EKF when the number of landmarks N reaches 600. At the end, with $N = 1,100$, each factorization took about 0.6 s, and the slope is nearly linear over time. In contrast, the computational requirements of the EKF increase quadratically with N, and by the end each update of the EKF took over a second.

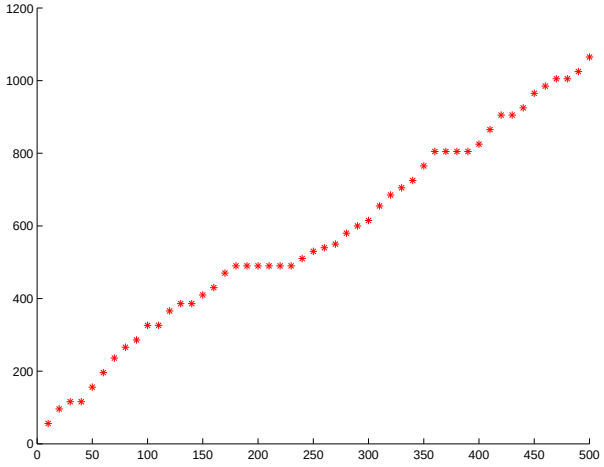
As implementation independent measures, we have also plotted N^2, as well as the number of non-zeros nz in the Cholesky triangle R. The behavior of the latter is exactly opposite to that of the EKF: when new, unexplored terrain is encountered, there is almost no correlation between new features and the past trajectory and/or map, and nz stays almost constant. In contrast, the EKF's computation is not affected when re-entering previously visited areas -closing the loop- whereas that is exactly when R fill-in occurs.

8 Experimental Results

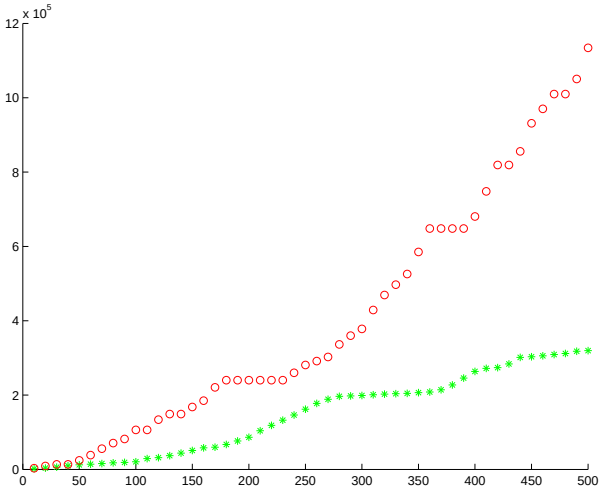
We have evaluated the non-linear incremental version of $\sqrt{\text{SAM}}$ on a very challenging, large-scale vision-based SLAM problem, in which a mobile robot equipped with 8 cameras traversed an indoor office environment. This is probably the most challenging data-set we have ever worked with, both because of the amount of data that needed to be dealt with, as well as the logistical and calibration issues that plague multi-camera rigs. In addition, dealing with visual features is complex and prone to failure.

In this case, the measurements are features extracted from 8 cameras mounted on top of an iRobot ATRV-Mini platform, as shown in Figure 15. They are matched between successive frames using RANSAC based on a trifocal camera-arrangement. The data was taken in an office environment, with a bounding box of about $30m$ by $50m$ for the robot trajectory, and the landmarks sought are a set of unknown 3D points.The measurements consisted of the odometry provided by the robot, as well as 260 joint images taken with variable distances of up to 2 meters between successive views, and an overall trajectory length of about $190m$. The unknown poses were modeled as having 6 degrees of freedom (DOF), three translational and three rotational. Even though 3DOF seems sufficient for a planar indoor office environment, it turns out that 6DOF with a prior on pitch, roll and height is necessary, since any small bump in the floor has a clearly visible effect on the images. The standard deviations on x and y are $0.02m$ and on ϕ (yaw) $0.02rad$. The priors on z, θ (pitch) and ψ (roll) are all 0 with standard deviations $0.01m$ and $0.02rad$ respectively. The camera rig was calibrated in advance.

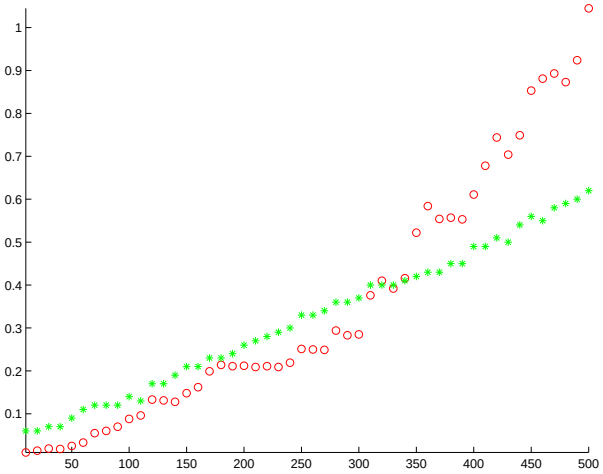
The $\sqrt{\text{SAM}}$ approach was able to comfortably deal with this large problem well within the real-time constraints of the application. Our approach was to invoke the batch $\sqrt{\text{SAM}}$ algorithm after every three joint images taken by the robot. In each of these invocations, the problem was repeatedly re-linearized and factorized to yield an optimal update. Again we used sparse LDL as the main factorization algorithm. More importantly, for ordering the variables we used *colamd* com-



(a) number of landmarks N seen



(b) N^2 (circles) and nz in R (asterisks)



(c) average time per call, in seconds for the EKF (circles) and LDL factorization (asterisks)

Figure 14: Timing results for incremental SAM in a simulated environment with 2000 landmarks, similar to the one in Figure 9, but 10 blocks on the side. As the number of landmarks seen increases, the EKF becomes quadratically slower. Note that the number of non-zeros nz increases faster when large loops are encountered around $i = 200$ and $i = 350$.

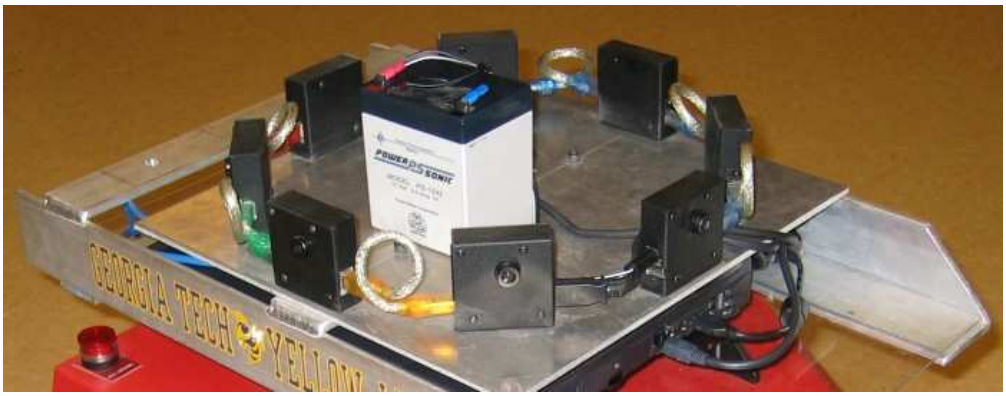


Figure 15: Custom made camera rig, mounted on top of an ATRV-Mini mobile robot. Eight FireWire cameras are distributed equally along a circle and connected to an on-board laptop.

bined with the block-structured ordering heuristic. The latter alone yielded a 15-fold improvement in the execution time of LDL with respect to *colamd* by itself.

Processing the entire sequence took a total of 11 minutes and 10 seconds on a 2GHz Pentium-M based laptop, and the final result can be seen in Figure 16. The correspondence matching yielded in 17780 measurements on a total of 4383 unknown 3D points, taken from 260 different robot locations. In the final step, the measurement matrix A had $36,337$ rows and $14,709$ columns, with about $350K$ non-zero entries.

The relevant timing results are shown in Figure 17. The unexpected result is that *the factorization, because of the good variable ordering, is now but a minor cost in the whole optimization*. Instead, the largest cost is now evaluating the measurement Jacobian A (linearizing the measurement equations), which was done a total of 453 times. Its computational demands over time are shown in the panel at the top. Next in line is the computation of the information matrix $\mathcal{I} = A^T A$, shown above by "hessian" in the bottom panel. This is done exactly as many times as LDL itself, i.e., a total of 734 times. By the end of the sequence, this sparse multiplication (yielding - by then - a $15K \times 15K$ matrix) takes about 0.6 secs. In contrast, factorizing the resulting information matrix $\mathcal{I}$ takes just 0.1 seconds.

Clearly, further improvement must come from avoiding linearization of the entire set of measurement equations, and hence keeping large parts of these expensive operations constant.

9 Discussion

The square root information smoothing approaches we presented in the current paper have several significant advantages over the EKF:

- They are much faster than EKF-based SLAM on large-scale problems
- They are exact, in contrast to approximate methods to deal with the EKF shortcomings
- They can be used in either batch or incremental mode
- If desired, they yield the *entire* smoothed robot trajectory

Figure 16: Projected trajectory and map after applying incremental $\sqrt{\text{SAM}}$ using visual input and odometry only. Each robot pose is shown as an outline of the ATRV-Mini platform. The recovered 3D structure is represented by green points. For comparison the manually aligned building map is shown in gray. Given that no loop-closing occurred and considering the large scale of the environment and the incremental nature of the reconstruction method, this result is of very high-quality. Note that features also occur along the ceiling, and that some features outside the building outline are caused by reflections.

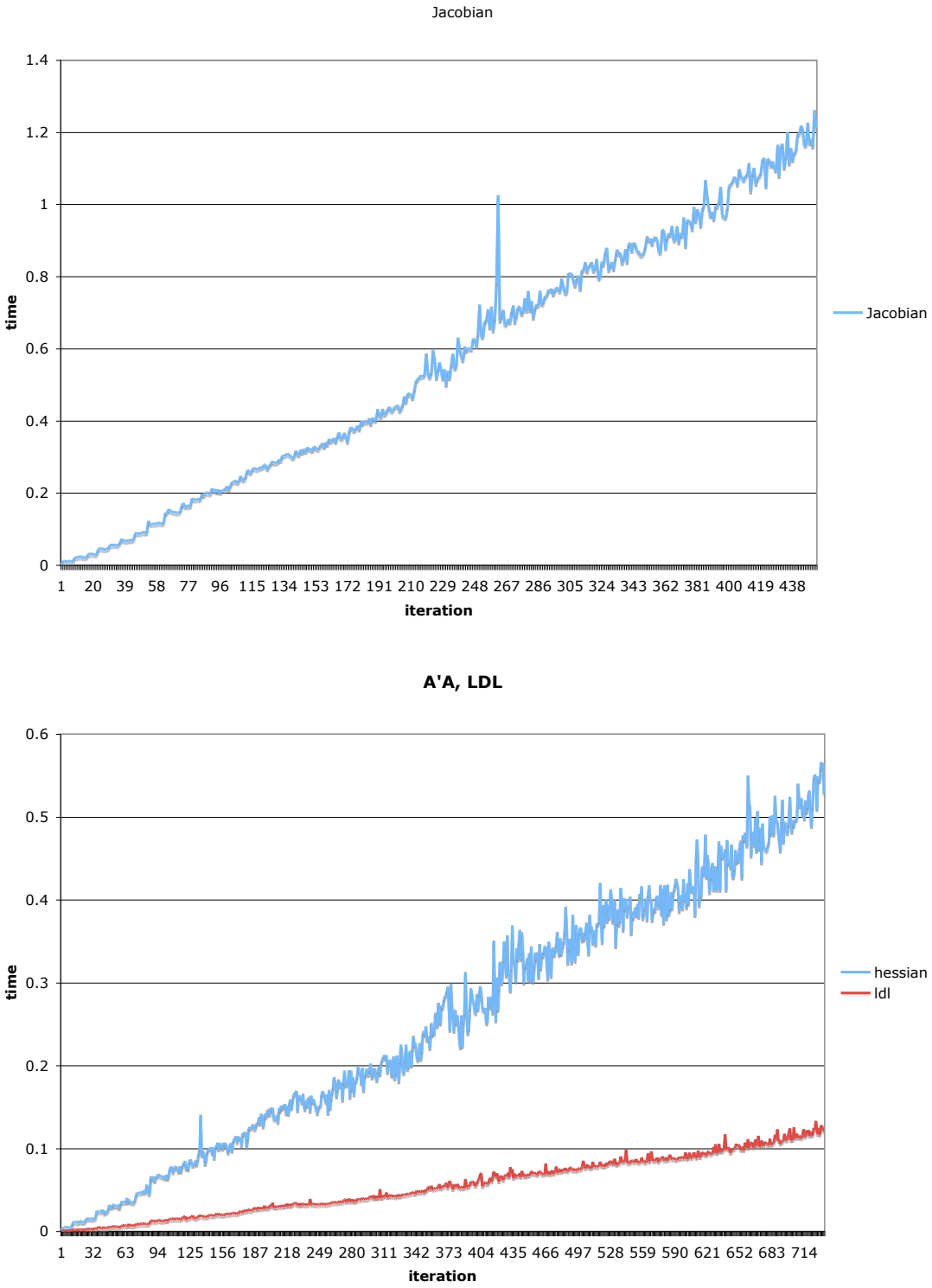


Figure 17: Timing results for incremental $\sqrt{\text{SAM}}$ on the sequence from Figure 16. Top: cost of forming the Jacobian A (linearization). Bottom: surprisingly, the cost for LDL factorization is now *less* than forming the Hessian or information matrix $\mathcal{I}$.

- They are much better equipped to deal with non-linear process and measurement models than the EKF
- They *automatically* exploit locality by means of variable ordering heuristics

However, there is also a price to pay:

- Because we smooth the entire trajectory, computational complexity grows without bound over time, as clearly illustrated in Figure 17. In many typical mapping scenarios, however, the computational and storage demands of the EKF information or covariance matrix will grow much faster still (in the same example, it would involve storing and manipulating a *dense* $15K \times 15K$ matrix)
- As with all information matrix approaches, it is expensive to recover the joint covariance matrix governing the unknowns. However, we *can* recover their marginal densities at a much lower cost [36]

In this paper we only reported on our initial experiences with this approach, and in particular the following leaves to be desired:

- We have not yet established any tight or amortized complexity bounds that predict the algorithm's excellent performance on problems of a given size.
- Comparison of our approach to more recent and faster SLAM variants, both approximate [49, 65, 78] and exact [38], is the object of future work.

Finally, in this paper we primarily concentrated on the large-scale optimization problem associated with SLAM. Many other issues are crucial in the practice of robot mapping, e.g. the data-association problem, exploitation vs. exploration, and the management of very-large scale mapping problems. As such, the techniques presented here are meant to complement, not replace methods that make judicious approximations in order to reduce the asymptotic complexity of SLAM.

10 Conclusion and Future Work

In conclusion, we believe square root information smoothing to be of great practical interest to the SLAM community. It recovers the entire trajectory and is exact, and even the sub-optimal incremental scheme we evaluated behaves much better than the EKF as the size of the environment grows. In addition, our experiments with real robot data offer proof that the possibility of re-linearizing the entire trajectory at each time-step makes $\sqrt{\text{SAM}}$ cope well with noisy measurements governed by non-linear measurement equations. In contrast, non-optimal linearization cannot be recovered from in an EKF, which inevitably *has* to summarize it in a quadratic (Gaussian) approximation.

Incremental implementations of the proposed method are of great interest for real-time applications. We are currently exploring Givens rotations as a means of incrementally updating the factor. Downdating could also allow selective changes of variables affected by re-linearization. Finally, our ongoing work indicates that an efficient retrieval of the marginal covariances is possible based on the square root factor. This is important for data association, an aspect of the SLAM problem that we have ignored in the present work.

Acknowledgments

We would like to thank Alexander Kipp and Peter Krauthausen for their valuable help in this project, without which our task would have been much harder. In addition, we would like to thank the anonymous reviewers for their valuable comments. Finally, we gratefully acknowledge support from the National Science Foundation through awards IIS - 0448111 "CAREER: Markov Chain Monte Carlo Methods for Large Scale Correspondence Problems in Computer Vision and Robotics" and No. IIS - 0534330 "Unlocking the Urban Photographic Record Through 4D Scene Understanding and Modeling".

References

[1] P. R. Amestoy, T. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[2] N. Ayache and O.D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Trans. Robot. Automat.*, 5(6):804–819, 1989.

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

[4] J.R.S. Blair and B.W. Peyton. An introduction to chordal graphs and clique trees. In J.A. George, J.R. Gilbert, and J.W-H. Liu, editors, *Graph Theory and Sparse Matrix Computations*, volume 56 of *IMA Volumes in Mathematics and its Applications*, pages 1–27. Springer-Verlag, 1993.

[5] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas framework for scalable mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2003.

[6] R.A. Brooks. Visual map making for a mobile robot. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 824 – 829, March 1985.

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

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

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

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

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

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

[13] T. A. Davis. Algorithm 8xx: a concise sparse Cholesky factorization package. Technical Report TR-04-001, Univ. of Florida, January 2004. Submitted to ACM Trans. Math. Software.

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

[15] T.A. Davis and K. Stanley. KLU: a "Clark Kent" sparse LU factorization algorithm for circuit matrices. In *2004 SIAM Conference on Parallel Processing for Scientific Computing (PP04)*, 2004.

[16] M. Deans and M. Hebert. Experimental comparison of techniques for localization and mapping using a bearings only sensor. In *Proc. of the ISER '00 Seventh International Symposium on Experimental Robotics*, December 2000.

[17] M. Deans and M. Hebert. Invariant filtering for simultaneous localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1042–7, April 2000.

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

[19] F. Dellaert, A. Kipp, and P. Krauthausen. A multifrontal QR factorization approach to distributed inference applied to multi-robot localization and mapping. In *AAAI Nat. Conf. on Artificial Intelligence*, 2005.

[20] J.E. Dennis and R.B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, 1983.

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

[22] T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, San Francisco, CA, 2000.

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

[24] R. Eustice, H. Singh, and J. Leonard. Exactly sparse delayed-state filters. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2428–2435, Barcelona, Spain, April 2005.

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

[26] J. Folkesson and H. I. Christensen. Graphical SLAM - a self-correcting map. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 383 – 390, 2004.

[27] J. Folkesson, P.Jensfelt, and H. I. Christensen. Graphical SLAM using vision and the measurement subspace. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[28] U. Frese. *An $O(\log n)$ Algorithm for Simultaneous Localization and Mapping of Mobile Robots in Indoor Environments*. PhD thesis, University of Erlangen-Nürnberg, 2004.

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

[30] U. Frese and T. Duckett. A multigrid approach for accelerating relaxation-based SLAM. In *Proc. of the IJCAI-03 on Reasoning with Uncertainty in Robotics*, 2003.

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

[32] J.A. George, J.R. Gilbert, and J.W-H. Liu, editors. *Graph Theory and Sparse Matrix Computations*, volume 56 of *IMA Volumes in Mathematics and its Applications*. Springer-Verlag, 1993.

[33] M. Golfarelli, D. Maio, and S. Rizzi. Elastic correction of dead-reckoning errors in map building. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 905–911, 1998.

[34] M. Golfarelli, D. Maio, and S. Rizzi. Correction of dead-reckoning errors in map building. *IEEE Trans. Robot. Automat.*, 17(1), 2001.

[35] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996.

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

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

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

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

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

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

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

[43] A. Howard, M.J. Matarić, and G.S. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1055 – 1060, Wailea, Hawaii, Oct 2001.

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

[45] S.J. Julier and J.K. Uhlmann. Simultaneous localisation and map building using split covariance intersection. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, volume 3, pages 1257–62, 2001.

[46] K. Konolige. Large-scale map-making. In *AAAI Nat. Conf. on Artificial Intelligence*, San Jose, 2004.

[47] Peter Krauthausen, Frank Dellaert, and Alexander Kipp. Exploiting locality by nested dissection for square root smoothing and mapping. In *Robotics: Science and Systems (RSS)*, 2006.

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

[49] J. J. Leonard and H. J. S. Feder. Decoupled stochastic mapping. *IEEE Journal of Oceanic Engineering*, pages 561–571, October 2001.

[50] J. J. Leonard and P. M. Newman. Consistent, convergent, and constant-time SLAM. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 2003.

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

[52] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IEEE Int. Workshop on Intelligent Robots and Systems*, pages 1442–1447, 1991.

[53] J.J. Leonard and H.F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic, Boston, 1992.

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

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

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

[57] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems*, page 249:275, April 1997.

[58] P. Matstoms. Sparse QR factorization in MATLAB. *ACM Trans. Math. Softw.*, 20(1):136–159, 1994.

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

[60] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI Nat. Conf. on Artificial Intelligence*, 2002.

[61] P. Moutarlier and R. Chatila. An experimental system for incremental environment modelling by an autonomous mobile robot. In *Experimental Robotics I, The First International Symposium, Montréal, Canada, June 19-21, 1989*, pages 327–346, 1989.

[62] P. Moutarlier and R. Chatila. Stochastic multisensor data fusion for mobile robot location and environment modelling. In *5th Int. Symp. Robotics Research*, 1989.

[63] K. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

[64] P. Newman. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. PhD thesis, The University of Sydney, 1999.

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

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

[67] D. Rodriguez-Losada, F. Matia, and A. Jimenez. Local maps fusion for real time multirobot indoor simultaneous localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 1308–1313, 2004.

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

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

[70] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *Int. Symp on Robotics Research*, 1987.

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

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

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

[74] Y. Tang and C. G. Lee. A geometric feature relation graph formulation for consistent sensor fusion. *IEEE Trans. Syst., Man, Cybern.*, 22:115–129, January/February 1992.

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

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

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

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

[79] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[80] S.B. Williams, G. Dissanayake, and H. Durrant-Whyte. An efficient approach to the simultaneous localisation and mapping problem. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 406–411, 2002.

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

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