iSAM: Incremental Smoothing and Mapping

Michael Kaess, *Student Member, IEEE*, Ananth Ranganathan, *Student Member, IEEE*, and Frank Dellaert, *Member, IEEE*

***Abstract*—In this paper, we present incremental smoothing and mapping (iSAM), which is a novel approach to the simultaneous localization and mapping problem that is based on fast incremental matrix factorization. iSAM provides an efficient and exact solution by updating a QR factorization of the naturally sparse smoothing information matrix, thereby recalculating only those matrix entries that actually change. iSAM is efficient even for robot trajectories with many loops as it avoids unnecessary fill-in in the factor matrix by periodic variable reordering. Also, to enable data association in real time, we provide efficient algorithms to access the estimation uncertainties of interest based on the factored information matrix. We systematically evaluate the different components of iSAM as well as the overall algorithm using various simulated and real-world datasets for both landmark and pose-only settings.**

***Index Terms*—Data association, localization, mapping, mobile robots, nonlinear estimation, simultaneous localization and mapping (SLAM), smoothing.**

I. Introduction

THE GOAL of simultaneous localization and mapping (SLAM) [1]–[3] is to provide an estimate after every step for both the robot trajectory and the map, given all available sensor data. In addition to being *incremental*, to be practically useful, a solution to SLAM has to perform in *real time*, be applicable to *large-scale environments*, and support *online data association*. Such a solution is essential for many applications, stretching from search and rescue, over reconnaissance to commercial products such as entertainment and household robots. While there has been much progress over the past decade, none of the work presented so far fulfills all of these requirements at the same time.

Our previous work, called *square root smoothing and mapping (SAM)* [4], [5], gets close to this goal by factorizing the information matrix of the smoothing problem. Formulating SLAM in a smoothing context adds the complete trajectory into the estimation problem, thereby simplifying its solution. While this seems counterintuitive at first, because more variables are added to the estimation problem, the simplification arises from the fact that the smoothing information matrix is naturally sparse. In contrast, in filtering approaches, the information matrix becomes dense when marginalizing out robot poses. As a consequence of applying smoothing, we are able to provide an exact, yet efficient solution based on a sparse matrix factorization of the smoothing information matrix in combination with back substitution. We call this matrix factor the *square root information matrix*, based on earlier work on square root information filtering (SRIF) and smoothing (SRIS), as recounted in [6] and [7].

In this paper, we present incremental smoothing and mapping (iSAM), which performs *fast incremental updates* of the square root information matrix yet is able to compute the full map and trajectory at any time. Our previous work, square root SAM, is a batch algorithm that first updates the information matrix when new measurements become available and then factors it completely. Hence, it performs unnecessary calculations when applied incrementally. In this paper, in contrast, we directly update the square root information matrix with new measurements as they arrive, using standard matrix update equations [8]. That means we reuse the previously calculated components of the square root factor and perform calculations only for entries that are actually affected by the new measurements. Thus, we obtain a local and constant-time operation for exploration tasks.

For trajectories with loops, *periodic variable reordering* prevents unnecessary fill-in in the square root factor that would otherwise slow down the incremental factor update, as well as the recovery of the current estimate by back substitution. Fill-in is a well-known problem for matrix factorization, as the resulting matrix factor can contain a large number of additional nonzero entries that reduce or even destroy the sparsity with associated negative consequences for the computational complexity. As the variable ordering influences the amount of fill-in obtained, it allows us to influence the computational complexity involved in solving the system. While finding the best variable ordering is infeasible, good heuristics are available. We perform incremental updates most of the time but periodically apply a variable reordering heuristic, followed by refactoring the resulting measurement Jacobian.

Incremental mapping also requires *online data association*; hence, we provide efficient algorithms to access the relevant estimation uncertainties from the incrementally updated square root factor. The key insight for efficient retrieval of these quantities is that only some entries of the full covariance matrix are needed, at least some of which can readily be accessed [9]. We present an efficient algorithm that avoids calculating all entries

Manuscript received November 9, 2007. First published November 18, 2008; current version published December 30, 2008. This paper was recommended for publication by Associate Editor A. Martinelli and Editor L. Parker upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant IIS-0448111 and by Defense Advanced Research Projects Agency (DARPA) under Grant FA8650-04-C-7131. This work was presented in part at the International Joint Conference on Artificial Intelligence, Hyderabad, India, January 2007 and in part at the IEEE International Conference on Robotics and Automation, Rome, Italy, April 2007.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This includes four multimedia audio video interleave (AVI) format movie clips, which visualize different aspects of the algorithms presented in this paper. This material is 4.3 MB in size.

M. Kaess is with the Massachusetts Institute of Technology, Cambridge, MA 02138 USA (e-mail: kaess@mit.edu).

A. Ranganathan is with Honda Research Institute, Cambridge, MA 02142 USA (e-mail: ananth@cc.gatech.edu).

F. Dellaert is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: dellaert@cc.gatech.edu).

Color versions of one or more of the figures are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2008.2006706

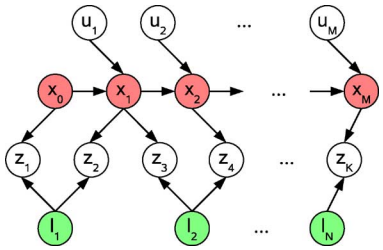


Fig. 1. Bayesian belief network representation of the SLAM problem. $\mathbf{x}_i$ is the state of the robot at time i, $\mathbf{l}_j$ the location of landmark j, $\mathbf{u}_i$ the control input at time i, and $\mathbf{z}_k$ the kth landmark measurement.

of the covariance matrix and instead focuses on the relevant parts by exploiting the sparsity of the square root factor. In addition to this exact solution, we also provide conservative estimates that are again derived from the square root factor.

We evaluate iSAM on simulated and real-world datasets for both landmark-based and pose-only settings. The pose-only setting is a special case of iSAM, in which no landmarks are used, general pose constraints between pairs of poses are considered in addition to odometry. The results show that iSAM provides an efficient and exact solution for both types of SLAM settings. They also show that the square root factor indeed remains sparse even for large-scale environments with a significant number of loops.

This paper is organized as follows. We continue in the next section with a review of the smoothing approach to SLAM as a least squares problem, providing a solution based on matrix factorization. We then present our incremental solution in Section III, addressing the topics of loops in the trajectory and nonlinear measurement functions in Section IV. For data association, we discuss efficient algorithms to retrieve the necessary components of the estimation uncertainty in Section V. We follow up with experimental results in Section VI, and finally, discuss related work in Section VII.

II. SAM: Smoothing Approach to SLAM

In this section, we review the formulation of the SLAM problem in the context of smoothing. The discussion mostly follows [4] but deviates in that it focuses on a solution based on QR factorization. We start with the probabilistic model underlying the smoothing approach to SLAM, and show how inference on this model leads to a least squares problem. We then obtain an equivalent linear formulation in matrix form by linearization of the measurement functions. We finally provide an efficient batch solution based on QR matrix factorization.

A. Probabilistic Model for SLAM

We formulate the SLAM problem in terms of the *belief network* model shown in Fig. 1. We denote the robot states by $X = \{\mathbf{x}_i\}$ with $i \in 0, \ldots, M$, the landmarks by $L = \{\mathbf{l}_j\}$ with $j \in 1, \ldots, N$, the control inputs by $U = \{\mathbf{u}_i\}$ for $i \in 1, \ldots, M$, and finally, the landmark measurements by $Z = \{\mathbf{z}_k\}$ with $k \in 1, \ldots, K$. The joint probability of all variables and measurements is given by

$$P(X, L, U, Z) \propto P(\mathbf{x}_0) \prod_{i=1}^{M} P(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i) \prod_{k=1}^{K} P(\mathbf{z}_k|\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) \tag{1}$$

where $P(\mathbf{x}_0)$ is a prior on the initial state, $P(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i)$ is the motion model, which is parametrized by the control input $\mathbf{u}_i$, and $P(\mathbf{z}_k|\mathbf{x}_{i_k}, \mathbf{l}_{j_k})$ is the landmark measurement model. Initially, we assume known correspondences (i_k, j_k) for each measurement $\mathbf{z}_k$. The problem of establishing correspondences, which is also called data association, is deferred until Section V.

We assume Gaussian measurement models, as is standard in the SLAM literature [10]. The process model

$$\mathbf{x}_i = f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) + \mathbf{w}_i \tag{2}$$

describes the odometry sensor or scan-matching process, where $\mathbf{w}_i$ is normally distributed zero-mean process noise with covariance matrix Λ_i. The Gaussian measurement equation

$$\mathbf{z}_k = h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) + \mathbf{v}_k \tag{3}$$

models the robot's landmark sensors, where $\mathbf{v}_k$ is normally distributed zero-mean measurement noise with covariance Γ_k.

B. SLAM as a Least Squares Problem

To obtain an optimal estimate for the set of unknowns given all available measurements, we convert the problem into an equivalent least squares formulation based on a *maximum a posteriori* (MAP) estimate. As we perform smoothing rather than filtering, we are interested in the MAP estimate for the *entire trajectory* X and the map of landmarks L, given the control inputs U and the landmark measurements Z. The MAP estimate X^*, L^* for trajectory and map is obtained by minimizing the negative log of the joint probability from (1)

$$\begin{aligned} X^*, L^* &= \arg\max_{X,L} P(X, L, U, Z) \\ &= \arg\min_{X,L} -\log P(X, L, U, Z). \end{aligned} \tag{4}$$

Combined with the process and measurement models, this leads to the following nonlinear least squares problem:

$$\begin{aligned} X^*, L^* = \arg\min_{X,L} \Bigg\{ &\sum_{i=1}^{M} \|f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i\|^2_{\Lambda_i} \\ &+ \sum_{k=1}^{K} \|h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \mathbf{z}_k\|^2_{\Gamma_k} \Bigg\} \end{aligned} \tag{5}$$

where we use the notation $\|\mathbf{e}\|^2_\Sigma = \mathbf{e}^T \Sigma^{-1} \mathbf{e}$ for the squared Mahalanobis distance with covariance matrix Σ. Note that we have dropped the prior $P(x_0)$ on the first pose for simplicity.

If the process models f_i and measurement functions h_k are nonlinear and a good linearization point is not available, nonlinear optimization methods are used, such as Gauss–Newton or the Levenberg–Marquardt algorithm, which solve a succession of linear approximations to (5) to approach the minimum [11]. This is similar to the extended Kalman filter (EKF) approach to SLAM as pioneered by [12] but allows for iterating multiple

times to convergence, thus avoiding the problems arising from wrong linearization points.

As derived in the Appendix, linearization of the measurement equations and subsequent collection of all components in one large linear system yields the following standard least squares problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|A\boldsymbol{\theta} - \mathbf{b}\|^2 \tag{6}$$

where the vector $\boldsymbol{\theta} \in \mathbb{R}^n$ contains all pose and landmark variables, the matrix $A \in \mathbb{R}^{m \times n}$ is a large but *sparse measurement Jacobian* with m measurement rows, and $\mathbf{b} \in \mathbb{R}^m$ is the right-hand side (RHS) vector. Such sparse least squares systems are converted into an ordinary linear equation system by setting the derivative of $\|A\boldsymbol{\theta} - \mathbf{b}\|^2$ to 0, resulting in the so-called normal equations $A^T A\boldsymbol{\theta} = A^T b$. This equation system can be solved by Cholesky decomposition of $A^T A$.

C. Solving by QR Factorization

We apply standard QR matrix factorization to the measurement Jacobian A to solve the least squares problem (6). In contrast to Cholesky factorization, this avoids having to calculate the information matrix $A^T A$ with the associated squaring of the matrix condition number. QR factorization of the measurement Jacobian A yields

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{7}$$

where $R \in \mathbb{R}^{n \times n}$ is the upper triangular square root information matrix (note that the information matrix is given by $R^T R = A^T A$) and $Q \in \mathbb{R}^{m \times m}$ is an orthogonal matrix. We apply this factorization to the least squares problem (6)

$$\begin{aligned} \|A\boldsymbol{\theta} - \mathbf{b}\|^2 &= \left\| Q \begin{bmatrix} R \\ 0 \end{bmatrix} \boldsymbol{\theta} - \mathbf{b} \right\|^2 \\ &= \left\| Q^T Q \begin{bmatrix} R \\ 0 \end{bmatrix} \boldsymbol{\theta} - Q^T \mathbf{b} \right\|^2 \\ &= \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} \boldsymbol{\theta} - \begin{bmatrix} \mathbf{d} \\ \mathbf{e} \end{bmatrix} \right\|^2 \\ &= \|R\boldsymbol{\theta} - \mathbf{d}\|^2 + \|\mathbf{e}\|^2 \end{aligned} \tag{8}$$

where we define $[\mathbf{d}, \mathbf{e}]^T := Q^T \mathbf{b}$ with $\mathbf{d} \in \mathbb{R}^n$ and $\mathbf{e} \in \mathbb{R}^{m-n}$. Equation (8) becomes minimal if and only if $R\boldsymbol{\theta} = \mathbf{d}$, leaving the second term $\|\mathbf{e}\|^2$ as the residual of the least squares problem. Therefore, QR factorization simplifies the least squares problem to a linear system with a single unique solution $\boldsymbol{\theta}^*$

$$R\boldsymbol{\theta}^* = \mathbf{d}. \tag{9}$$

Most of the work for solving this equation system has already been done by the QR decomposition, because R is upper triangular, so simple back substitution can be used. The result is the least squares estimate $\boldsymbol{\theta}^*$ for the complete robot trajectory as well as the map conditioned on all measurements.

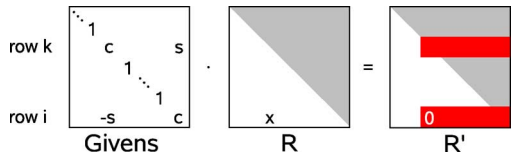


Fig. 2. Using a Givens rotation as a step in transforming a general matrix into upper triangular form. The entry marked "x" is eliminated, changing some of the entries marked in red (dark), depending on sparsity.

III. iSAM: Incremental Smoothing and Mapping

We present our incremental smoothing and mapping (iSAM) algorithm that avoids unnecessary calculations by directly updating the square root factor when a new measurement arrives. We begin with a review of Givens rotations for batch and incremental QR matrix factorization. We then apply this technique to update the square root factor and discuss how to retrieve the map and trajectory. We also analyze performance for exploration tasks in simulated environments.

A. Matrix Factorization by Givens Rotations

A standard approach to obtain the QR factorization of a matrix A uses *Givens rotations* [8] to clean out all entries below the diagonal, one at a time. While this is not the preferred way to do full QR factorization, we will later see that this approach readily extends to factorization updates, which are needed to incorporate new measurements. The process starts from the leftmost nonzero entry, and proceeds columnwise, by applying the Givens rotation

$$\Phi := \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \tag{10}$$

to rows i and k, with $i > k$, as shown in Fig. 2. The parameter ϕ is chosen so that a_{ik}, the (i,k) entry of A, becomes 0. After all entries below the diagonal are zeroed out in this manner, the upper triangular entries contain the R factor. The orthogonal rotation matrix Q is typically dense, which is why this matrix is never explicitly formed in practice. Instead, it is sufficient to update the RHS vector $\mathbf{b}$ with the same rotations that are applied to A.

Solving a least squares system $A\mathbf{x} = \mathbf{b}$ by matrix factorization using Givens rotations is numerically stable and accurate to machine precision if the rotations are determined as follows [8, Sec. 5.1]:

$$\begin{aligned} &(\cos\phi, \sin\phi) \\ &= \begin{cases} (1,0)\,, & \text{if } \beta = 0 \\ \left(\dfrac{-\alpha}{\beta\sqrt{1+(\alpha/\beta)^2}}, \dfrac{1}{\sqrt{1+(\alpha/\beta)^2}} \right), & \text{if } |\beta| > |\alpha| \\ \left(\dfrac{1}{\sqrt{1+(\beta/\alpha)^2}}, \dfrac{-\beta}{\alpha\sqrt{1+(\beta/\alpha)^2}} \right), & \text{otherwise} \end{cases} \end{aligned}$$

where $\alpha := a_{kk}$ and $\beta := a_{ik}$.

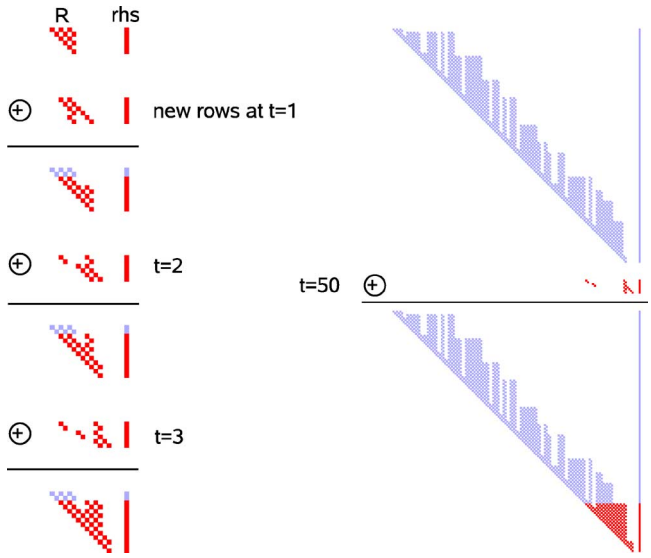


Fig. 3. Updating the factored representation of the smoothing information matrix for the example of an exploration task. New measurement rows are added to the upper triangular factor R and the RHS. The left column shows the updates for the first three steps, and the right column shows the update after 50 steps. The update operation is symbolically denoted by $\oplus$. Entries that remain unchanged are shown in light blue (gray). For the exploration task, the number of operations is bounded by a constant.

B. Incremental Updating

When a new measurement arrives, it is more efficient to modify the previous factorization directly by *QR-updating*, instead of updating and refactoring the matrix A. Adding a new measurement row $\mathbf{w}^T$ and RHS γ into the current factor R, RHS $\mathbf{d}$ yields a new system that is not yet in the correct factorized form

$$\begin{bmatrix} Q^T & \\ & 1 \end{bmatrix} \begin{bmatrix} A \\ \mathbf{w}^T \end{bmatrix} = \begin{bmatrix} R \\ \mathbf{w}^T \end{bmatrix}, \qquad \text{new RHS: } \begin{bmatrix} \mathbf{d} \\ \gamma \end{bmatrix}. \quad (11)$$

Note that this is the same system that is obtained by applying Givens rotations to the updated matrix A' to eliminate all entries below the diagonal, except for the last (new) row. Therefore, Givens rotations can be determined that zero out this new row, yielding the updated factor R'. As for the full factorization, we simultaneously update the RHS with the same rotations to obtain $\mathbf{d}'$. Several steps of this update process are shown in Fig. 3.

New variables are added to the QR factorization by expanding the factor R by the appropriate number of empty columns and rows. This expansion is simply done before new measurement rows containing the new variables are added. At the same time, the RHS $\mathbf{d}$ is augmented by the same number of zeros.

C. Incremental SAM

Applying the Givens-rotations-based updating process to the square root factor provides the basis for our efficient incremental solution to smoothing and mapping. In general, the maximum number of Givens rotations needed for adding a new measurement row is n. However, as both R and the new measurement row are sparse, only a constant number of Givens rotations are needed. Furthermore, new measurements typically refer to recently added variables so that often only the rightmost part of the new measurement row is (sparsely) populated.

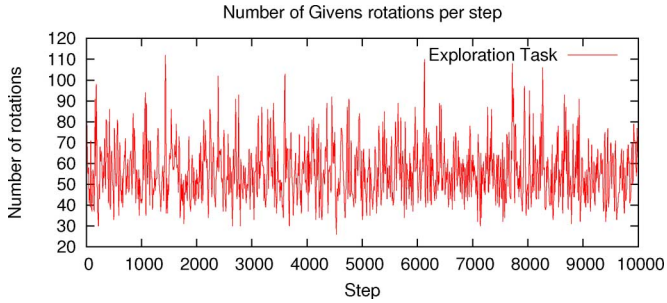


Fig. 4. Number of Givens rotations needed per step for a simulated linear exploration task. In each step, the square root factor is updated by adding the new measurement rows by Givens rotations. The number of rotations is independent of the length of the trajectory.

For a linear exploration task, incorporating a set of new landmark and odometry measurements takes constant time. Examples of such updates are shown in Fig. 3. The simulation results in Fig. 4 show that the number of rotations needed is independent of the size of the trajectory and the map. Updating the square root factor therefore takes $O(1)$ time, but this does not yet provide the current least squares estimate.

The current least squares estimate for the map and the full trajectory can be obtained at any time by back substitution in time linear in the number of variables. While back substitution has quadratic time complexity for general dense matrices, it is more efficient in the context of iSAM. For exploration tasks, the information matrix is band diagonal. Therefore, the square root factor has a constant number of entries per column, independent of the number of variables n that make up the map and trajectory. Therefore, back substitution requires $O(n)$ time in iSAM. In the earlier linear exploration example, this results in about 0.12 s computation time after $10\,000$ steps. In fact, for this special case of exploration, only a constant number of the most recent values has to be retrieved in each step to obtain the exact solution incrementally.

IV. Loops and Nonlinear Functions

We discuss how iSAM deals with loops in the robot trajectory, as well as with nonlinear sensor measurement functions. While realistic SLAM applications include much exploration, the robot often returns to previously visited places, closing loops in the trajectory. We discuss the consequences of loops on the matrix factorization, and show how to use periodic variable reordering to avoid unnecessary increases in computational complexity. Furthermore, real-world sensing often leads to nonlinear measurement functions, typically by means of angles such as bearing to a landmark or the robot heading. We show how iSAM allows relinearization of measurement equations and suggest a solution in connection with the periodic variable reordering.

A. Loops and Periodic Variable Reordering

Environments with loops do not have the nice property of local updates, resulting in increased complexity. In contrast to

pure exploration, where a landmark is only visible from a small part of the trajectory, a loop in the trajectory brings the robot back to a previously visited location. A loop introduces correlations between current poses and previously observed landmarks, which themselves are connected to earlier parts of the trajectory. An example based on a simulated environment with a robot trajectory in the form of a double 8-loop is shown in Fig. 5.

Loops in the trajectory can result in a significant increase of computational complexity through a large increase of nonzero entries in the factor matrix. Nonzero entries beyond the sparsity pattern of the information matrix are called *fill-in*. While the smoothing information matrix remains sparse even in the case of closing loops, the incremental updating of the factor matrix R leads to fill-in, as can be seen from Fig. 5(b).

We avoid fill-in by *variable reordering*, a technique well known in the linear algebra community, using a heuristic to efficiently find a good block ordering. The order of the columns in the information matrix influences the variable elimination order, and therefore, the resulting number of entries in the factor R also. While obtaining the best column variable ordering is NP-hard, efficient heuristics such as the column approximate minimum degree (COLAMD) ordering by Davis *et al.* [13] have been developed that yield good results for the SLAM problem, as shown in [4] and [5]. We apply this ordering heuristic to blocks of variables that correspond to the robot poses and landmark locations. As has been shown in [4], operating on these blocks leads to a further increase in efficiency as it exploits the special structure of the SLAM problem. The corresponding factor R after applying the COLAMD ordering shows negligible fill-in, as can be seen in Fig. 5(c).

We propose fast incremental updates with periodic variable reordering, combining the advantages of both methods. Factorization of the new measurement Jacobian after variable reordering is expensive when performed in each step. But combined with incremental updates, it avoids fill-in and still yields a fast algorithm as supported by the timing results in Fig. 5(d). In fact, as the log scale version of this figure shows, our solution is one to three orders of magnitude faster than either the purely incremental or the batch solution with the exception of occasional peaks caused by reordering of the variables and subsequent matrix factorization. In this example, we use a fixed interval of 100 steps, after which, we reorder the variables and refactor the complete matrix.

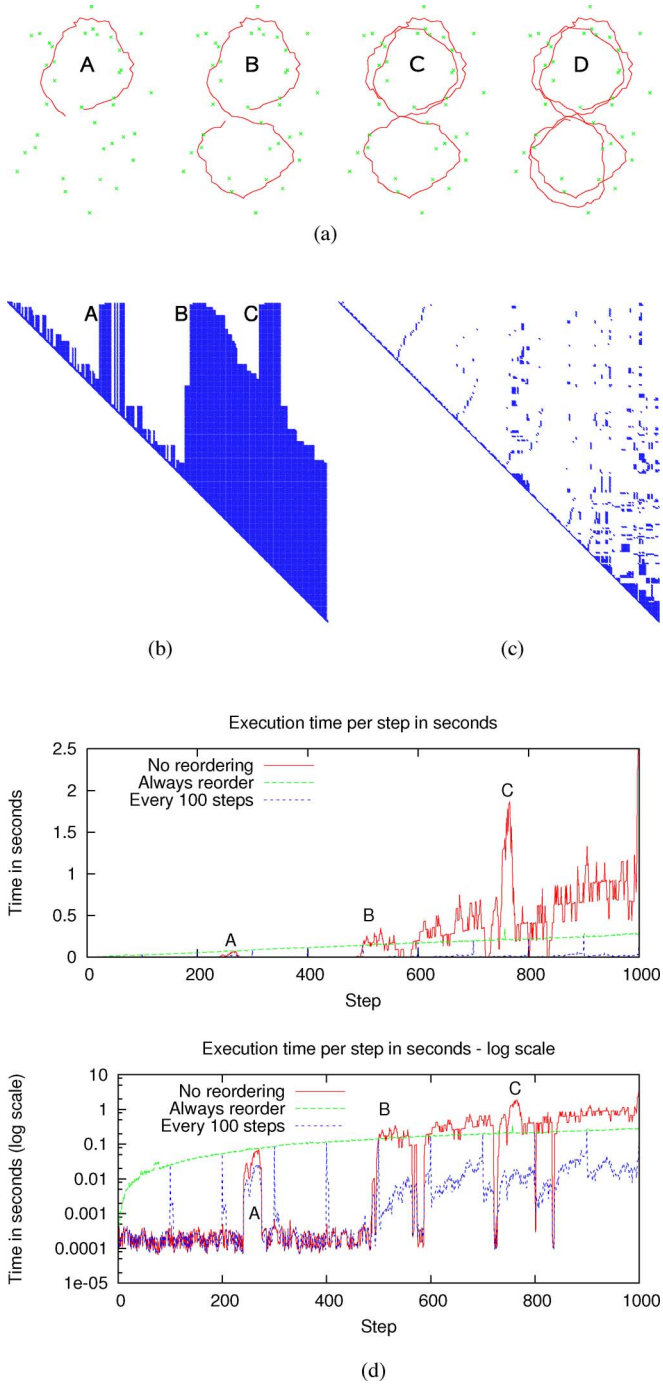


Fig. 5. For a simulated environment consisting of an 8-loop that is traversed twice (a), the upper triangular factor R shows significant fill-in (b), yielding bad performance (d, continuous red). Some fill-in occurs at the time of the first loop closing (A). Note that there are no negative consequences on the subsequent exploration along the second loop until the next loop closure occurs (B). However, the fill-in then becomes significant when the complete 8-loop is traversed for the second time, with a peak when visiting the center point of the 8-loop for the third time (C). After variable reordering according to an approximate minimum degree heuristic, the factor matrix again is completely sparse (c). In the presence of loops, reordering the variables after each step (d, dashed green) is sometimes less expensive than incremental updates. However, a considerable increase in efficiency is achieved by using fast incremental updates interleaved with only occasional variable reordering (d, dotted blue), here performed every 100 steps. (a) Simulated double 8-loop at interesting stages of loop closing (for simplicity, only a reduced example is shown here). (b) Factor R. (c) Same factor R after variable reordering. (d) Execution time per step for different updating strategies are shown in both (top) linear and (bottom) log scale.

B. Nonlinear Systems

While we have so far only discussed the case of linear measurement functions, SLAM applications are usually faced with nonlinear measurement functions. Angular measurements, such as the robot orientation or the bearing to a landmark, are the main source for nonlinear dependencies between measurements and variables. In that case, everything discussed so far is still valid. However, after solving the linearized version based on the current variable estimate, we might obtain a better estimate resulting in a modified Jacobian based on this new linearization point. For standard nonlinear optimization techniques, this process is iterated as explained in Section II-B until the change in

the estimate is sufficiently small. Convergence is guaranteed, at least to a local minimum, and the convergence speed is quadratic because we apply a second-order method.

As relinearization is not needed in every step, we propose combining it with periodic variable reordering. The SLAM problem is different from a standard nonlinear optimization as new measurements arrive sequentially. First, we already have a very good estimate for most if not all of the old variables. Second, measurements are typically fairly accurate on a local scale so that good estimates are available for a newly added robot pose as well as for newly added landmarks. We can therefore avoid calculating a new Jacobian and refactoring it in each step, which is a fairly expensive batch operation. Instead, for the results presented here, we combine relinearization with the periodic variable reordering for fill-in reduction as discussed in the previous section. In other words, in the variable reordering steps only, we also relinearize the measurement functions as the new measurement Jacobian has to be refactored anyway.

V. Data Association

As an incremental solution to the SLAM problem also requires incremental data association, we provide efficient algorithms to access the quantities of interest of the underlying estimation uncertainty. The data association problem in SLAM consists of matching measurements to their corresponding landmarks. While data association is required on a frame-to-frame basis, it is particularly problematic when closing large loops in the trajectory. We start with a general discussion of the data association problem, based on a maximum likelihood (ML) formulation. We discuss how to access the exact values of interest as well as conservative estimates from the square root factor of the smoothing information matrix. We then compare the performance of these algorithms to fast inversion of the information matrix and to nearest-neighbor (NN) matching.

A. ML Data Association

The often used NN approach to data association is not sufficient in many cases, as it does not take into account the estimation uncertainties. NN assigns each measurement to the closest predicted landmark measurement. The NN approach corresponds to a minimum cost assignment problem, based on a cost matrix that contains all the prediction errors. Details of this minimum cost assignment problem and how to deal with spurious measurements are given in [14].

Instead of NN, we use the ML solution to data association [15], which is more sophisticated in that it takes into account the relative uncertainties between the current robot location and the landmarks in the map. The ML formulation can again be reduced to a minimum cost assignment problem, using a Mahalanobis distance rather than the Euclidean distance. This Mahalanobis distance is based on the projection Ξ of the combined pose and landmark uncertainties Σ into the sensor measurement space

$$\Xi = J\Sigma J^T + \Gamma \tag{12}$$

where Γ is the measurement noise, and J is the Jacobian of the linearized measurement function h from (3). We use the Jonker–Volgenant–Castanon (JVC) assignment algorithm [16] to solve this assignment problem.

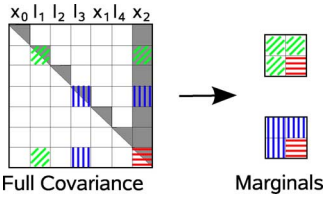


Fig. 6. Only a small number of entries of the dense covariance matrix are of interest for data association. In this example, the marginals between the latest pose $\mathbf{x}_2$ and the landmarks $\mathbf{l}_1$ and $\mathbf{l}_3$ are retrieved. The entries that need to be calculated in general are marked in gray. Only the triangular blocks along the diagonal and the rightmost block column are needed due to symmetry. Based on our factored information matrix representation, the last column can be obtained by simple back substitution. As we show here, the blocks on the diagonal can either be calculated exactly by calculating only the entries corresponding to nonzeros in the sparse factor R or approximated by conservative estimates for online data association.

B. Marginal Covariances

Knowledge of the relative uncertainties between the current pose $\mathbf{x}_i$ and any visible landmark $\mathbf{l}_j$ is needed for the ML solution to the data association problem. These marginal covariances

$$\Sigma = \begin{bmatrix} \Sigma_{jj} & \Sigma_{ij}^T \\ \Sigma_{ij} & \Sigma_{ii} \end{bmatrix} \tag{13}$$

contain blocks from the diagonal of the full covariance matrix, as well as the last block row and column, as is shown in Fig. 6. Note that the off-diagonal blocks are essential because the uncertainties are relative to an arbitrary reference frame, which is often fixed at the origin of the trajectory.

Calculating the full covariance matrix to recover these entries is not an option because the covariance matrix is always completely populated with n^2 entries, where n is the number of variables. However, calculating all entries is not necessary if we always add the most recent pose at the end, i.e., we first add the newly observed landmarks, then optionally perform variable reordering, and finally add the next pose. In that case, only some triangular blocks on the diagonal and the last block column are needed as indicated by the gray areas in Fig. 6, and the remaining entries are given by symmetry. Our factored representation allows us to retrieve the exact values of interest without having to calculate the complete dense covariance matrix, as well as to obtain a more efficient conservative estimate.

Common to both exact solution and conservative estimates are the recovery of the last columns of the covariance matrix from the square root factor, which can be done efficiently by back substitution. The exact pose uncertainty Σ_{ii} and the covariances Σ_{ij} can be recovered in linear time, based on the sparse R factor. As we choose the current pose to be the last variable in our factor R, the last block column X of the full covariance matrix $(R^T R)^{-1}$ contains Σ_{ii} as well as all Σ_{ij}, as observed in [9], but instead of having to keep an incremental estimate of these quantities, we can retrieve the exact values efficiently from the factor R by back substitution. With $d_{\mathbf{x}}$ the dimension

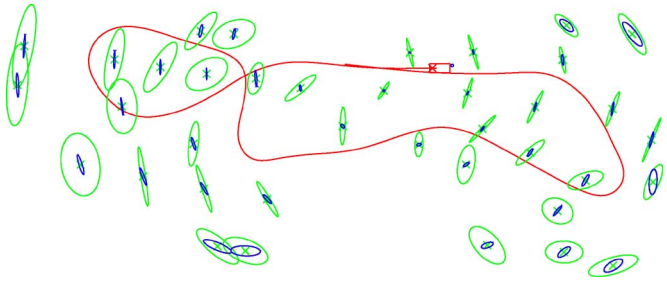


Fig. 7. Comparison of marginal covariance estimates *projected into the current robot frame* (robot indicated by red rectangle) for a short trajectory (red curve) and some landmarks (green crosses). Conservative covariances (green, large ellipses) as well as the exact covariances (blue, smaller ellipses) obtained by our fast algorithm are shown. Note that the exact covariances based on full inversion are also shown (orange, mostly hidden by blue).

of the last pose, we define $B \in \mathbb{R}^{n \times d_{\mathbf{x}}}$ as the last $d_{\mathbf{x}}$ unit vectors

$$B = \begin{bmatrix} 0_{(n-d_{\mathbf{x}}) \times d_{\mathbf{x}}} \\ I_{d_{\mathbf{x}} \times d_{\mathbf{x}}} \end{bmatrix} \tag{14}$$

and solve

$$R^T R X = B \tag{15}$$

by a forward and a back substitution

$$R^T Y = B, \qquad RX = Y. \tag{16}$$

The key to efficiency is that we never have to recover a full dense matrix, but due to R being upper triangular, we immediately obtain

$$Y = [0, \ldots, 0, R_{ii}^{-1}]^T. \tag{17}$$

Recovering these columns is efficient because only a constant number of $d_{\mathbf{x}}$ back substitutions are needed.

C. Conservative Estimates

Conservative estimates for the structure uncertainties Σ_{jj} are obtained from the initial uncertainties, as proposed by Eustice [9]. As the uncertainty can never grow when new measurements are added to the system, the initial uncertainties $\tilde{\Sigma}_{jj}$ provide conservative estimates. These are obtained by

$$\tilde{\Sigma}_{jj} = \bar{J} \begin{bmatrix} \Sigma_{ii} & \\ & \Gamma \end{bmatrix} \bar{J}^T \tag{18}$$

where $\bar{J}$ is the Jacobian of the linearized back-projection function (an inverse of the measurement function is not always available, for example, for the bearing-only case), and Σ_{ii} and Γ are the current pose uncertainty and the measurement noise, respectively. Fig. 7 provides a comparison of the conservative and exact covariances. A more tight conservative estimate on a landmark can be obtained after multiple measurements are available, or later in the process by means of the exact algorithm that is presented next.

D. Exact Covariances

Recovering the exact structure uncertainties Σ_{jj} is not straightforward, as they are spread out along the diagonal, but can still be done efficiently by again exploiting the sparsity structure of R. In general, the covariance matrix is obtained as the inverse of the information matrix

$$\Sigma := (A^T A)^{-1} = (R^T R)^{-1} \tag{19}$$

based on the factor R by noting that

$$R^T R \Sigma \; = \; I \tag{20}$$

and performing a forward, followed by a back substitution

$$R^T Y = I, \qquad R\Sigma = Y. \tag{21}$$

Because the information matrix is not band diagonal in general, this would seem to require calculating all n^2 entries of the fully dense covariance matrix, which is infeasible for any nontrivial problem. This is where the sparsity of the factor R is of advantage again. An efficient method for recovering exactly all entries σ_{ij} of the covariance matrix Σ that coincide with nonzero entries in the factor $R = (r_{ij})$ is presented in [17] and [18]

$$\sigma_{ll} = \frac{1}{r_{ll}} \left(\frac{1}{r_{ll}} - \sum_{j=l+1, r_{lj} \neq 0}^{n} r_{lj} \sigma_{jl} \right) \tag{22}$$

$$\sigma_{il} = \frac{1}{r_{ii}} \left(- \sum_{j=i+1, r_{ij} \neq 0}^{l} r_{ij} \sigma_{jl} - \sum_{j=l+1, r_{ij} \neq 0}^{n} r_{ij} \sigma_{lj} \right) \tag{23}$$

for $l = n, \ldots, 1$ and $i = l-1, \ldots, 1$, where the other half of the matrix is given by symmetry. Note that the summations apply only to nonzero entries of single columns or rows of the sparse matrix R. This means that in order to obtain the top leftmost entry of the covariance matrix, we at most have to calculate all other entries that correspond to nonzeros in R. The algorithm has $O(n)$ time complexity for band-diagonal matrices and matrices with only a small number of entries far from the diagonal but can be more expensive for general sparse R.

Based on a dynamic programming approach, our algorithm provides access to all entries of interest for data association. Fig. 7 shows the marginal covariances obtained by this algorithm for a small example. Note that they coincide with the exact covariances obtained by full inversion. As the upper triangular parts of the block diagonals of R are fully populated, and due to symmetry of the covariance matrix, we obtain all block diagonals, and therefore, the structure uncertainties Σ_{jj} also. Even if any of these populated entries in R happen to be zero, or if additional entries are needed that are outside the sparsity pattern of R, they are easily accessible. We use a dynamic programming approach that obtains the entries we need and automatically calculates any intermediate entries as required. This also allows efficient retrieval of additional quantities that may be required for other data association techniques, such as the joint compatibility test [19].

E. Evaluation

Table I compares the execution times of NN as well as ML data association for different methods of obtaining the marginal covariances. The results are based on a simulated environment

TABLE I
EXECUTION TIMES FOR DIFFERENT DATA ASSOCIATION TECHNIQUES FOR A SIMULATED LOOP

	Execution time		
	Overall	Avg./step	Max./step
NN	$2.03s$	$4.1ms$	$81ms$
ML conservative	$2.80s$	$5.6ms$	$95ms$
ML exact, efficient	$27.5s$	$55ms$	$304ms$
ML exact, full	$429s$	$858ms$	$3300ms$

The times include updating of the factorization, solving for all variables, and performing the respective data association technique, for every step.

with a 500-pose loop and 240 landmarks, with significant measurement noise added. Undetected landmarks and spurious measurements are simulated by replacing 3% of the measurements by random measurements. The NN approach is dominated by the time needed for factorization updates and back substitution in each step. As these same calculations are also performed for all covariance-based approaches that follow, these times are a close approximation to the overall calculation time without data association.

The numbers show that while our exact algorithm is much more efficient than full inversion, our conservative solution is better suited for real-time application. In the second and third rows, the ML approach is evaluated for our fast conservative estimate and our efficient exact solution. The conservative estimate adds only a small overhead to the NN time, which is mostly due to back substitution to obtain the last columns of the exact covariance matrix. This is computationally the same as the back substitution used for solving, except that it covers a number of columns equal to the dimension of a single pose instead of just one. Recovering the exact marginal covariances becomes fairly expensive in comparison as additionally the block-diagonal entries have to be recovered. Our exact efficient algorithm is an order of magnitude faster compared to the direct inversion of the information matrix, even though we have used an efficient algorithm based on a sparse LDL^T matrix factorization [8]. However, this does not change the fact that all n^2 entries of the covariance matrix have to be calculated for the full inversion. Nevertheless, even our fast algorithm will get expensive for large environments and cannot be calculated in every step. Instead, as the uncertainties can never grow when new measurements are added, exact values can be calculated when time permits in order to update the conservative estimates.

VI. EXPERIMENTAL RESULTS AND DISCUSSION

While we evaluate the individual components of iSAM in their respective sections, we now evaluate our overall algorithm on simulated data as well as real-world datasets. The simulated data allow comparison with ground truth, while the real-world data prove the applicability of iSAM to practical problems. We explore both landmark-based as well as pose-constraint-based SLAM.

We have implemented iSAM in the functional programming language OCaml, using exact, automatic differentiation [20] to obtain the Jacobians. All timing results in this section are obtained on a 2-GHz Pentium M laptop.

A. Landmark-Based iSAM

Landmark-based iSAM works well in real-world settings, even in the presence of many loops in the robot trajectory. We have evaluated iSAM on the Sydney Victoria Park dataset, which is a popular test dataset in the SLAM community that consists of laser-range data and vehicle odometry, recorded in a park with sparse tree coverage. It contains 7247 frames along a trajectory of 4 km length, recorded over a time frame of 26 min. As repeated measurements taken by a stopped vehicle do not add any new information, we have dropped these, leaving 6969 frames. We have extracted 3640 measurements of landmarks from the laser data by a simple tree detector.

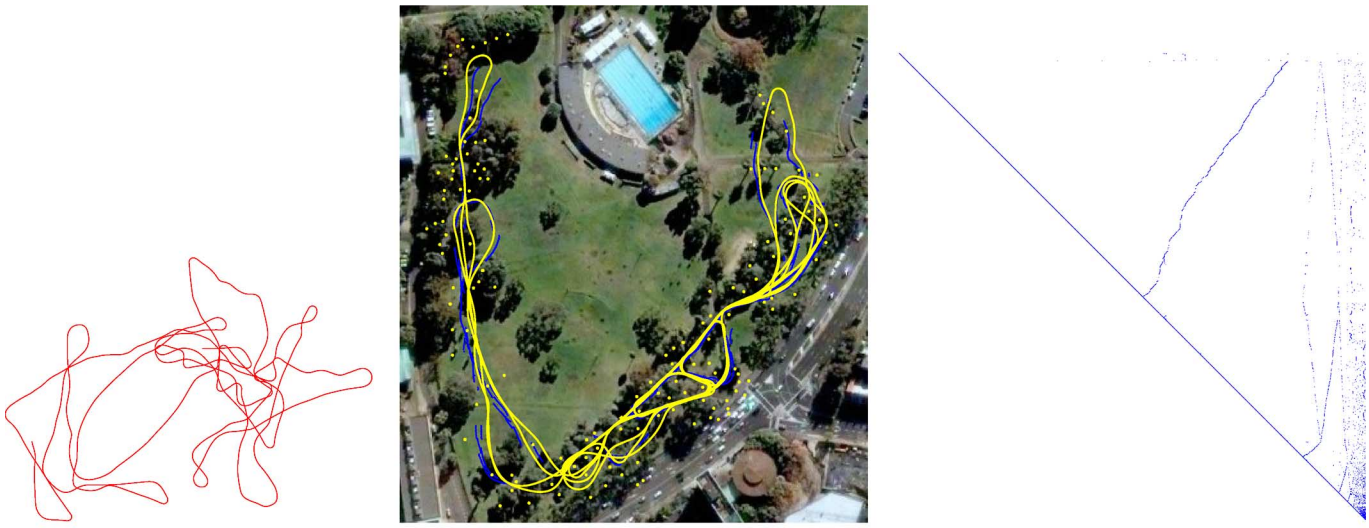
iSAM with unknown correspondences runs comfortably in real time. Performing data association based on conservative estimates, the incremental reconstruction, including solving for all variables after each new frame was added, took 464 s or 7.7 min, which is significantly less than the 26 min it took to record the data. That means that even though the Victoria Park trajectory contains a significant number of loops (several places are traversed eight times), increasing fill-in, iSAM is still over three times faster than real time. The resulting map contains 140 distinct landmarks, as shown in Fig. 8(b). Solving after every step is not necessary, as the measurements are fairly accurate locally, thereby providing good estimates. Calculating all variables only every ten steps yields a significant improvement to 270 s or 4.5 min.

Naturally, iSAM runs even faster with known data association, for example, due to uniquely identifiable landmarks. For this test, we use the correspondences that were automatically obtained before. Under known correspondences, the time reduces to 351 s or 5.9 min. The difference is mainly caused by the back substitution over the last three columns to obtain the off-diagonal entries in each step that are needed for data association. The decrease is not significant because a similar back substitution over a single column still has to be performed to solve for all variables in each step, but as a consequence, solving by back substitution only every ten steps now significantly reduces the time to 159 s or 2.7 min.

More importantly, iSAM still performs in real time toward the end of the trajectory, where the computations get more expensive. The average calculation time for the final 100 steps are the most expensive ones to compute as the number of variables is largest. Even for the slow case of retrieving the full solution after every step, iSAM takes in average 0.120 and 0.095 s per step, with and without data association, respectively. These results compare favorably to the 0.22 s needed for real-time performance. These computation times include a full linearization, COLAMD variable reordering step and matrix factorization, which took 1.8 s in total in both cases. Despite all the loops, the final factor R, as shown in Fig. 8(c), is still sparse, with 207 422 entries for 21 187 variables, yielding an average of only 9.79 entries per column.

B. Pose-Constraint-Based iSAM

iSAM can straightforwardly be applied to estimation problems without landmarks, purely based on pose constraints, as



(a) (b) (c)

Fig. 8. Results for the full Victoria Park sequence. Solving the complete problem including data association in each step took 7.7 min on a laptop computer. For known correspondences, the time reduces to 5.9 min. Since the dataset is from a 26-min-long robot run, iSAM with unknown correspondences is over *three times faster than real time* in this case, calculating the complete and exact solution in each step. The trajectory and landmarks are shown in yellow (light), manually overlaid on an aerial image for reference. Differential GPS was not used in obtaining our experimental results but is shown in blue (dark) for comparison—note that in many places, it is not available. (a) Trajectory based on odometry only. (b) Trajectory and map after incremental optimiziation. (c) Final R factor with side length 21 187.

we show in this section based on known correspondences. Such pose constraints most commonly arise from scan-matching dense laser range data but can also be generated from visual input [9]. Pose constraints either connect subsequent poses similar to odometry measurements, or they connect two arbitrary poses when closing loops. Pose constraints are incorporated in a way similar to the odometry measurements by introducing new terms that represent the error between the predicted and the measured difference between a pair of poses. We evaluate pose-only iSAM on simulated as well as real-world datasets, assuming known data association, as the generation of pose constraints by scan-matching has been well studied and good algorithms are available [22], [23].

The incremental solution of iSAM is comparable in quality to the solution obtained by full nonlinear optimization. To allow ground truth comparison, we use the simulated Manhattan world from [21], shown in Fig. 9(a) and (b). This dataset contains 3500 poses and 5598 constraints, 3499 of which are odometry measurements. While the result in [21] seems to be better as the left part is more straightened out, our solution has a slightly lower normalized χ^2-value of 1.0406, compared with 1.0412. After one extra relinearization and back substitution, the normalized χ^2 is 1.0375, which is the same value that we obtain by full nonlinear optimization until convergence. These results show that iSAM is comparable in accuracy to the exact solution provided by square root SAM.

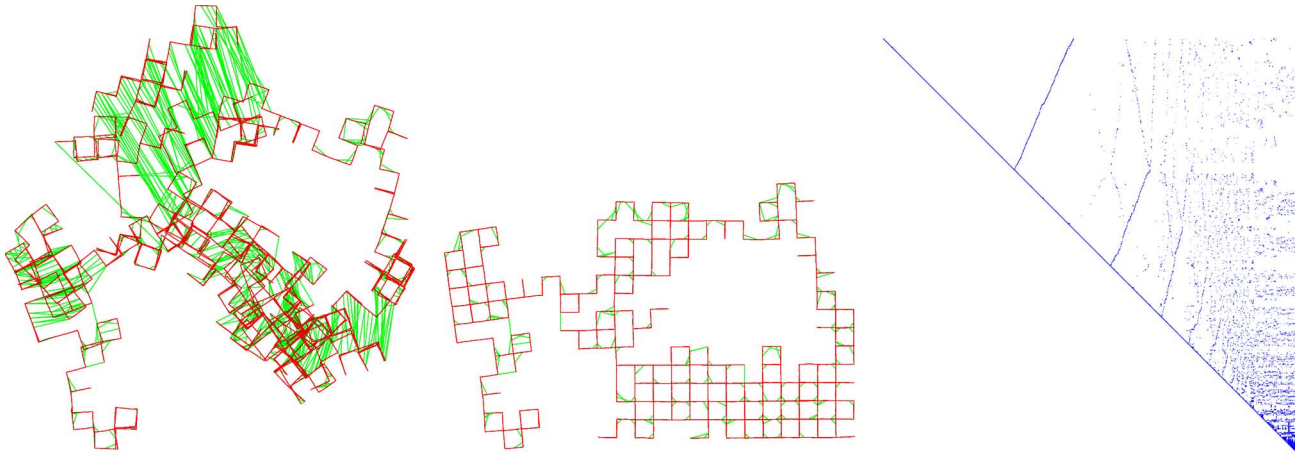
In terms of computational speed, iSAM also fares well for this dataset. Solving the full problem for each newly added pose, while reordering the variables and relinearizing the problem every 100 steps, takes iSAM 140.9 s, or an average of 40 ms/step. The last 100 steps take an average of 48 ms each, which includes 1.08 s for variable reordering and matrix factorization. The resulting R factor shown in Fig. 9(c) is sparse with 187 423 entries for a side length of 10 500.

iSAM also performs well on real-world data, both in quality of the solution as well as speed. We apply iSAM to two publicly available laser range datasets that appear in several publications. The first one is the Intel dataset shown in Fig. 10(b), providing a trajectory with many loops with continued exploration of the same environment in increasing detail. Preprocessing by scan matching results in 910 poses and 4453 constraints. iSAM obtains the full solution after each step, with variable reordering every 20 frames in 77.4 s, or about 85 ms/step. The last 100 steps take an average of 290 ms including 0.92 s for each of the five reordering and factorization steps. The final R factor is shown in Fig. 10(c), with 2730 variables containing 90 363 entries.

The second real-world pose-only example we use to evaluate iSAM is the Massachusetts Institute of Technology (MIT) Killian Court dataset shown in Fig. 11(b) that features much exploration with a few large-scale loops. Again, the dataset was preprocessed, yielding 1941 poses and 2190 pose constraints. iSAM takes 23.7 s for a complete solution after each step, or about 12.2 ms/step, with variable reordering after every 100 steps. The last 100 steps take an average of 31 ms including 0.36 s for reordering/refactorization. The final R factor with 52 414 entries is shown in Fig. 11(c).

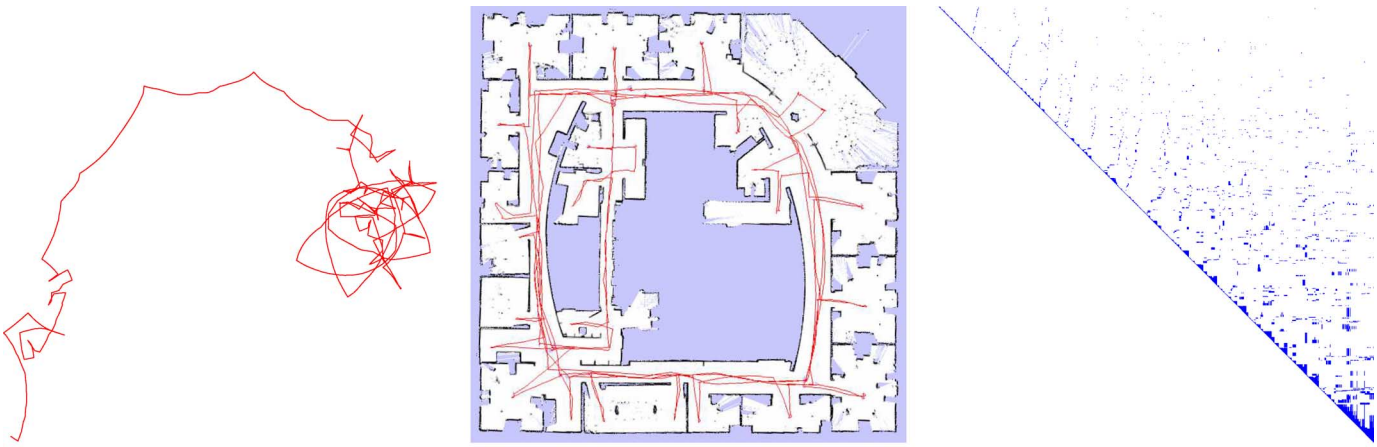
C. Sparsity of Square Root Factor

The complexity of iSAM depends heavily on the sparsity of the square root factor, as this affects both retrieving the solution as well as access to the covariances. Retrieving the solution from the square root factor requires back substitution, which usually has quadratic time complexity. However, if there are only a constant number of entries per column in the square root



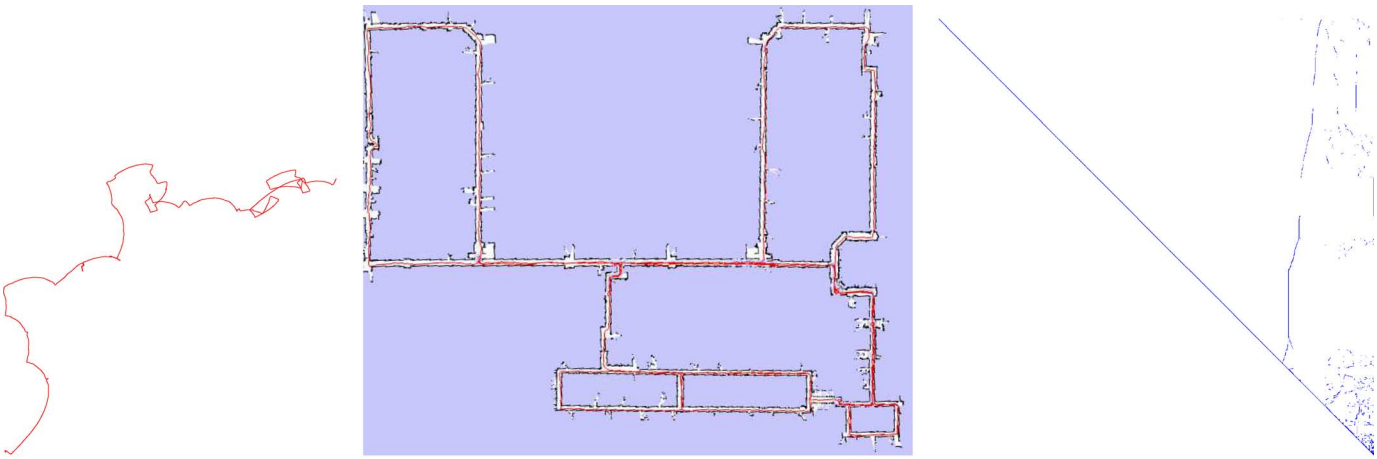
(a) (b) (c)

Fig. 9. iSAM results for the simulated Manhattan world from [21] takes about 40 ms/step. The resulting R factor has 187 423 entries, which corresponds to 0.34% or an average of 17.8 entries per column. (a) Original noisy dataset. (b) Trajectory after incremental optimization. (c) Final R factor with side length 10 500.



(a) (b) (c)

Fig. 10. Results from iSAM applied to the Intel dataset. iSAM calculates the full solution for 910 poses and 4453 constraints with an average of 85 ms/step while reordering the variables every 20 steps. The problem has $910 \times 3 = 2730$ variables and $4453 \times 3 = 13\,359$ measurement equations. The R factor contains 90 363 entries, which corresponds to 2.42% or 33.1 entries per column. (a) Trajectory based on odometry only. (b) Final trajectory and evidence grid map. (c) Final R factor with side length 2730.



(a) (b) (c)

Fig. 11. iSAM results for the MIT Killian Court dataset. iSAM calculates the full solution for the 1941 poses and 2190 pose constraints with an average of 12.2 ms per step. The R factor contains 52 414 entries for 5823 variables, which corresponds to 0.31% or 9.0 per column. (a) Trajectory based on odometry only. (b) Final trajectory and evidence grid map. (c) Final R factor with side length 5823.

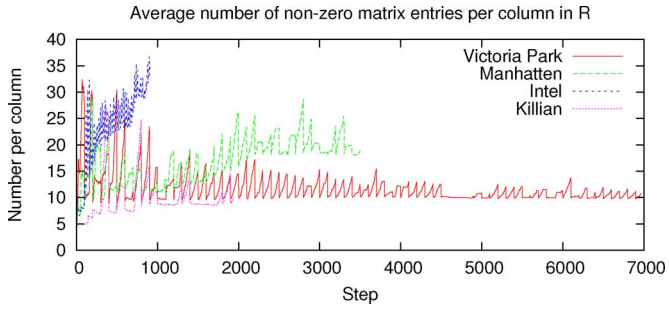


Fig. 12. Average number of entries per column in the R factor over time for the different datasets in this section. Even though the environments contain many loops, the average converges to a low constant in most cases, confirming our assumption that the number of entries per column is approximately independent of the number of variables n.

factor, then back substitution requires only $O(n)$ time. The same applies to retrieval of the last columns of the covariance matrix, which is the dominant cost for our conservative estimates.

Our results show that the number of entries per column is typically bound by a low constant. Fig. 12 shows how the density of the factor matrix R develops over time for each dataset used in this section. The densities increase initially, showing very large changes: increases are caused by incremental updating of the matrix factor, while sudden drops are the consequence of the periodic variable reordering. Except for the Intel sequence, all curves clearly converge to a low constant, explaining the good performance of iSAM. For the Intel dataset, the density increases more significantly because 1) the trajectory starts with a coarse run through the building, followed by more detailed exploration, and 2) there are unnecessarily many pose constraints to previous parts of the trajectory that lead to fill-in. This is also the reason for choosing a shorter interval for the periodic variable reordering (20 steps) than for all other datasets (100 steps). Nevertheless, as the results for that sequence show, iSAM still performs faster than needed for real time.

From a theoretical point of view, some bounds can be specified depending on the nature of the environment. Krauthausen *et al.* [24] provides an upper bound of $O(n \log n)$ for the fill-in of the square root factor for planar mapping with restricted sensor range. This means that the average fill-in per column is bound by $O(\log n)$. One special case is a pure exploration task, in which the robot never returns to previously mapped environments. In this case, the information matrix is band diagonal, and therefore, the factor matrix without variable reordering has a constant number of entries per column. Another special case is a robot that remains in the same restricted environment, continuously observing the same landmarks. In this case, the optimal solution is given by the variable ordering that puts all landmarks at the end, and iSAM performs a calculation similar to the Schur complement. The result is a system that requires computation time linear in the length of the trajectory but with a large constant that is quadratic in the constant number of landmarks. However, once the complete environment of the robot is mapped, there is no longer any need for SLAM, but rather localization based on the obtained map is sufficient.

VII. Related Work

There is a large body of literature in the field of robot localization and mapping, and we will only address closely related work as well as some of the most influential algorithms. A general overview of the area of SLAM can be found in [3], [10], [25], and [26]. Initial work on probabilistic SLAM was based on the EKF and is due to Smith *et al.* [12], building on earlier work [1], [27], [28]. It was soon shown that filtering is inconsistent in nonlinear SLAM settings [29], and much later work [30], [31] focused on reducing the effect of nonlinearities and providing more efficient but typically approximate solutions to deal with larger environments.

Smoothing in the SLAM context avoids these problems by keeping the complete robot trajectory as part of the estimation problem. It is also called the *full* SLAM problem [10] and is closely related to bundle adjustment [18] in photogrammetry, and to structure from motion (SFM) [32] in computer vision. While these are typically solved by batch processing of all measurements, SLAM by nature is an incremental problem. Also, SLAM provides additional constraints in the form of odometry measurements and an ordered sequence of poses that are not present in general SFM problems. The first smoothing approach to the SLAM problem is presented in [33], where the estimation problem is formulated as a network of constraints between robot poses. The first implementation [34] was based on matrix inversion.

A number of improved and numerically more stable algorithms have since been developed, based on well-known iterative techniques such as relaxation [10], [35], [36], gradient descent [37], [38], conjugate gradient [39], and, more recently, multilevel relaxation [40], [41]. The latter is based on a general multigrid approach that has proven very successful in other fields for solving systems of equations. While most of these approaches represent interesting solutions to the SLAM problem, they all have in common that it is very expensive to recover the uncertainties of the estimation process.

Recently, the information form of SLAM has become very popular. Filter-based approaches include the sparse-extended information filter (SEIF) [42] and the thin junction tree filter (TJTF) [43]. On the smoothing side, Treemap [44] exploits the information form, but applies multiple approximations to provide a highly efficient algorithm. Square root SAM [4], [5] provides an efficient and exact solution based on a batch factorization of the information matrix but does not address how to efficiently access the marginal covariances.

While iSAM includes the complete trajectory and map, this is not always the case when smoothing is applied. Instead, the complexity of the estimation problem can be reduced by omitting the trajectory altogether, for example, pursued by D-SLAM [45], where measurements are transformed into relative constraints between landmarks. Similarly, parts of the trajectory can be omitted as done by Folkesson and Christensen [37], where parts of the underlying graph are collapsed into so-called star nodes. Alternatively, the problem can be stated as estimating the trajectory only, which leads to an exactly sparse information matrix [46], [47], where image measurements are

converted to relative pose constraints. While this approach is similar to our pose-only case, they employ iterative methods to solve the estimation problem. While conservative estimates are available in Eustice's work [46] and, in fact, were the inspiration for our work, efficient access to the exact covariances is not possible based on the iterative solver.

Recently, some SLAM algorithms employ direct equation solvers based on Cholesky or QR factorization. Treemap [48] uses Cholesky factors to represent probability distributions in a tree-based algorithm. However, multiple approximations are employed to reduce the complexity, while iSAM solves the full and exact problem, and therefore allows relinearization of all variables at any time. The problem of data association is not addressed in [48]. Square root SAM [4] solves the estimation problem by factorization of the naturally sparse information matrix. However, the matrix has to be factored completely after each step, resulting in unnecessary computational burden. We have recently presented our incremental solution [49] and its extension to unknown data association [50] in conference versions of this extended paper.

To the best of our knowledge, updating of matrix factorizations has not been applied in the context of SLAM yet. However, it is a well-known technique in many areas, with applications such as computer vision [18], [51] and signal processing [52]. Golub and Van Loan [8] present general methods for updating matrix factorizations based on [53], [54], including the Givens rotations we use in this paper. Davis has done much work in the areas of variable ordering and factorization updates and provides highly optimized software libraries [13], [55] for various such tasks.

VIII. Conclusion

We have presented iSAM, which is a fast incremental solution to the SLAM problem that updates a sparse matrix factorization. By employing smoothing, we obtain a naturally sparse information matrix. As our approach is based on a direct equation solver using QR factorization, it has multiple advantages over iterative methods. Most importantly, iSAM allows access to the underlying estimation uncertainties, and we have shown how to access these efficiently, both the exact values as well as conservative estimates. We have further evaluated iSAM for both simulated and real-world data. In addition to the typical landmark-based application, we have also presented results for trajectory-only estimation problems.

iSAM compares favorably with other methods in terms of computational speed. Even though some other algorithms are faster, they either only provide approximations, or they do not provide access to the exact estimation uncertainties. iSAM in contrast, combines exact recovery of the map and trajectory with efficient retrieval of the covariances needed for data association, while providing real-time processing on readily available hardware. Therefore, we expect that approximate solutions will become less important in the future, and methods like iSAM that are based on direct equation solvers will take their place.

There is still potential for improvements in several aspects of iSAM. An incremental variable ordering that balances fill-in and the cost for incrementally updating the matrix could prove beneficial. This could allow completely eliminating any batch steps, because relinearization can also be performed incrementally, as typically only a small number of variables are affected. Finally, for both recovery of the solution and access to the covariances, back substitution could be restricted to only access the entries that are actually needed.

Our incremental solution should also be of interest beyond the applications presented here. One potential application is tracking a sensor in unknown settings for augmented reality as a cheap alternative to instrumenting the environment. We are working on visual SLAM applications of iSAM that will benefit from scalable and exact solutions, especially for unstructured outdoor environments. Furthermore, the real-time properties of iSAM allow for autonomous operation when mapping buildings or entire cities for virtual reality applications.

Appendix

For the completeness of this paper, we review how to linearize the measurement functions and collect all components of the nonlinear least squares objective function (5) into one general least squares formulation, following [4]. We linearize the measurement functions in (5) by Taylor expansion, assuming that either a good linearization point is available or that we are working on one iteration of a nonlinear optimization method. In either case, the first-order linearization of the process term in (5) is given by

$$\begin{aligned} & f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i \\ & \quad \approx \left\{ f_i\left(\mathbf{x}_{i-1}^0, \mathbf{u}_i\right) + F_i^{i-1} \delta \mathbf{x}_{i-1} \right\} - \left\{ \mathbf{x}_i^0 + \delta \mathbf{x}_i \right\} \\ & \quad = \left\{ F_i^{i-1} \delta \mathbf{x}_{i-1} - \delta \mathbf{x}_i \right\} - \mathbf{a}_i \end{aligned} \tag{24}$$

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

$$F_i^{i-1} := \left. \frac{\partial f_i(\mathbf{x}_{i-1}, \mathbf{u}_i)}{\partial \mathbf{x}_{i-1}} \right|_{\mathbf{x}_{i-1}^0} \tag{25}$$

and $\mathbf{a}_i := \mathbf{x}_i^0 - f_i(\mathbf{x}_{i-1}^0, \mathbf{u}_i)$ is the odometry prediction error (note that $\mathbf{u}_i$ here is given and, hence, is constant). The first-order linearizations of the measurement term in (5) are obtained similarly

$$\begin{aligned} & h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \mathbf{z}_k \\ & \quad \approx \left\{ h_k(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0) + H_k^{i_k} \delta \mathbf{x}_{i_k} + J_k^{j_k} \delta \mathbf{l}_{j_k} \right\} - \mathbf{z}_k \\ & \quad = \left\{ H_k^{i_k} \delta \mathbf{x}_{i_k} + J_k^{j_k} \delta \mathbf{l}_{j_k} \right\} - \mathbf{c}_k \end{aligned} \tag{26}$$

where $H_k^{i_k}$ and $J_k^{j_k}$ are, respectively, the Jacobians of the measurement function $h_k(.)$ with respect to a change in $\mathbf{x}_{i_k}$ and $\mathbf{l}_{j_k}$ evaluated at the linearization point $(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0)$

$$H_k^{i_k} := \left. \frac{\partial h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})}{\partial \mathbf{x}_{i_k}} \right|_{(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0)} \tag{27}$$

$$J_k^{j_k} := \left. \frac{\partial h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})}{\partial \mathbf{l}_{j_k}} \right|_{(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0)} \tag{28}$$

and $\mathbf{c}_k := \mathbf{z}_k - h_k(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0)$ is the measurement prediction error.

Using the linearized process and measurement models (24) and (26), respectively, (5) becomes

$$\delta\boldsymbol{\theta}^* = \arg\min_{\delta\boldsymbol{\theta}} \left\{ \sum_{i=1}^{M} \left\| F_i^{i-1} \delta\mathbf{x}_{i-1} + G_i^i \delta\mathbf{x}_i - \mathbf{a}_i \right\|_{\Lambda_i}^2 + \sum_{k=1}^{K} \left\| H_k^{i_k} \delta\mathbf{x}_{i_k} + J_k^{j_k} \delta\mathbf{l}_{j_k} - \mathbf{c}_k \right\|_{\Gamma_k}^2 \right\} \quad (29)$$

i.e., we obtain a *linear* least squares problem in $\delta\boldsymbol{\theta}$ that needs to be solved efficiently. To avoid treating $\delta\mathbf{x}_i$ in a special way, we introduce the matrix $G_i^i = -I_{d_\mathbf{x} \times d_\mathbf{x}}$.

By a simple change of variables, we can drop the covariance matrices Λ_i and Γ_k. With $\Lambda^{-1/2}$ the matrix square root of Λ, we can rewrite the Mahalanobis norm as follows:

$$\|\mathbf{e}\|_\Lambda^2 := \mathbf{e}^T \Lambda^{-1} \mathbf{e} = (\Lambda^{-T/2}\mathbf{e})^T (\Lambda^{-T/2}\mathbf{e}) = \left\| \Lambda^{-T/2}\mathbf{e} \right\|^2 \quad (30)$$

i.e., we can always eliminate Λ_i from (29) by premultiplying F_i^{i-1}, G_i^i, and $\mathbf{a}_i$ in each term with $\Lambda_i^{-T/2}$, and similarly eliminate Γ_k from the measurement terms. For scalar measurements, this simply means dividing each term by the measurement standard deviation. Next, we assume that this has been done and drop the Mahalanobis notation.

Finally, after collecting the Jacobian matrices into one large matrix A, and the vectors $\mathbf{a}_i$ and $\mathbf{c}_k$ into one right-hand side vector $\mathbf{b}$, we obtain the following standard least squares problem:

$$\delta\boldsymbol{\theta}^* = \arg\min_{\delta\boldsymbol{\theta}} \|A\delta\boldsymbol{\theta} - \mathbf{b}\|^2 \quad (31)$$

where we drop the $\delta\cdot$ notation for simplicity outside of this Appendix.

ACKNOWLEDGMENT

The authors are grateful to U. Frese, C. Stachniss, and E. Olson for helpful discussions and W. Burgard's group, especially G. Grisetti, for making their code available. They also thank E. Olson for sharing the Manhattan world dataset, D. Haehnel for the Intel dataset, M. Bosse for the Killian Court dataset, and E. Nebot and H. Durrant-Whyte for the Victoria Park dataset. Finally, they thank the reviewers for their valuable comments.

REFERENCES

[1] R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. J. Robot. Res.*, vol. 5, no. 4, pp. 56–68, 1987.

[2] J. Leonard, I. Cox, and H. Durrant-Whyte, "Dynamic map building for an autonomous mobile robot," *Int. J. Robot. Res.*, vol. 11, no. 4, pp. 286–289, 1992.

[3] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millennium*. San Mateo, CA: Morgan Kaufmann, 2003, pp. 1–35.

[4] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1203, Dec. 2006.

[5] F. Dellaert, "Square root SAM: Simultaneous location and mapping via square root information smoothing," presented at the Robot.: Sci. Syst. (RSS) Conf., 2005, Cambridge, MA.

[6] G. Bierman, *Factorization methods for discrete sequential estimatio* (ser. Mathematics in Science and Engineering 128). New York: Academic, 1977.

[7] P. Maybeck, *Stochastic Models, Estimation and Control.* vol. 1, New York: Academic, 1979.

[8] G. Golub and C. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.

[9] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard, "Visually navigating the RMS titanic with SLAM information filters," in *Proc. Robot.: Sci. Syst. (RSS) Conf.*, Jun. 2005, pp. 57–64.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[11] J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice–Hall, 1983.

[12] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in Robotics," in *Autonomous Robot Vehicles*, I. Cox and G. Wilfong, Eds. Berlin, Germany: Springer-Verlag, 1990, pp. 167–193.

[13] T. Davis, J. Gilbert, S. Larimore, and E. Ng, "A column approximate minimum degree ordering algorithm," *ACM Trans. Math. Softw.*, vol. 30, no. 3, pp. 353–376, 2004.

[14] F. Dellaert, "Monte Carlo EM for data association and its applications in computer vision," Ph.D. dissertation, School Comput. Sci., Carnegie Mellon, Pittsburgh, PA, Tech. Rep. CMU-CS-01-153, Sep. 2001.

[15] Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques and Software*. Boston, MA: Artech House, 1993.

[16] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[17] G. Golub and R. Plemmons, "Large-scale geodetic least squares adjustment by dissection and orthogonal decomposition," *Linear Algebra Its Appl.*, vol. 34, pp. 3–28, Dec. 1980.

[18] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment-A modern synthesis," in *Vision Algorithms: Theory and Practice* (ser. Lecture Notes in Computer Science), W. Triggs, A. Zisserman, and R. Szeliski, Eds. Berlin, Germany: Springer-Verlag, Sep. 1999, pp. 298–375.

[19] J. Neira and J. Tardos, "Data association in stochastic mapping using the joint compatibility test," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 890–897, Dec. 2001.

[20] A. Griewank, "On automatic differentiation," in *Mathematical Programming: Recent Developments and Applications*, M. Iri and K. Tanabe, Eds. Norwell, MA: Kluwer, 1989, pp. 83–108.

[21] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2006, pp. 2262–2269.

[22] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao–Blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[23] D. Hähnel, W. Burgard, D. Fox, and S. Thrun, "A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2003, pp. 206–211.

[24] P. Krauthausen, F. Dellaert, and A. Kipp, "Exploiting locality by nested dissection for square root smoothing and mapping," presented at the Robot.: Sci. Syst. (RSS) Conf., Philadelphia, PA, 2006.

[25] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robot. Autom. Mag.*, vol. 13, pp. 99–110, Jun. 2006.

[26] T. Bailey and H. Durrant-Whyte, "Simultaneous localisation and mapping (SLAM): Part II state of the art," *Robot. Autom. Mag.*, vol. 13, no. 3, pp. 108–117, Sep. 2006.

[27] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," presented at the Int. Symp. Robot. Res., Santa Cruz, CA, 1987.

[28] H. Durrant-Whyte, "Uncertain geometry in robotics," *IEEE Trans. Robot. Autom.*, vol. 4, no. 1, pp. 23–31, Feb. 1988.

[29] S. Julier and J. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2001, vol. 4, pp. 4238–4243.

[30] J. Guivant and E. Nebot, "Optimization of the simultaneous localization and map building algorithm for real time implementation," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 242–257, Jun. 2001.

[31] J. Knight, A. Davison, and I. Reid, "Towards constant time SLAM using postponement," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2001, pp. 405–413.
[32] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[33] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auton. Robots*, vol. 4, pp. 333–349, Apr. 1997.
[34] J.-S. Gutmann and B. Nebel, "Navigation mobiler roboter MIT laserscans," in *Autonome Mobile Systeme*. Berlin, Germany: Springer-Verlag, 1997.
[35] T. Duckett, S. Marsland, and J. Shapiro, "Fast, on-line learning of globally consistent maps," *Auton. Robots*, vol. 12, no. 3, pp. 287–300, 2002.
[36] M. Bosse, P. Newman, J. Leonard, and S. Teller, "Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework," *Int. J. Robot. Res.*, vol. 23, no. 12, pp. 1113–1139, Dec. 2004.
[37] J. Folkesson and H. Christensen, "Graphical SLAM—A self-correcting map," in *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*, 2004, vol. 1, pp. 383–390.
[38] J. Folkesson and H. Christensen, "Closing the loop with graphical SLAM," *IEEE Trans. Robot.*, vol. 23, no. 4, pp. 731–741, Aug. 2007.
[39] K. Konolige, "Large-scale map-making," in *Proc.* 21th *AAAI Nat. Conf. AI*, San Jose, CA, 2004, pp. 457–463.
[40] U. Frese, "An $O(\log n)$ algorithm for simultaneous localization and mapping of mobile robots in indoor environments," Ph.D. dissertation, Univ. Erlangen-Nürnberg, Erlangen, Germany, 2004.
[41] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 196–207, Apr. 2005.
[42] S. Thrun and Y. Liu, "Multirobot SLAM with sparse extended information filters," in *Proc. 11th Int. Symp. Robot. Res. (ISRR'03)*. Sienna, Italy: Springer-Verlag, pp. 114–141.
[43] M. Paskin, "Thin junction tree filters for simultaneous localization and mapping," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2003, pp. 1157–1164.
[44] U. Frese, "Treemap: An $O(\log n)$algorithm for indoor simultaneous localization and mapping," *Auton. Robots*, vol. 21, no. 2, pp. 103–122, 2006.
[45] Z. Wang, S. Huang, and G. Dissanayake, "DSLAM: Decoupled localization and mapping for autonomous robots," presented at the Int. Symp. Robot. Res. (ISRR), San Francisco, CA, Oct. 2005.
[46] R. Eustice, H. Singh, and J. Leonard, "Exactly sparse delayed-state filters," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Apr. 2005, pp. 2417–2424.
[47] R. Eustice, H. Singh, J. Leonard, and M. Walter, "Visually mapping the RMS titanic: Conservative covariance estimates for SLAM information filters," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1223–1242, Dec. 2006.
[48] U. Frese and L. Schröder, "Closing a million-landmarks loop," in *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst. (IROS)*, Oct. 2006, pp. 5032–5039.
[49] M. Kaess, A. Ranganathan, and F. Dellaert, "Fast incremental square root information smoothing," in *Proc. Int. Joint Conf. AI (IJCAI)*, Hyderabad, India, 2007, pp. 2129–2134.
[50] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Rome, Italy, Apr. 2007, pp. 1670–1677.
[51] Z. Khan, T. Balch, and F. Dellaert, "MCMC data association and sparse factorization updating for real time multitarget tracking with merged and multiple measurements," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 1960–1972, Dec. 2006.
[52] F. Ling, "Givens rotation based least squares lattice related algorithms," *IEEE Trans. Signal Process.*, vol. 39, no. 7, pp. 1541–1551, Jul. 1991.
[53] P. Gill, G. Golub, W. Murray, and M. Saunders, "Methods for modifying matrix factorizations," *Math. Comput.*, vol. 28, no. 126, pp. 505–535, 1974.
[54] W. Gentleman, "Least squares computations by Givens transformations without square roots," *IMA J. Appl. Math.*, vol. 12, pp. 329–336, 1973.
[55] T. Davis and W. Hager, "Modifying a sparse Cholesky factorization," *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 3, pp. 606–627, 1996.

Michael Kaess (S'02) received the Ph.D. degree in computer science from the College of Computing, Georgia Institute of Technology, Atlanta, in 2008.

He is currently a Postdoctoral Associate with the Massachusetts Institute of Technology, Cambridge, MA. His current research interests include probabilistic methods in mobile robotics and computer vision, in particular efficient localization and large-scale mapping, as well as data association.

Ananth Ranganathan (S'02) received the B. Tech. degree in computer science from the Indian Institute of Technology, Roorkee, India, in 2002 and the Ph.D. degree in computer science from the College of Computing, Georgia Institute of Technology, Atlanta, in 2008.

He is currently a Senior Research Scientist with the Honda Research Institute, Cambridge, MA. His current research interests include probabilistic methods in robotics and computer vision, specifically for mapping and planning in robotics, and scene and object recognition in computer vision.

Frank Dellaert (M'96) received the M.S. degree in computer science and engineering from Case Western Reserve University, Cleveland, OH, in 1995, the equivalent of the M.S. degree in electrical engineering from the Catholic University of Leuven, Leuven, Belgium, in 1989, and the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, in 2001.

He is currently an Associate Professor with the College of Computing, Georgia Institute of Technology, Atlanta. His current research interests include probabilistic methods in robotics and vision. He has published more than 60 articles in journals and refereed conference proceedings, as well as several book chapters.

Dr. Dellaert is an Associate Editor for the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE.