2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

iSAM: Fast Incremental Smoothing and Mapping with Efficient Data Association

Michael Kaess, Ananth Ranganathan, and Frank Dellaert
Center for Robotics and Intelligent Machines, College of Computing
Georgia Institute of Technology, Atlanta, GA 30332
{kaess,ananth,dellaert}@cc.gatech.edu

***Abstract*— We introduce incremental smoothing and mapping (iSAM), a novel approach to the problem of simultaneous localization and mapping (SLAM) that addresses the data association problem and allows real-time application in large-scale environments. We employ smoothing to obtain the complete trajectory and map without the need for any approximations, exploiting the natural sparsity of the smoothing information matrix. A QR-factorization of this information matrix is at the heart of our approach. It provides efficient access to the exact covariances as well as to conservative estimates that are used for online data association. It also allows recovery of the exact trajectory and map at any given time by back-substitution. Instead of refactoring in each step, we update the QR-factorization whenever a new measurement arrives. We analyze the effect of loops, and show how our approach extends to the non-linear case. Finally, we provide experimental validation of the overall non-linear algorithm based on the standard Victoria Park data set with unknown correspondences.**

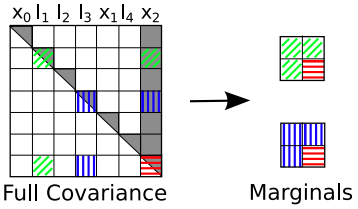


Fig. 1. Only a small number of entries of the dense covariance matrix are of interest for data association. In this example, the marginals between the last pose x_2 and the landmarks l_1 and l_3 are retrieved. The entries that need to be calculated in general are marked in gray: Only the triangular blocks on the diagonal and the last block column are needed, due to symmetry. Based on our factored information matrix representation, the last column can be obtained by simple back-substitution. The blocks on the diagonal can either be calculated exactly by only calculating the entries corresponding to non-zeros in the sparse factor R, or approximated by conservative estimates for online data association.

I. Introduction

The goal of simultaneous localization and mapping (SLAM) [1], [2], [3] is to provide a full solution for both the robot trajectory and the map, given the sensor data, in every time step. In addition, to be practically useful, the solution should be real-time, applicable to large-scale environments, and should include 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 house-hold robots. It also allows keeping track of a sensor in unknown settings, providing a cheap alternative to instrumenting the environment for augmented reality. Furthermore, it allows for autonomous operation when mapping buildings or entire cities for virtual reality applications.

What makes SLAM difficult is the uncertainty arising from integrating noisy local sensor data into a global frame. The underlying problem has two components: a discrete correspondence problem and a continuous estimation problem. While they are largely orthogonal, knowledge about the uncertainty in the continuous estimation can substantially simplify the solution of the correspondence problem. This is most important for loop closing, which is the problem of identifying previously visited locations in the face of localization uncertainty. To date, successful SLAM algorithms have overwhelmingly used probabilistic approaches because the probabilistic framework deals effectively with the uncertainty introduced by the noisy sensor data.

Filtering algorithms, which only compute the current pose of the robot, are the most widely used approach to real-time SLAM, and are typically based on the extended Kalman filter (EKF) [4]. However, filtering algorithms have significant restrictions [5]. While filters work well for linear problems, they are not suitable for most real-world problems, which are inherently non-linear. The reason for this is well-known: Marginalization over previous poses bakes any linearization errors into the system in a way that cannot be undone, leading to unbounded errors for large-scale problems. Moreover, repeated marginalization also complicates the problem by making the naturally sparse dependencies between poses and landmarks, which are summarized by the information matrix, dense. Approximations are required to deal with this added complexity, as for example employed in the sparse extended information filter [6] and the thin junction tree filter [7].

Smoothing approaches, in contrast, recover the *complete* robot trajectory and the map and therefore avoid the problems inherent to filters. In particular, the information matrix is sparse and remains sparse over time, without the need for any approximations. Even though the number of variables increases continuously for smoothing, in many real-world scenarios, especially those involving exploration, the information matrix still contains far less entries than in filter-based methods [8]. When repeatedly observing a small number of landmarks, filters seem to have an advantage, at least when ignoring the linearization issues. However, a better way of dealing with this situation is to switch to localization after a map of sufficient quality is obtained.

The *major contribution* of this paper is a fast incremental smoothing approach to the SLAM problem, that at the same time provides efficient mechanisms for data association. Our incremental smoothing and mapping algorithm (iSAM) combines the advantages of factorization-based square-root SAM [8], [9] with real-time performance for adding new measurements and obtaining the trajectory and the map [10]. This is in contrast to existing solutions to the smoothing problem, that do not provide efficient access to the quantities needed for data association, or require batch processing, making them unsuitable for real-time applications.

We show how to perform *data association* based on a factorization of the information matrix. Our method allows efficient access to the exact marginal covariances, without having to calculate the full covariance matrix. This is in contrast to filters, that produce overconfident results, and in contrast to existing smoothing algorithms, that cannot efficiently access these quantities. For online data association, more efficient conservative estimates are calculated, again based on our factored representation.

In the remaining part of this section we discuss related work. In Section II, we revisit the probabilistic formulation of the full SLAM problem, the equivalent non-linear optimization process, and a factorization-based solution. Section III discusses data association with a focus on how to efficiently obtain the marginal covariances from the factorization. In Section IV, we present an incremental approach to factorization, and show how it applies to loopy and non-linear environments. We finally provide experimental validation for our overall approach in Section V.

A. Related Work

Smoothing in the SLAM context is often referred to as the *full* SLAM problem [6]. It is closely related to bundle adjustment [11] in photogrammetry, and to structure from motion (SFM) [12] in computer vision. The first smoothing approach to the SLAM problem is presented in [13], where the estimation problem is formulated as a network of constraints between robot poses. The first implementation [14] was based on matrix inversion. A number of improved and numerically more stable algorithms have been developed, such as LRGC [15], [16], Atlas [17], Graphical SLAM [18], multi-level relaxation [19], square root SAM [8], GraphSLAM [6], and Treemap [20], [21].

We briefly compare iSAM to closely related techniques. [21] 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 [21].

Square-root smoothing and mapping (square-root SAM) [9] solves the estimation problem by factorization of the naturally sparse information matrix, while exploiting the special structure of the SLAM problem for efficiency. However, the matrix has to be factored completely after each step, resulting in unnecessary computational burden. We have recently presented an incremental solution [10], that we extend in this work to deal with unknown data association.

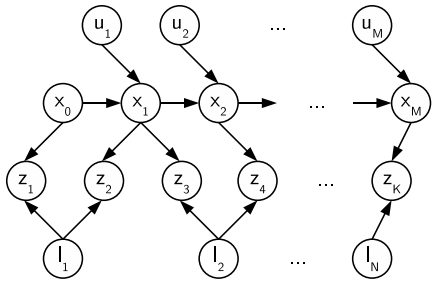


Fig. 2. Bayesian belief network representation of the SLAM problem, where x_i is the state of the robot at time i, l_j the location of landmark j, u_i the control input at time i and z_k the k^{th} measurement.

II. Smoothing and Mapping (SAM)

In this section we review the formulation of the SLAM problem in a smoothing framework, following the notation of [9]. In contrast to filtering methods, no marginalization is performed, and all pose variables are retained. We describe the underlying probabilistic model of this full SLAM problem, show how inference on this model leads to a least-squares problem, and provide a solution based on a matrix factorization.

A. A Probabilistic Model for SLAM

We formulate the SLAM problem in terms of the belief network model shown in Fig. 2. We denote the robot state at the i^{th} time step by x_i, with $i \in 0 \dots M$, a landmark by l_j, with $j \in 1 \dots N$, and a measurement by z_k, with $k \in 1 \dots K$. The joint probability is given by

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

where $P(x_0)$ is a prior on the initial state, $P(x_i|x_{i-1}, u_i)$ is the motion model, parametrized by the control input u_i, and $P(z_k|x_{i_k}, l_{j_k})$ is the landmark measurement model, assuming known correspondences (i_k, j_k) for each measurement z_k.

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

$$x_i = f_i(x_{i-1}, u_i) + w_i \tag{2}$$

describes the robot behavior in response to control input, where w_i is normally distributed zero-mean process noise with covariance matrix Λ_i. The Gaussian measurement equation

$$z_k = h_k(x_{i_k}, l_{j_k}) + v_k \tag{3}$$

models the robot's sensors, where v_k is normally distributed zero-mean measurement noise with covariance Σ_k.

B. SLAM as a Least Squares Problem

We now discuss how to obtain an optimal estimate for the set of unknowns given all available measurements. As we perform smoothing rather than filtering, we are interested in the *maximum a posteriori* (MAP) estimate for the entire trajectory $X = \{x_i\}$ and the map of landmarks $L = \{l_j\}$, given the measurements $Z = \{z_k\}$ and the control inputs $U = \{u_i\}$. Collecting all unknowns from X and L in the vector $\Theta = (X, L)$, the MAP estimate Θ^* is obtained by minimizing the negative log of the joint probability $P(X, L, Z)$ from (1):

$$\Theta^* = \arg\min_{\Theta} -\log P(X, L, Z). \tag{4}$$

Combined with the process and measurement models, this leads to the following non-linear least-squares problem

$$\begin{aligned}\Theta^* \quad = \quad \arg\min_{\Theta} \Bigg\{ & \sum_{i=1}^{M} ||f_i(x_{i-1}, u_i) - x_i||^2_{\Lambda_i} \\ & + \sum_{k=1}^{K} ||h_k(x_{i_k}, l_{j_k}) - z_k||^2_{\Sigma_k} \Bigg\}\end{aligned} \tag{5}$$

where we use the notation $||e||^2_{\Sigma} = e^T \Sigma^{-1} e$ for the squared Mahalanobis distance given a covariance matrix Σ.

In practice one always considers a linearized version of this problem. 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 solve a succession of linear approximations to this equation in order to approach the minimum. We therefore linearize the least-squares problem by assuming that either a good linearization point is available or that we are working on one iteration of a non-linear optimization method, see [8] for the derivation:

$$\begin{aligned}\delta\Theta^* = \arg\min_{\delta\Theta} \Bigg\{ & \sum_{i=1}^{M} ||F_i^{i-1} \delta x_{i-1} + G_i^i \delta x_i - a_i||^2_{\Lambda_i} \\ & + \sum_{k=1}^{K} ||H_k^{i_k} \delta x_{i_k} + J_k^{j_k} \delta l_{j_k} - c_k||^2_{\Sigma_k} \Bigg\}.\end{aligned} \tag{6}$$

where $H_k^{i_k}$, $J_k^{j_k}$ are the Jacobians of h_k with respect to a change in x_{i_k} and l_{j_k} respectively, F_i^{i-1} the Jacobian of f_i at x_{i-1}, and $G_i^i = I$ for symmetry. a_i and c_k are the odometry and observation measurement prediction errors, respectively.

Collecting all Jacobian matrices into a single matrix A, and all the vectors into a right-hand side vector b, we obtain a standard least-squares problem:

$$\theta^* = \arg\min_{\theta} ||A\theta - b||^2. \tag{7}$$

C. Solving by QR Factorization

We apply the standard QR matrix factorization [22] to solve the least-squares problem (7). In the linear case, the measurement Jacobian A of this least-squares problem is independent of the current estimate θ. We can therefore rewrite the least-squares problem (7) as

$$||Q \begin{bmatrix} R \\ 0 \end{bmatrix} \theta - b||^2 \quad = \quad ||R\theta - d||^2 + ||e||^2 \tag{8}$$

where Q is an orthogonal matrix and we define $[d, e]^T \triangleq Q^T b$. Note that if the Jacobian A is an $m \times n$ matrix, then Q and R are $m \times m$ and $n \times n$ matrices respectively. The first term $||R\theta - d||^2$ vanishes for the least-squares solution θ^*, leaving the second term $||e||^2$ as the residual of the least-squares problem.

The solution for the complete robot trajectory as well as the entire map can be recovered efficiently at any given time during the mapping process. This is simply achieved by back-substitution using the current factor R and right-hand side d to obtain an update for all model variables θ based on

$$R\theta = d. \tag{9}$$

While the algorithmic complexity of back-substitution is $O(n^2)$ for general dense matrices, it is more efficient in our case. Throughout the paper, we will assume that the number of entries per column of R does not depend on the number of variables n that make up the map and trajectory. Even though there can be a dependency on n for loopy environments, it is typically so small that it can be ignored. This is confirmed by our results in Section V for a very loopy real-world dataset. Under this assumption, back-substitution requires $O(n)$ time. To further improve efficiency, we can also restrict the calculation to a subset of variables by stopping the back-substitution process when all variables of interest are obtained. This constant time operation is typically sufficient unless loops are closed.

III. Covariances and Data Association

We now describe a method to perform maximum likelihood data association, which requires marginal covariances, in an efficient manner. The data association problem in SLAM consists of matching measurements to their corresponding landmarks. This is especially problematic when closing large loops in the environment. A simple and commonly used data association technique is the nearest neighbor (NN) approach, in which each measurement is matched to the landmark that minimizes the prediction error. This corresponds to a minimum cost assignment problem, based on a cost matrix containing all the prediction errors. Details of the method, including incorporation of spurious measurements, can be found in [23]. We use the Jonker-Volgenant-Castanon (JVC) assignment algorithm [24] to solve the minimum cost assignment problem.

The *maximum likelihood* (ML) solution to data association is more sophisticated than the NN approach in that it takes into account the relative uncertainties between the current robot location and the landmarks in the map. This can again be reduced to a minimum cost assignment problem, where we use a Mahalanobis distance rather than the Euclidean distance. Again a threshold, typically 3 sigma, is used to identify new landmarks or spurious measurements. The Mahalanobis distance needed for the ML solution is defined in the measurement space. It is therefore based on the projection Ξ of the combined pose and landmark uncertainties Σ into the measurement space

$$\Xi \quad = \quad H_p(H_t \Sigma H_t^T + \Gamma) H_p^T \tag{10}$$

where H_t is the Jacobian of the transformation into the robot system, H_p the Jacobian of the projection process in robot coordinates, and Γ the measurement noise.

The ML solution thus requires knowledge of the relative uncertainties between the current pose m_i and any visible landmark x_j. These marginal covariances

$$\Sigma_{ij} = \begin{bmatrix} \Sigma_{jj} & \Sigma_{ij}^T \\ \Sigma_{ij} & \Sigma_{ii} \end{bmatrix} \tag{11}$$

contain blocks from the diagonal of the full covariance matrix, as well as the last block row and column, as is shown in Fig. 1. Note that the off-diagonal blocks are essential, because the uncertainties are relative to an arbitrary reference frame, that is typically fixed at the origin of the trajectory. Calculating the full covariance matrix in order to recover these entries of interest is not an option, because the covariance matrix is always completely populated with n^2 entries. However, this is also not necessary, as only some triangular blocks on the diagonal and the last block column are needed, as shown in gray in Fig. 1. The remaining entries are obtained 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 efficiently obtain a conservative estimate. The exact pose uncertainty Σ_{ii} and the covariances Σ_{ij} can be recovered in linear time. We will choose the current pose to be the last variable in our factor R. Therefore, the $\dim(m_i)$ last columns X of the full covariance matrix $(R^T R)^{-1}$ contain Σ_{ii} as well as all Σ_{ij}, as observed in [25]. But instead of having to keep an incremental estimate of these entries, we can retrieve the exact values efficiently from the factor R by back-substitution. We define B as the last $\dim(m_i)$ unit vectors and solve

$$R^T R X = B \tag{12}$$

by a forward and a back-substitution

$$R^T Y = B, \quad RX = Y. \tag{13}$$

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

$$Y = [0, ..., 0, R_{ii}^{-1}]^T. \tag{14}$$

Hence only $\dim(m_i)$ back-substitutions are needed, which only requires $O(n)$ time based on our assumption that the number of entries per column in R is independent of the number of variables n. But we can do even better: Typically only a small number of landmarks in the map can be visible from the current robot location. As we only need the marginal covariances of those, we can use a dynamic programming approach that also calculates any intermediate entries that might be needed. The result is constant time for exploration tasks, but can be linear for loopy environments.

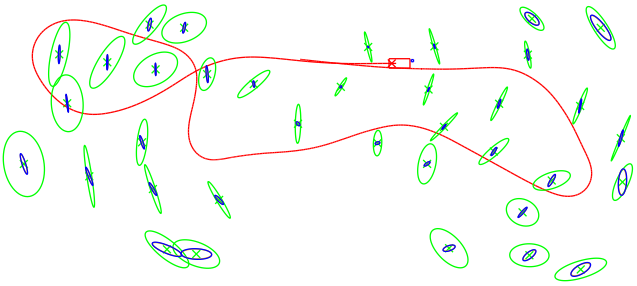


Fig. 3. Comparison of marginal covariance estimates *projected into the current robot frame* (indicated by red rectangle), for a short trajectory (red line) and some landmarks (green crosses). Conservative covariances (green, large ellipses) are shown as well as the exact covariances (blue, smaller ellipses) obtained by our fast algorithm. Note that the exact covariances based on full inversion are also shown (orange, mostly hidden by blue).

A. Conservative Estimates

Conservative estimates for the structure uncertainties Σ_{jj} can be obtained as proposed by [25]. As the covariance can only shrink over time during the smoothing process, we can use the initial uncertainties $\tilde{\Sigma}_{jj}$ as conservative estimates. These are obtained by

$$\tilde{\Sigma}_{jj} = H_{-pt} \begin{bmatrix} \Sigma_{ii} & \\ & \Gamma \end{bmatrix} H_{-pt}^T \tag{15}$$

where H_{-pt} is the Jacobian of the back-projection, including the transformation into map coordinates, and Σ_{ii} and Γ are the current pose uncertainty and the measurement noise, respectively. Fig. 3 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.

B. Exact Covariances

Recovering the exact structure uncertainties Σ_{jj} is not straightforward, as they are spread out along the diagonal, but can still be performed efficiently in this context by exploting the sparsity structure of R. In general, the inverse

$$\Sigma \triangleq (A^T A)^{-1} = (R^T R)^{-1} \tag{16}$$

is obtained based on the factor R by noting that

$$R^T R \Sigma \quad = \quad I \tag{17}$$

and performing a forward, followed by a back-substitution

$$R^T Y = I, \quad R\Sigma = Y. \tag{18}$$

As the information matrix is not band-diagonal in general, this would seem to require calculating all $O(n^2)$ entries of the fully dense covariance matrix, which is infeasible for any non-trivial problem. Here is where the sparsity of the factor R is of advantage again. Both, [26] and [11] present an efficient method for recovering exactly all entries σ_{ij} of

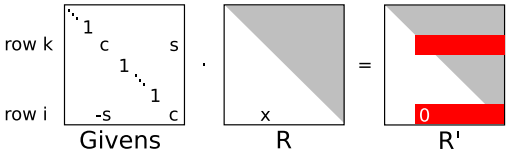


Fig. 4. Using a Givens rotation to transform a matrix into upper triangular form. The entry marked 'x' is eliminated, changing some of the entries marked in red (dark), depending on sparseness.

the covariance matrix Σ that coincide with non-zero entries in the factor R:

$$\sigma_{ll} = \frac{1}{r_{ll}}(\frac{1}{r_{ll}} - \sum_{j=l+1, r_{lj}\neq 0}^{n} r_{lj}\sigma_{jl}), \tag{19}$$

$$\sigma_{il} = \frac{1}{r_{ii}}(- \sum_{j=i+1, r_{ij}\neq 0}^{l} r_{ij}\sigma_{jl} - \sum_{j=l+1, r_{ij}\neq 0}^{n} r_{ij}\sigma_{lj}) \tag{20}$$

for $l = n, \ldots, 1$ and $i = l-1, \ldots, 1$. Note that the summations only apply to non-zero entries of single columns or rows of the sparse matrix R. The algorithm therefore has $O(n)$ time complexity for band-diagonal matrices and matrices with only a constant number of entries far from the diagonal, but can be more expensive for general sparse R.

As the upper triangular parts of the block diagonals of R are fully populated, and due to symmetry of the covariance matrix, this algorithm provides access to all block diagonals, which includes the structure uncertainties Σ_{jj}. Fig. 3 shows marginal covariances obtained by this algorithm for a small example. Note that they coincide with the exact covariances obtained by full inversion.

IV. Incremental SAM (iSAM)

We revisit our incremental solution to the full SLAM problem [10] based on updating a factored representation of the information matrix of the least-squares problem from (7). For simplicity we initially only consider the case of linear process and measurement models, and return to the non-linear case later in this section.

A. Givens Rotations

A standard approach to obtain the QR factorization of the measurement Jacobian A uses Givens rotations [22] to clean out all entries below the diagonal, one at a time. As we will see later, this approach readily extends to factorization updates, as will be needed to incorporate new measurements. The process starts from the bottom left-most non-zero entry, and proceeds either column- or row-wise, by applying the Givens rotation

$$\begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \tag{21}$$

to rows i and k, with $i > k$. The parameter ϕ is chosen so that the (i,k) entry of A becomes 0, as shown in Fig. 4. Givens rotations are numerically stable and accurate to machine precision, if implemented correctly [22, 5.1]. After all entries below the diagonal are zeroed out, the upper triangular entries contain the R factor. Note that a sparse measurement Jacobian A will result in a sparse R factor, at least for an appropriate variable ordering, as will be discussed later. However, the orthogonal rotation matrix Q is typically dense, which is why this matrix is never explicitly stored or even formed in practice. Instead, it is sufficient to update the right-hand side (rhs) b with the same rotations that are applied to A.

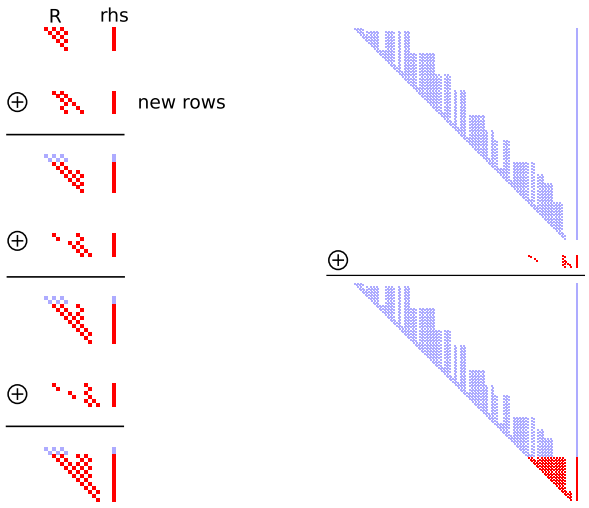


Fig. 5. Updating a factored representation of the smoothing information matrix: New measurement rows are added to the upper triangular factor R and the right-hand side (rhs). The left column shows the first three updates, the right column shows the update after 50 steps. The update operation is symbolically denoted by $\oplus$. Entries that remain unchanged are shown in light blue (gray). For a typical exploration task the number of operations is bounded by a constant.

Instead of factorizing the updated measurement Jacobian A when a new measurement arrives, it is more efficient to modify the previous factorization by QR-updating. Adding a new measurement row w^T and rhs γ into the current factor R and rhs d yields a new system that is not in the correct factorized form:

$$\begin{bmatrix} Q^T & \\ & 1 \end{bmatrix} \begin{bmatrix} A \\ w^T \end{bmatrix} = \begin{bmatrix} R \\ w^T \end{bmatrix}, \text{ new rhs: } \begin{bmatrix} d \\ \gamma \end{bmatrix}. \tag{22}$$

Note that this is the same system that would be obtained by applying Givens rotations 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'. In the same way as for the full factorization, we simultaneously update the right-hand side with the same rotations to obtain d'.

For iSAM, QR-updating is efficient. In general, the maximum number of Givens rotations needed for adding a new row is n. However, as 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. An example demonstrating the locality of the update process is shown in Fig. 5.

It is easy to add new landmark and pose variables to the QR factorization, as we can just expand the factor R by the appropriate number of zero columns and rows, before updating with the new measurement rows. Similarly, the right-hand side d is augmented by the same number of zero entries. The ordering of the variables requires some careful consideration. In each step, we first add the new landmarks for the current pose, and only then add the next pose. This assures that the last variable in the system is always the most recent pose, which is needed for an efficient recovery of the covariances, as described in Section III.

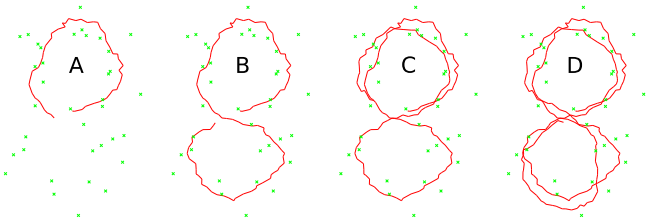
For an exploration task in the linear case, the number of rotations needed to incorporate a set of new landmark and odometry measurements is independent of the size of the trajectory and map [10]. Updating therefore has $O(1)$ time complexity for exploration tasks. Recovering all variables after each step requires $O(n)$ time, but is still very efficient even after $10\,000$ steps, at about 0.12 seconds per step. Furthermore, a constant time approximation can be obtained, as only the most recent variables change significantly enough to warrant recalculation. This is achieved by stopping the back-substitution once the change in the variable estimate drops below a threshold.

B. Loops and Variable Reordering

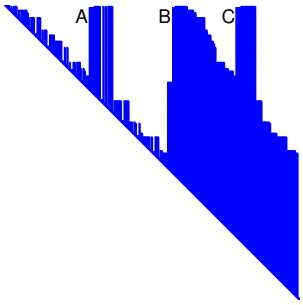
In contrast to pure exploration, where landmarks are only observed in consecutive frames, loops can lead to fill-in of the factor R. A loop is a cycle in the trajectory that brings the robot back to a previously visited location. This introduces correlations between current poses and previously observed landmarks, which themselves are connected to earlier parts of the trajectory. Results based on a simulated environment with multiple loops are shown in Fig. 6. Even though the information matrix remains sparse in the process, the incremental updating of the factor R leads to fill-in. This fill-in is local and does not affect further exploration, as is evident from the example.

However, this fill-in can be avoided, as it depends on the ordering of the variables. While obtaining the best ordering is NP hard, efficient heuristics like *colamd* [27] have been developed in linear algebra, that yield good results for the SLAM problem as evaluated in [9]. The same factor R after reordering shows no signs of fill-in. However, reordering of the variables and subsequent factorization of the new measurement Jacobian itself is also expensive when performed in each step. We therefore propose fast incremental updates interleaved with periodic reordering, yielding a fast algorithm as supported by the dashed blue curve in Fig. 6(d).

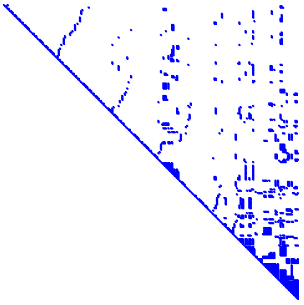
When the robot continuously observes the same landmarks, for example by remaining in one small room, this approach will eventually fail, as the information matrix itself will become dense. However, in this case filters will also fail due to underestimation of uncertainties that will finally converge to 0. A better solution to deal with this scenario is to eventually switch to localization.



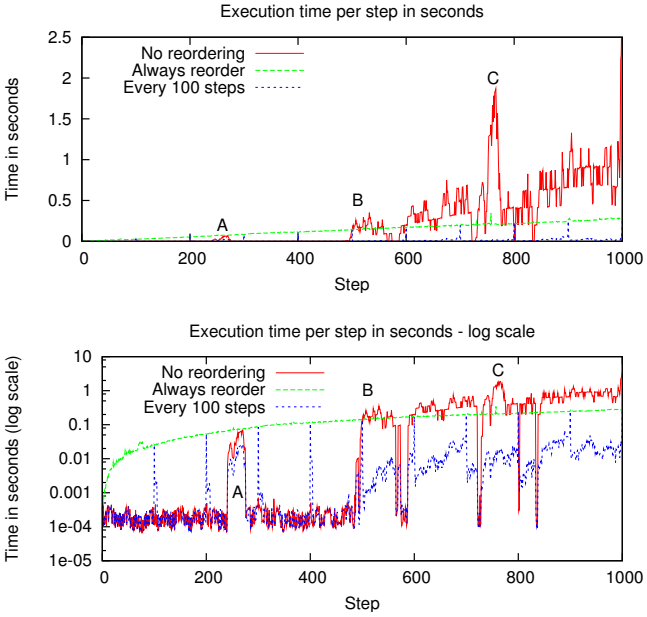
(a) Simulated double 8-loop at interesting stages of loop closing.



(b) Factor R.



(c) The same factor R after variable reordering.



(d) Execution time per step for different updating strategies are shown in both linear and log scale.

Fig. 6. 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 when the first loop is closed (A). Note that this has 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, the factor matrix again is completely sparse (c). Reordering after each step (d, dashed green) can be less expensive in the case of multiple loops. A considerable increase in efficiency is achieved by using fast incremental updates interleaved with occasional reordering (d, dotted blue), here every 100 steps.

	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	$429s$	$858ms$	$3\,300ms$

Fig. 7. Execution times for different data association techniques for a simulated 500-pose loop with 240 landmarks. The execution times include updating of the factorization, solving for all variables, and performing the respective data association technique, for every step.

C. Non-linear Systems

Updating the linearization point based on a new variable estimate changes the measurement Jacobian. One way then to obtain a new factorization is to refactor the measurement Jacobian. However, in many situations it is not necessary to perform relinearization [28]. First, measurements are typically fairly accurate on a local scale, so that relinearization is not needed in every step. Therefore, to avoid additional computation, the relinearization can be performed together with the periodic variable reordering. Second, measurements are also local and only affect a small number of variables directly, with their effects rapidly declining while propagating through the constraint graph. An exception is a situation such as a loop closing, that can affect many variables at once. In any case it is sufficient to perform selective relinearization only over variables whose estimate has changed by more than some threshold. Relinearization can then also be applied incrementally, by first removing the affected measurement rows from the current factor by QR-downdating (eg. hyperbolic rotations [22]), followed by adding the relinearized measurement rows by QR-updating (eg. Givens rotations).

V. Experiments and Results

A. Evaluation of Data Association Techniques

To compare the different data association techniques, we have created a simulated environment 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 execution times of the different data association techniques are shown in Fig. 7. The nearest neighbor (NN) approach is dominated by the time needed for factorization and backsubstitution in each step. The maximum likelihood (ML) approach is evaluated for our fast conservative estimate (Section III-A) and our efficient exact solution (Section III-B). The conservative estimate only adds a small overhead, which is mostly due to back-substitution to obtain the last columns of the exact covariance matrix. Recovering the exact matrix in comparison is fairly expensive, as the block-diagonal entries have to be recovered. To show the advantage of our exact efficient algorithm, we included results based on the direct inversion of the information matrix. Note that we have used a fast algorithm based on a sparse LDL^T matrix factorization [22], but this does not change the fact that all n^2 entries of the covariance matrix have to be calculated.

Fig. 8. Optimized map of the full Victoria Park sequence. Solving the complete problem including data association in each step took 4.5 minutes on a laptop. For known correspondences the time reduces to 2.7 minutes. Since the dataset is from a 26 minute long robot run, iSAM with correspondences is over **5 times faster than real time** on a laptop computer. The trajectory and landmarks are shown in yellow (light), manually overlayed on an aerial image for reference. Differential GPS was not used in obtaining our experimental results, but is shown in blue (dark) for comparison purposes. Note that GPS is not available in many places.

B. Victoria Park Dataset

We have applied iSAM to the Sydney Victoria Park dataset (available at `http://www.acfr.usyd.edu.au/homepages/academic/enebot/dataset.htm`), a popular test dataset in the SLAM community. The data consists of 7 247 frames along a trajectory of 4 kilometer length, recorded over a time frame of 26 minutes. 6 969 frames are left after removing all measurements where the robot was stationary. We have extracted 3 640 measurements of landmarks from the laser data by a simple tree detector.

We have evaluated the performance of iSAM for both, known and automatic data association. The final optimized trajectory and map are shown in Fig. 8. The incremental reconstruction includes solving for all variables after each new frame is added. All timing results are obtained on a 2 GHz Pentium M laptop computer. Using conservative estimates for data association, it took $464s$ (7.7 minutes) to calculate, which is significantly less than the 26 minutes it took to record the data. The resulting reconstruction contains 140 distinct landmarks. Under known correspondences, the time reduces to $351s$ (5.9 minutes). The difference is mainly caused by the back-substitution of the last three columns

in order to obtain the off-diagonal entries in each step. A similar back-substitution, but only over a single column, is performed to solve for all variables in each step. This is in fact not really needed, as the measurements are fairly accurate locally. Calculating all variables only every 10 steps yields a significant improvement to $270s$ (4.5 minutes) with and $159s$ (2.7 minutes) without data association.

Even though this trajectory contains a significant number of loops (several places are traversed 8 times), iSAM is over 5 times faster than real time. The average calculation time for the final 100 steps, which are the most expensive ones to compute, is $0.120s$ and $0.095s$, with and without data association, respectively. This compares favorably to the $0.22s$ needed for real time. These times include a full factorization and variable reordering, which took $1.8s$ in total in both cases. Despite all the loops, the final factor R is still sparse, with 207 422 entries for 21 187 variables, yielding an average of 9.79 entries per column. This supports our assumption that the number of entries per column is approximately independent of the number of variables n, even for very loopy environments.

VI. Conclusions

We presented a real-time SLAM algorithm that overcomes the problems inherent to filters. The key is a factored representation of the information matrix, that provides access to the marginal covariances needed for data association, and allows retrieving the exact trajectory and map in linear time. Efficiency arises from incrementally updating the factorized system while periodically reordering the variables to avoid fill-in. We have finally demonstrated that our algorithm is capable of solving large-scale SLAM problems in real-time, including data association.

Future work includes the search for a good incremental variable ordering, so that full matrix factorizations can be completely avoided. For very large-scale environments it seems likely that the complexity can be bounded by approaches similar to submaps or multi-level representations, that have proven successful in combination with other SLAM algorithms.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. IIS - 0448111. We would like to thank Eduardo Nebot and Hugh Durrant-Whyte for sharing their Victoria Park data set.

References

[1] R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Intl. J. of Robotics Research*, 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," *Intl. J. of Robotics Research*, vol. 11, no. 4, pp. 286–289, 1992.

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

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

[5] S. Julier and J. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 4, 2001, pp. 4238–4243.

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

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

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

[9] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, no. 12, Dec 2006.

[10] M. Kaess, A. Ranganathan, and F. Dellaert, "Fast incremental square root information smoothing," in *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007, pp. 2129–2134.

[11] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice*, ser. LNCS, W. Triggs, A. Zisserman, and R. Szeliski, Eds. Springer Verlag, 2000, pp. 298–375.

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

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

[14] J.-S. Gutmann and B. Nebel, "Navigation mobiler roboter mit laserscans," in *Autonome Mobile Systeme*. Berlin: Springer Verlag, 1997.

[15] J.-S. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *IEEE Intl. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*, 1999, pp. 318–325.

[16] K. Konolige, "Large-scale map-making," in *Proc. 21^{th} AAAI National Conference on AI*, San Jose, CA, 2004.

[17] 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.

[18] J. Folkesson and H. I. Christensen, "Graphical SLAM - a self-correcting map," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 1, 2004, pp. 383 – 390.

[19] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Trans. Robotics*, vol. 21, no. 2, pp. 196–207, April 2005.

[20] U. Frese, "Treemap: An $O(\log n)$ algorithm for indoor simultaneous localization and mapping," *Autonomous Robots*, vol. 21, no. 2, pp. 103–122, 2006.

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

[22] G. Golub and C. V. Loan, *Matrix Computations*, 3rd ed. Baltimore: Johns Hopkins University Press, 1996.

[23] F. Dellaert, "Monte Carlo EM for data association and its applications in computer vision," Ph.D. dissertation, School of Computer Science, Carnegie Mellon, September 2001, also available as Technical Report CMU-CS-01-153.

[24] 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.

[25] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard, "Visually navigating the RMS titanic with SLAM information filters," in *Robotics: Science and Systems (RSS)*, Cambridge, USA, June 2005.

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

[27] 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.

[28] D. Steedly, I. Essa, and F. Dellaert, "Spectral partitioning for structure from motion," in *ICCV*, 2003.