Fast Incremental Square Root Information Smoothing*

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 propose a novel approach to the problem of simultaneous localization and mapping (SLAM) based on incremental smoothing, that is suitable for real-time applications in large-scale environments. The main advantages over filter-based algorithms are that we solve the full SLAM problem without the need for any approximations, and that we do not suffer from linearization errors. We achieve efficiency by updating the square-root information matrix, a factored version of the naturally sparse smoothing information matrix. We can efficiently recover the exact trajectory and map at any given time by back-substitution. Furthermore, our approach allows access to the exact covariances, as it does not suffer from under-estimation of uncertainties, which is another problem inherent to filters. We present simulation-based results for the linear case, showing constant time updates for exploration tasks. We further evaluate the behavior in the presence of loops, and discuss how our approach extends to the non-linear case. Finally, we evaluate the overall non-linear algorithm on the standard Victoria Park data set.

1 Introduction

The problem of simultaneous localization and mapping (SLAM) has received considerable attention in mobile robotics, as it is one way to enable a robot to explore and navigate previously unknown environments. For a wide range of applications, such as commercial robotics, search-and-rescue and reconnaissance, the solution must be incremental and real-time in order to be useful. The problem consists of two components: a discrete correspondence and a continuous estimation problem. In this paper we focus on the continuous part, while keeping in mind that the two components are not completely independent, as information from the estimation part can simplify the correspondence problem.

*This work was supported in part by the National Science Foundation award IIS - 0448111 "CAREER: Markov Chain Monte Carlo Methods for Large Scale Correspondence Problems in Computer Vision and Robotics".

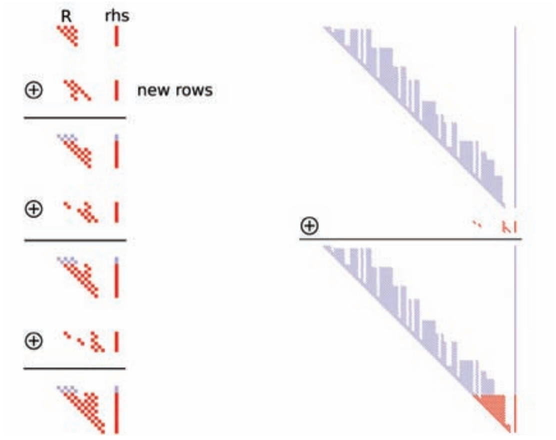


Figure 1: Updating the upper triangular factor R and the right hand side (rhs) with new measurement rows. The left column shows the first three updates, the right column shows the update after 50 steps. The update operation is symbolically denoted by $\oplus$, and entries that remain unchanged are shown in light blue.

Current real-time approaches to SLAM are typically based on filtering. While filters work well for linear problems, they are not suitable for most real-world problems, which are inherently non-linear [Julier and Uhlmann, 2001]. The reason for this is well known: Marginalization over previous poses bakes linearization errors into the system in a way that cannot be undone, leading to unbounded errors for large-scale applications. But repeated marginalization also complicates the problem by making the naturally sparse dependencies between poses and landmarks, which are summarized in the information matrix, dense. Sparse extended information filters [Thrun *et al.*, 2005] and thin junction tree filters [Paskin, 2003] present approximations to deal with this complexity.

Smoothing approaches to SLAM recover the *complete* robot trajectory and the map, avoiding 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 [Dellaert, 2005]. When repeatedly observing a small number of landmarks, filters seem to have an advantage, at least when ignoring the linearization issues. However, the correct way of dealing with this situation is to switch to localization after a map of sufficient quality is obtained.

In this paper, we present a fast incremental smoothing approach to the SLAM problem. We revisit the underlying probabilistic formulation of the smoothing problem and its equivalent non-linear optimization formulation in Section 2. The basis of our work is a factorization of the smoothing information matrix, that is directly suitable for performing optimization steps. Rather than recalculating this factorization in each step from the measurement equations, we incrementally update the factorized system whenever new measurements become available, as described in Section 3. This fully exploits the natural sparsity of the SLAM problem, resulting in constant time updates for exploration tasks. We apply variable reordering similar to [Dellaert, 2005] in order to keep the factored information matrix sparse even when closing multiple loops. All linearization choices can be corrected at any time, as no variables are marginalized out. From our factored representation, we can apply back-substitution at any time to obtain the complete map and trajectory in linear time. We evaluate the incremental algorithm based on the standard Victoria Park dataset in Section 4.

As our approach is incremental, it is suitable for real-time applications. In contrast, many other smoothing approaches like GraphSLAM [Thrun *et al.*, 2005] perform batch processing, solving the complete problem in each step. Even though explicitly exploiting the structure of the SLAM problem can yield quite efficient batch algorithms [Dellaert, 2005], for large-scale applications an incremental solution is needed. Examples of such systems that incrementally add new measurements are Graphical SLAM [Folkesson and Christensen, 2004] and multi-level relaxation [Frese *et al.*, 2005]. However, these and similar solutions have no efficient way of obtaining the marginal covariances needed for data association. In Section 5 we describe how our approach allows access to the exact marginal covariances without a full matrix inversion, and how to obtain a more efficient conservative estimate.

2 SLAM and Smoothing

In this section we review the formulation of the SLAM problem in a smoothing framework, following the notation of [Dellaert, 2005]. 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, and show how inference on this model leads to a least-squares problem.

2.1 A Probabilistic Model for SLAM

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

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

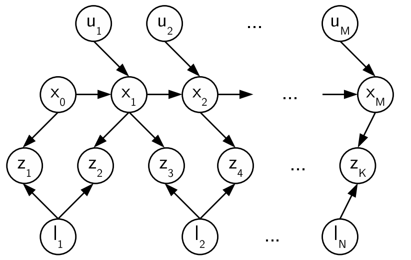


Figure 2: Bayesian belief network representation of the SLAM problem, where x is the state of the robot, l the landmark locations, u the control input and z the measurements.

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.

As is standard in the SLAM literature, we assume Gaussian process and measurement models. The process model $x_i = f_i(x_{i-1},u_i) + w_i$ 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 $z_k = h_k(x_{i_k},l_{j_k}) + v_k$ models the robot's sensors, where v_k is normally distributed zero-mean measurement noise with covariance Σ_k.

2.2 SLAM as a Least Squares Problem

In this section we discuss how to obtain an optimal estimate for the set of unknowns given all measurements available to us. As we perform smoothing rather than filtering, we are interested in the maximum a posterior (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 $U = \{u_i\}$. Let us collect all unknowns in X and L in the vector $\Theta = (X, L)$. The MAP estimate Θ^* is then obtained by minimizing the negative log of the joint probability $P(X,L,Z)$ from equation (1):

$$\Theta^* = \arg\min_{\Theta} - \log P(X,L,Z) \tag{2}$$

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

$$\begin{aligned}\Theta^* \quad = \quad & \arg\min_{\Theta} \left\{ \sum_{i=1}^{M} ||f_i(x_{i-1},u_i) - x_i||^2_{\Lambda_i} \right. \\ & \left. + \sum_{k=1}^{K} ||h_k(x_{i_k},l_{j_k}) - z_k||^2_{\Sigma_k} \right\}\end{aligned} \tag{3}$$

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

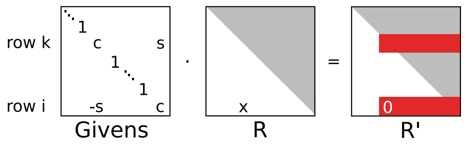


Figure 3: Using a Givens rotation to transform a matrix into upper triangular form. The entry marked 'x' is eliminated, changing some or all of the entries marked in red (dark), depending on sparseness.

optimization method, see [Dellaert, 2005] for the derivation:

$$\delta\Theta^* = \arg\min_{\delta\Theta} \left\{ \sum_{i=1}^{M} ||F_i^{i-1}\delta x_{i-1} + G_i^i \delta x_i - a_i||^2_{\Lambda_i} + \sum_{k=1}^{K} ||H_k^{i_k}\delta x_{i_k} + J_k^{j_k}\delta l_{j_k} - c_k||^2_{\Sigma_k} \right\} \quad (4)$$

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.

We combine all summands into one least-squares problem after dropping the covariance matrices Λ_i and Σ_k by pulling their matrix square roots inside the Mahalanobis norms:

$$||e||^2_\Sigma = e^T\Sigma^{-1}e = \left(\Sigma^{-T/2}e\right)^T \left(\Sigma^{-T/2}e\right) = ||\Sigma^{-T/2}e||^2 \quad (5)$$

For scalar matrices this simply means dividing each term by the measurement standard deviation. We then collect all Jacobian matrices into a single matrix A, the vectors a_i and c_k into a right hand side vector b, and all variables into the vector θ, to obtain the following standard least-squares problem:

$$\theta^* = \arg\min_\theta ||A\theta - b||^2 \quad (6)$$

3 Updating a Factored Representation

We present an incremental solution to the full SLAM problem based on updating a matrix factorization of the measurement Jacobian A from (6). For simplicity we initially only consider the case of linear process and measurement models. In the linear case, the measurement Jacobian A is independent of the current estimate θ. We therefore obtain the least squares solution θ^* in (6) directly from the standard QR matrix factorization [Golub and Loan, 1996] of the Jacobian $A \in \mathbb{R}^{m\times n}$:

$$A = Q\begin{bmatrix} R \\ 0 \end{bmatrix} \quad (7)$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal, and $R \in \mathbb{R}^{n\times n}$ is upper triangular. Note that the information matrix $\mathcal{I} \triangleq A^TA$ can also be expressed in terms of this factor R as $\mathcal{I} = R^TR$, which we therefore call the *square-root information matrix* R. We rewrite the least-squares problem, noting that multiplying with the orthogonal matrix Q^T does not change the norm:

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

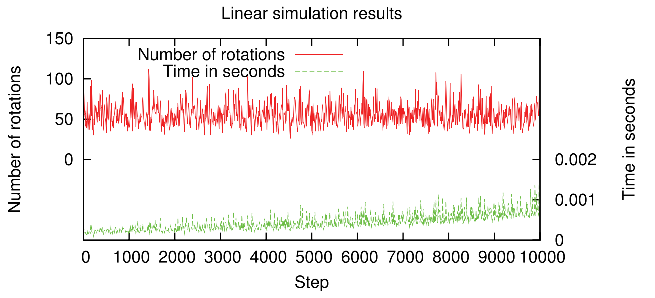


Figure 4: Complexity of a simulated linear exploration task, showing 10-step averages. In each step, the square-root factor is updated by adding the new measurement rows by Givens rotations. The average number of rotations (red) remains constant, and therefore also the average number of entries per matrix row. The execution time per step (dashed green) slightly increases for our implementation due to tree representations of the underlying data structures.

where we defined $[d, e]^T \triangleq Q^Tb$. 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 square-root factor allows us to efficiently recover the complete robot trajectory as well as the map at any given time. This is simply achieved by back-substitution using the current factor R and right hand side (rhs) d to obtain an update for all model variables θ based on

$$R\theta = d \quad (9)$$

Note that this is an efficient operation for sparse R, under the assumption of a (nearly) constant average number of entries per row in R. We have observed that this is a reasonable assumption even for loopy environments. Each variable is then obtained in constant time, yielding $O(n)$ time complexity in the number n of variables that make up the map and trajectory. 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.

3.1 Givens Rotations

One standard way to obtain the QR factorization of the measurement Jacobian A is by using Givens rotations [Golub and Loan, 1996] 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:

$$\begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \quad (10)$$

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 Figure 3. After all entries below the diagonal are zeroed out, the upper triangular entries will contain the R factor. Note that a sparse measurement Jacobian A will result in a sparse R factor. 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 rhs b with the same rotations that are applied to A.

When a new measurement arrives, it is cheaper to update the current factorization than to factorize the complete measurement Jacobian A. 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:

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

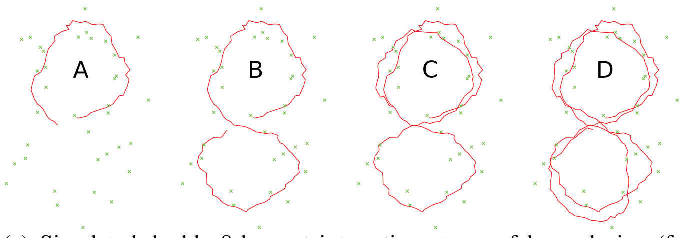
Note that this is the same system that is obtained by applying Givens rotations to eliminate all entries below the diagonal, except for the last (new) row. Therefore Givens rotations can be determined that zero out this new row, yielding the updated factor R'. In the same way as for the full factorization, we simultaneously update the right hand side with the same rotations to obtain d'. In general, the maximum number of Givens rotations needed to add a new measurement is n. However, as R and the new measurement rows 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 right most part of the new measurement row is (sparsely) populated. An example of the locality of the update process is shown in Figure 1.

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 rhs d is augmented by the same number of zero entries.

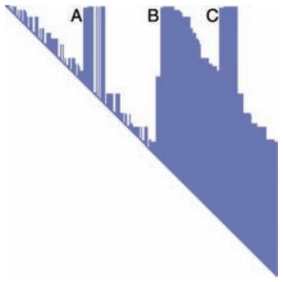
For an exploration task in the linear case, the number of rotations needed to incorporate a set of new landmark and odometry measurements is independent of the size of the trajectory and map, as the simulation results in Figure 4 show. Our algorithm therefore has $O(1)$ time complexity for exploration tasks. Recovering all variables after each step requires $O(n)$ time, but is still very efficient even after 10000 steps, at about 0.12 seconds per step. However, only the most recent variables change significantly enough to warrant recalculation, providing a good approximation in constant time.

3.2 Loops and Variable Reordering

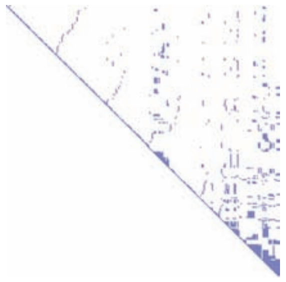
Loops can be dealt with by a periodic reordering of variables. A loop is a cycle in the trajectory that brings the robot back to a previously visited location. This introduces correlations between current poses and previously observed landmarks, which themselves are connected to earlier parts of the trajectory. Results based on a simulated environment with multiple loops are shown in Figure 5. 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 by variable reordering, as has been described in [Dellaert, 2005]. 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 occasional reordering, yielding a fast algorithm as supported by the dotted blue curve in Figure 5(d).



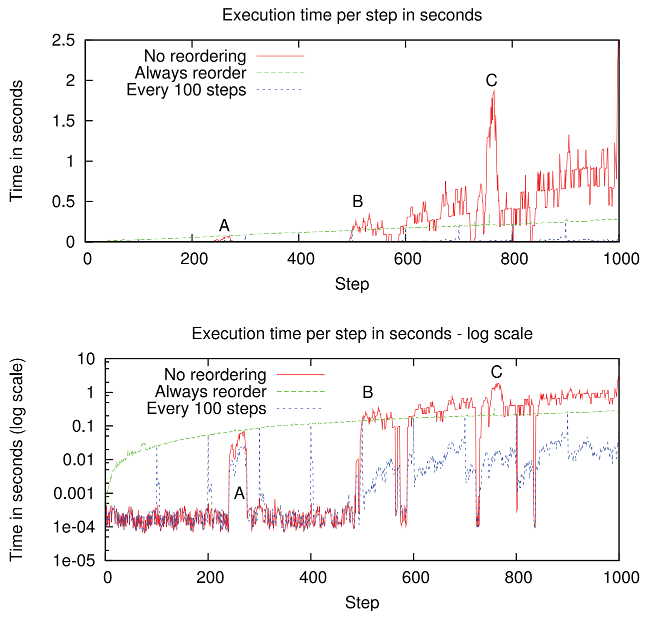
(a) Simulated double 8-loop at interesting stages of loop closing (for simplicity only a quarter of the poses and landmarks are shown).



(b) Cholesky factor R.



(c) Cholesky factor after variable reordering.



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

Figure 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 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 periodic reordering (d, dotted blue), here every 100 steps.

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 completely dense. However, in this case filters will also fail due to underestimation of uncertainties that will finally converge to 0. The correct solution to deal with this scenario is to eventually switch to localization.

3.3 Non-linear Systems

Our factored representation allows changing the linearization point of any variable at any time. 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 this matrix. For the results presented here, we combine this with the periodic reordering of the variables for fill-in reduction as discussed earlier.

However, in many situations it is not necessary to perform relinearization [Steedly *et al.*, 2003]. Measurements are typically fairly accurate on a local scale, so that relinearization is not needed in every step. 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. In this case, the affected measurement rows are first removed from the current factor by QR-downdating [Golub and Loan, 1996], followed by adding the relinearized measurement rows by QR-updating as described earlier.

4 Results

We have applied our approach 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 trajectory consists of 7247 frames along a trajectory of 4 kilometer length, recorded over a time frame of 26 minutes. 6968 frames are left after removing all measurements where the robot is stationary. We have extracted 3631 measurements of 158 landmarks from the laser data by a simple tree detector.

The final optimized trajectory and map are shown in Figure 6. For known correspondences, the full incremental reconstruction, including solving for all variables after each new frame is added, took $337s$ (5.5 minutes) to calculate on a Pentium M 2 GHz laptop, which is significantly less than the 26 minutes it took to record the data. The average calculation time for the final 100 steps is $0.089s$ per step, which includes a full factorization including variable reordering, which took $1.9s$. This shows that even after traversing a long trajectory with a significant number of loops, our algorithm still performs faster than the $0.22s$ per step needed for real-time. However, we can do even better by selective reordering based on a threshold on the running average over the number of Givens rotations, and back-substitution only every 10 steps, resulting in an execution time of only $116s$. Note that this still provides good map and trajectory estimates after each step, due to the measurements being fairly accurate locally.

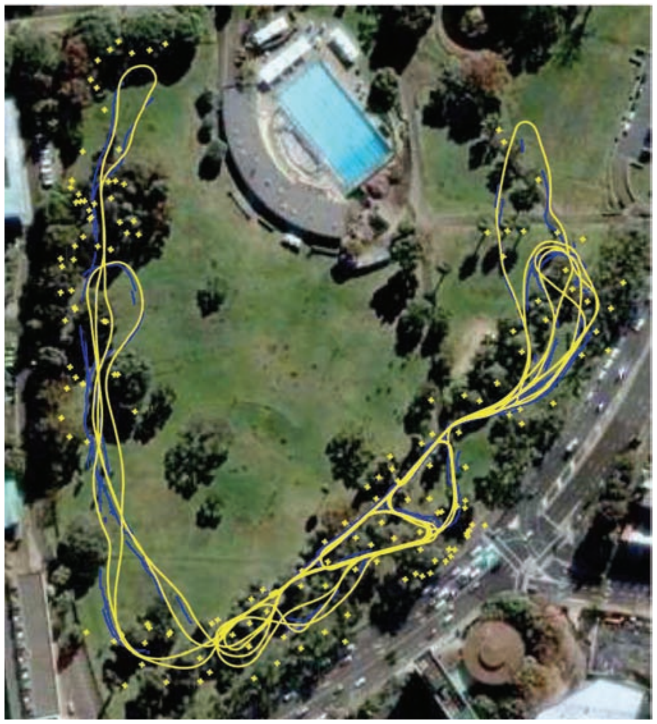


Figure 6: Optimized map of the full Victoria Park sequence. Solving the complete problem after every step takes less than 2 minutes on a laptop for known correspondences. The trajectory and landmarks are shown in yellow (light), manually overlayed on an aerial image for reference. Differential GPS was not used in obtaining the results, but is shown in blue (dark) where available. Note that in many places GPS is not available.

5 Covariances for Data Association

Our algorithm is helpful for performing data association, which generally is a difficult problem due to noisy data and the resulting uncertainties in the map and trajectory estimates. First it allows to undo data association decisions, but we have not exploited this yet. Second, it allows for recovering the underlying uncertainties, thereby reducing the search space.

In order to reduce the ambiguities in matching newly observed features to already existing landmarks, it is advantageous to know the projection Ξ of the combined pose and landmark uncertainty Σ' into the measurement space:

$$\Xi = H\Sigma' H^T + \Gamma \tag{12}$$

where H is the Jacobian of the projection process, and Γ measurement noise. This requires knowledge of the marginal covariances

$$\Sigma'_{ij} = \left[\begin{array}{cc} \Sigma_{jj} & \Sigma_{ij}^T \\ \Sigma_{ij} & \Sigma_{ii} \end{array} \right] \tag{13}$$

between the current pose m_i and any visible landmark x_j, where Σ_{ij} are the corresponding blocks of the full covariance matrix $\Sigma = \mathcal{I}^{-1}$. Calculating this covariance matrix in order to recover all entries of interest is not an option, because it is always completely populated with n^2 entries.

Our 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. By design, the most recent pose is always the last variable in our factor R – if variable reordering is performed, it is done before the next pose is added. 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 [Eustice *et al.*, 2005]. 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 $R^T RX = B$ by two back-substitutions $R^T Y = B$ and $RX = Y$. The key to efficiency is that we never have to recover a full dense matrix, but due to R being upper triangular immediately obtain $Y = [0, ..., 0, R_{ii}^{-1}]^T$. Hence only one back-substitution is needed to recover $\dim(m_i)$ dense vectors, which only requires $O(n)$ time due to the sparsity of R.

Conservative estimates for the structure uncertainties Σ_{jj} can be obtained as proposed by [Eustice *et al.*, 2005]. As the covariance can only shrink over time during the smoothing process, we can use the initial uncertainties as conservative estimates. A more tight conservative estimate can be obtained after multiple measurements are available.

Recovering the *exact* structure uncertainties Σ_{jj} is more tricky, as they are spread out along the diagonal. As the information matrix is not band-diagonal in general, this would seem to require calculating all entries of the fully dense covariance matrix, which is infeasible. Here is where our factored representation, and especially the sparsity of the factor R are useful. Both, [Golub and Plemmons, 1980] and [Triggs *et al.*, 2000] present an efficient method of recovering exactly all entries of the covariance matrix that coincide with non-zero entries in the factor 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 blocks on the diagonal that correspond to the structure uncertainties Σ_{jj}. The inverse $Z = (A^T A)^{-1}$ is obtained based on the factor R in the standard way by noting that $A^T AZ = R^T RZ = I$, and performing two back-substitutions $R^T Y = I$ and $RZ = Y$. As the covariance matrix is typically dense, this would still require calculation of $O(n^2)$ elements. However, we exploit the fact that many entries obtained during back-substitution of one column are not needed in the calculation of the next column. To be exact, only the ones are needed that correspond to non-zero entries in R, which leads to:

$$z_{ll} = \frac{1}{r_{ll}} \left(\frac{1}{r_{ll}} - \sum_{j=l+1, r_{lj} \neq 0}^{n} r_{lj} z_{jl} \right) \tag{14}$$

$$z_{il} = \frac{1}{r_{ii}} \left(- \sum_{j=i+1, r_{ij} \neq 0}^{l} r_{ij} z_{jl} - \sum_{j=l+1, r_{ij} \neq 0}^{n} r_{ij} z_{lj} \right) \tag{15}$$

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 R matrix. 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.

6 Conclusion

We presented a new SLAM algorithm that overcomes the problems inherent to filters and is suitable for real-time applications. Efficiency arises from incrementally updating a factored representation of the smoothing information matrix. We have shown that at any time the exact trajectory and map can be retrieved in linear time. We have demonstrated that our algorithm is capable of solving large-scale SLAM problems in real-time. We have finally described how to obtain the uncertainties needed for data association, either exact or as a more efficient conservative estimate.

An open research question at this point is if a good incremental variable ordering can be found, so that full matrix factorizations can be 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. We plan to test our algorithm in a landmark-free setting, that uses constraints between poses instead, as provided by dense laser matching.

References

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

[Eustice *et al.*, 2005] 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.

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

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

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

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

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

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

[Steedly *et al.*, 2003] D. Steedly, I. Essa, and F. Dellaert. Spectral partitioning for structure from motion. In *ICCV*, 2003.

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

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