A Multifrontal QR Factorization Approach to Distributed Inference applied to Multi-robot Localization and Mapping

Frank Dellaert, Alexander Kipp, and Peter Krauthausen
College of Computing, Georgia Institute of Technology
Atlanta, Georgia 30332
Email: dellaert@cc.gatech.edu

Abstract

QR factorization is most often used as a "black box" algorithm, but is in fact an elegant computation on a factor graph. By computing a rooted clique tree on this graph, the computation can be parallelized across subtrees, which forms the basis of so-called multifrontal QR methods. By judiciously choosing the order in which variables are eliminated in the clique tree computation, we show that one straightforwardly obtains a method for performing inference in distributed sensor networks. One obvious application is distributed localization and mapping with a team of robots. We phrase the problem as inference on a large-scale Gaussian Markov Random Field induced by the measurement factor graph, and show how multifrontal QR on this graph solves for the global map and all the robot poses in a distributed fashion. The method is illustrated using both small and large-scale simulations, and validated in practice through actual robot experiments.

Introduction

In this paper we show how multifrontal QR factorization, a state of the art method for sparse matrix factorization, can be used to efficiently implement distributed *maximum a posteriori* (MAP) inference on a network of intelligent sensor nodes. The approach is most suited to coarse-grained parallelism, for example, a network of nodes equipped with enough computing power to perform floating point math.

One important application is distributed simultaneous localization and mapping (D-SLAM) with a team of autonomous robots, where each robot collects a large amount of sensor readings and the goal is to build a global map of the environment. As discussed in (Thrun 2001), successful mobile robot systems require maps for their operation, e.g. delivery robots in hospitals, and museum tour-guide robots. In some applications, e.g. teams of autonomous reconnaissance robots, detailed maps of the environment are the end-product. Ideally, in such cases, the sensor data should be processed locally and integrated into the global computation with a minimum of communication. In what follows, we show that multifrontal QR methods are ideal in this respect.

Here we focus on distributed inference tasks that can be phrased in terms of continuous-valued graphical models, which lead to least-squares sub-problems. QR orthogonalization is the method of choice for solving these (Golub & Loan 1996). While expensive for dense problems, QR is quite competitive for large *sparse* problems such as those that arise in sensor networks. While often used as a "black box" algorithm, QR factorization is in fact an elegant computation on a graph. It can be argued that in many cases, a straightforward problem statement in terms of matrices obscures the graph-based nature of the computation. In contrast, the state of the art in linear algebra is often a blend of numerical methods and advanced graph theory, sharing many characteristics with inference algorithms typically found in the graphical models literature (Cowell *et al.* 1999).

One such class of algorithms are the so-called *multifrontal QR factorization* (MFQR) methods (Matstoms 1994; Lu & Barlow 1996). The basic idea is the use of efficient *dense* QR methods, e.g. LAPACK Householder-based QR, to attack small sub-problems in the larger factorization. Because these dense methods make use of heavily optimized level-3 BLAS routines, MFQR can be quite efficient. However, *the* property of MFQR that makes it eminently suited for distributed inference is that it is easily parallelized (Amestoy, Duff, & Puglisi 1994; Lu & Barlow 1996). In sensor networks, the spatial nature of the inference problem induces a natural way of parallelizing the computation.

The central data structure that underlies multifrontal QR is the *clique tree* (Pothen & Sun 1992; Blair & Peyton 1993), also known as the junction tree in the AI literature (Cowell *et al.* 1999). In what follows, we formalize the distributed inference problem in terms of factor graphs (Kschischang, Frey, & Loeliger 2001). After a factor graph is triangulated one obtains a chordal graph with cliques that can be structured into a tree-based data structure, the clique tree. In multifrontal QR factorization, computation progresses in this tree from the leaves to the root to factorize a sparse matrix, and then from the root to the leaves to perform a backward substitution step. If the clique-tree structure respects the topology of the distributed network of sensor nodes, the computation can be done in a completely distributed fashion.

In terms of related work, the closest is that of Paskin & Guestrin, (Paskin & Guestrin 2004; Guestrin *et al.* 2004; Paskin, Guestrin, & McFadden 2005), which is very elegant and complimentary to ours. In particular, they have examined in detail how, given a topology of a sensor network, a communication spanning tree and a clique tree can be built via local message-passing only. The architecture described in (Paskin, Guestrin, & McFadden 2005) supports optimizing the spanning trees to minimize communication and separator width in the clique trees. They then perform distributed inference via belief propagation in the junction tree.

In our approach we perform a different calculation on the clique tree, but our work is very similar in spirit, and can indeed benefit from their robust architecture. We are focused on solving large non-linear inference problems using successive QR factorization phases, and are less interested in the clique marginals obtained by belief propagation. The update messages sent in the QR computation are generally much smaller than the ones for belief propagation, as they represent partial square-root factorizations. However, we do not obtain clique marginals, only the MAP estimate.

To the best of our knowledge, our approach represents the first comprehensive approach to distributed, landmark-based SLAM. Both Howard (Howard, Matarić, & Sukhatme 2002) and Dellaert (Dellaert, Alegre, & Martinson 2003) used MLE approaches to a SLAM problem where the robots themselves function as landmarks. However, the computation in these approaches takes place on a central server, introducing a severe communication problem. A distributed version of the robot configuration problem (Howard, Matarić, & Sukhatme 2003) uses a gradient descent method and hence suffers from slow convergence, in contrast to the quadratic convergence provided by MFQR.

MAP Inference on Factor Graphs/MRFs

As the underlying graphical representation of the distributed inference problem we choose factor graphs, a class of bipartite graphical models that can be used to represent factored distributions (Kschischang, Frey, & Loeliger 2001). There are nodes for unknowns and nodes for probability factors defined on them. The graph structure expresses which unknowns are involved in each factor. Without loss of generality, below we consider only single and pairwise cliques, leading to the following factored probability:

$$P(X) \propto \prod_i \phi(x_i) \prod_{\{i,j\}} \psi(x_i, x_j) \qquad (1)$$

Typically the potentials $\phi(x_i)$ encode a prior or a single measurement constraint at an unknown $x_i \in X$, whereas the pairwise potentials $\psi(x_i, x_j)$ relate to measurements that involve the relationship between two unknowns x_i and x_j. Note that the second product is over pairwise cliques $\{i, j\}$, counted once. The form (1) is also exactly the expression for a pairwise Markov random field (Yedidia, Freeman, & Y.Weiss 2000): MRFs and factor graphs are equivalent representations. In the case of Gaussian priors $\phi_i(x_i) \propto P(x_i)$ and Gaussian measurement models $\psi_k(x_i, x_j) \propto P(z_{ij}|x_i, x_j)$ we obtain a Gaussian MRF:

$$\begin{aligned} \phi_i(x_i) &= \exp\left\{-\frac{1}{2}\|x_i - \mu_i\|^2_{P_i}\right\} \\ \psi_k(x_i, x_j) &= \exp\left\{-\frac{1}{2}\|A_{ki}x_i + A_{kj}x_j - b_k\|^2_{\Sigma_k}\right\} \end{aligned}$$

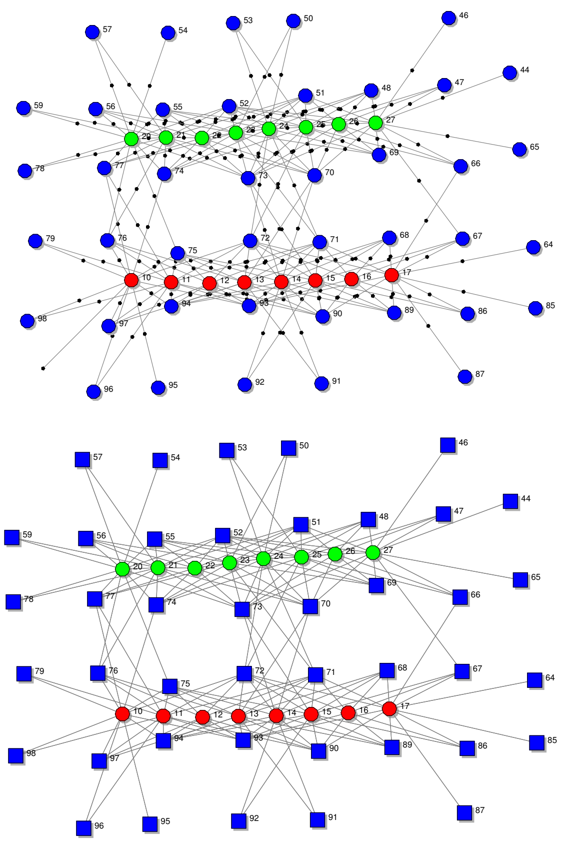


Figure 1: Equivalent factor graph (top) and MRF (bottom) representations for a distributed SLAM problem. Two robots move from left to right and perceive a set of landmarks, with each measurement inducing a factor node that relates pose and landmark unknowns. In the MRF we have rendered the landmarks as squares.

where k is the index of the measurement corresponding to the edge $\{i, j\}$. Ignoring the priors, this corresponds exactly to a least-squares problem of the form

$$\underset{X}{\operatorname{argmin}} \;\; \|AX - b\|^2_\Sigma \qquad (2)$$

where A is the sparse measurement matrix, b the right-hand side (RHS), and Σ a block-diagonal covariance matrix. Typically the graphs are very sparse: as an example, the factor graph and the MRF associated with a small distributed SLAM problem are shown in Figure 1.

For a full-rank $m \times n$ matrix A, with $m \geq n$, the unique LS solution to (2) can be found by computing the QR-factorization of A itself along with its corresponding right hand side (Golub & Loan 1996):

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

Here Q is an $m \times m$ orthogonal matrix, and R is the $n \times n$ upper-triangular *Cholesky triangle*. The orthogonal matrix Q is not usually formed: instead, the transformed RHS $Q^T b$

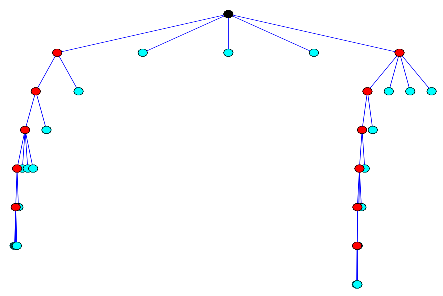


Figure 2: A clique tree, out of many possible, for the example from Figure 1. In this example, the two larger subtrees correspond to both robots and the structure they alone see, whereas he root node contains the structure seen by both.

is computed by appending b as an extra column to A. Because the Q factor is orthogonal, we have:

$$\|A\theta - b\|_2^2 = \left\|Q^T A\theta - Q^T b\right\|_2^2 = \|R\theta - d\|_2^2 + \|e\|_2^2$$

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

$$R\theta = d \tag{3}$$

using efficient back-substitution, as R is upper-triangular.

Multifrontal QR Factorization

For sparse QR factorization, the key to efficiency is exploiting the sparsity structure of the matrix A. In particular, *multifrontal QR factorization* (MFQR) (Matstoms 1994; Lu & Barlow 1996) builds a clique tree by eliminating either the factor graph or the MRF (efficient algorithms exist for both), and then recursively factorizes the matrix from the leaves up, using a sequence of dense QR factorizations. By structuring the computation as a series of small but dense factorizations, it is possible to use highly optimized dense QR subroutines for the bulk of the computation, while maximally exploiting sparsity at the structural level. In many ways, the structure of the computation is similar to that of belief propagation in junction trees.

The key insight is that Householder reflections, when used to eliminate column j of a partially factored matrix,

$$H_j\left(H_{j-1}\cdots H_1 A\right) = \begin{bmatrix} R_{1:j,1:j} & R_{1:j,j+1:n} \\ 0 & A_{22} \end{bmatrix}$$

only affect those rows in A with (below-diagonal) non-zero entries in the column. When these rows do not intersect for two given columns i and j, they can be eliminated in any order by two smaller, dense QR factorizations.

Which columns interact and in what way can be computed in advance, exactly by building the clique tree. The structure of the tree encodes the structural dependencies between the columns of A (Pothen & Sun 1992; Blair & Peyton 1993). In particular, factorizations in different subtrees can be done in parallel. As an example, a clique tree for the example from Figure 1 is shown in Figure 2. Importantly, the clique tree has to be computed only once, and can then be reused to factorize matrices with the same sparsity pattern over and over. This is well suited to non-linear inference, where the linearized system will have the same structure at every non-linear minimization iteration. An example is D-SLAM, where both bearing and range measurements are non-linear functions in pose and landmark unknowns.

After this *analysis phase*, the clique tree is used to drive a recursive, depth-first tree traversal that constitutes the *numerical phase* of the algorithm, described in detail as Algorithm 1. For each node j a *frontal matrix* F_j is factorized into a row R_j of the final product R, and an update matrix U_j, used to create the frontal matrix of its parent $\pi(j)$. The name "frontal" originates as MFQR proceeds on several "fronts" in parallel.

Algorithm 1 Multifrontal QR, adapted from (Lu & Barlow 1996).

Given a full-rank $m \times n$ matrix A and its clique tree T, call the following recursive procedure once for the root n:

- Given node j, calculate R_j and an update matrix U_j:
  1. Compute the update matrices $\{U_c\}$ for all s children c of j by making s recursive calls to this procedure
  2. Let $A[j]$ be the row-slice of A which has non-zeroes associated with unknowns in the clique j, excluding unknowns in the separator between j and its clique $\pi(j)$.
  3. Form the the frontal matrix F_j as
  $$\begin{bmatrix} A[j] \\ U_{c_1} \\ \vdots \\ U_{c_s} \end{bmatrix}$$
  4. Factorize the frontal matrix F_j using dense QR as
  $$Q_j^T F_j = \begin{bmatrix} R_{jj} & R_{j,j+1:n} \\ 0 & U_j \end{bmatrix}$$
  5. Use $R_j = \begin{bmatrix} R_{jj} & R_{j,j+1:n} \end{bmatrix}$ as the j^{th} slice of R, and return U_j as the update matrix for node j

Distributed MFQR Inference for SLAM

SLAM refers to the problem of localizing a robot while simultaneously mapping its environment. For a thorough introduction to the typical Extended Kalman Filter (EKF) approaches to SLAM, see (Smith, Self, & Cheeseman 1990; Leonard, Cox, & Durrant-Whyte 1992). In contrast, in this paper we take a non-linear smoothing approach, where we will be interested in obtaining the entire trajectory for each robot. This is a large-scale inference problem, given the number of unknowns involved, and would typically be done on a central server with ample computing power. Not counting the related 3D reconstruction in vision and photogrammetry (Triggs *et al.* 2000), we do not know of any robotics work that solves distributed SLAM problems this way. However, it is a fairly simple extension of single-robot smoothing and mapping as in (Lu & Milios 1997).

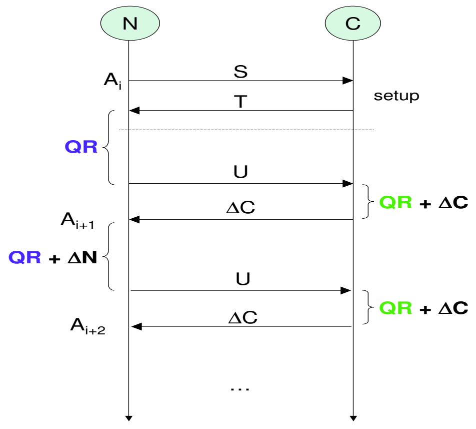


Figure 3: A sequence diagram describing the data flow for the distributed SLAM inference algorithm. In the figure, N refers to a generic network node, and C refers to the coordinating node. See text for a detailed explanation.

```
Algorithm 2 Distributed SLAM.
As a one-time setup procedure:
  1. All nodes $N$ send a symbolic version $S$ of the local measurement matrix $A$ to the root $C$.
  2. The coordinator $C$ computes the clique tree $T$ and communicates it to all nodes $N$.
Then, we perform multiple iterations until convergence:
  1. Node $N$ calculates $R_N$ corresponding to the subtree rooted by $N$, and sends an update matrix $U_N$ to $C$.
  2. Given the update matrices $\{U_N\}$ for all nodes $N$, the coordinator $C$ performs the factorization at the clique tree root to produce $R_C$.
  3. The coordinator $C$ computes the solution $\Delta C$ for the unknowns in the root clique, and sends this to all nodes $N$
  4. The nodes $N$ use this to compute the solutions $\Delta N$ for all the unknowns in their subtree.
```

We introduce the use of MFQR as a novel way in which to perform the inference in a completely distributed fashion, cutting down on both communication and the need for a highly capable central server. No measurements will have to be communicated between robots or robots and a server. Instead, the communication is limited to QR update messages, which condense the entire measurement history on the individual robots into a small upper trapezoidal matrix.

A complete characterization of the problem is outside the scope of the paper, but below we introduce our notation and the high-level equations involved. In the distributed version of SLAM, we shall denote the state of a robot r at the i^{th} time step by x_{ri}, with $r \in 1..R$ and $i \in 0..M$, a landmark by l_j, with $j \in 1..N$, and a measurement by z_k, with $k \in 1..K$. The joint probability model corresponding to SLAM is exactly modeled by a factor graph or equivalently an MRF:

$$\prod_{r=1}^{R}\left\{P(x_{r0})\prod_{i=1}^{M}P(x_{ri}|x_{ri-1},u_{ri})\right\}\prod_{k=1}^{K}P(z_k|x_{ri_k},l_{j_k})$$

Here $P(x_{r0})$ is a prior on the initial state of robot r, $P(x_{ri}|x_{ri-1},u_{ri})$ is the *motion model*, parameterized by a control input u_{ri}, and $P(z|x,l)$ is the *landmark measurement model*. The above assumes that the data-association problem has been solved, i.e., that the indices ri_k and j_k corresponding to each measurement z_k are known.

As is standard in the SLAM literature we assume Gaussian process and measurement models, defined by

$$x_i = f_i(x_{i-1},u_i) + w_i \tag{4}$$

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

$$z_k = h_k(x_{i_k},l_{j_k}) + v_k \tag{5}$$

where $h_k(.)$ is a measurement equation, and v_k is normally distributed zero-mean measurement noise with covariance Σ_k. The equations above model the robot's behavior in response to control input, and its sensors, respectively.

In practice one always considers a linearized version of the SLAM problem as the process models f_i and measurement equations h_k are typically non-linear and a succession of linear approximations will need to be considered. Below we assume that we are working on one of these iterations.

The data-flow of our current implementation is shown in Figure 3. We used a simple scheme where one of the robots is chosen as the coordinating node C, and other nodes N are connected in a trivial spanning tree with C as the root. Note that any of the schemes in (Paskin, Guestrin, & McFadden 2005) can be used instead: this will not substantially modify the algorithm. To achieve a clique tree topology that allows for this distributed computation, we first eliminate all structure seen by one robot, then the trajectory unknowns for each robot, and finally the structure seen by more than one robot. This leads to a clique tree where the root (or top tree) contains the common structure, with subtrees hanging off the root corresponding to each robot.

The computation then proceeds as in Algorithm 2. As a one-time setup procedure, a clique tree T is computed on the coordinator node C and communicated to all nodes N in the network. This incurs low communication overhead as only a symbolic version S of the measurement matrices A needs to be sent over the network. Then, each iteration executes a factorization and a back-substitution phase. In the non-linear case, the solutions computed are update vectors for the current linearization, hence they are denoted by ΔC and ΔN. As a useful fact, after the local factorization is computed on a node N, it can already update a local view of the world, e.g. to perform navigation or obstacle avoidance. When finally the update ΔC from the coordinator arrives, the local view is updated to align with the global solution. Note that multifrontal QR is used both as the top-layer distributed computation scheme, as well as to perform the factorizations on the nodes N and C. Only, in this case the clique tree spans multiple network nodes, and hence some of the edges in the tree are in fact links across the network.

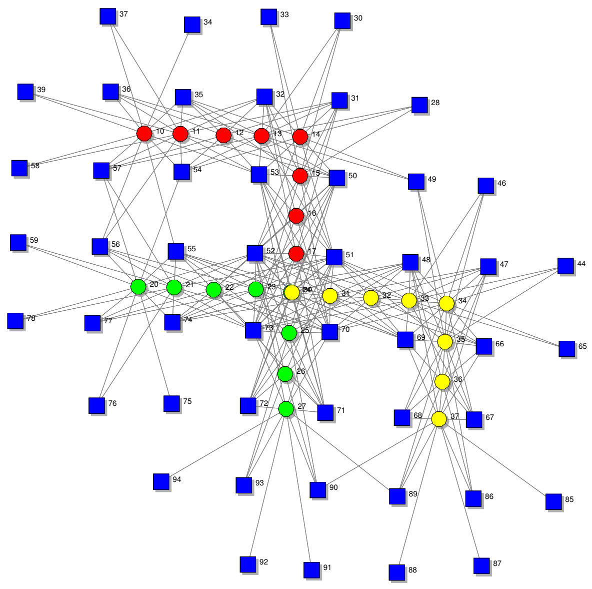


Figure 4: MRF structure and final solution for a simulated scenario in which three robots all pass through the same central location, but at three different times. Hence, they never actually see the same landmarks at the same time.

Experimental Results

Simulations We performed multiple experiments with simulated data, including the simple example from Figures 1 and 2. Here we discuss a different, representative simulation involving three robots. The MRF topology induced by the simulated trajectories and measurements is shown in Figure 4. In this scenario, three robots all pass through the same central location, but at three different times. Hence, they never actually see the same landmarks at the same time.

The quality of the solution is very good despite non-trivial simulated measurement noise: in our simulated environment, both the landmarks and robot were aligned along the main 2D axes. The global rotation that can be seen in the figure is a fundamental ambiguity that cannot be recovered by any algorithm that uses relative landmark measurements only. The clique tree computed by our implementation is shown in Figure 5. The root, shown in black, in fact contains all the landmarks corresponding to the one central location that was visited by all robots, as well as some of the other landmarks that were seen by at least two robots. Note that the algorithm has no trouble at all dealing with the multiple loops in this example. The three main subtrees correspond to one robot each, with the leaf nodes in each subtree corresponding to the landmarks seen only by that robot.

Using 10 Real Robots We also tested the algorithm in practice, using 10 four-legged Sony Aibo robots in a lab setting. The 10 robots where placed at regular intervals in a "hallway" around a 5×5 meter field of obstacles, as shown in Figure 6a. The hallway also contained 32 regularly placed artificial landmarks, consisting of cylinders wrapped with

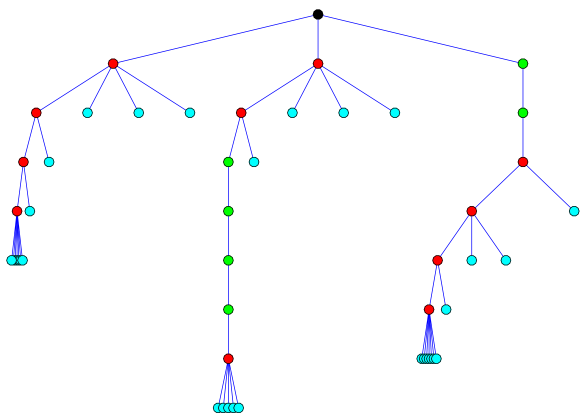


Figure 5: The clique tree computed in our distributed implementation for the simulation from Figure 4.

colored paper, as frequently used in the Robocup competitions. Each robot was instructed to walk 3m back and forth, while its built-in camera took images and stored them on a memory stick. The Aibo robots are challenging for SLAM applications as their odometry is notoriously unreliable, differs from robot to robot, and depends heavily on factors such as the floor texture or even the battery level.

We did not implement the MFQR algorithm on the Aibos, but have tested the entire distributed computation that would take place on the robots, using 10 executables, one for each robot. The images were processed off-line to detect landmarks and generate the measurement data for each robot. Data association was not an issue as each landmark was locally unique and confusion was not possible. After landmarks were detected in the image, bearing and range of each landmark were determined using the centroid image coordinates and the size of the landmarks.

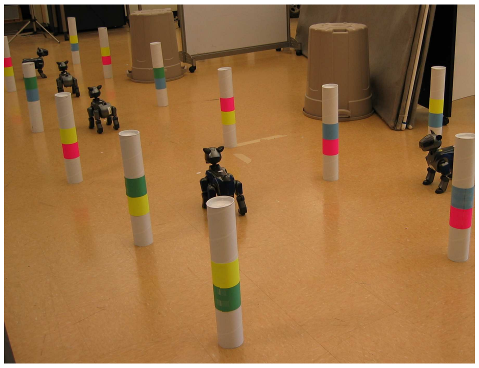
The clique tree used and the final result are also shown in Figure 6. The reconstruction is noisy but reasonable given the unreliable odometry and the relatively few measurement sightings. Still, the loop is closed and the computation was flawless, matching that of a single server implementation.

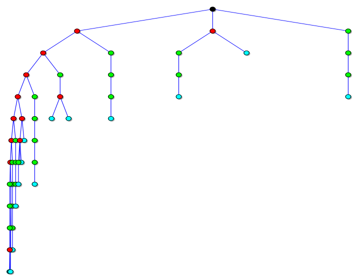
Conclusion and Future Work

We believe that MFQR will play a large role in the future of distributed inference such as needed in intelligent sensor networks and more involved computations such as distributed SLAM with a team of intelligent robots. We demonstrated the viability of the approach in the domain of distributed SLAM, but the framework applies more generally to any distributed computation whose sub-problems can be mapped to a Gaussian MRF.

Our implementation was tested both in simulation as well as using a team of 10 robots, collaboratively mapping an environment of which a single robot sees only a small part. We believe that this is a first in landmark-based SLAM.

In future work we will examine more sophisticated schemes to build the clique tree, to reconcile the conflict-


(a) Experimental Setup


(b) Clique Tree

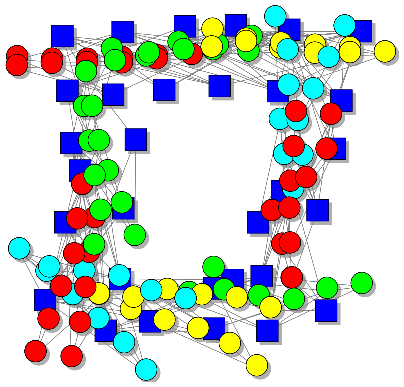

(c) Final D-SLAM Result

Figure 6: A Distributed SLAM experiment with 10 legged robots in a lab setting. A photograph conveys the experimental setup, and also shown are the clique tree used in the computation (10 subtrees), and the final result. See also the text.

ing objectives of reducing fill-in of the graph (leading to increased computation) and keeping the measurements local (to keep communication costs down). In this respect. the robust distributed inference architecture from (Paskin, Guestrin, & McFadden 2005) is an obvious candidate for us to implement and adapt to the MFQR method.

References

Amestoy, P. R.; Duff, I. S.; and Puglisi, C. 1994. Multifrontal QR factorization in a multiprocessor environment. Technical Report TR/PA/94/09, ENSEEIHT, Toulouse, France.

Blair, J., and Peyton, B. 1993. An introduction to chordal graphs and clique trees. In George, J.; Gilbert, J.; and Liu, J.-H., eds., *Graph Theory and Sparse Matrix Computations*. Springer-Verlag.

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

Dellaert, F.; Alegre, F.; and Martinson, E. 2003. Intrinsic localization and mapping with 2 applications: Diffusion mapping and Marco Polo localization. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Golub, G., and Loan, C. V. 1996. *Matrix Computations*. Baltimore: Johns Hopkins University Press, third edition.

Guestrin, C.; Bodik, P.; Thibaux, R.; Paskin, M.; and Madden, S. 2004. Distributed regression: an efficient framework for modeling sensor network data. In *Information Processing in Sensor Networks (IPSN)*.

Howard, A.; Matarić, M.; and Sukhatme, G. 2002. Localization for mobile robot teams using maximum likelihood estimation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 434–459.

Howard, A.; Matarić, M.; and Sukhatme, G. 2003. Localization for mobile robot teams: A distributed MLE approach. In Siciliano, B., and Dario, P., eds., *Experimental Robotics VIII*, Advanced Robotics Series. Springer-Verlag. 146–155.

Kschischang, F.; Frey, B.; and Loeliger, H.-A. 2001. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory* 47(2).

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

Lu, S.-M., and Barlow, J. L. 1996. Multifrontal computation with the orthogonal factors of sparse matrices. *SIAM Journal on Matrix Analysis and Applications* 17(3):658–679.

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

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

Paskin, M. A., and Guestrin, C. E. 2004. Robust probabilistic inference in distributed systems. In *Conf. on Uncertainty in Artificial Intelligence*.

Paskin, M.; Guestrin, C.; and McFadden, J. 2005. A robust architecture for distributed inference in sensor networks. In *Fourth International Conference on Information Processing in Sensor Networks, ISPN*.

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

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

Thrun, S. 2001. A probabilistic online mapping algorithm for teams of mobile robots. *Intl. J. of Robotics Research* 20(5):335–363.

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

Yedidia, J.; Freeman, W.; and Y.Weiss. 2000. Generalized belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*, 689–695.