Incremental Collaborative Trajectory Estimation Using WSN Based on Multifrontal QR Factorization

Daniel I. M. Quiñones
Escola Politécnica
Universidade de São Paulo
São Paulo, Brazil
Email: dquinones@usp.br

Cíntia B. Margi
Escola Politécnica
Universidade de São Paulo
São Paulo, Brazil
Email: cintia@usp.br

***Abstract*—Wireless Sensor Networks (WSN) are used for a variety of applications, including the target's trajectory estimation. Most proposed solutions are based on sequential estimation. However, in this paper we present a new solution to the trajectory estimation problem using the batch estimation approach. In our solution, we model the problem as a system of equations $AX = b$, with matrix A being sparse and vector X being the trajectory. Next, through multifrontal QR factorization, factorization $A = QR$ is distributed between the sensors, which calculate it collaboratively and incrementally. Simulation results show that our solution has the same performance as the centralized estimator. Also, we demonstrate its implementation viability by showing that the processing and memory requirements are compatible to generic motes characteristics.**

I. Introduction

In many WSN applications, it is necessary to estimate the trajectory of moving objects. This problem is related to the target tracking application, in which the goal is to detect and track moving objects such as animals, vehicles or humans.

Consider the problem of n homogeneous sensors $(s_1, s_2, \ldots, s_n)$ having to estimate a robot's trajectory. At each sample time k, observations of the robot's position are collected by the sensors. Equation (1) is the robot's state model, where $x(k)$ is the state vector and equation (2) is the observation model that appears in every sensor $s = 1, \ldots S$ that observes the robot. Here, we consider that we are working with linear models with F and $H_S = H$ constant. Also, $v(k)$ and $w_S(k)$ are Gaussian white noise vectors with known covariance matrices $Q(k) = Q$ and $R_S(k) = R$ respectivelly.

$$x(k) = Fx(k-1) + v(k) \quad (1)$$

$$z_S(k) = H_S x(k) + w_S(k) \quad (2)$$

Most proposed solutions to this problem use the sequential estimation approach. Another way to estimate the trajectory is the batch estimation approach [8] that assumes all trajectory observations are available to be processed simultaneously. To formulate the problem, after the WSN observes the robot's trajectory for a time window $\triangle T$, we will have several model and observation equations that can be combined to form a single system of equations $AX = b$, where vector X is the robot's trajectory, matrix $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ and vector $b \in \mathbb{R}^m$ [2], [13]. This is a least-squares problem, whose solution is obtained by the criteria in equation (3).

$$\hat{x} = \underset{X}{argmin} \left\| AX - b \right\|_2 \quad (3)$$

In this paper we present a new incremental and collaborative algorithm for trajectory estimation with WSN. We will present our solution in the context of the collaboration between WSN and a mobile robot, in which the network helps the robot to estimate its trajectory. To solve the system $AX = b$, the sensor nodes factorize matrix A in an incremental and collaborative way by using multifrontal QR factorization. Then the robot estimates its trajectory by back substitution.

The remainder of the paper is outlined as follows: in Section II we provide a brief description of the mathematical background used to develop our solution. Section III describes our incremental solution, while Section IV presents our experimental results and discussion and Sections V and VI present related work and conclusions.

II. Background

To solve equation (3) we use QR factorization of matrix A . The QR factorization calculates $A = QR$, where $Q \in \mathbb{R}^{m \times m}$ is an orthogonal matrix and $R \in \mathbb{R}^{n \times n}$ is an upper triangular matrix [7]. By using $A = QR$, equation (3) is transformed into $Rx = Q^T b$, which is solved by back substitution. Usually, matrix Q is not formed: the vector $Q^T b$ is calculated by working directly on the augmented matrix $[A|b]$ instead. This is because Q is orthogonal and factorizing $[A|b]$ results in $[A|b] = Q\left[R|Q^T b\right]$. From now on, when we refer to matrix A, it is assumed that we are working with the extended matrix $[A|b]$.

Because matrix A is sparse, we factorize it using *multifrontal QR factorization* (MFQR), which is a well known algorithm for calculating sparse matrix QR factorizations [10]. The MFQR algorithm has two phases: analysis phase and numerical factorization phase. In the analysis phase, the algorithm determines a graph $T(A)$ called elimination tree of matrix A, whose nodes follow the important property (among others) that, if two nodes belong to different subtrees, can be eliminated in parallel. The numerical factorization phase calculates R using $T(A)$ as a guide, beginning on the leaf

978-1-4799-3780-6/14/$31.00 ©2014 IEEE

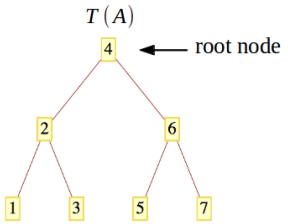

Figure 1: Elimination tree of matrix A of the system of equations $Ax = b$ formed after a WSN observed 7 states of the robot trajectory and ordered by Nested Dissection algorithm.

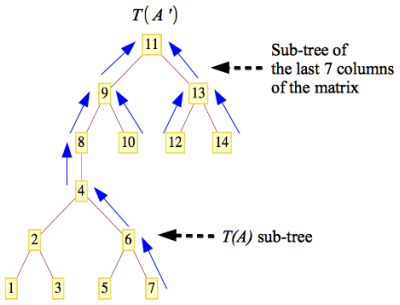

Figure 2: $T(A')$ which allows the incremental factorization. The blue arrows show the path of the MFQR algorithm to update the R factor incrementally.

nodes and ending on the root. For every node i, a frontal matrix F_i is calculated, which is composed by rows of matrix A and one update matrix U_{ch} for every children of node i in $T(A)$. Then, the algorithm calculates $F_i = Q_i R_i$, in which the first row of R_i corresponds to the ith row of R and the rest of the matrix is the update matrix U_i which is used by the parent of node i to form its frontal matrix.

In our solution, we exploit the parallelism obtained by $T(A)$. This parallelism depends on the structure of the sparse matrix and a parallel factorization may not be possible [4]. This is the case for the problem at hand, for which $T(A)$ is a chain, as a result of the Markov property followed by the robot's state model [1]. To achieve parallel factorization, we used the Nested Dissection column ordering algorithm [5] due to the fact that, for the trajectory estimation problem, it results a $T(A)$ with less levels and more subtrees than the obtained with other algorithms.

Matrix A grows with time and has a known structure. To develop our solution, we exploit these knowledge to propose a way to factorize it incrementally. The main idea is to find a column ordering of matrix A whose $T(A)$ allows *incremental factorization*. This column ordering is possible owing to the fact that the ordering of the nodes of $T(A)$ is topological [6]. To ilustrate this idea, consider the $T(A)$, as depicted in Figure 1, which corresponds to the principal matrix of a system of equations $AX = b$ formed after a WSN observed 7 robot's states and after its columns being reordered by Nested Dissection. If 7 new states are observed, 7 new columns will be added to matrix A at its right side, resulting in a new matrix A'. If we order the new columns using Nested Dissection, leaving the old columns intact, the resulting $T(A')$ resemble two concatenated sub-trees, as shown in Figure 2. This happens because the ordering is topological and the child nodes (columns) in the tree are always placed at the left side in the matrix. Consequently, the root of $T(A)$ has to be child of at least one of the new nodes introduced in $T(A')$. In the example, node 4 (the old root) is child of node 8, which is a leaf of the sub-tree formed by the new nodes.

The advantage of ordering the columns in this manner is that $T(A')$ allows us to factorize matrix A' incrementally. Having factorized matrix A using $T(A)$, to factorize matrix A' we do not need to eliminate all nodes of $T(A')$ because most of them have been eliminated in the first factorization. However, to exploit that, we have to analyze the structure of matrix A', because $T(A')$ does not show clearly the relations of the columns of matrix A'. In our example, because of the Markov property of the state model, state 8 is related to state 7 in matrix A'. This is not seen in $T(A')$ although it has to be considered to properly factorize matrix A'. As a consequence, we have to re-eliminate node 7 and all nodes within the path from node 7 to node 4 to calculate U_4, which is required by node 8 to form its frontal matrix. Figure 2 shows the algorithm path in solid blue arrows.

III. Incremental Collaborative Algorithm

Here, we present our incremental collaborative algorithm to estimate a mobile robot's trajectory. It is collaborative because the sensors have to collaborate with each other to factorize matrix A. It is incremental because at every time window ΔT we update matrix R incrementally.

Our solution consists of three phases, executed every ΔT: data collection phase; factorization phase and back substitution phase.

Data collection phase: In this phase, during the time window ΔT, the sensors collect observations of the robot's position and, for each observation, the sensors send a detection notification message to the robot. With these messages the robot forms a table that we call *Table of Observed States*. This table contains the information of the states and the group of sensors that have observed the robot.

Factorization phase: This phase begins at the end of ΔT and consists of three steps: determination of $T(A)$; cluster formation; and numerical factorization. In the determination of $T(A)$ step, the robot uses the ordering described in section II to determine $T(A)$ in order to calculate the incremental factorization. Then, in the cluster formation step, the robot maps $T(A)$ to the WSN by assigning a cluster of sensors to every node of the tree. For doing so, the robot uses the *Table of Observed States*. In the end, the robot sends this information to the WSN to proceed to the factorization. In the numerical factorization step, the WSN uses the MFQR algorithm to factorize matrix A. The procedure was described in section II.

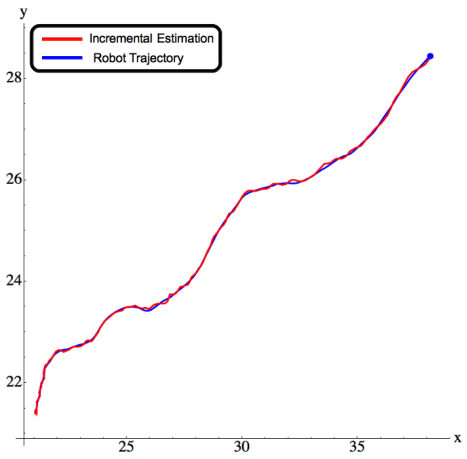

Figure 3: Incremental collaborative trajectory estimation

Beginning at the leaves, each cluster head will collect all observations of the cluster's members and all update matrices of its children in $T(A)$ to form its corresponding frontal matrix, then calculates the line r_i which is sent to the robot and U_i which is sent to its parent cluster head in $T(A)$.

Back substitution phase: After the factorization phase, the robot estimates its trajectory by solving the system $RX' = Q^T b$ using back substitution and re-ordering vector X' following the column ordering used in the factorization phase.

IV. Experimental Results & Discussion

We performed experiments with simulated data and real hardware. Simulations were used to evaluate the estimation performance of our solution and were done using Mathematica 9. To validate the implementation feasibility, we conducted experiments using a TelosB mote.

A. Simulations

We modeled a WSN with $n = 361$ omni-directional sensors evenly spaced in a square grid. Despite this configuration, our algorithm does not depend on any WSN topology to work. Also, we assume that if the robot is detected between two sensors, they are able to communicate and share information. For the robot's state model, we used the Discrete White Noise Acceleration (DWNA) model [1]. This model facilitates the demonstration of the incremental and collaborative characteristics of our algorithm due to its simplicity.

To explore our solution quality we compared it to the centralized estimator, which collects all observations of the WSN, forms the system $Ax = b$ and solves it by least-squares. Figure 3 shows an estimated trajectory obtained by our solution that closely follows the robot's trajectory. In addition, when the same set of observations of the robot's trajectory is used by the centralized estimator, the estimated trajectory is exactly the same.

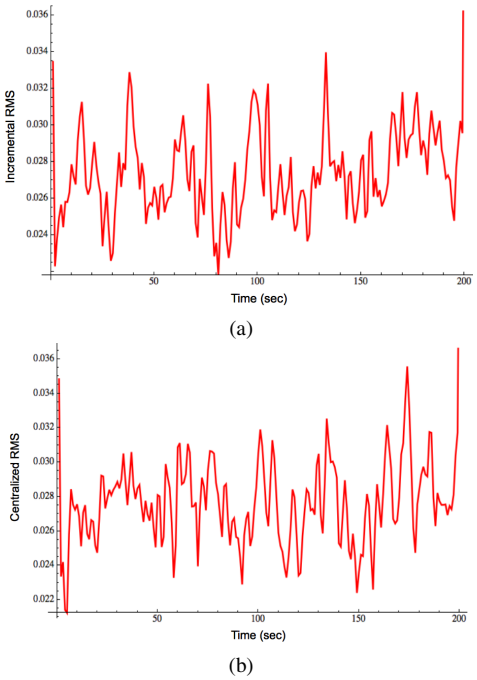

Figure 4: RMS error for the incremental and centralized solutions: (a) Incremental RMS error with an average value of 0.0275 and (b) Centralized RMS error with an average value of 0.0277

Table I: Memory space used and average factorization time spent by a Crossbow TelosB mote

Matrix Size ($m \times n$)	F_i Memory Space (KiB)	Avg. Factorization Time (sec)
28×13	11.38	2.452
30×13	12.19	2.771
32×13	13.00	3.104

The reason for obtaining the same estimated trajectory with both estimators is the fact that, with the MFQR factorization, we can distribute the same system solved by the centralized estimator into the WSN, without reformulations or minimizations.

Moreover, this result is corroborated by the Root Mean Square (RMS) error of both estimators, shown in Figure 4, where we see that the average incremental RMS error is equal to 0.0275 and that the average centralized RMS error is equal to 0.0277. The RMS error results demonstrate that both solutions have the same performance.

B. Implementation viability

To verify that a mote has enough processing power and memory to store the frontal matrices, we conducted a $N = 100$ Monte Carlo run with the simulator and, for every run, we extracted the largest frontal matrix. Then we used a TelosB mote to factor every matrix and to calculate the factorization time. In this experiment, matrix elements are represented by a 32-bit floating point data type.

Table I shows the results, in which for the largest matrix

the TelosB used 13 KiB of memory and spent 3.104 sec in the factorization. This demonstrates that the mote could execute the factorization and have enough memory to store F_i, but suggests that we have to be careful with the factorization time. If the WSN has to use a large $T(A)$ at a particular time window, the factorization time will be too large for a good operation of the algorithm.

C. Communication

Our solution has three specific communication patterns. The first occurs in the data collection phase, in which the sensors send the detection notification message to the robot. This corresponds to 1 byte message with its corresponding acknowledgement message sent by the robot.

The second pattern occurs after the cluster formation, when the robot sends the column ordering, $T(A)$ and the cluster information to the cluster heads. This information is represented by vectors that varies in size with the configuration of the network. Concerning our simulations configuration, a single message of 100 bytes and its acknowledgement message are sufficient to send this information.

The third pattern occurs in the numerical factorization step, in which there are 4 types of communication:

1) In the begining, the communication between the robot and the root cluster head, which corresponds to 1 start factorization message, that also serves as a request message for the r_{root} line of R, and its response.
2) The communication between the cluster head and sensors in the cluster, which corresponds to 1 request of observation message and the observation response.
3) The communication between cluster heads, which consist in a request message for U_i and its response
4) The communication between cluster heads and robot, which correspond to the transmission of the r_i line of the R factor and the reception acknowledgement message

V. Related Work

Most proposed solutions to the WSN target tracking problem use filters (such as Kalman filter) that have been adapted to fit to the restricted WSN resources [11], [12], [14], [15]. Our work differs from these solutions in that we use a batch estimation approach. Our work was inspired by the use of the MFQR factorization to solve the distributed multi-robot Simultaneous Localization and Mapping (SLAM) problem [3], in which the MFQR factorization was used to distribute, between a system of mobile robots, the estimation of the map and their trajectories. In subsequent work, Kaess et. al. presented iSAM2 [9], an incremental solution to the Smoothing and Mapping (SAM) problem, where the authors use a Bayes tree to store and compute the Cholesky factor R and to allow incremental reordering and fluid relinearization of the problem. Our solution differs from these works in that the trajectory estimation problem with WSN has a very different structure than the SLAM and SAM problems. Additionally, they use robots, which have more resources and power than WSN nodes. For this reason, based on our knowledge of the structure of the trajectory estimation problem and to minimize the processing and memory usage of the sensor, we use the elimination tree to maintain frontal matrices small, instead of other data structures such as the clique tree and the Bayes tree, which could lead to larger frontal matrices.

VI. Conclusion

We presented a new algorithm for trajectory estimation with WSN. Our approach use the MFQR algorithm to distribute the solution between the sensor nodes, which incrementally factorize the principal matrix A of the problem. Our results show that this new method has the same estimation performance that the centralized approach. Additionally, we showed that a mote with low resources, a TelosB, could store the frontal matrix F_i in memory and perform the factorization of it. This factorization is the principal calculation of our algorithm.

In future work, we plan to implement our algorithm on an actual robot-WSN collaboration system.

References

[1] Yaakov Bar-Shalom, Thiagalingam Kirubarajan, and X.-Rong Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[2] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research*, 25(12):1181–1203, Dec 2006.

[3] Frank Dellaert, Alexander Kipp, and Peter Krauthausen. A multifrontal qr factorization approach to distributed inference applied to multi-robot localization and mapping. In *in Proceedings of the American Association for Artificial Intelligence 9-13 July*, pages 1261–1266, 2005.

[4] James W. Demmel, Michael T. Heath, and Henk A. van der Vorst. Parallel numerical linear algebra. Technical report, UC Berkelly, 1992.

[5] Alan George. Nested dissection of a regular finite element mesh. *SIAM Journal of Numerical Analisys*, 10(2):345–363, 1973.

[6] Alan George. On finding and analizing the structure of the cholesky factor. In G. Winter Althaus and E. Spedicato, editors, *Algorithms for Large Scale Linear Algebraic Systems: Applications in Science and Engineering*. Kluwer Academic Publishers, 1998.

[7] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3 edition, 1996.

[8] David L. Hall. Estimation and kalman filters. In *Distributed Sensor Networks*. Chapman and Hall/CRC, 2004.

[9] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 30(14), 2011.

[10] Pontus Matstoms. Sparse qr factorization in matlab. *ACM Trans. Math. Softw.*, 20(1):136–159, 1994.

[11] A. Ribeiro, I. Schizas, S. Roumeliotis, and G. Giannakis. Kalman filtering in wireless sensor networks. *Control Systems, IEEE*, 30(2):66 – 86, April 2010.

[12] X. Sheng, Y.-H. Hu, and Parameswaran Ramanathan. Distributed particle filter with gmm approximation for multiple targets localization and tracking in wireless sensor network. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 181–188, April 2005.

[13] Gilbert Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge, 1986.

[14] Yan Zhou and Jianxun Li. Quantized measurement fusion for target tracking in wireless sensor networks. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 6835 – 6840, Dec. 2009b.

[15] Shanying Zhu, Cailian Chen, and Xinping Guan. Distributed optimal consensus filter for target tracking in heterogeneous sensor networks. In *Control Conference (ASCC), 2011 8th Asian*, pages 806 –811, May 2011.