2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

A Sparsity-aware QR Decomposition Algorithm for Efficient Cooperative Localization

Ke X. Zhou* and Stergios I. Roumeliotis†
*Dept. of Electrical and Computer Engineering, †Dept. of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455 Email: {kezhou|stergios}@cs.umn.edu

Abstract— **This paper focuses on reducing the computational complexity of the extended Kalman filter (EKF)-based multi-robot cooperative localization (CL) by taking advantage of the sparse structure of the measurement Jacobian matrix H. In contrast to the standard EKF update, whose complexity is up to $\mathcal{O}(N^4)$ (N is the number of robots in a team), we introduce a Modified Householder QR algorithm which fully exploits the sparse structure of the matrix H, and prove that the overall complexity of the EKF update, based on our QR factorization scheme, reduces to $\mathcal{O}(N^3)$. Finally, we validate the Modified Householder QR algorithm through extensive simulations, and demonstrate its superior performance both in terms of accuracy and CPU runtime, as compared to the current state-of-the-art QR decomposition algorithm for sparse matrices.**

I. Introduction

Multi-robot teams (sensor networks) have recently attracted significant interest in the research community because of their robustness, versatility, speed, and potential applications, such as environmental monitoring [1], surveillance [2], human-robot interaction [3], and target tracking [4]. Regardless of the application, every robot in the team must be able to accurately localize itself in an unknown environment to ensure successful execution of its tasks. While each robot can independently estimate its own pose (position and orientation) by integrating its linear and rotational velocities, e.g., from wheel encoders [5], the uncertainty of the estimates generated using this technique (dead-reckoning) increases fast, and eventually renders them unreliable. Although one can overcome this limitation by equipping every robot with absolute positioning sensors, such as the Global Positioning System (GPS) receivers, GPS signals are unreliable in urban environments and unavailable in space and underwater. On the other hand, by performing cooperative localization (CL), where communicating robots use relative measurements (distance, bearing, and orientation) to *jointly* estimate the robots' poses, the accuracy of the robot pose estimates can significantly improve [6], even in the absence of GPS.

As shown in [5], in CL, each robot can process its own proprioceptive measurements independently and distributively. However, since CL involves joint-state estimation, the processing of exteroceptive measurements (i.e., relative robot-to-robot measurements) requires the robots to communicate with each other and update the covariance matrix corresponding to all pose estimates in a centralized fashion. Using the extended Kalman filter (EKF) framework, the computational cost for processing each exteroceptive measurement is $\mathcal{O}(N^2)$ (N being the number of robots) [5]. Since at every time step the maximum possible number of exteroceptive measurements is $(N-1)N$, the overall cost for updating the estimate's covariance, in the worst case, becomes $\mathcal{O}(N^4)$ per time step. As N grows, this high computational complexity may prohibit real-time performance.

In this paper, we investigate the computational complexity[1] of the centralized EKF-based CL algorithm, considering the most challenging case where the total number of relative measurements per time step is $(N-1)N$. The main contributions of this work are the following: We show that the computational cost of the covariance update can be reduced to $\mathcal{O}(N^3)$ by employing the Information filter (see Sec. III-C). To further improve the numerical stability of the Information filter, we present an EKF update approach based on the QR factorization of the measurement Jacobian $\mathbf{H}$ in Sec. III-D. While the Standard Householder QR algorithm requires $\mathcal{O}(N^4)$ operations for decomposing $\mathbf{H}$, we present the Modified Householder QR algorithm (see Sec. IV), which exploits the sparse structure of $\mathbf{H}$ to reduce the cost of QR factorization to $\mathcal{O}(N^3)$. As a result, the overall computational complexity of the EKF-based CL is also reduced from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^3)$ per update step.

II. Literature Review

Multi-robot cooperative localization (CL) has received considerable attention in the literature (e.g., [8], [9], [10], [11]). Various system architectures, such as centralized [5], [12] or distributed [13], [14], [15], have been proposed for CL. These system architectures use various estimation algorithms, such as the EKF [5], the maximum likelihood estimator (MLE) [12], the maximum a posteriori estimator (MAP) [15], and particle filters (PF) [9]. In this paper, we focus our discussion on centralized or distributed EKF-based CL. In what follows, we denote as N the number of robots in the team and consider the worst-case computational complexity where the total number of relative measurements per time step is $(N-1)N$.

This work was supported by the University of Minnesota through the Digital Technology Center (DTC), the National Science Foundation (IIS-0643680), and AFOSR (FA9550-10-1-0567).

[1]In the remainder of the paper, computational complexity refers exclusively to time (or processing) complexity. For the analysis of space (or memory) complexity, the interested reader can refer to [7] for more details.

In [5], Roumeliotis and Bekey showed that within the EKF framework, each robot can independently propagate its own state and covariance in a fully distributed fashion. However, during the update step, the observing robot needs to broadcast its relative measurements to the rest of the robots in the team, and update the covariance matrix corresponding to all pose estimates in a centralized fashion. In this approach, the computational complexity for processing each exteroceptive measurement is $\mathcal{O}(N^2)$. Therefore, in the worst case, the overall processing cost for sequentially updating the state and covariance becomes $\mathcal{O}(N^4)$ per time step. In order to reduce the computational requirements of the EKF-based CL, several *approximate* algorithms have been proposed in the literature.

Panzieri *et al.* [16] presented a fully-decentralized algorithm based on the Interlaced EKF [17], where each robot only processes relative measurements taken by itself to update its own pose estimate, while treating the other robots' poses as deterministic parameters. The computational cost per robot and time step is $\mathcal{O}(N)$ [hence the overall cost per time step is $\mathcal{O}(N^2)$] when relative measurements are processed sequentially. Similarly to [16], Karam *et al.* [18] proposed a distributed EKF-based method for CL. However, contrary to [16], here the robots are restricted to exchange their state estimates with others within communication range. The computational cost per robot and time step is $\mathcal{O}(N^2)$, resulting in an overall cost of $\mathcal{O}(N^3)$ per time step.

Martinelli [19] developed a distributed approach for CL that uses hierarchical EKF filters to estimate the robots' poses. In particular, the N robots in a team are divided into L groups, each comprising M robots ($N = LM$). Every group contains a leader who processes all the relative measurements between any two robots in that group and only updates the pose estimates of the robots belonging to its group. A team leader is in charge of processing all observations between any two robots from different groups and only updates the pose estimates of the L group leaders. The computational cost per time step for each group leader and the team leader are $\mathcal{O}(M^4)$ and $\mathcal{O}(N(N-M)L^2)$, respectively.

The main drawback of the aforementioned approximate algorithms (see [16], [18], [19]) is that in order to reduce the computational complexity of the EKF-based CL, these approaches ignore cross-correlations amongst robots, which often leads to overly optimistic and inconsistent estimates. In this paper, we present an algorithm that reduces the complexity of the EKF-based CL to $\mathcal{O}(N^3)$, without introducing any approximations, but, instead, by taking advantage of the specific sparse structure of the measurement Jacobian matrix.

III. Problem Formulation

Consider a group of N mobile robots performing CL in 2-D by processing relative *distance and bearing* measurements. In this paper, we study the case of *global* localization, i.e., the pose (position and orientation) of each robot is described with respect to a fixed (global) frame of reference.

The pose of the ith robot (or robot-i) at time-step k is denoted as $\mathbf{x}_k^i = [(\mathbf{p}_k^i)^{\mathrm{T}} \ \phi_k^i]^{\mathrm{T}}$, where $\mathbf{p}_k^i = [x_k^i \ y_k^i]^{\mathrm{T}}$ and ϕ_k^i represent the global position and orientation of the ith robot at time-step k, respectively. The state vector for the robot team at time-step k is defined as $\mathbf{x}_k = [(\mathbf{x}_k^1)^{\mathrm{T}} \ (\mathbf{x}_k^2)^{\mathrm{T}} \ \ldots \ (\mathbf{x}_k^N)^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{3N}$. We assume that each robot is equipped with proprioceptive sensors (e.g., wheel encoders), which measure its linear and rotational velocities, as well as exteroceptive sensors (e.g., laser scanners), which can detect, identify, and measure the relative distance and bearing to other robots.

A. State Propagation

The discrete-time state propagation equation for robot-i from time-step $k-1$ to k is

$$\mathbf{x}_k^i = \mathbf{f}_{k-1}^i(\mathbf{x}_{k-1}^i, \mathbf{u}_{k-1}^i, \mathbf{w}_{k-1}^i), \ i = 1, \ldots, N,$$

where the control input $\mathbf{u}_{k-1}^i = [v_{k-1}^i \ \omega_{k-1}^i]^{\mathrm{T}}$, consisting of the linear velocity measurement v_{k-1}^i and rotational velocity measurement ω_{k-1}^i recorded by the robot's odometric sensors, is corrupted by zero-mean, white Gaussian process noise $\mathbf{w}_{k-1}^i = [w_{k-1,v}^i \ w_{k-1,\omega}^i]^{\mathrm{T}}$ with covariance $\mathbf{C}_{\mathbf{w}}^i$.

In this work, we employ the extended Kalman filter (EKF) for recursively estimating the robot's pose $\mathbf{x}_k^i, \ i = 1, \ldots, N$. Thus the estimate of the ith robot pose is propagated by[2]

$$\hat{\mathbf{x}}_{k|k-1}^i = \mathbf{f}_{k-1}^i(\hat{\mathbf{x}}_{k-1|k-1}^i, \mathbf{u}_{k-1}^i, \mathbf{0}), \ i = 1, \ldots, N,$$

where $\hat{\mathbf{x}}_{\ell|j}$ is the state estimate at time-step ℓ, after measurements up to time-step j have been processed.

The covariance matrix corresponding to the state estimate $\hat{\mathbf{x}}_{k|k-1}$ is propagated as

$$\mathbf{P}_{k|k-1} = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{\Phi}_{k-1}^{\mathrm{T}} + \mathbf{G}_{k-1}\mathbf{C}_{\mathbf{w}}\mathbf{G}_{k-1}^{\mathrm{T}}, \quad (1)$$

where $\mathbf{\Phi}_{k-1} = \mathrm{diag}(\mathbf{\Phi}_{k-1}^1, \ldots, \mathbf{\Phi}_{k-1}^N)$ with $\mathbf{\Phi}_{k-1}^i = \mathbf{\nabla}_{\mathbf{x}_{k-1}^i}\mathbf{f}_{k-1}^i$, and $\mathbf{G}_{k-1} = \mathrm{diag}(\mathbf{G}_{k-1}^1, \ldots, \mathbf{G}_{k-1}^N)$ with $\mathbf{G}_{k-1}^i = \mathbf{\nabla}_{\mathbf{w}_{k-1}^i}\mathbf{f}_{k-1}^i, \ i = 1, \ldots, N$. The overall process noise covariance is $\mathbf{C}_{\mathbf{w}} = \mathrm{diag}(\mathbf{C}_{\mathbf{w}}^1, \ldots, \mathbf{C}_{\mathbf{w}}^N)$.

Note that due to the block diagonal structures of $\mathbf{\Phi}_{k-1}$ and $\mathbf{G}_{k-1}$, the overall processing cost of (1) is $\mathcal{O}(N^2)$ [5].

B. Measurement Model

At time-step k, the relative distance and bearing observations recorded by the exteroceptive sensors of robot-i measuring robot-j ($1 \leq i \neq j \leq N$) are given by

$$\mathbf{z}_k^{i,j} = \mathbf{h}_k^{i,j}(\mathbf{x}_k^i, \mathbf{x}_k^j) + \mathbf{n}_k^{i,j}, \quad (2)$$

where $\mathbf{h}_k^{i,j} = [d_k^{i,j} \ \theta_k^{i,j}]^{\mathrm{T}}$, with $d_k^{i,j}$ and $\theta_k^{i,j}$ denoting the true distance and bearing from robot-i to robot-j at time-step k, and $\mathbf{n}_k^{i,j} = [n_{k,d}^{i,j} \ \ n_{k,\theta}^{i,j}]^{\mathrm{T}}$ is zero-mean white Gaussian measurement noise with covariance $\mathbf{C}_{\mathbf{n}}^{i,j}$. Without loss of generality, we assume $\mathbf{C}_{\mathbf{n}}^{i,j} = \mathbf{I}_2$ ($1 \leq i \neq j \leq N$) in the remainder of the paper.

[2]In the remainder of the paper, the "hat" symbol $\hat{}$ is used to denote the estimated value of a quantity, while the "tilde" symbol $\tilde{}$ is used to signify the error between the actual value of a quantity and its estimate. The relationship between a variable x and its estimate $\hat{x}$, is $\tilde{x} = x - \hat{x}$. Additionally, $\mathbf{0}_{m\times n}$ and $\mathbf{I}_n$ represent the $m \times n$ zero matrix and $n \times n$ identity matrix, and $\mathbf{e}_1$ is the unit vector with a 1 in the 1st coordinate and 0's elsewhere.

The measurement-error equation for $\mathbf{z}_k^{i,j}$, obtained by linearizing (2) around the current best estimate $\hat{\mathbf{x}}_{k|k-1}$, is

$$\begin{aligned}\tilde{\mathbf{z}}_{k|k-1}^{i,j} &= \mathbf{z}_k^{i,j} - \mathbf{h}_k^{i,j}(\hat{\mathbf{x}}_{k|k-1}^{i}, \hat{\mathbf{x}}_{k|k-1}^{j}) \qquad (3)\\ \approx \boldsymbol{\Psi}_k^{i,j}\tilde{\mathbf{x}}_{k|k-1}^{i} &+ \boldsymbol{\Upsilon}_k^{i,j}\tilde{\mathbf{x}}_{k|k-1}^{j} + \mathbf{n}_k^{i,j} = \mathbf{H}_k^{i,j}\tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_k^{i,j},\end{aligned}$$

where $\boldsymbol{\Psi}_k^{i,j} = \boldsymbol{\nabla}_{\mathbf{x}_k^i}\mathbf{h}_k^{i,j}$ and $\boldsymbol{\Upsilon}_k^{i,j} = \boldsymbol{\nabla}_{\mathbf{x}_k^j}\mathbf{h}_k^{i,j}$ are 2×3 matrices, and the measurement Jacobian matrix $\mathbf{H}_k^{i,j}$ (of dimensions $2 \times 3N$) has a sparse structure (without loss of generality, we assume $i < j$)

$$\mathbf{H}_k^{i,j} = \begin{bmatrix}\mathbf{0}_{2\times 3(i-1)} & \boldsymbol{\Psi}_k^{i,j} & \mathbf{0}_{2\times 3(j-i-1)} & \boldsymbol{\Upsilon}_k^{i,j} & \mathbf{0}_{2\times 3(N-j)}\end{bmatrix}. \quad (4)$$

In fact, for the relative distance and bearing measurement model, there are at most 9 nonzero elements in $\mathbf{H}_k^{i,j}$ (4 from the distance, and 5 from the bearing measurement) [7].

In this paper, we consider the most challenging, in terms of computational requirements, scenario where the sensing range of the exteroceptive sensors is sufficiently large so that each robot can detect, identify, and measure the relative distance and bearing to the remaining $N-1$ robots at every time step.[3] Thus, the total number of relative measurements per time step is $(N-1)N$.

The measurement-error equation for the robot team, obtained by stacking all the measurement residuals $\tilde{\mathbf{z}}_{k|k-1}^{i,j}$ $(1 \leq i \neq j \leq N)$ [see (3)] into a column vector, is

$$\tilde{\mathbf{z}}_{k|k-1} \approx \mathbf{H}_k\tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_k,$$

where $\tilde{\mathbf{z}}_{k|k-1} = [(\tilde{\mathbf{z}}_{k|k-1}^{1,2})^{\mathrm{T}} \dots (\tilde{\mathbf{z}}_{k|k-1}^{i,j})^{\mathrm{T}} \dots (\tilde{\mathbf{z}}_{k|k-1}^{N,N-1})^{\mathrm{T}}]^{\mathrm{T}}$ is the measurement residual error vector of dimension $2(N-1)N$, and $\mathbf{n}_k = [(\mathbf{n}_k^{1,2})^{\mathrm{T}} \dots (\mathbf{n}_k^{i,j})^{\mathrm{T}} \dots (\mathbf{n}_k^{N,N-1})^{\mathrm{T}}]^{\mathrm{T}}$ is zero-mean, white Gaussian measurement noise with covariance $\mathbf{C}_{\mathbf{n}} = \mathbf{I}_{2(N-1)N}$.

The overall measurement Jacobian matrix $\mathbf{H}_k = [(\mathbf{H}_k^{1,2})^{\mathrm{T}} \dots (\mathbf{H}_k^{i,j})^{\mathrm{T}} \dots (\mathbf{H}_k^{N,N-1})^{\mathrm{T}}]^{\mathrm{T}}$, whose dimensions are $2(N-1)N \times 3N$, has the following sparse structure [see (4)]

$$\mathbf{H}_k = \begin{bmatrix}
\boldsymbol{\Psi}_k^{1,2} & \boldsymbol{\Upsilon}_k^{1,2} & & & & \\
\boldsymbol{\Psi}_k^{1,3} & & \boldsymbol{\Upsilon}_k^{1,3} & & & \\
\vdots & & & & & \\
\boldsymbol{\Psi}_k^{1,N} & & & & & \boldsymbol{\Upsilon}_k^{1,N} \\
\boldsymbol{\Upsilon}_k^{2,1} & \boldsymbol{\Psi}_k^{2,1} & & & & \\
 & \boldsymbol{\Psi}_k^{2,3} & \boldsymbol{\Upsilon}_k^{2,3} & & & \\
 & \vdots & & & & \\
 & \boldsymbol{\Psi}_k^{2,N} & & & & \boldsymbol{\Upsilon}_k^{2,N} \\
 & & \vdots & \vdots & \vdots & \\
\boldsymbol{\Upsilon}_k^{N,1} & & & & & \boldsymbol{\Psi}_k^{N,1} \\
 & \boldsymbol{\Upsilon}_k^{N,2} & & & & \boldsymbol{\Psi}_k^{N,2} \\
 & & & & & \vdots \\
 & & & & \boldsymbol{\Upsilon}_k^{N,N-1} & \boldsymbol{\Psi}_k^{N,N-1}
\end{bmatrix}. \quad (5)$$

Remark 1: We highlight two important properties of $\mathbf{H}_k$. Firstly, due to its sparse structure [see (4)-(5)], each row of $\mathbf{H}_k$ has at most 5 non-zero elements. Per column, there exist at most $4(N-1)$ non-zeros [7], e.g., the nonzero elements of the first column of $\mathbf{H}_k$ originate from the first columns of the matrices $\boldsymbol{\Psi}_k^{1,j}$ and $\boldsymbol{\Upsilon}_k^{j,1}$, $j = 2, \dots, N$. Therefore $\mathbf{nnz}(\mathbf{H}_k) \sim \mathcal{O}(N^2)$, where $\mathbf{nnz}(\mathbf{H})$ denotes the number of nonzero elements of $\mathbf{H}$. Secondly, since all the exteroceptive measurements are relative observations between each pair of robots, and absolute pose measurements (such as GPS) are unavailable, it can be shown [7] that $\mathbf{rank}(\mathbf{H}_k) \leq 3N-2$, i.e., $\mathbf{H}_k$ is not full (column) rank.

[3]For the general case when the number of measurements is less than $(N-1)N$ due to the limited sensing range of each robot, our proposed method (see Sec. IV) can be readily applied without modifications.

C. State and Covariance Update

Once all the relative measurements $\mathbf{z}_k^{i,j}$, $1 \leq i \neq j \leq N$, become available, the state estimate and its covariance can be updated as

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{z}}_{k|k-1}, \quad (6)$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^{\mathrm{T}}, \quad (7)$$

where $\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}}\mathbf{S}_k^{-1}$ is the Kalman gain, $\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{C}_{\mathbf{n}}$ is the measurement residual covariance.

Unfortunately, the computational complexity of (6)-(7) is $\mathcal{O}(N^6)$, due to the inversion of a dense matrix $\mathbf{S}_k$ whose dimensions are $2(N-1)N \times 2(N-1)N$. Therefore, the real-time implementation of (6)-(7) becomes prohibitive as the number of robots, N, increases.

It is possible to avoid the inversion of $\mathbf{S}_k$ by processing the $(N-1)N$ exteroceptive measurements *sequentially*. In this case, and since the computational cost for processing every relative measurement $\mathbf{z}_k^{i,j}$ is $\mathcal{O}(N^2)$ [5], the total computational cost per time step becomes $\mathcal{O}(N^4)$. Note, however, that due to the nonlinearity of the measurement model [see (2)], the robot pose estimates obtained by sequential updates are less accurate than these computed by concurrent updates [see (6)-(7)].

Another alternative approach for updating the state and covariance is to employ the Information filter [20, Ch. 5, Sec. 5.6], i.e., for $\mathbf{C}_{\mathbf{n}} = \mathbf{I}_{2(N-1)N}$,

$$\mathbf{P}_{k|k} = \left(\mathbf{P}_{k|k-1}^{-1} + \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k\right)^{-1}, \quad (8)$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{k|k}\mathbf{H}_k^{\mathrm{T}}\tilde{\mathbf{z}}_{k|k-1}. \quad (9)$$

It is worth noting that computing $\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k$ and $\mathbf{P}_{k|k}\mathbf{H}_k^{\mathrm{T}}\tilde{\mathbf{z}}_{k|k-1}$ has computational cost only $\mathcal{O}(N^2)$ [7], due to the sparse structure of $\mathbf{H}_k$ [see (5)]. Hence, the most computationally demanding operation of (8)-(9) involves the inversions of two matrices (i.e., $\mathbf{P}_{k|k-1}$ and $\mathbf{P}_{k|k-1}^{-1} + \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k$), whose dimensions are linear in N. Thus, the total computational complexity of the state and covariance updates using (8)-(9) is $\mathcal{O}(N^3)$, a significant complexity reduction as compared to the standard EKF updates (6)-(7).

Unfortunately, (8) often suffers from numerical instability, due to $\kappa(\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k) = \kappa^2(\mathbf{H}_k)$ [where $\kappa(\mathbf{H})$ is the condition number of $\mathbf{H}$], which renders the algorithm (8)-(9) numerically less robust [21], [22].

D. State and Covariance Update through QR Factorization

An effective strategy to overcome the numerical instability of the Information filter [see (8)-(9)] is to apply *thin* QR factorization [22] on $\mathbf{H}_k$ [21], i.e.,

$$\mathbf{H}_k = \mathbf{Q}_k \mathbf{R}_k, \tag{10}$$

where $\mathbf{Q}_k$ is a $2(N-1)N \times 3N$ matrix with orthonormal columns (i.e., $\mathbf{Q}_k^{\mathrm{T}}\mathbf{Q}_k = \mathbf{I}_{3N}$), and $\mathbf{R}_k$ is upper triangular.[4]

Substituting (10) into (8), we obtain the EKF update equations based on QR factorization, i.e.,

$$\begin{aligned}\mathbf{P}_{k|k} &= \left(\mathbf{P}_{k|k-1}^{-1} + \mathbf{R}_k^{\mathrm{T}}\mathbf{R}_k\right)^{-1}\\ &= \mathbf{P}_{k|k-1} - \mathbf{P}_{k|k-1}\mathbf{R}_k^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}\mathbf{R}_k\mathbf{P}_{k|k-1}, \qquad (11)\\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{k|k}\mathbf{H}_k^{\mathrm{T}}\tilde{\mathbf{z}}_{k|k-1}, \qquad (12)\end{aligned}$$

where $\boldsymbol{\Sigma}_k = \mathbf{R}_k\mathbf{P}_{k|k-1}\mathbf{R}_k^{\mathrm{T}} + \mathbf{I}_{3N}$, and the second equality in (11) is established using the matrix inversion lemma [22].

Note that in contrast to $\mathbf{S}_k$ [see (7)], whose dimensions are $2(N-1)N \times 2(N-1)N$, the matrix $\boldsymbol{\Sigma}_k$ in (11) has dimensions only $3N \times 3N$. Thus, assuming that $\mathbf{R}_k$ is given, the total computational cost of the state and covariance updates using (11)-(12) is $\mathcal{O}(N^3)$, the same order of complexity as when using (8)-(9). Most importantly, $\kappa(\mathbf{R}_k) = \kappa(\mathbf{H}_k)$ [22]. Hence the numerical stability of (11)-(12) is significantly better as compared to (8)-(9).

Therefore, in order to ensure that the computational complexity for the state and covariance updates through QR factorization [see (11)-(12)] remains $\mathcal{O}(N^3)$, we conclude that the maximum number of operations to implement (10) should also be within $\mathcal{O}(N^3)$.

There exist several methods for performing QR factorization (or QR decomposition), such as the Cholesky decomposition (CHO), the modified Gram–Schmidt process (MGS), the Givens rotations (GIV), and the Householder transformations (or Householder reflections) [22]. Due to its simplicity and numerical stability, we adopt the QR factorization algorithm utilizing Householder transformations, which is termed as the Standard Householder QR in this paper.[5] In what follows, we present an overview of the main steps of the Standard Householder QR algorithm. The purpose of this is to show that when applying the Standard Householder QR factorization to the matrix $\mathbf{H}_k$ appearing in CL, the computational complexity becomes $\mathcal{O}(N^4)$. For clarity, the time-step index k is dropped from (10) throughout the rest of the paper, with $m = 2(N-1)N$ and $n = 3N$ denoting the number of rows and columns of $\mathbf{H}_k$, respectively.

E. Overview of the Standard Householder QR Algorithm

We first introduce an orthogonal and symmetric Householder (reflection) matrix $\mathbf{Q} = \mathbf{I} - \beta\mathbf{v}\mathbf{v}^{\mathrm{T}}$, where $\beta = \frac{2}{\|\mathbf{v}\|_2^2}$, and the nonzero vector $\mathbf{v}$ is called a Householder vector. The Standard Householder QR algorithm [22, Algorithm 5.2.1] applies a sequence of Householder matrices (multiplies $\mathbf{H}$ from the left with $\mathbf{Q}$) to gradually transform $\mathbf{H}$ into an upper triangular form $\mathbf{R}$. Specifically, suppose that after $(\ell-1)$ Householder matrices $\{\mathbf{Q}_i = \mathbf{I}_m - \beta_i\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}},\ i = 1, \ldots, \ell-1\}$ have left-multiplied the original $\mathbf{H}$, the resulting matrix $\mathbf{H}^{(\ell-1)}$ takes the following block form:

$$\mathbf{H}^{(\ell-1)} = \Big(\prod_{i=1}^{\ell-1}(\mathbf{I}_m - \beta_i\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}})\Big)\mathbf{H} = \begin{bmatrix}\mathbf{H}_{1,1}^{(\ell-1)} & \mathbf{H}_{1,2}^{(\ell-1)}\\ \mathbf{0} & \mathbf{H}_{2,2}^{(\ell-1)}\end{bmatrix}, \tag{13}$$

where $\mathbf{H}_{1,1}^{(\ell-1)}$ is an *upper triangular* matrix of dimensions $(\ell-1)\times(\ell-1)$.

At the ℓth iteration, we seek a Householder matrix $\mathbf{Q}_\ell = \mathbf{I}_m - \beta_\ell\mathbf{v}_\ell\mathbf{v}_\ell^{\mathrm{T}}$ such that the *first ℓ columns* of $\mathbf{H}^{(\ell)} = (\mathbf{I}_m - \beta_\ell\mathbf{v}_\ell\mathbf{v}_\ell^{\mathrm{T}})\mathbf{H}^{(\ell-1)}$ become *upper triangular*. This can be achieved [22] by selecting $\mathbf{v}_\ell = [\mathbf{0}_{1\times(\ell-1)}\ \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$, with

$$\mathbf{v} = \mathbf{u} + \mathrm{sign}(u_1)\|\mathbf{u}\|_2\mathbf{e}_1, \tag{14}$$

where the vector $\mathbf{u}$ is *the first column* of $\mathbf{H}_{2,2}^{(\ell-1)}$, and u_1 is the first element of $\mathbf{u}$. Furthermore, $\beta_\ell = \frac{1}{\|\mathbf{u}\|_2^2 + |u_1|\|\mathbf{u}\|_2}$.

For clarity, we decompose the matrices $\mathbf{H}_{1,2}^{(\ell-1)}$ and $\mathbf{H}_{2,2}^{(\ell-1)}$ as follows: $\mathbf{H}_{1,2}^{(\ell-1)} = [\mathbf{t}\ \ \overline{\mathbf{H}}_{1,2}^{(\ell-1)}]$, where the vector $\mathbf{t}$ represents the first column of $\mathbf{H}_{1,2}^{(\ell-1)}$ and the matrix $\overline{\mathbf{H}}_{1,2}^{(\ell-1)}$ consists of the remaining columns; similarly $\mathbf{H}_{2,2}^{(\ell-1)} = \begin{bmatrix}\mathbf{u} & \overline{\mathbf{H}}_{2,2}^{(\ell-1)}\end{bmatrix} = \begin{bmatrix}u_1 & \bar{\mathbf{s}}^{\mathrm{T}}\\ \mathbf{u}_{-1} & \underline{\mathbf{H}}_{2,2}^{(\ell-1)}\end{bmatrix}$, where $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$ consists of all but the first column of $\mathbf{H}_{2,2}^{(\ell-1)}$, $\mathbf{u}_{-1}$ denotes the vector obtained by removing the first element of $\mathbf{u}$, the *row* vector $\bar{\mathbf{s}}^{\mathrm{T}}$ corresponds to the first row of $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$, and the matrix $\underline{\mathbf{H}}_{2,2}^{(\ell-1)}$ is obtained by removing $\bar{\mathbf{s}}^{\mathrm{T}}$ from $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$. Following this notation, it can be shown that $\mathbf{H}^{(\ell)}$ has the following block structure [7]

$$\mathbf{H}^{(\ell)} = \mathbf{Q}_\ell\mathbf{H}^{(\ell-1)} = \begin{bmatrix}\mathbf{H}_{1,1}^{(\ell)} & \mathbf{H}_{1,2}^{(\ell)}\\ \mathbf{0} & \mathbf{H}_{2,2}^{(\ell)}\end{bmatrix}, \tag{15}$$

where $\mathbf{H}_{1,1}^{(\ell)} = \begin{bmatrix}\mathbf{H}_{1,1}^{(\ell-1)} & \mathbf{t}\\ \mathbf{0} & \alpha\end{bmatrix}$ is an *upper triangular* matrix with $\alpha = -\mathrm{sign}(u_1)\|\mathbf{u}\|_2$; $\mathbf{H}_{1,2}^{(\ell)} = \Big[\big(\overline{\mathbf{H}}_{1,2}^{(\ell-1)}\big)^{\mathrm{T}}\ \ \mathbf{s}\Big]^{\mathrm{T}}$, where

$$\mathbf{s} = \bar{\mathbf{s}} - v_1\boldsymbol{\delta}, \tag{16}$$

$$\mathbf{H}_{2,2}^{(\ell)} = \underline{\mathbf{H}}_{2,2}^{(\ell-1)} - \mathbf{u}_{-1}\boldsymbol{\delta}^{\mathrm{T}}, \tag{17}$$

and $\mathbf{v} = [v_1\ \mathbf{v}_{-1}^{\mathrm{T}}]^{\mathrm{T}}$, $\boldsymbol{\delta} = \beta_\ell\big(\overline{\mathbf{H}}_{2,2}^{(\ell-1)}\big)^{\mathrm{T}}\mathbf{v}$.

[4]In the implementation of QR decomposition, it is necessary to invoke column pivoting (or permutation) techniques since $\mathbf{H}_k$ is not full-column rank (see Remark 1). Specifically, (10) is modified to $\mathbf{H}_k\boldsymbol{\Pi}_k = \mathbf{Q}_k\mathbf{R}_k$, where $\boldsymbol{\Pi}_k$ is a column permutation matrix, and the absolute value of the diagonal elements of $\mathbf{R}_k$ are arranged in decreasing order. In practice, column pivoting can be efficiently achieved using pointers. Furthermore, it can be shown that the computational overhead associated with column pivoting is $\mathcal{O}(N^2)$ [7], thanks to the effective updating of the column norms discovered by Businger and Golub [23]. Since the most dominant computational cost is $\mathcal{O}(N^3)$ (see Sec. IV-B), without loss of generality, in the following analysis we assume $\boldsymbol{\Pi}_k = \mathbf{I}_{3N}$. The simulation results shown in Sec. V are based on the Modified Householder QR algorithm with column pivoting techniques, whose implementation is described in [7].

[5]We have also conducted computational analysis when employing CHO, MGS, and GIV. It can be shown [7] that the complexity of CHO is $\mathcal{O}(N^3)$, due to the sparse structure of $\mathbf{H}_k$. However, CHO is not applicable since it requires $\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k$ to be positive definite, or equivalently, $\mathbf{H}_k$ to be full column rank [22]. On the other hand, both MGS and GIV require $\mathcal{O}(N^4)$ arithmetic operations [7].

Remark 2: A key observation of (14) is that the vectors $\mathbf{v}$ and $\mathbf{u}$ differ only by the first element, i.e., $\mathbf{v}_{-1} = \mathbf{u}_{-1}$.

The above process terminates at $\ell = n$, with outputs $\mathbf{Q}$ corresponding to the first n columns of the matrix product $\mathbf{Q}_1 \cdots \mathbf{Q}_n$ and $\mathbf{R}$ selected as the first n rows of $\mathbf{H}^{(n)}$.

We have conducted a detailed computational complexity analysis of applying the Standard Householder QR to factorize $\mathbf{H}$ [see (10)]. Unfortunately, it turns out that the Standard Householder QR requires $\mathcal{O}(N^4)$ arithmetic operations, due to the fact that the original sparse structure of $\mathbf{H}$ is destroyed and $\mathbf{nnz}(\mathbf{H}_{2,2}^{(\ell)})$ increases quadratically at every iteration ℓ [7]. Due to space limitations, the detailed computational complexity analysis of the Standard Householder QR algorithm is provided in [7]. This motivates us to develop the Modified Householder QR algorithm, which is based on the Standard Householder QR algorithm but explicitly exploits the sparse structure of $\mathbf{H}$, to achieve QR decomposition in $\mathcal{O}(N^3)$. In the next section, we describe the main idea behind the Modified Householder QR algorithm, as well as its complexity analysis.

IV. Modified Householder QR Algorithm

The Modified Householder QR algorithm is derived from [24], where Kaufman proposed an idea that exploits the sparsity of the original matrix. However, in [24], Kaufman assumes that the Householder reflection matrices (or equivalently, the Householder vectors) are known in advance, which is not the case in our scenario. We make several modifications to the original algorithm proposed in [24], and term this new algorithm as the Modified Householder QR in Sec. IV-A. We further analyze its complexity when applied to $\mathbf{H}$ [see (5)] and show in Sec. IV-B that the computational cost is reduced from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^3)$.

A. *Description of the Modified Householder* QR *Algorithm*

To facilitate the description and derivation of the Modified Householder QR algorithm, we adopt the same notation used in Sec. III-E. Furthermore, we use $\mathbf{h}_i$ and $\mathbf{h}_i^{(\ell)}$, $i = 1, \ldots, n$, to denote the ith columns of the original matrix $\mathbf{H}$ and the updated matrix $\mathbf{H}^{(\ell)}$ after the ℓth Householder reflection is processed, respectively. From (13), we have

$$\mathbf{h}_j^{(\ell-1)} = \Big(\prod_{i=1}^{\ell-1}(\mathbf{I}_m - \beta_i \mathbf{v}_i \mathbf{v}_i^{\mathrm{T}})\Big)\mathbf{h}_j,\ j = \ell, \ldots, n. \tag{18}$$

Hence $\mathbf{h}_j^{(\ell-1)}$ $(j = \ell, \ldots, n)$ is a linear combination of $\mathbf{h}_j$ and the Householder vectors $\{\mathbf{v}_i, i = 1, \ldots, \ell-1\}$. Additionally, a crucial property of the Householder transformation (see Remark 2) is $(\mathbf{v}_i)_{-i} = (\mathbf{h}_i^{(i-1)})_{-i}, i = 1, \ldots, \ell-1$, where $\mathbf{v}_{-i}$ denotes the vector obtained by removing the first i components of $\mathbf{v}$. Accordingly, $(\mathbf{v}_i)_{-(\ell-1)} = (\mathbf{h}_i^{(i-1)})_{-(\ell-1)}$ for $1 \le i \le \ell - 1$. Hence, $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}$ $(j = \ell, \ldots, n)$ can be expressed as a linear combination of $(\mathbf{h}_j)_{-(\ell-1)}$ and $\{(\mathbf{h}_i^{(i-1)})_{-(\ell-1)}, i = 1, \ldots, \ell-1\}$, i.e.,

$$(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)} = (\mathbf{h}_j)_{-(\ell-1)} + \sum_{i=1}^{\ell-1}(\mathbf{h}_i^{(i-1)})_{-(\ell-1)}\ \chi_{i,j}^{(\ell-1)},$$

for some coefficients $\chi_{i,j}^{(\ell-1)}$. Furthermore, notice that every vector $\mathbf{h}_i^{(i-1)}, i = 2, \ldots, \ell-1$, itself is a linear combination of $\mathbf{h}_i$ and the Householder vectors $\{\mathbf{v}_\eta, \eta = 1, \ldots, i-1\}$ [see (18)], and recall that $(\mathbf{v}_\eta)_{-(\ell-1)} = (\mathbf{h}_\eta^{(\eta-1)})_{-(\ell-1)}$ for $\eta = 1, \ldots, i-1$. Thus, $(\mathbf{h}_i^{(i-1)})_{-(\ell-1)}$ $(i = 2, \ldots, \ell-1)$ can be expressed as a linear combination of $(\mathbf{h}_i)_{-(\ell-1)}$ and $\{(\mathbf{h}_\eta^{(\eta-1)})_{-(\ell-1)}, \eta = 1, \ldots, i-1\}$. In addition, $\mathbf{h}_1^{(0)} = \mathbf{h}_1$ by definition, and hence, $(\mathbf{h}_1^{(0)})_{-(\ell-1)} = (\mathbf{h}_1)_{-(\ell-1)}$. Therefore, we conclude by *recursion* that each vector $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}, j = \ell, \ldots, n$, can be expressed as a *linear combination* of $(\mathbf{h}_j)_{-(\ell-1)}$ and $\{(\mathbf{h}_i)_{-(\ell-1)}, i = 1, \ldots, \ell-1\}$, i.e.,

$$(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)} = (\mathbf{h}_j)_{-(\ell-1)} + \sum_{i=1}^{\ell-1}(\mathbf{h}_i)_{-(\ell-1)}\ \gamma_{i,j}^{(\ell-1)}, \tag{19}$$

where the coefficients $\gamma_{i,j}^{(\ell-1)},\ i = 1, \ldots, \ell-1,\ j = \ell, \ldots, n$, need to be determined at each iteration. In what follows, we will provide a formula [see (30)] that updates $\gamma_{i,j}^{(\ell-1)}$ recursively.

Notice that the matrix $\mathbf{H}_{2,2}^{(\ell-1)}$ [see (13)] comprises all the column vectors $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)},\ j = \ell, \ldots, n$. Hence, (19) can be summarized into a compact matrix form

$$\mathbf{H}_{2,2}^{(\ell-1)} = \mathbf{H}_{-(\ell-1)} \begin{bmatrix} \mathbf{\Gamma}^{(\ell-1)} \\ \mathbf{I}_{n-\ell+1} \end{bmatrix}, \tag{20}$$

where $\mathbf{H}_{-(\ell-1)} = [(\mathbf{h}_1)_{-(\ell-1)} \ \ldots \ (\mathbf{h}_n)_{-(\ell-1)}]$ is the sub-matrix of $\mathbf{H}$ resulting by removing its first $(\ell-1)$ rows. The $(\ell-1) \times (n-\ell+1)$ matrix $\mathbf{\Gamma}^{(\ell-1)} = [\gamma_{i,j}^{(\ell-1)}]$ is termed the coefficient matrix. In order to facilitate the presentation of the ensuing derivations, $\mathbf{\Gamma}^{(\ell-1)}$ is written as $[\boldsymbol{\gamma}_\ell^{(\ell-1)} \ \ \overline{\mathbf{\Gamma}}^{(\ell-1)}]$, where $\boldsymbol{\gamma}_\ell^{(\ell-1)}$ is the first column of $\mathbf{\Gamma}^{(\ell-1)}$.

As shown in [7], in contrast to the original vector $(\mathbf{h}_j)_{-(\ell-1)},\ j = \ell, \ldots, n$, which has at most $\mathcal{O}(N)$ non-zero elements [see Remark 1], the vector $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}$, obtained after the $(\ell-1)$th Householder reflection is processed, has $\mathbf{nnz}((\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}) \sim \mathcal{O}(\ell N)$, which results in the computational cost of $\mathcal{O}(N^4)$ when employing the Standard Householder QR algorithm. In order to preserve the original sparsity of $\mathbf{H}$, and at the same time reduce the memory usage, our modification to the Standard Householder QR algorithm is that the *explicit form* of $\mathbf{H}_{2,2}^{(\ell-1)}$ in (20) [or equivalently, the explicit form of the vectors $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)},\ j = \ell, \ldots, n$, in (19)] is *not calculated*. Instead, $\mathbf{H}_{2,2}^{(\ell-1)}$ is *stored and represented implicitly* by the coefficient matrix $\mathbf{\Gamma}^{(\ell-1)}$ [see (20)].

In what follows, we address two key issues in the Modified Householder QR algorithm. Firstly, computing the matrices $\mathbf{H}_{1,1}^{(\ell)}$, $\mathbf{H}_{1,2}^{(\ell)}$ in (15); Secondly, deriving the recursive rule for obtaining $\mathbf{\Gamma}^{(\ell)}$ from $\mathbf{\Gamma}^{(\ell-1)}$. Or equivalently, we seek the expression of $\mathbf{\Gamma}^{(\ell)}$ such that

$$\mathbf{H}_{2,2}^{(\ell)} = \mathbf{H}_{-\ell} \begin{bmatrix} \mathbf{\Gamma}^{(\ell)} \\ \mathbf{I}_{n-\ell} \end{bmatrix}, \tag{21}$$

where $\mathbf{H}_{-\ell} = [(\mathbf{h}_1)_{-\ell} \ \ldots \ (\mathbf{h}_n)_{-\ell}]$ is the sub-matrix of $\mathbf{H}$

resulting after removing its first ℓ rows.

To proceed, we first note that the vector $\mathbf{u} = (\mathbf{h}_{\ell}^{(\ell-1)})_{-(\ell-1)}$, i.e., the first column of $\mathbf{H}_{2,2}^{(\ell-1)}$, plays an important role in generating the Householder vector and the subsequent process. Hence, we *explicitly* compute $\mathbf{u}$ using (20), i.e.,

$$\mathbf{u} = (\mathbf{h}_{\ell}^{(\ell-1)})_{-(\ell-1)} = (\mathbf{h}_{\ell})_{-(\ell-1)} + \mathbf{H}^{1}_{-(\ell-1)}\, \boldsymbol{\gamma}_{\ell}^{(\ell-1)}\ , \quad (22)$$

where $\mathbf{H}^{1}_{-(\ell-1)} = [(\mathbf{h}_1)_{-(\ell-1)}\ \dots\ (\mathbf{h}_{\ell-1})_{-(\ell-1)}]$ consists of the first $(\ell-1)$ columns of $\mathbf{H}_{-(\ell-1)}$. Note that we *explicitly* compute only the vector $\mathbf{u} = (\mathbf{h}_{\ell}^{(\ell-1)})_{-(\ell-1)}$, the first column of $\mathbf{H}_{2,2}^{(\ell-1)}$, while the remaining columns $(\mathbf{h}_{j}^{(\ell-1)})_{-(\ell-1)}$, $j = \ell+1, \dots, n$, are *not explicitly* computed. Instead, they are represented *implicitly* by $\overline{\mathbf{\Gamma}}^{(\ell-1)}$.

Once the *explicit* form of $\mathbf{u}$, computed by (22), is known, $\|\mathbf{u}\|_2$ can be calculated as well. Thus, the scalar β_ℓ and the Householder vector $\mathbf{v}_\ell$ (or equivalently $\mathbf{v}$) are readily available. Notice that computing $\mathbf{v}$ from $\mathbf{u}$ only requires updating the first element of $\mathbf{u}$ [see (14)].

Now we are ready to present the recursive formulas for computing $\mathbf{H}_{1,1}^{(\ell)}$, $\mathbf{H}_{1,2}^{(\ell)}$, and $\mathbf{\Gamma}^{(\ell)}$.

1) Computing $\mathbf{H}_{1,1}^{(\ell)}$*:* Notice that the terms $\mathbf{H}_{1,1}^{(\ell-1)}$ as well as $\mathbf{t}$ are already available after the $(\ell-1)$th Householder transformation. Hence the only unknown in $\mathbf{H}_{1,1}^{(\ell)}$ [see (15)] is the scalar α, which can be calculated from $\|\mathbf{u}\|_2$ in a fixed number of arithmetic operations.

2) Computing $\mathbf{H}_{1,2}^{(\ell)}$*:* Since $\overline{\mathbf{H}}_{1,2}^{(\ell-1)}$ is known after the $(\ell-1)$th Householder transformation, updating $\mathbf{H}_{1,2}^{(\ell)}$ is equivalent to calculating the vector $\mathbf{s}$ from (16), which requires $\bar{\mathbf{s}}$ and $\boldsymbol{\delta} = \beta_\ell \big(\overline{\mathbf{H}}_{2,2}^{(\ell-1)}\big)^{\mathrm{T}}\mathbf{v}$. Remember that we do not have the *explicit* forms of $\bar{\mathbf{s}}$ and $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$. However, using the fact that $\bar{\mathbf{s}}^{\mathrm{T}}$ is the first row of $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$ and based on (20), we can rewrite $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$ and $\bar{\mathbf{s}}$ as follows

$$\overline{\mathbf{H}}_{2,2}^{(\ell-1)} = \mathbf{H}^{2}_{-(\ell-1)} + \mathbf{H}^{1}_{-(\ell-1)}\ \overline{\mathbf{\Gamma}}^{(\ell-1)}, \quad (23)$$

$$\bar{\mathbf{s}} = \boldsymbol{\rho}_2 + \big(\overline{\mathbf{\Gamma}}^{(\ell-1)}\big)^{\mathrm{T}}\boldsymbol{\rho}_1, \quad (24)$$

where $\mathbf{H}^{2}_{-(\ell-1)} = [(\mathbf{h}_{\ell+1})_{-(\ell-1)}\ \dots\ (\mathbf{h}_{n})_{-(\ell-1)}]$ comprises the last $(n-\ell)$ columns of $\mathbf{H}_{-(\ell-1)}$, and $\boldsymbol{\rho}_1^{\mathrm{T}}$ and $\boldsymbol{\rho}_2^{\mathrm{T}}$ are the first rows of $\mathbf{H}^{1}_{-(\ell-1)}$ and $\mathbf{H}^{2}_{-(\ell-1)}$, respectively. From (23), we compute the vector

$$\boldsymbol{\delta} = \beta_\ell \big(\overline{\mathbf{H}}_{2,2}^{(\ell-1)}\big)^{\mathrm{T}}\mathbf{v} = \boldsymbol{\delta}_2 + \big(\overline{\mathbf{\Gamma}}^{(\ell-1)}\big)^{\mathrm{T}}\boldsymbol{\delta}_1, \quad (25)$$

where $\boldsymbol{\delta}_1 = \beta_\ell\big(\mathbf{H}^{1}_{-(\ell-1)}\big)^{\mathrm{T}}\mathbf{v}$ and $\boldsymbol{\delta}_2 = \beta_\ell\big(\mathbf{H}^{2}_{-(\ell-1)}\big)^{\mathrm{T}}\mathbf{v}$.

Substituting (24) and (25) in (16), we arrive at the following update equation for $\mathbf{s}$ (or equivalently, $\mathbf{H}_{1,2}^{(\ell)}$), i.e.,

$$\mathbf{s} = \big[\boldsymbol{\rho}_2 - v_1\boldsymbol{\delta}_2\big] + \big(\overline{\mathbf{\Gamma}}^{(\ell-1)}\big)^{\mathrm{T}}\big[\boldsymbol{\rho}_1 - v_1\boldsymbol{\delta}_1\big]. \quad (26)$$

3) Computing $\mathbf{\Gamma}^{(\ell)}$*:* To determine $\mathbf{\Gamma}^{(\ell)}$, we begin with (21) and (17). Recall that we do not have the *explicit* expression of $\underline{\mathbf{H}}_{2,2}^{(\ell-1)}$. However, since $\underline{\mathbf{H}}_{2,2}^{(\ell-1)}$ corresponds to $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$ with the first row removed, we obtain [from (23)],

$$\underline{\mathbf{H}}_{2,2}^{(\ell-1)} = \mathbf{H}^{2}_{-\ell} + \mathbf{H}^{1}_{-\ell}\ \overline{\mathbf{\Gamma}}^{(\ell-1)}, \quad (27)$$

where $\mathbf{H}^{1}_{-\ell} = [(\mathbf{h}_1)_{-\ell}\ \dots\ (\mathbf{h}_{\ell-1})_{-\ell}]$ and $\mathbf{H}^{2}_{-\ell} = [(\mathbf{h}_{\ell+1})_{-\ell}\ \dots\ (\mathbf{h}_{n})_{-\ell}]$.

Next we explore the property $\mathbf{v}_{-1} = \mathbf{u}_{-1}$ [see Remark 2]. In particular, based on (22), we have

$$\mathbf{v}_{-1} = \mathbf{u}_{-1} = (\mathbf{h}_\ell)_{-\ell} + \mathbf{H}^{1}_{-\ell}\boldsymbol{\gamma}_{\ell}^{(\ell-1)}. \quad (28)$$

Finally, we substitute (27), (28), and (25) into (17), and notice that $\mathbf{H}_{-\ell} = [\mathbf{H}^{1}_{-\ell}\ \ (\mathbf{h}_\ell)_{-\ell}\ \ \mathbf{H}^{2}_{-\ell}]$, to arrive at

$$\begin{aligned}\mathbf{H}_{2,2}^{(\ell)} &= \mathbf{H}^{1}_{-\ell}\big(\overline{\mathbf{\Gamma}}^{(\ell-1)} - \boldsymbol{\gamma}_{\ell}^{(\ell-1)}\boldsymbol{\delta}^{\mathrm{T}}\big) - (\mathbf{h}_\ell)_{-\ell}\boldsymbol{\delta}^{\mathrm{T}} + \mathbf{H}^{2}_{-\ell} \\ &= \mathbf{H}_{-\ell}\begin{bmatrix}\overline{\mathbf{\Gamma}}^{(\ell-1)} - \boldsymbol{\gamma}_{\ell}^{(\ell-1)}\boldsymbol{\delta}^{\mathrm{T}} \\ -\boldsymbol{\delta}^{\mathrm{T}} \\ \mathbf{I}_{n-\ell}\end{bmatrix}. \quad (29)\end{aligned}$$

Comparing (29) and (21), we immediately obtain

$$\mathbf{\Gamma}^{(\ell)} = \begin{bmatrix}\overline{\mathbf{\Gamma}}^{(\ell-1)} - \boldsymbol{\gamma}_{\ell}^{(\ell-1)}\boldsymbol{\delta}^{\mathrm{T}} \\ -\boldsymbol{\delta}^{\mathrm{T}}\end{bmatrix}. \quad (30)$$

Note that the upper part of $\mathbf{\Gamma}^{(\ell)}$ is a *rank-one modification* of the *existing matrix* $\overline{\mathbf{\Gamma}}^{(\ell-1)}$, which has a relatively low computational cost. Furthermore, (30) *affirms* that every vector $(\mathbf{h}_{j}^{(\ell)})_{-\ell}, j = \ell+1, \dots, n$, is a linear combination of $(\mathbf{h}_j)_{-\ell}$ and $\{(\mathbf{h}_i)_{-\ell}, i = 1, \dots, \ell\}$.

In summary, we outline the algorithmic flow chart of the Modified Householder QR in Algorithm 1.

Algorithm 1 Modified Householder QR

```
Require: $\mathbf{H}$
Ensure: $\mathbf{R}$ [see (10)].
1: for $\ell = 1, \dots, n,$ do
2:     Compute $\mathbf{u}$ from (22) and calculate $\|\mathbf{u}\|_2$.
3:     Calculate $\mathbf{v}, \beta_\ell, \alpha$ [see Sec. III-E], and update $\mathbf{H}_{1,1}^{(\ell)}$.
4:     Determine $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2$, and $\boldsymbol{\delta}$ in (25).
5:     Calculate $\mathbf{s}$ from (26), and update $\mathbf{H}_{1,2}^{(\ell)}$.
6:     Update $\mathbf{\Gamma}^{(\ell)}$ based on (30).
7: end for
8: Return $\mathbf{R} = \mathbf{H}_{1,1}^{(n)}$.
```

B. Computational Complexity Analysis

In this section, we briefly analyze the computational cost of applying the Modified Householder QR algorithm to factorize $\mathbf{H}$ [see (10)]. We claim that its complexity is $\mathcal{O}(N^3)$. To prove it, we will identity the number of flops for *every* line inside the "for loop" in Algorithm 1, and show that the number of arithmetic operations required per iteration is bounded above by $\mathcal{O}(N^2)$. Since the maximum number of iterations is $3N$, the overall computational cost is $\mathcal{O}(N^3)$.

- Computational cost of Line 2:
  Recall that $\mathbf{nnz}(\mathbf{h}_j) \sim \mathcal{O}(N),\ j = 1, \dots, 3N$ [see Remark 1], hence $\mathbf{nnz}((\mathbf{h}_i)_{-(\ell-1)}) \sim \mathcal{O}(N), i = 1, \dots, \ell$. Therefore, computing $\mathbf{u}$ from (22) requires $\mathcal{O}(\ell N)$ operations. Additionally, $\mathbf{nnz}(\mathbf{u}) \sim \mathcal{O}(\ell N)$, thus calculating $\|\mathbf{u}\|_2$ is of $\mathcal{O}(\ell N)$ operations. Hence, the cost of performing Line 2 is of $\mathcal{O}(\ell N)$, where $\ell \leq 3N$.

- Computational cost of Line 3:
  Once $\mathbf{u}$ and $\|\mathbf{u}\|_2$ become available, determining the scalars α and β_ℓ requires constant number of arithmetic operations. Furthermore, updating $\mathbf{v}$ from $\mathbf{u}$ only requires the modification of the first element of $\mathbf{u}$, which requires $\mathcal{O}(1)$ running time. In summary, the cost of performing Line 3 is of $\mathcal{O}(1)$.
- Computational cost of Line 4:
  Since each column of $\mathbf{H}^1_{-(\ell-1)}$ and $\mathbf{H}^2_{-(\ell-1)}$ has $\mathcal{O}(N)$ non-zeros, which is attained by preserving the original sparse structure of $\mathbf{H}$, computing $\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$ has a cost of $\mathcal{O}(N^2)$, regardless of the structure of $\mathbf{v}$. Additionally, calculating $\boldsymbol{\delta}$ from $\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$ has a cost of $\mathcal{O}(\ell N)$, since (25) involves an $(3N-\ell)\times(\ell-1)$ matrix multiplied by an $(\ell-1)\times 1$ vector and a vector-vector addition of dimension $3N-\ell$. In summary, the cost of performing Line 4 is of $\mathcal{O}(N^2)$.
- Computational cost of Line 5:
  Since (26) involves a $(\ell-1)\times 1$ vector multiplying with an $(3N-\ell)\times(\ell-1)$ matrix and a vector-vector addition of dimension $3N-\ell$, the overall cost of performing Line 5 is of $\mathcal{O}(\ell N)$.
- Computational cost of Line 6:
  In (30), the vector $\boldsymbol{\delta}$ [see (25)] is available from Line 4 and *does not need to be recomputed*. Hence, we only need to focus on the upper part of (30), which is a *rank-one update* of the *existing matrix* $\overline{\boldsymbol{\Gamma}}^{(\ell-1)}$. Since the vectors $\boldsymbol{\gamma}_\ell^{(\ell-1)}$ and $\boldsymbol{\delta}$ are of dimensions $\ell-1$ and $3N-\ell$, respectively, we conclude that the overall cost of performing Line 6 is of $\mathcal{O}(\ell N)$.

In summary, the most demanding computational cost of Algorithm 1, from Line 2 to Line 6, is of $\mathcal{O}(N^2)$. Therefore, we have shown that the number of arithmetic operations required per iteration is bounded above by $\mathcal{O}(N^2)$, and the worst-case computational complexity of applying the Modified Householder QR algorithm on $\mathbf{H}$ is $\mathcal{O}(N^3)$.

Furthermore, we would like to point out that the Modified Householder QR algorithm is applicable to any sparse matrix $\mathbf{H}$ of sparsity pattern other than that of the measurement Jacobian matrix [see (5)]. In particular, suppose that the dimensions of $\mathbf{H}$ are $m \times n$, and denote $\tau = \max(\mathbf{nnz}(\mathbf{h}_1),\ldots,\mathbf{nnz}(\mathbf{h}_n))$. We have shown that the computational complexity of the Modified Householder QR algorithm is of $\mathcal{O}(\tau n^2)$ [7], in contrast to $\mathcal{O}(mn^2)$ of the Standard Householder QR algorithm [22].

V. Simulation Results

In the previous section, we have shown that the worst-case computational complexity of the Modified Householder QR algorithm on the sparse matrix $\mathbf{H}$ is $\mathcal{O}(N^3)$. In order to corroborate our theoretical analysis, we have evaluated the running time required by the Modified Householder QR algorithm for a team of N robots performing CL. Specifically, we randomly generate the poses of the robots and assume that each robot is able to detect, identify, and measure both relative distance and bearing to the remaining $N-1$ robots. Hence, $\mathbf{H}$ has dimensions $2(N-1)N \times 3N$.

TABLE I
CPU RUNTIME (SEC)

N	SS-QR	MH-QR
6	7.5378×10^{-5}	4.9683×10^{-5}
21	1.3985×10^{-3}	1.4737×10^{-3}
26	2.8506×10^{-3}	2.8315×10^{-3}
51	2.8017×10^{-2}	2.2363×10^{-2}
101	3.2935×10^{-1}	1.8164×10^{-1}
201	4.3048×10^{0}	1.5436×10^{0}
301	1.9804×10^{1}	6.7462×10^{0}
401	5.7972×10^{1}	1.8523×10^{1}
501	1.5869×10^{3}	3.8437×10^{1}
551	N/A	5.3501×10^{1}
601	N/A	7.0800×10^{1}
701	N/A	1.1808×10^{2}
801	N/A	1.8626×10^{2}
901	N/A	2.6653×10^{2}
1001	N/A	3.7570×10^{2}

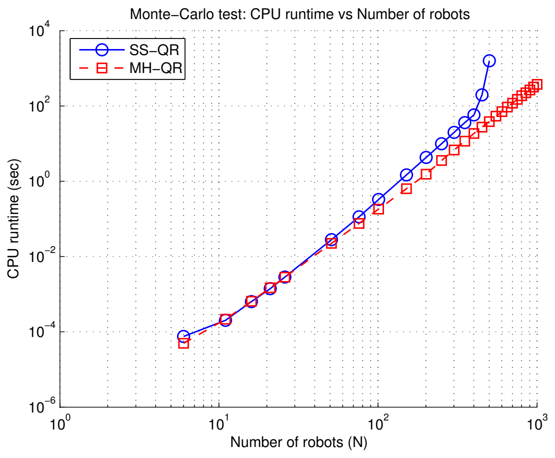

Fig. 1. Average runtime of QR decomposition of $\mathbf{H}$. Comparison between SuiteSparseQR (SS-QR) and the Modified Householder QR (MH-QR).

We have examined the scalability of our algorithm by varying N from 6 to 1001, and for every value of N, we have conducted 120 simulations. We count the CPU running time for a complete QR decomposition of (10) when employing the Modified Householder QR algorithm. The average running times are summarized in Figure 1, as well as in Table I.[6] Furthermore, we compared our results with the CPU running time when employing SuiteSparseQR [25], the current state-of-the-art QR decomposition package for sparse matrices, which is an implementation of the multi-frontal sparse QR factorization algorithm. All simulations were run on a Linux (kernel 2.6.32) desktop computer with a 2.66 GHz Intel Core-i5 Quadcore CPU and 4 GB of RAM.

The results presented in Table I and Figure 1 illustrate

[6]Due to space limitations, we only list partial results of the average CPU runtime in Table I. For complete results, please refer to [7].

that when the number of robots is small ($N \leq 21$), both SuiteSparseQR and the Modified Householder QR achieve indistinguishable performances, with SuiteSparseQR slightly faster as compared to the Modified Householder QR. However, as N increases ($N \geq 26$), the Modified Householder QR algorithm significantly outperforms SuiteSparseQR. Additionally, we were unable to run QR decomposition using SuiteSparseQR when $N \geq 551$, due to memory shortage. In contrast, the Modified Householder QR is applicable even when the number of robots increases to 1001, and it successfully performs QR factorization on $\mathbf{H}$, whose dimensions are around 2 million by 3 thousand, in about 375 seconds.

Furthermore, we have examined the accuracy of the proposed Modified Householder QR. In particular, we computed the Frobenius norm of $\mathbf{H}^{\mathrm{T}}\mathbf{H} - \mathbf{R}^{\mathrm{T}}\mathbf{R}$, which is 0 in the ideal case. We have compared the Frobenius norms of SuiteSparseQR and the Modified Householder QR. The results demonstrate that the Modified Householder QR attains higher arithmetic accuracy than SuiteSparseQR. Due to space limitations, we are unable to include the results in the paper. The interested reader can refer to [7] for more details.

VI. Conclusion

In this paper, we have developed an efficient algorithm for QR decomposition of sparse matrices, namely the Modified Householder QR. The proposed algorithm has been successfully applied to 2-D multi-robot CL. In particular, we have shown that the overall computational complexity per EKF update using QR factorization, when implemented using the Modified Householder QR algorithm, is of $\mathcal{O}(N^3)$, i.e., at least one order of magnitude reduction as compared to the standard EKF update process. Simulation results demonstrate that for large number of robots, the Modified Householder QR algorithm attains higher accuracy and significantly outperforms SuiteSparseQR, the current state-of-the-art QR decomposition algorithm of sparse matrices, in terms of CPU runtime.

In our future work, we plan to extend our current approach and apply it to CL in 3-D. Finally, we intend to investigate distributed and decentralized implementations of the Modified Householder QR algorithm to ensure that the overall computational load is evenly shared among every robot in the team [15], as well as to account for limitations on the robots' communication bandwidth and range [13], [26], [27].

References

[1] J. Polastre, "Design and implementation of wireless sensor networks for habitat monitoring," Master's thesis, University of California, Berkeley, Berkeley, CA, May 2003.

[2] L. E. Parker, B. Birch, and C. Reardon, "Indoor target intercept using an acoustic sensor network and dual wavefront path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Las Vegas, NV, Oct. 27-31 2003, pp. 278–283.

[3] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, "Towards robotic assistants in nursing homes: challenges and results," *Robot. Auton. Syst.*, vol. 42, no. 3-4, pp. 271–281, Mar. 2003.

[4] K. Zhou and S. I. Roumeliotis, "Multi-robot active target tracking with combinations of relative observations," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 678–695, Aug. 2011.

[5] S. I. Roumeliotis and G. Bekey, "Distributed multirobot localization," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 781–795, Oct. 2002.

[6] A. I. Mourikis and S. I. Roumeliotis, "Performance analysis of multirobot cooperative localization," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 666–681, Aug. 2006.

[7] K. Zhou and S. I. Roumeliotis, "A sparsity-aware QR decomposition algorithm for efficient cooperative localization," Dept. Comput. Sci. Eng. Univ. Minnesota, Minneapolis, MN, Tech. Rep. TR-2011-004, Sep. 2011, http://mars.cs.umn.edu/tr/reports/Ke11b.pdf.

[8] R. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Diego, CA, May 8-13 1994, pp. 1250–1257.

[9] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Auton. Robots*, vol. 8, no. 3, pp. 325–344, Jun. 2000.

[10] S. I. Roumeliotis and I. Rekleitis, "Propagation of uncertainty in cooperative multirobot localization: analysis and experimental results," *Auton. Robots*, vol. 17, no. 1, pp. 41–54, Jul. 2004.

[11] N. Trawny, S. I. Roumeliotis, and G. B. Giannakis, "Cooperative multirobot localization under communication constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, Kobe, Japan, May 12-17 2009, pp. 4394–4400.

[12] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Localization for mobile robot teams using maximum likelihood estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Lausanne, Switzerland, Sep. 30 - Oct. 5 2002, pp. 434–459.

[13] K. Y. K. Leung, T. D. Barfoot, and H. H. T. Liu, "Decentralized localization of sparsely-communicating robot networks: a centralized-equivalent approach," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 62–77, Feb. 2010.

[14] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Localization for mobile robot teams: a distributed MLE approach," in *Experimental Robotics VIII*, ser. Springer Tracts in Advanced Robotics, B. Siciliano and P. Dario, Eds. Berlin, German: Springer-Verlag, 2003, vol. 5, pp. 146–155.

[15] E. D. Nerurkar, S. I. Roumeliotis, and A. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *Proc. IEEE Int. Conf. Robot. Autom.*, Kobe, Japan, May 12-17 2009, pp. 1402–1409.

[16] S. Panzieri, F. Pascucci, and R. Setola, "Multirobot localisation using interlaced extended Kalman filter," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, Oct. 9-15 2006, pp. 2816–2821.

[17] L. Glielmo, R. Setola, and F. Vasca, "An interlaced extended Kalman filter," *IEEE Trans. Autom. Control*, vol. 44, no. 8, pp. 1546–1549, Aug. 1999.

[18] N. Karam, F. Chausse, R. Aufrere, and R. Chapuis, "Localization of a group of communicating vehicles by state exchange," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, Oct. 9-15 2006, pp. 519–524.

[19] A. Martinelli, "Improving the precision on multi robot localization by using a series of filters hierarchically distributed," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Diego, CA, Oct. 29-Nov. 2 2007, pp. 1053–1058.

[20] P. S. Maybeck, *Stochastic models, estimation and control,* Volume 1. New York, NY: Academic Press, 1979.

[21] D. S. Bayard and P. B. Brugarolas, "On-board vision-based spacecraft estimation algorithm for small body exploration," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 1, pp. 243–260, Jan. 2008.

[22] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins University Press, 1996.

[23] P. A. Businger and G. H. Golub, "Linear least squares solutions by Householder transformations," *Numer. Math.*, vol. 7, no. 3, pp. 269–276, Jun. 1965.

[24] L. Kaufman, "Application of dense Householder transformations to a sparse matrix," *ACM Trans. Math. Softw.*, vol. 5, no. 4, pp. 442–450, Dec. 1979.

[25] T. A. Davis, "Algorithm 915: Multifrontal multithreaded rank-revealing sparse QR factorization," *ACM Trans. Math. Softw.*, vol. 38, no. 1, Nov. 2011.

[26] E. D. Nerurkar and S. I. Roumeliotis, "Asynchronous multi-centralized cooperative localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Taipei, Taiwan, Oct. 18-22 2010, pp. 4352–4359.

[27] E. D. Nerurkar, K. X. Zhou, and S. I. Roumeliotis, "A hybrid estimation framework for cooperative localization under communication constraints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Francisco, CA, Sep. 25–30 2011, pp. 502–509.