A Sparsity-aware QR Decomposition Algorithm for Efficient Cooperative Localization

Ke Zhou and Stergios I. Roumeliotis
{kezhou|stergios}@cs.umn.edu

Department of Computer Science & Engineering
University of Minnesota

Multiple Autonomous
Robotic Systems Labroratory

Technical Report
Number -2011-004
September 2011

Dept. of Computer Science & Engineering
University of Minnesota
4-192 EE/CS Building
200 Union St. S.E.
Minneapolis, MN 55455
Tel: (612) 625-2217
Fax: (612) 625-0572

A Sparsity-aware QR Decomposition Algorithm for Efficient Cooperative Localization

Ke 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 $\mathbf{H}$. In contrast to the standard EKF update, whose time 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 $\mathbf{H}$, and prove that the overall time complexity of the EKF update, based on our QR factorization scheme, reduces to $\mathcal{O}(N^3)$. Additionally, the space complexity of the proposed Modified Householder QR algorithm is $\mathcal{O}(N^2)$. Finally, we validate the Modified Householder QR algorithm through extensive simulations and experiments, and demonstrate its superior performance both in accuracy and in CPU runtime, as compared to the current state-of-the-art QR decomposition algorithm.**

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], defense applications [4], as well as target tracking [5], [6].

Regardless of the applications, 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 encoder [7], the uncertainty of the robot pose estimates generated using this technique (dead-reckoning) increases fast, and eventually renders these estimates useless. Although one can overcome this limitation by equipping every robot with absolute positioning sensors such as the Global Positioning System (GPS), 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, it has been shown that the accuracy and performance of the robot pose estimates can significantly improve [8], even in the absence of GPS.

As shown in [7], 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 time complexity for processing each exteroceptive measurement is $\mathcal{O}(N^2)$ (N being the number of robots) [7]. Since the maximum possible number of exteroceptive measurements is $(N-1)N$ at every time step, the overall time complexity for updating the state and covariance, in the worst case, becomes $\mathcal{O}(N^4)$ per time step. As N grows, this high (time) complexity may prohibit real-time performance.

In this paper, we investigate the computational complexity (i.e., *time* complexity and *space* complexity) of the centralized EKF-based CL algorithm, considering the most challenging situation 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 time complexity of state and covariance updates can be reduced to $\mathcal{O}(N^3)$ by employing the Information filter (see Section III-C). To further improve the numerical stability of the Information filter, we present an EKF update scheme based on the QR factorization (also called QR decomposition) of the measurement Jacobian $\mathbf{H}$ (see Section III-D). While the Standard Householder QR algorithm has time complexity $\mathcal{O}(N^4)$ and space complexity $\mathcal{O}(N^3)$ for decomposing (or factorizing) $\mathbf{H}$, we develop the Modified Householder QR algorithm (see Section V), which exploits the sparse structure of $\mathbf{H}$ and reduces the time complexity and space complexity of QR decomposition to $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$, respectively. As a result, the overall computational complexity of the EKF-based CL is reduced by an order of magnitude (from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^3)$ for time complexity and $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$ for space complexity) per update step.

Following a brief review of related work in Section II, we present the formulation of the EKF-based CL, as well as an EKF update scheme based on the QR factorization of $\mathbf{H}$ in Section III. In Section IV, we briefly describe the QR decomposition based on the Standard Householder QR algorithm, and show its time complexity and space complexity of factorizing $\mathbf{H}$ are $\mathcal{O}(N^4)$ and $\mathcal{O}(N^3)$, respectively. We then develop the Modified Householder QR algorithm in Section V, which exploits the sparse structure of $\mathbf{H}$ and achieves QR decomposition with time complexity $\mathcal{O}(N^3)$ and space complexity $\mathcal{O}(N^2)$. Simulation and real-world experimental results are presented in Sections VI and VII, respectively, followed by the conclusions and future directions in Section VIII.

This work was supported by the University of Minnesota through the Digital Technology Center (DTC), the National Science Foundation (IIS-0643680, IIS-0811946, IIS-0835637), and AFOSR (FA9550-10-1-0567).

II. Literature Review

Multi-robot cooperative localization (CL) has received considerable attention in the literature (e.g., [9], [10], [11], [12]). Various system architectures, such as centralized [7], [13] or distributed [14], [15], [16], have been proposed for CL. These system architectures use various estimation algorithms, such as the EKF [7], the maximum likelihood estimator (MLE) [13], the maximum a posteriori estimator (MAP) [16], and particle filters [10]. In this paper, we focus our discussion on centralized or distributed approaches that fuse data in batch mode (i.e., MLE or MAP, see Section II-A) or sequentially (i.e., EKF, see Section II-B). The latter case is the main focus of our work. In what follows, we denote as K the total number of time steps addressed in batch mode, 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 $(N-1)N$.

A. MLE or MAP-based Estimator

Howard *et al.* [13] presented a centralized MLE-based algorithm to address the CL problem. The solution of the underlying non-convex optimization problem, formulated by maximizing the conditional probability density function (pdf), is computed iteratively by the conjugate gradient (CG) algorithm with time complexity per CG iteration of $\mathcal{O}(KN^2)$.

Contrary to [13], where all measurements are transmitted and processed at a fusion center, the authors [15] proposed to solve the MLE in a distributed fashion. The original non-convex optimization problem is decomposed into N sub-problems, one for each robot. In particular, the ith robot (i.e., robot-i) minimizes only a part of the cost function that involves terms corresponding to its own proprioceptive measurements, as well as the relative measurements between robot-i and other robots. The optimization process in this case approximates other robots' poses as constants. However, there exists no proof that the proposed approach guarantees convergence to even one of the local minima. Additionally, the computational complexity of the algorithm is not addressed in the paper.

Similar to [13], Dellaert *et al.* [17] addressed MAP-based centralized CL by incorporating prior information about the initial poses of robots. Contrary to [13], the resulting non-convex optimization problem is solved iteratively by the Levenberg-Marquardt (LM) algorithm. The solution of the system of linear equations at each iteration of the LM method is determined through a sparse QR solver. Although some insights were provided in the selection of initial estimates, the algorithm does not guarantee convergence to the global optimum. Furthermore, the paper does not provide information about the worst-case time complexity of the sparse QR solver.

Recently, Nerurkar *et al.* [16] introduced a fully distributed CG algorithm to address the MAP-based CL. Similar to [17], the resulting non-convex optimization problem is solved iteratively using the LM algorithm. In contrast to [13], the incremental solution at each iteration of the LM method is distributively computed by the CG algorithm. The time complexity per robot and per iteration of the LM method is of $\mathcal{O}((KN)^2)$.

The main drawback of the previous gradient-based iterative algorithms (see [13], [16], [17]) is that the total number of iterations required for convergence has not been addressed, as well as no guarantees of convergence to global optimum, due to the existence of multiple local minima of the resulting non-convex optimization problem arising from CL. Furthermore, the above batch-mode estimators can be implemented only when all data become available, and are unable to provide on-line estimations when new data arrives. This motivates us to consider EKF-based estimators in CL.

B. EKF-based Estimator

In [7], 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 stage, 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 time complexity for processing each exteroceptive measurement is $\mathcal{O}(N^2)$. Therefore, in the worst case, the overall time complexity for sequentially updating state and covariance becomes $\mathcal{O}(N^4)$ per time step. To reduce the cost of EKF-based CL, several *approximate* algorithms have been proposed.

Panzieri *et al.* [18] presented a fully decentralized algorithm based on the Interlaced EKF [19], 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 time complexity per robot and per time step is $\mathcal{O}(N)$ (hence the overall time complexity per time step is $\mathcal{O}(N^2)$) when relative measurements are processed sequentially. Similar to [18], Karam *et al.* [20] proposed a distributed EKF-based method for CL. However, contrary to [18], here the robots are restricted to exchange their state estimates with others within communication range. The time complexity per robot and per time step is $\mathcal{O}(N^2)$, resulting in the overall time complexity $\mathcal{O}(N^3)$ per time step.

Martinelli [21] developed a distributed approach for CL that uses hierarchical EKF filters to estimate the robots' poses. In this case, N robots in a team are divided into L groups, each consisting of M robots ($N = LM$). Every group contains a group 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 leaders. The time complexity 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 above approximate algorithms (see [18], [20], [21]) is that in order to reduce the computational complexity of 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 time complexity of 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 $\mathbf{x}_k^i = [(\mathbf{p}_k^i)^{\mathrm{T}}\ \varphi_k^i]^{\mathrm{T}}$, where $\mathbf{p}_k^i = [x_k^i\ y_k^i]^{\mathrm{T}}$ and φ_k^i represent the position and orientation of robot-i at time-step k with respect to the global frame of reference, 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 proprioceptive 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 robot-i's pose is propagated by[1]:

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

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

Note that due to the block diagonal structures of $\mathbf{\Phi}_{k-1}$ and $\mathbf{G}_{k-1}$, the overall computational complexity (both time and space complexity) of implementing (1) is $\mathcal{O}(N^2)$ [7].

B. Measurement Model

At time-step k, the relative distance and bearing observations recorded by the exteroceptive sensors from 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 $\mathbf{h}_k^{i,j}(\mathbf{x}_k^i, \mathbf{x}_k^j) = [d_k^{i,j}\ \theta_k^{i,j}]^{\mathrm{T}}$, with

$$d_k^{i,j} = \|\mathbf{p}_k^j - \mathbf{p}_k^i\|_2\ ,$$
$$\theta_k^{i,j} = \arctan\left(\frac{y_k^j - y_k^i}{x_k^j - x_k^i}\right) - \varphi_k^i$$

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

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

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

where

$$\mathbf{\Psi}_k^{i,j} = \boldsymbol{\nabla}_{\mathbf{x}_k^i}\mathbf{h}_k^{i,j} = \begin{bmatrix} -\frac{(\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i)^{\mathrm{T}}}{\|\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i\|_2} & 0 \\ -\frac{(\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i)^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}}{\|\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i\|_2^2} & -1 \end{bmatrix}\ , \quad (4)$$

$$\mathbf{\Upsilon}_k^{i,j} = \boldsymbol{\nabla}_{\mathbf{x}_k^j}\mathbf{h}_k^{i,j} = \begin{bmatrix} \frac{(\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i)^{\mathrm{T}}}{\|\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i\|_2} & 0 \\ \frac{(\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i)^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}}{\|\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i\|_2^2} & 0 \end{bmatrix} \quad (5)$$

are of dimensions 2×3, and $\mathbf{J} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$. Furthermore, note that the measurement Jacobian matrix $\mathbf{H}_k^{i,j}$ (of dimensions $2 \times 3N$) has a sparse structure (without loss of generality, we assume $i < j$):

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

In fact, there are at most 9 nonzero elements in $\mathbf{H}_k^{i,j}$ (5 from $\mathbf{\Psi}_k^{i,j}$ and 4 from $\mathbf{\Upsilon}_k^{i,j}$).

In this paper, we consider the worst-case 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. Thus, the total number of relative measurements per time step is $(N-1)N$, which corresponds to the most challenging case in terms of computational complexity. 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 Section V) can be readily applied without modifications.

The measurement-error equation for the robot team, obtained by stacking all the measurement residuals $\tilde{\mathbf{z}}_{k|k-1}^{i,j}$ [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 $\tilde{\mathbf{z}}_{k|k-1} = [(\tilde{\mathbf{z}}_{k|k-1}^{1,2})^{\mathrm{T}} \ldots (\tilde{\mathbf{z}}_{k|k-1}^{i,j})^{\mathrm{T}} \ldots (\tilde{\mathbf{z}}_{k|k-1}^{N,N-1})^{\mathrm{T}}]^{\mathrm{T}}$ is the measurement residual error vector of dimension $2(N-1)N$, and $\mathbf{n}_k = [(\mathbf{n}_k^{1,2})^{\mathrm{T}} \ldots (\mathbf{n}_k^{i,j})^{\mathrm{T}} \ldots (\mathbf{n}_k^{N,N-1})^{\mathrm{T}}]^{\mathrm{T}}$

[1]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, respectively. $\mathbf{1}_n$ denotes the n dimensional (column) vector with all elements being 1, and $\mathbf{e}_j$ is the unit (column) vector with a 1 in the jth coordinate and 0's elsewhere.

is zero-mean, white Gaussian measurement noise with covariance $\mathbf{C_n} = \mathbf{I}_{2(N-1)N}$.

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:

$$\mathbf{H}_k = \begin{bmatrix} \boldsymbol{\Psi}_k^{1,2} & \boldsymbol{\Upsilon}_k^{1,2} & & & & & \\ \boldsymbol{\Upsilon}_k^{2,1} & \boldsymbol{\Psi}_k^{2,1} & & & & & \\ \vdots & & & & & & \\ \boldsymbol{\Psi}_k^{1,N} & & & & & \boldsymbol{\Upsilon}_k^{1,N} & \\ \boldsymbol{\Upsilon}_k^{N,1} & & & & & \boldsymbol{\Psi}_k^{N,1} & \\ & \boldsymbol{\Psi}_k^{2,3} & \boldsymbol{\Upsilon}_k^{2,3} & & & & \\ & \boldsymbol{\Upsilon}_k^{3,2} & \boldsymbol{\Psi}_k^{3,2} & & & & \\ & \vdots & & & & & \\ & \boldsymbol{\Psi}_k^{2,N} & & & & \boldsymbol{\Upsilon}_k^{2,N} & \\ & \boldsymbol{\Upsilon}_k^{N,2} & & & & \boldsymbol{\Psi}_k^{N,2} & \\ & & \vdots & \vdots & \vdots & & \\ & & & & \boldsymbol{\Psi}_k^{N-1,N} & \boldsymbol{\Upsilon}_k^{N-1,N} & \\ & & & & \boldsymbol{\Upsilon}_k^{N,N-1} & \boldsymbol{\Psi}_k^{N,N-1} & \end{bmatrix}. \quad (6)$$

Remark 1: We highlight two important properties of $\mathbf{H}_k$. Firstly, due to its sparse structure [see (6)], each row of $\mathbf{H}_k$ has at most 5 nonzero elements [see (4)-(5)]. Per column, there exist at most $4(N-1)$ non-zeros, 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 $\boldsymbol{\Upsilon}_k^{j,1}$, $j = 2, \dots, N$, where each column of $\boldsymbol{\Psi}_k^{1,j}$ (or $\boldsymbol{\Upsilon}_k^{j,1}$) has at most 2 nonzero elements [see (4)-(5)]. Therefore $\mathbf{nnz}(\mathbf{H}_k) \sim \mathcal{O}(N^2)$, where $\mathbf{nnz}(\mathbf{H})$ denotes the number of nonzero elements of $\mathbf{H}$, and the storage of $\mathbf{H}_k$ can be efficiently achieved using $\mathcal{O}(N^2)$ memory cells. Secondly, since all the exteroceptive measurements are *relative* observations between each pair of robots, and *absolute* pose measurements (such as GPS) are unavailable, $\mathbf{H}_k$ is *rank deficiency*. In particular, we have the following proposition.

Proposition 1: $\mathbf{rank}(\mathbf{H}_k) \leq 3N-2$, hence, $\mathbf{H}_k$ is *not full (column) rank*.

Proof: To show $\mathbf{rank}(\mathbf{H}_k) \leq 3N-2$, we first construct a $3N \times 2$ matrix $\boldsymbol{\Xi} = \mathbf{1}_N \otimes \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{0}_{1\times 2} \end{bmatrix}$, where $\otimes$ denotes the Kronecker tensor product. Next, for every sub-matrix $\mathbf{H}_k^{i,j}, 1 \leq i \neq j \leq N$, it is straightforward to verify that

$$\mathbf{H}_k^{i,j}\boldsymbol{\Xi} = \begin{bmatrix} -\frac{(\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i)^{\mathrm{T}}}{\|\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i\|_2} \\ -\frac{(\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i)^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}}{\|\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i\|_2^2} \end{bmatrix} + \begin{bmatrix} \frac{(\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i)^{\mathrm{T}}}{\|\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i\|_2} \\ \frac{(\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i)^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}}{\|\hat{\mathbf{p}}_{k|k-1}^j - \hat{\mathbf{p}}_{k|k-1}^i\|_2^2} \end{bmatrix} = \mathbf{0}.$$

Hence, $\mathbf{H}_k\boldsymbol{\Xi} = \mathbf{0}_{2(N-1)N\times 2}$. Since $\mathbf{rank}(\boldsymbol{\Xi}) = 2$, we conclude $\mathbf{rank}(\mathbf{H}_k) \leq 3N-2$, and thus $\mathbf{H}_k$ is not full rank. ■

C. State and Covariance Updates

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 are updated as

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{z}}_{k|k-1}, \quad (7)$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^{\mathrm{T}}, \quad (8)$$

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

Unfortunately, the time complexity and space complexity of the state and covariance updates using (7)-(8) are $\mathcal{O}(N^6)$ and $\mathcal{O}(N^4)$, respectively, due to the inversion and storage 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 (7)-(8) becomes prohibitive as the number of robots increases.

It is possible to avoid the inversion and storage of $\mathbf{S}_k$ by processing $(N-1)N$ measurements *sequentially*. In this case, since the time complexity for processing every relative measurement $\mathbf{z}_k^{i,j}$ is $\mathcal{O}(N^2)$ [7], the total time complexity per time step becomes $\mathcal{O}(N^4)$. In addition, space complexity can be reduced to $\mathcal{O}(N^2)$ [7]. However, due to the nonlinearity of the measurement model [see (2)], the robot pose estimates obtained by sequential updates are less accurate than the estimates obtained by concurrent updates using (7)-(8).

Another alternative approach for updating the state and covariance is to employ the Information filter [22, Chapter 5, Section 5.6] (for $\mathbf{C_n} = \mathbf{I}_{2(N-1)N}$):

$$\mathbf{P}_{k|k} = \left(\mathbf{P}_{k|k-1}^{-1} + \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k\right)^{-1}, \quad (9)$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{k|k}\mathbf{H}_k^{\mathrm{T}}\tilde{\mathbf{z}}_{k|k-1}\ . \quad (10)$$

It is worth noting that computing $\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k$ and $\mathbf{P}_{k|k}\mathbf{H}_k^{\mathrm{T}}\tilde{\mathbf{z}}_{k|k-1}$ requires basic computational steps only $\mathcal{O}(N^2)$, due to the sparse structure of $\mathbf{H}_k$ (See Appendix A). Hence, the most computationally demanding operation of (9)-(10) 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 time complexity of the state and covariance updates using (9)-(10) is $\mathcal{O}(N^3)$, a significant complexity reduction as compared to the standard EKF updates using (7)-(8). Additionally, since both $\mathbf{P}_{k|k-1}$ and $\mathbf{H}_k$ occupy $\mathcal{O}(N^2)$ memory cells (because of $\mathbf{nnz}(\mathbf{H}_k) \sim \mathcal{O}(N^2)$ and we only need to store non-zeros of $\mathbf{H}_k$), the space complexity of updating the state and covariance using (9)-(10) is reduced from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^2)$.

Unfortunately, (9) often suffers from numerical instability, due to $\kappa(\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k) = \kappa^2(\mathbf{H}_k)$ ($\kappa(\mathbf{H})$ is the condition number of $\mathbf{H}$), which renders the state and covariance updates using (9)-(10) numerically less robust [23].

D. State and Covariance Updates through QR Factorization

An effective strategy to overcome the numerical instability of the Information filter [see (9)-(10)] is to apply *thin* QR factorization with *column pivoting* on $\mathbf{H}_k$ [23], i.e.,

$$\mathbf{H}_k = \mathbf{Q}_{\mathbf{H}_k}\mathbf{R}_{\mathbf{H}_k}\boldsymbol{\Pi}_{\mathbf{H}_k}^{\mathrm{T}}\ , \quad (11)$$

where $\mathbf{Q}_{\mathbf{H}_k}$ is a $2(N-1)N \times 3N$ matrix with orthonormal columns (i.e., $\mathbf{Q}_{\mathbf{H}_k}^{\mathrm{T}}\mathbf{Q}_{\mathbf{H}_k} = \mathbf{I}_{3N}$), and $\mathbf{R}_{\mathbf{H}_k}$ is an *upper triangular* matrix with the absolute value of its diagonal elements arranged in decreasing order. Furthermore, $\boldsymbol{\Pi}_{\mathbf{H}_k}$ is a (column) permutation matrix generated from column pivoting techniques, satisfying $\boldsymbol{\Pi}_{\mathbf{H}_k}\boldsymbol{\Pi}_{\mathbf{H}_k}^{\mathrm{T}} = \boldsymbol{\Pi}_{\mathbf{H}_k}^{\mathrm{T}}\boldsymbol{\Pi}_{\mathbf{H}_k} = \mathbf{I}_{3N}$. Note that in order to ensure orthogonality of $\mathbf{Q}_{\mathbf{H}_k}$, it is necessary to invoke column pivoting on $\mathbf{H}_k$ [24, Section 5.4.1], due to the fact that $\mathbf{H}_k$ is rank deficient (see Proposition 1).

Substituting (11) into (9), we obtain the EKF update equa-

tions based on QR factorization

$$\mathbf{P}_{k|k} = \left(\mathbf{P}_{k|k-1}^{-1} + \mathbf{\Pi}_{\mathbf{H}_k}\mathbf{R}_{\mathbf{H}_k}^{\mathrm{T}}\mathbf{R}_{\mathbf{H}_k}\mathbf{\Pi}_{\mathbf{H}_k}^{\mathrm{T}}\right)^{-1} \tag{12}$$

$$= \mathbf{P}_{k|k-1} - \mathbf{P}_{k|k-1}\mathbf{\Pi}_{\mathbf{H}_k}\mathbf{R}_{\mathbf{H}_k}^{\mathrm{T}}\mathbf{\Sigma}_k^{-1}\mathbf{R}_{\mathbf{H}_k}\mathbf{\Pi}_{\mathbf{H}_k}^{\mathrm{T}}\mathbf{P}_{k|k-1} \ ,$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{k|k}\mathbf{H}_k^{\mathrm{T}}\tilde{\mathbf{z}}_{k|k-1} \ , \tag{13}$$

where $\mathbf{\Sigma}_k = \mathbf{R}_{\mathbf{H}_k}\mathbf{\Pi}_{\mathbf{H}_k}^{\mathrm{T}}\mathbf{P}_{k|k-1}\mathbf{\Pi}_{\mathbf{H}_k}\mathbf{R}_{\mathbf{H}_k}^{\mathrm{T}} + \mathbf{I}_{3N}$, and the last equality in (12) is established by the matrix inversion lemma.

Note that in contrast to $\mathbf{S}_k$ [see (8)], whose dimensions are $2(N-1)N \times 2(N-1)N$, the matrix $\mathbf{\Sigma}_k$ in (12) has dimensions only $3N \times 3N$. Thus, assuming both $\mathbf{R}_{\mathbf{H}_k}$ and $\mathbf{\Pi}_{\mathbf{H}_k}$ are given, the total time complexity and space complexity of the state and covariance updates using (12)-(13) are $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$, respectively, the same order of computational complexity as of using (9)-(10). More importantly, $\kappa(\mathbf{R}_{\mathbf{H}_k}) = \kappa(\mathbf{H}_k)$ [24]. Hence the numerical stability of (12)-(13) is improved as compared to (9)-(10).

Therefore, in order to ensure that the time complexity and space complexity for the state and covariance updates through QR factorization [see (12)-(13)] remain $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$, we conclude that the maximum number of arithmetic operations and memory usage to implement (11) should be $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$, respectively.

In what follows, we focus on the *Householder QR* algorithm, one of the most widely adopted numerical techniques for QR factorization [24, Section 5.2.1]. We analyze the computational complexity of QR decomposition of $\mathbf{H}_k$ using the Householder QR algorithm (see Section IV), and develop a modified version of Householder QR that exploits the sparsity of $\mathbf{H}_k$ and reduces the overall computational complexity by an order of magnitude (see Section V).

IV. Standard Householder QR

As mentioned in Section III-D, a numerically robust and efficient QR decomposition algorithm with time complexity and space complexity at most $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$, respectively, is a prerequisite for successfully implementing the state and covariance updates through (12)-(13). Several methods exist for performing QR factorization, such as the Cholesky decomposition (CHO), the modified Gram-Schmidt process (MGS), the Givens rotations (GIV), or the Householder transformations (also called Householder reflections) [24, Section 5.2]. Due to its simplicity and numerical stability, we adopt the column pivoted QR factorization algorithm utilizing Householder transformations, which is termed as Standard Householder QR in this paper.[2] In what follows, we present a brief overview of Householder reflections (see Section IV-A), as well as the essence of the Standard Householder QR algorithm (see Section IV-B). Additionally, the computational complexity analysis conducted in Sections IV-C reveals that the QR decomposition of $\mathbf{H}_k$, when applying the Standard Householder QR, has time complexity $\mathcal{O}(N^4)$ and space complexity $\mathcal{O}(N^3)$. For clarity, the time-step index k is dropped from (11) throughout the rest of the paper, with $m_{\mathbf{H}} = 2(N-1)N$ and $n_{\mathbf{H}} = 3N$ denoting the number of rows and columns of $\mathbf{H}$, respectively.

[2]We have conducted computational complexity analysis when employing CHO, MGS, and GIV. More specifically, it is shown in Appendix B that the time complexity and space complexity of CHO are $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$, respectively, 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. On the other hand, both MGS and GIV require $\mathcal{O}(N^4)$ arithmetic operations and $\mathcal{O}(N^3)$ memory cells (see Appendix B).

A. Householder Reflection

We first introduce an *orthogonal* and *symmetric* Householder (reflection) matrix $\mathbf{Q} = \mathbf{I} - \beta\mathbf{v}\mathbf{v}^{\mathrm{T}}$, where $\beta = 2/\|\mathbf{v}\|_2^2$, and the *nonzero* vector $\mathbf{v}$ is called a Householder vector.

Any matrix $\mathbf{H}$, multiplied by $\mathbf{Q}$, can be computed as

$$\mathbf{Q}\mathbf{H} = (\mathbf{I} - \beta\mathbf{v}\mathbf{v}^{\mathrm{T}})\mathbf{H} = \mathbf{H} - \mathbf{v}\left(\beta\mathbf{H}^{\mathrm{T}}\mathbf{v}\right)^{\mathrm{T}} \ . \tag{14}$$

It is worth emphasizing that Householder updates [see (14)] "never entail the explicit formation of the Householder matrix $\mathbf{Q}$. Instead, an Householder update involves a matrix-vector multiplication $\boldsymbol{\zeta} = \beta\mathbf{H}^{\mathrm{T}}\mathbf{v}$ and an outer product update $\mathbf{H} - \mathbf{v}\boldsymbol{\zeta}^{\mathrm{T}}$" to reduce computational complexity [24, Section 5.1.4].

Now suppose for a given matrix $\mathbf{H}$, our objective is to seek a Householder matrix $\mathbf{Q}$ (or corresponding Householder vector $\mathbf{v}$), such that the first column of $\mathbf{Q}\mathbf{H}$ has zeros below its first component. In other words, letting $\mathbf{u}$ denote the first column of $\mathbf{H}$, we select a vector $\mathbf{v}$ and a scalar α, such that

$$\mathbf{Q}\mathbf{u} = \mathbf{u} - \beta(\mathbf{v}^{\mathrm{T}}\mathbf{u})\mathbf{v} = r\mathbf{e}_1 \ . \tag{15}$$

The solution of (15) is provided by [24, Section 5.1.2]

$$r = -\mathrm{sign}(u_1)\|\mathbf{u}\|_2 \ , \ \beta = \left(\|\mathbf{u}\|_2^2 + |u_1|\|\mathbf{u}\|_2\right)^{-1} \ , \tag{16}$$

$$\mathbf{v} = \mathbf{u} + \mathrm{sign}(u_1)\|\mathbf{u}\|_2\mathbf{e}_1 \ , \tag{17}$$

where u_1 is the first element of $\mathbf{u}$. For notational convenience, we use $\mathbf{u}_{-1}$ to denote the vector obtained by removing the first element of $\mathbf{u}$, i.e., $\mathbf{u} = [u_1 \ \mathbf{u}_{-1}^{\mathrm{T}}]^{\mathrm{T}}$.

Remark 2: A key observation of (17) is that the vectors $\mathbf{v}$ and $\mathbf{u}$ differ only by their first elements ($v_1 = u_1 - r$). In other words, $\mathbf{v}_{-1} = \mathbf{u}_{-1}$. This property plays a pivotal role in the development of the Modified Householder QR in Section V.

B. Description of the Standard Householder QR

In the previous section (see Section IV-A), we provide a framework to generate a Householder matrix $\mathbf{Q}$ such that the product $\mathbf{Q}\mathbf{H}$ eliminates all except the first element of the first column of $\mathbf{H}$. The Standard Householder QR algorithm (with column pivoting) [24, Algorithm 5.4.1] extends the above strategy by applying *a sequence of* Householder matrices (multiplying $\mathbf{H}$ from *left* by $\mathbf{Q}$), as well as *a sequence of* column permutation matrices (multiplying $\mathbf{H}$ from *right* by $\mathbf{\Pi}$), to gradually transform $\mathbf{H}$ into an upper triangular form $\mathbf{R}_{\mathbf{H}}$, whose diagonal elements $r_{i,i}, \ i = 1, \ldots, n_{\mathbf{H}}$, satisfying $|r_{i,i}| \geq |r_{j,j}|$ for $1 \leq i < j \leq n_{\mathbf{H}}$.

More specifically, suppose that after $(\ell - 1)$ Householder matrices $\{\mathbf{Q}_i = \mathbf{I}_{m_{\mathbf{H}}} - \beta_i\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}, i = 1, \ldots, \ell - 1\}$ have left-multiplied $\mathbf{H}$, as well as $(\ell - 1)$ column permutation matrices $\{\mathbf{\Pi}_i, i = 1, \ldots, \ell - 1\}$ have right-multiplied $\mathbf{H}$, the resulting matrix $\mathbf{H}^{(\ell-1)}$ takes the following block form

$$\mathbf{H}^{(\ell-1)} = \mathbf{Q}_{\ell-1}\cdots\mathbf{Q}_1\mathbf{H}\mathbf{\Pi}_1\cdots\mathbf{\Pi}_{\ell-1} = \begin{bmatrix} \mathbf{H}_{1,1}^{(\ell-1)} & \mathbf{H}_{1,2}^{(\ell-1)} \\ \mathbf{0} & \mathbf{H}_{2,2}^{(\ell-1)} \end{bmatrix}, \tag{18}$$

where $\mathbf{H}_{1,1}^{(\ell-1)}$ is an *upper triangular* matrix of dimensions $(\ell-1)\times(\ell-1)$ with the absolute value of its diagonal elements arranged in decreasing order.

At the ℓth iteration, the Standard Householder QR algorithm firstly performs column pivoting (i.e., right-multiplying $\mathbf{H}^{(\ell-1)}$ with $\mathbf{\Pi}_\ell$), followed by an Householder update by left-multiplying $\mathbf{H}^{(\ell-1)}\mathbf{\Pi}_\ell$ with $\mathbf{Q}_\ell$.

Specifically, for column pivoting, we compute the squared norm of every column of $\mathbf{H}_{2,2}^{(\ell-1)}$, denoted as $c_\ell,\ldots,c_{n_\mathbf{H}}$, and seek $\ell^*=\arg\max_j\{c_j,\ j=\ell,\ldots,n_\mathbf{H}\}$ and $c^*=c_{\ell^*}$. Note that $c_\ell,\ldots,c_{n_\mathbf{H}}$ can be updated recursively by a formula (see Algorithm 1, Line 11) discovered by Businger and Golub [25], which significantly reduces the computational complexity [24, Section 5.4.1]. The permutation matrix $\mathbf{\Pi}_\ell$ is generated by exchanging canonical vectors $\mathbf{e}_\ell$ and $\mathbf{e}_{\ell^*}$ in $\mathbf{I}_{n_\mathbf{H}}$. Furthermore, $\mathbf{H}^{(\ell-1)}\mathbf{\Pi}_\ell$ is equivalent to performing column permutations by swapping the ℓth column and ℓ^*th column of $\mathbf{H}^{(\ell-1)}$. Without loss of generality, in what follows, we assume $\mathbf{\Pi}_\ell=\mathbf{I}_{n_\mathbf{H}}$.

For Householder update, we seek $\mathbf{Q}_\ell=\mathbf{I}_{m_\mathbf{H}}-\beta_\ell\mathbf{v}_\ell\mathbf{v}_\ell^{\mathrm{T}}$ such that the *first ℓ columns* of $\mathbf{H}^{(\ell)}=(\mathbf{I}_{m_\mathbf{H}}-\beta_\ell\mathbf{v}_\ell\mathbf{v}_\ell^{\mathrm{T}})\mathbf{H}^{(\ell-1)}\mathbf{\Pi}_\ell$ are *upper triangular*. Since $\mathbf{H}_{1,1}^{(\ell-1)}$ is upper triangular, the Householder matrix $\mathbf{Q}_\ell$ only needs to zero all but the 1st component of $\mathbf{u}$, where $\mathbf{u}$ is *the first column* of $\mathbf{H}_{2,2}^{(\ell-1)}$, while leaving the previous ℓ–1 columns of $\mathbf{H}^{(\ell-1)}$ *intact* [24, Section 5.2.1]. This is achieved by selecting $\mathbf{v}_\ell=[\mathbf{0}_{1\times(\ell-1)}\ \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$, with

$$\mathbf{v}=\mathbf{u}+\mathrm{sign}(u_1)(c^*)^{1/2}\,\mathbf{e}_1\ , \tag{19}$$

where $c^*=\|\mathbf{u}\|_2^2$ and $\beta_\ell=(c^*+|u_1|(c^*)^{1/2})^{-1}$. Note that $c^*=c_{\ell^*}$ is available after the column pivoting stage and needs *not* to be recomputed through $c^*=\mathbf{u}^{\mathrm{T}}\mathbf{u}$.

Now let us examine the structure of $\mathbf{H}^{(\ell)}=\mathbf{Q}_\ell\mathbf{H}^{(\ell-1)}\mathbf{\Pi}_\ell$, after the ℓth iteration of the Standard Householder QR. For clarity, we partition 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 $\mathbf{t}$ is the first column of $\mathbf{H}_{1,2}^{(\ell-1)}$, and $\overline{\mathbf{H}}_{1,2}^{(\ell-1)}$ consists of its 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 $\mathbf{H}_{2,2}^{(\ell-1)}$, the *row* vector $\bar{\mathbf{s}}^{\mathrm{T}}$ represents the first row of $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$, and $\underline{\mathbf{H}}_{2,2}^{(\ell-1)}$ is obtained by removing $\bar{\mathbf{s}}^{\mathrm{T}}$ from $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$. Using this notation, we establish Proposition 2.

Proposition 2: $\mathbf{H}^{(\ell)}$ has the following block structure

$$\mathbf{H}^{(\ell)}=\mathbf{Q}_\ell\cdots\mathbf{Q}_1\mathbf{H}\mathbf{\Pi}_1\cdots\mathbf{\Pi}_\ell=\begin{bmatrix}\mathbf{H}_{1,1}^{(\ell)}&\mathbf{H}_{1,2}^{(\ell)}\\ \mathbf{0}&\mathbf{H}_{2,2}^{(\ell)}\end{bmatrix}\ , \tag{20}$$

where $\mathbf{H}_{1,1}^{(\ell)}=\begin{bmatrix}\mathbf{H}_{1,1}^{(\ell-1)}&\mathbf{t}\\ \mathbf{0}&r_\ell\end{bmatrix}$ is an *upper triangular* matrix of dimensions $\ell\times\ell$, and $\mathbf{H}_{1,2}^{(\ell)}=\begin{bmatrix}\big(\overline{\mathbf{H}}_{1,2}^{(\ell-1)}\big)^{\mathrm{T}}&\mathbf{s}\end{bmatrix}^{\mathrm{T}}$, with

$$r_\ell=-\mathrm{sign}(u_1)(c^*)^{1/2}\ , \tag{21}$$

$$\mathbf{s}=\bar{\mathbf{s}}-v_1\boldsymbol{\delta}\ , \tag{22}$$

$$\mathbf{H}_{2,2}^{(\ell)}=\underline{\mathbf{H}}_{2,2}^{(\ell-1)}-\mathbf{u}_{-1}\boldsymbol{\delta}^{\mathrm{T}}\ . \tag{23}$$

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

Proof: The proof is shown in Appendix C. ■

In summary, we have briefly described column pivoting and Householder update at the ℓth iteration of the Standard Householder QR algorithm. The overall algorithm terminates when $c^*<\epsilon$, where ϵ is a small positive scalar chosen to take into account of machine roundoff errors (ideally the algorithm terminates when $c^*=0$). Denoting the overall number of iterations as l, the output $\mathbf{R}_\mathbf{H}$ is selected as the first $n_\mathbf{H}$ rows of $\mathbf{H}^{(l)}$ (Note that the explicit expression of each individual $\mathbf{Q}_\ell,\ \ell=1,\ldots,l$, as well as $\mathbf{Q}_\mathbf{H}=\mathbf{Q}_1\cdots\mathbf{Q}_l$, is *not* computed, since the covariance update [see (12)] *does not involve* $\mathbf{Q}_\mathbf{H}$. In addition, we *never* entail the explicit formation of $\mathbf{\Pi}_\ell,\ \ell=1,\ldots,l$, as well as $\mathbf{\Pi}_\mathbf{H}=\mathbf{\Pi}_1\cdots\mathbf{\Pi}_l$. Instead, $\mathbf{\Pi}_\mathbf{H}$ is represented and updated by a *permutation vector* $[\pi_1\ \ldots\ \pi_{n_\mathbf{H}}]$, i.e., $\mathbf{\Pi}_\mathbf{H}=[\mathbf{e}_{\pi_1}\ \ldots\ \mathbf{e}_{\pi_{n_\mathbf{H}}}]$. For completeness, the flow chart of the Standard Householder QR is summarized in Algorithm 1.

Algorithm 1 Standard Householder QR

Require: $\mathbf{H}^{(0)}=\mathbf{H}$ of dimensions $m_\mathbf{H}\times n_\mathbf{H}$ ($m_\mathbf{H}\geq n_\mathbf{H}$).
Ensure: $\mathbf{R}_\mathbf{H}$ and $\mathbf{\Pi}_\mathbf{H}$ of dimensions $n_\mathbf{H}\times n_\mathbf{H}$.
1: **for** $i=1$ **to** $n_\mathbf{H}$, **do**
2: Initialize $\pi_i=i$, and compute c_i, the squared norm of the ith column of $\mathbf{H}_{2,2}^{(0)}=\mathbf{H}^{(0)}$.
3: **end for**
4: Set $\ell=1$, determine $\ell^*=\arg\max_j\{c_j,\ j=\ell,\ldots,n_\mathbf{H}\}$ and $c^*=c_{\ell^*}$.
5: **while** $c^*>\epsilon$ **do**
6: Swap the ℓth and ℓ^*th columns of $\mathbf{H}^{(\ell-1)}$, exchange c_ℓ and c_{ℓ^*}, and then interchange π_ℓ with π_{ℓ^*}.
7: Calculate $\beta_\ell,\mathbf{v}$ from $\mathbf{u}$ ($\mathbf{u}$ is the first column of $\mathbf{H}_{2,2}^{(\ell-1)}$) using (19), and r_ℓ from c^* using (21).
8: Compute $\boldsymbol{\delta}$ and $\mathbf{s}=[s_1\ \ldots\ s_{n_\mathbf{H}-\ell}]^{\mathrm{T}}$ [see (22)].
9: Update the sub-matrix $\mathbf{H}_{2,2}^{(\ell)}$ [see (23)].
10: **for** $j=\ell+1$ **to** $n_\mathbf{H}$, **do**
11: Modify $c_j\Leftarrow c_j-s_{j-\ell}^2$.
12: **end for**
13: Seek $\ell^*=\arg\max_j\{c_j,\ j=\ell+1,\ldots,n_\mathbf{H}\}$ and $c^*=c_{\ell^*}$, and then $\ell\Leftarrow\ell+1$.
14: **end while**
15: **return** $\mathbf{\Pi}_\mathbf{H}=[\mathbf{e}_{\pi_1}\ \ldots\ \mathbf{e}_{\pi_{n_\mathbf{H}}}]$, and $\mathbf{R}_\mathbf{H}$ selected as the first $n_\mathbf{H}$ rows of $\mathbf{H}^{(l)}$, l being the total number of iterations.

C. Complexity Analysis of the Standard Householder QR

We now analyze the computational complexity of factorizing $\mathbf{H}$ using the Standard Householder QR algorithm. The main result is that the time complexity and space complexity, when employing the Standard Householder QR algorithm, are $\mathcal{O}(N^4)$ and $\mathcal{O}(N^3)$, respectively. As explained later on, this high computational cost is due to the fact that the sparse structure of $\mathbf{H}$ is destroyed by the Householder transformations.

To proceed, we notice that the computational complexity of the Standard Householder QR algorithm is contributed from two sources, (1) column pivoting in association with $\mathbf{\Pi}_\ell$ and (2) Householder update in association with $\mathbf{Q}_\ell,\ell=1,\ldots,n_\mathbf{H}$. Specifically, we decompose Algorithm 1 into two categories,

where Lines 1-4, 6, 10-13 corresponds to column pivoting, while Householder update consists of Lines 7-9. We examine the overhead associated with each individual source separately. Thus, the total complexity of the algorithm is *dominated* by the higher overhead from these two sources.

By exploiting the sparse structure of $\mathbf{H}$, we can show the time complexity and space complexity associated with column pivoting in the Standard Householder QR are $\mathcal{O}(N^2)$ and $\mathcal{O}(N)$, respectively. More details are provided in Appendix D.

Remark 3: As will become clear in following sections, the computational overhead associated with column pivoting is at least one order of magnitude less than that of Householder update. Furthermore, in practice, column pivoting can be efficiently implemented using pointers, without physically swapping data in memory. In what follows, without loss of generality, we assume $\mathbf{\Pi}_\ell = \mathbf{I}_{n_\mathbf{H}}, \ell = 1, \ldots, l$, hence $\mathbf{\Pi_H} = \mathbf{I}_{n_\mathbf{H}}$ and $[\pi_1 \ \ldots \ \pi_{n_\mathbf{H}}] = [1 \ \ldots \ n_\mathbf{H}]$.

In what follows, we address the computational complexity associated with Householder update in the Standard Householder QR (see Lines 7-9 in Algorithm 1). For the purpose of presentation, we introduce a matrix $\mathbf{\Lambda}$ of dimensions $m_\mathbf{\Lambda} \times n_\mathbf{\Lambda}$, with $m_\mathbf{\Lambda} = (N-1)N/2 = m_\mathbf{H}/4$ and $n_\mathbf{\Lambda} = N = n_\mathbf{H}/3$ [see (24)]. Specifically, $\mathbf{\Lambda}$ is constructed by firstly removing from $\mathbf{H}$ the measurement Jacobian matrices $\mathbf{H}^{i,j}$ for all $1 \leq j < i \leq N$, which correspond to all even block rows of $\mathbf{H}$ in (6). Secondly, every pair of matrices $\mathbf{\Psi}^{i,j}$ and $\mathbf{\Upsilon}^{i,j}$ ($i < j$) of dimensions 2×3 in $\mathbf{H}$ [see (6)] are replaced by *nonzero scalars* $\psi^{i,j}, \upsilon^{i,j}$ in $\mathbf{\Lambda}$ [see (24)]. Since both $m_\mathbf{H}$ and $m_\mathbf{\Lambda}$ are quadratic in N, as well as $n_\mathbf{H}$ and $n_\mathbf{\Lambda}$ both linear in N, and most importantly, each matrix $\mathbf{\Psi}^{i,j}, \mathbf{\Upsilon}^{i,j}$ has dimensions *independent* of the size of robot team N, the computational complexity of decomposing $\mathbf{\Lambda}$ in (24) is in the *same order* as of decomposing $\mathbf{H}$ in (6). Thus, for simplicity, we present the computational complexity analysis of factorizing $\mathbf{\Lambda}$.

$$\mathbf{\Lambda} = \begin{bmatrix} \psi^{1,2} & \upsilon^{1,2} & & & & & \\ \psi^{1,3} & & \upsilon^{1,3} & & & & \\ \vdots & & & \ddots & & & \\ \psi^{1,N} & & & & & & \upsilon^{1,N} \\ & \psi^{2,3} & \upsilon^{2,3} & & & & \\ & \psi^{2,4} & & \upsilon^{2,4} & & & \\ & \vdots & & & \ddots & & \\ & \psi^{2,N} & & & & & \upsilon^{2,N} \\ & & \vdots & \cdots & & & \vdots \\ & & & & \psi^{N-1,N} & & \upsilon^{N-1,N} \end{bmatrix}. \quad (24)$$

To facilitate the discussion afterwards, the following proposition proves to be useful, which shows the time complexity of Householder update (see Algorithm 1, Lines 7-9) at the ℓth iteration is *proportional* to the number of non-zeros of $\mathbf{u}$ representing the first column of $\mathbf{\Lambda}_{2,2}^{(\ell-1)}$, and is *independent* of the structure of $\mathbf{\Lambda}^{(\ell-1)}$.

Proposition 3: Let $\mathbf{u}$ represents the first column of $\mathbf{\Lambda}_{2,2}^{(\ell-1)}$ and suppose $\mathbf{nnz}(\mathbf{u}) = \tau_{\ell-1}$, then the time complexity of implementing Lines 7-9 in Algorithm 1 at the ℓth iteration of the Standard Householder QR is $\mathcal{O}((n_\mathbf{\Lambda} - \ell)\tau_{\ell-1})$.

Proof: The proof is provided in Appendix E. ■

From Proposition 3, the overall time complexity associated with Householder update is in the order of $\sum_{\ell=1}^{n_\mathbf{\Lambda}} (n_\mathbf{\Lambda} - \ell)\tau_{\ell-1}$. In what follows, we firstly seek the expressions of $\tau_{\ell-1}, \ell = 1, \ldots, n_\mathbf{\Lambda}$, to determine the time complexity associated with Householder update (see Section IV-C1), followed by examining the space complexity in Section IV-C2.

1) Time Complexity: In this section, we focus on analyzing the number of arithmetic operations for Householder update per iteration (see Algorithm 1, Lines 7-9). Additionally, we examine the evolution of the sparsity pattern of $\mathbf{\Lambda}$.

Initially, $\mathbf{\Lambda}^{(0)} = \mathbf{\Lambda}$ has a sparse structure [see (24)]. More specifically, *each* column of $\mathbf{\Lambda}$ has $\bar{\tau}_0 = \tau_0 = N-1$ non-zeros, and $\mathbf{nnz}(\mathbf{\Lambda}) = (N-1)N$.

• At the 1st iteration

Since $\mathbf{u}$, the first column of $\mathbf{\Lambda}_{2,2}^{(0)} = \mathbf{\Lambda}^{(0)}$, satisfies $\mathbf{nnz}(\mathbf{u}) = \tau_0 = N-1$, based on Proposition 3, the time complexity of Householder update at the 1st iteration is $\mathcal{O}((N-1)^2)$.

Now let us examine the structure of $\mathbf{\Lambda}^{(1)}$, after the 1st Householder matrix $\mathbf{Q}_1$ (generated by the Householder vector $\mathbf{v}$) has been applied to $\mathbf{\Lambda}^{(0)}$. Due to the nonzero elements of $\mathbf{u}$ (i.e., $\psi^{1,j}, j = 2, \ldots, N$), the first $N-1$ elements of all except for the first column of $\mathbf{\Lambda}^{(1)}$ become nonzero [see (25)]. Therefore the Householder update *does not preserve* the original sparsity of $\mathbf{\Lambda}^{(0)}$, and the resulting matrix $\mathbf{\Lambda}^{(1)} = \mathbf{Q}_1\mathbf{\Lambda}^{(0)}$ has the following dense structure[3]

$$\mathbf{\Lambda}^{(1)} = \begin{bmatrix} \times & \times & \boxtimes & \boxtimes & \cdots & \boxtimes \\ & \boxtimes & \times & \boxtimes & \cdots & \boxtimes \\ & \vdots & \vdots & \vdots & \ddots & \vdots \\ & \boxtimes & \boxtimes & \boxtimes & \cdots & \times \\ & \psi^{2,3} & \upsilon^{2,3} & & & \\ & \psi^{2,4} & & \upsilon^{2,4} & & \\ & \vdots & & & \ddots & \\ & \psi^{2,N} & & & & \upsilon^{2,N} \\ & & \vdots & \cdots & & \vdots \\ & & & & \psi^{N-1,N} & \upsilon^{N-1,N} \end{bmatrix}. \quad (25)$$

Furthermore, in contrast to $N-1$ nonzero elements per column of $\mathbf{\Lambda}^{(0)}$ and $\mathbf{nnz}(\mathbf{\Lambda}^{(0)}) = (N-1)N$, we observe that all but the first column of $\mathbf{\Lambda}^{(1)}$ has $\bar{\tau}_1 = \bar{\tau}_0 + N - 2 = \sum_{i=1}^{2}(N-i)$ nonzero elements [see (25)], as well as $\mathbf{nnz}(\mathbf{\Lambda}^{(1)}) = (N-1)\bar{\tau}_1 + 1 = (N-1)\sum_{i=1}^{2}(N-i) + 1$.

• At the 2nd iteration

At the 2nd iteration, the vector $\mathbf{u}$ (the first column of $\mathbf{\Lambda}_{2,2}^{(1)}$) has $\mathbf{nnz}(\mathbf{u}) = \tau_1 = \bar{\tau}_1 - 1 = \sum_{i=1}^{2}(N-i) - 1$. From Proposition 3, the time complexity of Householder update at the 2nd iteration is $\mathcal{O}((N-2)[\sum_{i=1}^{2}(N-i) - 1])$.

After the 2nd Householder matrix $\mathbf{Q}_2$ has been applied, the resulting matrix $\mathbf{\Lambda}^{(2)} = \mathbf{Q}_2\mathbf{\Lambda}^{(1)}$ has a dense structure shown in (26). Similarly, $\mathbf{Q}_2$ further *destroys* the sparse pattern of $\mathbf{\Lambda}^{(1)}$ by introducing extra nonzero elements through $\psi^{2,j}, j = 3, \ldots, N$, to the corresponding rows of the submatrix $\left[(\mathbf{\Lambda}_{1,2}^{(1)})^\mathrm{T} \ (\mathbf{\Lambda}_{2,2}^{(1)})^\mathrm{T}\right]^\mathrm{T}$, which represents the last $N-2$ columns of $\mathbf{\Lambda}^{(1)}$ [see (18)]. Thus, all but the first two columns

[3]We use " $\times$ " to denote a nonzero element, whereas " $\boxtimes$ " highlights a nonzero filling.

of $\mathbf{\Lambda}^{(2)}$ has $\bar{\tau}_2 = \bar{\tau}_1 + N - 3 = \sum_{i=1}^{3}(N-i)$ nonzero elements, and $\mathbf{nnz}(\mathbf{\Lambda}^{(2)}) = (N-2)\bar{\tau}_2 + \sum_{i=1}^{2} i = (N-2)\sum_{i=1}^{3}(N-i) + \sum_{i=1}^{2} i$.

$$\mathbf{\Lambda}^{(2)} = \begin{bmatrix} \times & \times & \times & \times & \cdots & \times \\ & \times & \times & \times & \cdots & \times \\ & & \vdots & \vdots & \ddots & \vdots \\ & & \times & \times & \cdots & \times \\ & & \times & \boxtimes & \cdots & \boxtimes \\ & & \boxtimes & \times & \cdots & \boxtimes \\ & & \vdots & \vdots & \ddots & \vdots \\ & & \boxtimes & \boxtimes & \cdots & \times \\ & & \psi^{3,4} & \upsilon^{3,4} & & \\ & & \vdots & \cdots & & \vdots \\ & & & & \psi^{N-1,N} & \upsilon^{N-1,N} \end{bmatrix} . \quad (26)$$

• At the ℓth iteration

Similar to the above analysis, we can draw conclusions by induction. More specifically, after $\ell - 1$ Householder updates have been applied to $\mathbf{\Lambda}^{(0)}$, *every* jth column ($j = \ell, \ldots, N$) of $\mathbf{\Lambda}_{2,2}^{(\ell-1)}$ has $\tau_{\ell-1} = \bar{\tau}_{\ell-1} - \ell + 1 = \sum_{i=1}^{\ell}(N-i) - \ell + 1$ nonzero elements, resulting in the time complexity of Householder update at the ℓth iteration of $\mathcal{O}((N-\ell)[\sum_{i=1}^{\ell}(N-i) - \ell + 1])$.

Furthermore, compared to $\mathbf{\Lambda}^{(\ell-1)}$, extra non-zeros are introduced to those rows of the sub-matrix $\left[(\mathbf{\Lambda}_{1,2}^{(\ell-1)})^{\mathrm{T}}\ (\mathbf{\Lambda}_{2,2}^{(\ell-1)})^{\mathrm{T}}\right]^{\mathrm{T}}$ (corresponding to the last $N-\ell$ columns of $\mathbf{\Lambda}^{(\ell-1)}$ [see (18)]) whose indices match the linear indices of the non-zeros $\psi^{\ell,j}, j = \ell+1, \ldots, N$. Hence, the number of nonzero elements for *each* of the last $N - \ell$ columns of $\mathbf{\Lambda}^{(\ell)}$ is $\bar{\tau}_\ell = \bar{\tau}_{\ell-1} + N - \ell - 1 = \sum_{i=1}^{\ell+1}(N-i)$, and $\mathbf{nnz}(\mathbf{\Lambda}^{(\ell)}) = (N-\ell)\bar{\tau}_\ell + \sum_{i=1}^{\ell} i = (N-\ell)\sum_{i=1}^{\ell+1}(N-i) + \sum_{i=1}^{\ell} i$.

• Summary of time complexity

In summary, we conclude that the overall time complexity of QR decomposition of $\mathbf{\Lambda}$ using the Standard Householder QR algorithm is in the order of

$$\sum_{\ell=1}^{N}\left[(N-\ell)\Big(\sum_{i=1}^{\ell}(N-i) - \ell + 1\Big)\right] \sim \mathcal{O}(N^4) .$$

2) Space Complexity: To analyze space complexity, we focus on the storage of $\mathbf{\Lambda}^{(\ell)}, \ell = 1, \ldots, N$, which is the most dominant source in terms of memory cell usage. Since $\mathbf{nnz}(\mathbf{\Lambda}^{(\ell)}) = (N-\ell)\sum_{i=1}^{\ell+1}(N-i) + \sum_{i=1}^{\ell} i$, we conclude that the space complexity of QR decomposition of $\mathbf{\Lambda}$ using the Standard Householder QR algorithm is in the order of

$$\max_{1\le\ell\le N}\left[\mathbf{nnz}(\mathbf{\Lambda}^{(\ell)}) = (N-\ell)\sum_{i=1}^{\ell+1}(N-i) + \sum_{i=1}^{\ell} i\right] \sim \mathcal{O}(N^3) .$$

3) Summary of Complexity Analysis: In summary, we have shown that factorizing $\mathbf{H}$ by employing the Standard Householder QR has time complexity $\mathcal{O}(N^4)$ and space complexity $\mathcal{O}(N^3)$. Therefore, we conclude that the overall time complexity and space complexity for the state and covariance updates through standard Householder QR factorization become $\mathcal{O}(N^4)$ and $\mathcal{O}(N^3)$, an order of magnitude increase than that of using (9)-(10). 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 $\mathcal{O}(N^3)$ time complexity and $\mathcal{O}(N^2)$ space complexity. In the next section, we describe the main idea behind the Modified Householder QR algorithm, as well as its complexity analysis.

V. Modified Householder QR Algorithm

The Modified Householder QR is derived from [26], where Kaufman proposed an idea that exploits the sparsity of the original matrix. However, in [26], 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 [26], and term this new algorithm as Modified Householder QR. In what follows, we provide a detailed description of the Modified Householder QR algorithm in Section V-A, and analyze its computational complexity when applied to $\mathbf{H}$ [see (6)] in Section V-B to conclude that the overall time complexity is reduced from $O(N^4)$ to $O(N^3)$, and the space complexity from $O(N^3)$ to $O(N^2)$.

A. Description and Derivation of Modified Householder QR

In this section, we present a general description of the Modified Householder QR and focus on key ideas and derivations to develop the algorithm. For the purpose of generality, we do *not* consider the specific structure of $\mathbf{H}$. Instead, the sparsity of $\mathbf{H}$ [see (6)] will be exploited in Section V-B when we analyze the computational complexity.

To facilitate the description and derivation of the Modified Householder QR algorithm, we adopt the same notation used in Section IV-B. Furthermore, we use $\mathbf{h}_i$ and $\mathbf{h}_i^{(\ell)}, i = 1, \ldots, n_{\mathbf{H}}$, to denote the ith columns of the original matrix $\mathbf{H}$ and the updated matrix $\mathbf{H}^{(\ell)}$ after the ℓth iteration of the Householder QR is processed, respectively, with $m_{\mathbf{H}}$ and $n_{\mathbf{H}}$ representing the number of rows and columns of $\mathbf{H}$.

As mentioned in Section IV-B, the ℓth iteration of the Householder QR is decomposed into two separated stages, column pivoting and Householder update. The Modified Householder QR algorithm performs column pivoting *identically* to that of the Standard Householder QR, with difference in the implementation of Householder update. Similar to the discussion in Section IV-B, due to the minimum computational overhead associated with column pivoting (see Remark 3), in the following description of the Modified Householder QR, we assume $\mathbf{\Pi}_\ell = \mathbf{I}_{n_{\mathbf{H}}}, \ell = 1, \ldots, n_{\mathbf{H}}$, and focus on the modifications of the implementation of Householder update.

From (18), we have (under the assumption $\mathbf{\Pi}_\ell = \mathbf{I}_{n_{\mathbf{H}}}, \forall \ell$)

$$\mathbf{h}_j^{(\ell-1)} = \left(\prod_{i=1}^{\ell-1}\left(\mathbf{I}_{m_{\mathbf{H}}} - \beta_i \mathbf{v}_i \mathbf{v}_i^{\mathrm{T}}\right)\right)\mathbf{h}_j \ , \ j = \ell, \ldots, n_{\mathbf{H}}. \quad (27)$$

Hence $\mathbf{h}_j^{(\ell-1)}$ ($j = \ell, \ldots, n_{\mathbf{H}}$) 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 $1 \leq i \leq \ell-1$. Hence $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}$ $(j = \ell, \ldots, n_{\mathbf{H}})$ can be expressed as a linear combination of $(\mathbf{h}_j)_{-(\ell-1)}$ and $\{(\mathbf{h}_i^{(i-1)})_{-(\ell-1)}, i = 1, \ldots, \ell-1\}$. 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 $\{\mathbf{v}_\eta, \eta = 1, \ldots, i-1\}$, and recalling $(\mathbf{v}_\eta)_{-(\ell-1)} = (\mathbf{h}_\eta^{(\eta-1)})_{-(\ell-1)}$ for $1 \leq \eta \leq i-1$, hence $(\mathbf{h}_i^{(i-1)})_{-(\ell-1)}$ $(i = 2, \ldots, \ell-1)$ 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 (thus, $(\mathbf{h}_1^{(0)})_{-(\ell-1)} = (\mathbf{h}_1)_{-(\ell-1)}$). Therefore, we conclude by *recursion* that each vector $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}, j = \ell, \ldots, n_{\mathbf{H}}$, can be expressed as a *linear combination* of $(\mathbf{h}_j)_{-(\ell-1)}$ and $\{(\mathbf{h}_i)_{-(\ell-1)}, i = 1, \ldots, \ell-1\}$:

$$(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)} = (\mathbf{h}_j)_{-(\ell-1)} - \sum_{i=1}^{\ell-1} (\mathbf{h}_i)_{-(\ell-1)} \ \gamma_{i,j}^{(\ell-1)} \ , \qquad (28)$$

where the coefficients $\gamma_{i,j}^{(\ell-1)}; i = 1, \ldots, \ell-1, j = \ell, \ldots, n_{\mathbf{H}}$, need to be sought at each iteration. In what follows, we will provide a formula [see (39)] that updates $\gamma_{i,j}^{(\ell-1)}$ recursively.

Notice that the matrix $\mathbf{H}_{2,2}^{(\ell-1)}$ [see (18)] comprises all the column vectors $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}, j = \ell, \ldots, n_{\mathbf{H}}$. Hence, (28) can be summarized into a compact matrix form

$$\mathbf{H}_{2,2}^{(\ell-1)} = \mathbf{H}_{-(\ell-1)} \begin{bmatrix} \mathbf{\Gamma}^{(\ell-1)} \\ \mathbf{I}_{n_{\mathbf{H}}-\ell+1} \end{bmatrix} \ , \qquad (29)$$

where $\mathbf{H}_{-(\ell-1)} = [(\mathbf{h}_1)_{-(\ell-1)} \ \ldots \ (\mathbf{h}_{n_{\mathbf{H}}})_{-(\ell-1)}]$ is the sub-matrix of $\mathbf{H}$ resulting by removing its first $\ell-1$ rows. The $(\ell-1) \times (n_{\mathbf{H}}-\ell+1)$ 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 $\mathbf{\Gamma}^{(\ell-1)}$.

In contrast to the original vector $(\mathbf{h}_j)_{-(\ell-1)}, j = \ell, \ldots, n_{\mathbf{H}}$, which has at most $\mathcal{O}(N)$ non-zero elements (see Remark 1), by following the analysis conducted on $\mathbf{\Lambda}$ in Section IV-C, it can be shown that the vector $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}, j = \ell, \ldots, n_{\mathbf{H}}$, obtained after the $(\ell-1)$th iteration of the Householder QR is processed, has $\mathbf{nnz}((\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}) \sim \mathcal{O}(\ell N - \frac{\ell^2}{2})$, which results in $\mathcal{O}(N^4)$ time complexity and $\mathcal{O}(N^3)$ space complexity when employing the Standard Householder QR algorithm (see Section IV-C). 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 is that the *explicit form* of $\mathbf{H}_{2,2}^{(\ell-1)}$ [see (29)] (or equivalently, the explicit form of the vectors $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}, j = \ell, \ldots, n_{\mathbf{H}}$ [see (28)]) is *not calculated*. Instead, $\mathbf{H}_{2,2}^{(\ell-1)}$ is *stored and represented implicitly* by the coefficient matrix $\mathbf{\Gamma}^{(\ell-1)}$ [see (29)].

Next, we address two key issues in the Modified Householder QR. Firstly, computing the matrices $\mathbf{H}_{1,1}^{(\ell)}, \mathbf{H}_{1,2}^{(\ell)}$ in (20); Secondly, deriving the recursive rule for obtaining $\mathbf{\Gamma}^{(\ell)}$ from $\mathbf{\Gamma}^{(\ell-1)}$. Equivalently, we seek the expression of $\mathbf{\Gamma}^{(\ell)}$ such that

$$\mathbf{H}_{2,2}^{(\ell)} = \mathbf{H}_{-\ell} \begin{bmatrix} \mathbf{\Gamma}^{(\ell)} \\ \mathbf{I}_{n_{\mathbf{H}}-\ell} \end{bmatrix} \ , \qquad (30)$$

where $\mathbf{H}_{-\ell} = [(\mathbf{h}_1)_{-\ell} \ \ldots \ (\mathbf{h}_{n_{\mathbf{H}}})_{-\ell}]$ 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 (29),

$$\mathbf{u} = (\mathbf{h}_\ell^{(\ell-1)})_{-(\ell-1)} = (\mathbf{h}_\ell)_{-(\ell-1)} + \mathbf{H}_{-(\ell-1)}^1 \ \boldsymbol{\gamma}_\ell^{(\ell-1)} \ , \qquad (31)$$

where $\mathbf{H}_{-(\ell-1)}^1 = [(\mathbf{h}_1)_{-(\ell-1)} \ \ldots \ (\mathbf{h}_{\ell-1})_{-(\ell-1)}]$ 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 $(\mathbf{h}_j^{(\ell-1)})_{-(\ell-1)}, j = \ell+1, \ldots, n_{\mathbf{H}}$, are *not explicitly* computed. Instead, they are represented *implicitly* by $\overline{\mathbf{\Gamma}}^{(\ell-1)}$.

Once the *explicit* form of $\mathbf{u}$, computed by (31), is known, and c^* is retrieved from column pivoting, the Householder vector $\mathbf{v}_\ell$ (or equivalently $\mathbf{v}$), as well as β_ℓ and r_ℓ, are readily available through (19) and (21). Notice that computing $\mathbf{v}$ from $\mathbf{u}$ only requires updating the first element of $\mathbf{u}$ [see Remark 2].

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 iteration of the Householder QR. Hence the only unknown in $\mathbf{H}_{1,1}^{(\ell)}$ [see (20)] is the scalar r_ℓ, which can be calculated from c^* and u_1 in a fixed number of arithmetic operations [see (21)].

2) Computing $\mathbf{H}_{1,2}^{(\ell)}$*:* Since $\overline{\mathbf{H}}_{1,2}^{(\ell-1)}$ is known after the $(\ell-1)$th iteration of the Householder QR, updating $\mathbf{H}_{1,2}^{(\ell)}$ is equivalent to calculating the vector $\mathbf{s}$ from (22), 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 $\bar{\mathbf{s}}^{\mathrm{T}}$ corresponds to the first row of $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$ and based on (29), we can rewrite $\overline{\mathbf{H}}_{2,2}^{(\ell-1)}$ and $\bar{\mathbf{s}}$ as follows

$$\overline{\mathbf{H}}_{2,2}^{(\ell-1)} = \mathbf{H}_{-(\ell-1)}^2 + \mathbf{H}_{-(\ell-1)}^1 \ \overline{\mathbf{\Gamma}}^{(\ell-1)} \ ; \qquad (32)$$

$$\bar{\mathbf{s}} = \boldsymbol{\rho}_2 + \big(\overline{\mathbf{\Gamma}}^{(\ell-1)}\big)^{\mathrm{T}} \boldsymbol{\rho}_1 \ ; \qquad (33)$$

where $\mathbf{H}_{-(\ell-1)}^2 = [(\mathbf{h}_{\ell+1})_{-(\ell-1)} \ \ldots \ (\mathbf{h}_{n_{\mathbf{H}}})_{-(\ell-1)}]$ comprises the last $n_{\mathbf{H}} - \ell$ columns of $\mathbf{H}_{-(\ell-1)}$, and $\boldsymbol{\rho}_1^{\mathrm{T}}$ and $\boldsymbol{\rho}_2^{\mathrm{T}}$ are the first rows of $\mathbf{H}_{-(\ell-1)}^1$ and $\mathbf{H}_{-(\ell-1)}^2$, respectively. From (32), we compute the vector

$$\boldsymbol{\delta} = \beta_\ell \big(\overline{\mathbf{H}}_{2,2}^{(\ell-1)}\big)^{\mathrm{T}} \mathbf{v} = \boldsymbol{\delta}_2 + \big(\overline{\mathbf{\Gamma}}^{(\ell-1)}\big)^{\mathrm{T}} \boldsymbol{\delta}_1 \ , \qquad (34)$$

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

Substituting (33) and (34) in (22), we arrive at the following update equation for $\mathbf{s}$ (or equivalently, $\mathbf{H}_{1,2}^{(\ell)}$):

$$\mathbf{s} = \big[\boldsymbol{\rho}_2 - v_1 \boldsymbol{\delta}_2\big] + \big(\overline{\mathbf{\Gamma}}^{(\ell-1)}\big)^{\mathrm{T}} \big[\boldsymbol{\rho}_1 - v_1 \boldsymbol{\delta}_1\big] \ . \qquad (35)$$

3) Computing $\mathbf{\Gamma}^{(\ell)}$*:* To determine $\mathbf{\Gamma}^{(\ell)}$, we begin with (30) and (23). 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 (32)],

$$\underline{\mathbf{H}}_{2,2}^{(\ell-1)} = \mathbf{H}_{-\ell}^2 + \mathbf{H}_{-\ell}^1 \ \overline{\mathbf{\Gamma}}^{(\ell-1)} \ , \qquad (36)$$

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

Next we explore the property $\mathbf{v}_{-1} = \mathbf{u}_{-1}$ [see Remark 2]. In particular, based on (31), we have

$$\mathbf{v}_{-1} = \mathbf{u}_{-1} = (\mathbf{h}_\ell)_{-\ell} + \mathbf{H}^1_{-\ell}\boldsymbol{\gamma}_\ell^{(\ell-1)} \ . \tag{37}$$

Finally, we substitute (36), (37), and (34) into (23), and notice that $\mathbf{H}_{-\ell} = [\mathbf{H}^1_{-\ell} \ \ (\mathbf{h}_\ell)_{-\ell} \ \ \mathbf{H}^2_{-\ell}]$, to arrive at

$$\begin{aligned}\mathbf{H}^{(\ell)}_{2,2} &= \mathbf{H}^1_{-\ell}\big(\overline{\boldsymbol{\Gamma}}^{(\ell-1)} - \boldsymbol{\gamma}_\ell^{(\ell-1)}\boldsymbol{\delta}^{\mathrm{T}}\big) - (\mathbf{h}_\ell)_{-\ell}\,\boldsymbol{\delta}^{\mathrm{T}} + \mathbf{H}^2_{-\ell} \\ &= \mathbf{H}_{-\ell}\begin{bmatrix}\overline{\boldsymbol{\Gamma}}^{(\ell-1)} - \boldsymbol{\gamma}_\ell^{(\ell-1)}\boldsymbol{\delta}^{\mathrm{T}} \\ -\boldsymbol{\delta}^{\mathrm{T}} \\ \mathbf{I}_{n_{\mathbf{H}}-\ell}\end{bmatrix} \ . \end{aligned} \tag{38}$$

Comparing (38) and (30), we immediately obtain the expression of the $\ell \times (n_{\mathbf{H}} - \ell)$ matrix $\boldsymbol{\Gamma}^{(\ell)}$:

$$\boldsymbol{\Gamma}^{(\ell)} = \begin{bmatrix}\overline{\boldsymbol{\Gamma}}^{(\ell-1)} - \boldsymbol{\gamma}_\ell^{(\ell-1)}\boldsymbol{\delta}^{\mathrm{T}} \\ -\boldsymbol{\delta}^{\mathrm{T}}\end{bmatrix} \ . \tag{39}$$

Note that the upper part of $\boldsymbol{\Gamma}^{(\ell)}$ is a *rank-one modification* of the *existing matrix* $\overline{\boldsymbol{\Gamma}}^{(\ell-1)}$, which has a relatively low time complexity. Furthermore, (39) *affirms* that every vector $(\mathbf{h}_j^{(\ell)})_{-\ell}, j = \ell+1, \ldots, n_{\mathbf{H}}$, is a linear combination of $(\mathbf{h}_j)_{-\ell}$ and $\{(\mathbf{h}_i)_{-\ell}, i = 1, \ldots, \ell\}$.

In summary, we outline the algorithmic flow chart of Modified Householder QR in Algorithm 2. Note that column pivoting implemented by the Modified Householder QR is *identical* to that of the Standard Householder QR algorithm.

Algorithm 2 Modified Householder QR

Require: $\mathbf{H}^{(0)} = \mathbf{H}$ of dimensions $m_{\mathbf{H}} \times n_{\mathbf{H}}$ ($m_{\mathbf{H}} \geq n_{\mathbf{H}}$).
Ensure: $\mathbf{R}_{\mathbf{H}}$ and $\boldsymbol{\Pi}_{\mathbf{H}}$ of dimensions $n_{\mathbf{H}} \times n_{\mathbf{H}}$.
1: **for** $i = 1$ **to** $n_{\mathbf{H}}$, **do**
2: Initialize $\pi_i = i$, and compute c_i, the squared norm of the ith column of $\mathbf{H}^{(0)}_{2,2} = \mathbf{H}^{(0)}$. Additionally, initialize $\boldsymbol{\Gamma}^{(0)}$ as an empty matrix.
3: **end for**
4: Set $\ell = 1$, determine $\ell^* = \arg\max_j\{c_j, \ j = \ell, \ldots, n_{\mathbf{H}}\}$ and $c^* = c_{\ell^*}$.
5: **while** $c^* > \epsilon$ **do**
6: Swap the ℓth and ℓ^*th columns of $\mathbf{H}^{(\ell-1)}$, as well as the 1st and $(\ell^* - \ell + 1)$th columns of $\boldsymbol{\Gamma}^{(\ell-1)}$, exchange c_ℓ and c_{ℓ^*}, and then interchange π_ℓ with π_{ℓ^*}.
7: Calculate $\mathbf{u}$ from (31).
8: Compute $\mathbf{v}, \beta_\ell, r_\ell$ and update the sub-matrix $\mathbf{H}^{(\ell)}_{1,1}$.
9: Determine $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2$, and $\boldsymbol{\delta}$ using (34).
10: Calculate $\mathbf{s} = [s_1 \ \ldots \ s_{n_{\mathbf{H}}-\ell}]^{\mathrm{T}}$ from (35) and update the sub-matrix $\mathbf{H}^{(\ell)}_{1,2}$.
11: Update $\boldsymbol{\Gamma}^{(\ell)}$ based on (39).
12: **for** $j = \ell + 1$ **to** $n_{\mathbf{H}}$, **do**
13: Modify $c_j \Leftarrow c_j - s^2_{j-\ell}$.
14: **end for**
15: Seek $\ell^* = \arg\max_j\{c_j, \ j = \ell+1, \ldots, n_{\mathbf{H}}\}$ and $c^* = c_{\ell^*}$, and then $\ell \Leftarrow \ell + 1$.
16: **end while**
17: **return** $\boldsymbol{\Pi}_{\mathbf{H}} = [\mathbf{e}_{\pi_1} \ \ldots \ \mathbf{e}_{\pi_{n_{\mathbf{H}}}}]$, and $\mathbf{R}_{\mathbf{H}}$ selected as the first $n_{\mathbf{H}}$ rows of $\mathbf{H}^{(l)}$, l being the total number of iterations.

B. Computational Complexity Analysis

In Section V-A, we provide a general description of the Modified Householder QR algorithm which is *applicable* to decompose arbitrary matrix. In this section, we apply the Modified Householder QR to perform QR factorization on $\mathbf{H}$ [see (6)], and analyze the overall computational complexity by taking into account the sparse structure of $\mathbf{H}$.

Since the Modified Householder QR only differs from the Standard Householder QR algorithm in the implementation of Householder update, while column pivoting is performed identically by both algorithms, we conclude that the computational overhead associated with column pivoting in the Modified Householder QR has the time complexity of $\mathcal{O}(N^2)$ and the space complexity of $\mathcal{O}(N)$. Therefore, in what follows, we analyze the computational complexity associated with Householder update in the Modified Householder QR, which corresponds to Lines 7-11 in Algorithm 2. We start by analyzing the time complexity, followed by the space complexity.

1) Time Complexity: In this section, we briefly analyze the time complexity associated with Householder update when applying the Modified Householder QR algorithm to decompose $\mathbf{H}$. We claim that the worst-case time complexity is $\mathcal{O}(N^3)$. To prove it, we will identify the number of flops for *every* line between Lines 7-11 in Algorithm 2, and show that the total number of arithmetic operations required per iteration of the Modified Householder QR is bounded above by $\mathcal{O}(N^2)$. Since the maximum number of iterations is $n_{\mathbf{H}} = 3N$, the overall time complexity associated with Householder update is $\mathcal{O}(N^3)$.

• Computational cost of Line 7:

Recall that $\mathbf{nnz}(\mathbf{h}_j) \sim \mathcal{O}(N), j = 1, \ldots, 3N$ [see Remark 1], hence $\mathbf{nnz}((\mathbf{h}_i)_{-(\ell-1)}) \sim \mathcal{O}(N), i = 1, \ldots, \ell$. Thus, computing $\mathbf{u}$ from (31) requires $\mathcal{O}(\ell N)$ operations. Therefore, the cost of performing Line 7 is bounded above by $\mathcal{O}(N^2)$.

• Computational cost of Line 8:

Once $\mathbf{u}$ becomes available from Line 7, as well as $c^* = \mathbf{u}^{\mathrm{T}}\mathbf{u}$ retrieved from column pivoting, determining the scalars β_ℓ and r_ℓ invokes constant number of arithmetic operations. Furthermore, updating $\mathbf{v}$ from $\mathbf{u}$ only requires the modification of the 1st element of $\mathbf{u}$, which can be achieved in $\mathcal{O}(1)$ process time. In summary, the cost of performing Line 8 is of $\mathcal{O}(1)$.

• Computational cost of Line 9:

Since each column of $\mathbf{H}^1_{-(\ell-1)}$ and $\mathbf{H}^2_{-(\ell-1)}$ has $\mathcal{O}(N)$ nonzeros, 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 (34) involves an $(n_{\mathbf{H}} - \ell) \times (\ell - 1)$ matrix multiplied by an $(\ell - 1) \times 1$ vector and a vector-vector addition of dimension $n_{\mathbf{H}} - \ell$. In summary, the cost of performing Line 9 is of $\mathcal{O}(N^2)$.

• Computational cost of Line 10:

Since (35) involves a $(\ell - 1) \times 1$ vector multiplying with an $(n_{\mathbf{H}} - \ell) \times (\ell - 1)$ matrix and a vector-vector addition of dimension $(n_{\mathbf{H}} - \ell)$, the overall cost of performing Line 10 is bounded above by $\mathcal{O}(N^2)$.

• Computational cost of Line 11:

In (39), the vector $\boldsymbol{\delta}$ [see (34)] is available from Line 9 and *does not need to be recomputed*. Hence, we only need to focus on the upper part of (39), 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 $(n_{\mathbf{H}}-\ell)$, respectively, we conclude that the overall cost of performing Line 11 is of $\mathcal{O}(\ell N)$, which is bounded above by $\mathcal{O}(N^2)$.

In summary, we have shown that the number of arithmetic operations required per iteration of Householder update in the Modified Householder QR algorithm is bounded above by $\mathcal{O}(N^2)$. Therefore the worst-case time complexity of applying the Modified Householder QR algorithm on $\mathbf{H}$ is $\mathcal{O}(N^3)$.

2) Space Complexity: To address space complexity, we focus on the memory requirement for $\mathbf{H}$, $\mathbf{u}$, and $\boldsymbol{\Gamma}^{(\ell)}, \ell = 1, \ldots, n_{\mathbf{H}}$, which are the most dominant sources in terms of memory cell usage. Recall from Remark 1 $\mathbf{nnz}(\mathbf{H}) \sim \mathcal{O}(N^2)$. In addition, notice that the dimension of $\mathbf{u}$ is *at most* $m_{\mathbf{H}} \sim \mathcal{O}(N^2)$. On the other hand, since the dimensions of $\boldsymbol{\Gamma}^{(\ell)}$ are $\ell \times (n_{\mathbf{H}}-\ell)$, thus $\max\{\mathbf{nnz}(\boldsymbol{\Gamma}^{(\ell)}); \ell = 1, \ldots, n_{\mathbf{H}}\} \sim \mathcal{O}(N^2)$. Therefore, we conclude that the space complexity of QR decomposition of $\mathbf{H}$ using the Modified Householder QR algorithm is in the order of $\mathcal{O}(N^2)$.

3) Summary of Complexity Analysis: In summary, we have shown that factorizing $\mathbf{H}$ by employing the Modified Householder QR algorithm has time complexity of $\mathcal{O}(N^3)$ and space complexity of $\mathcal{O}(N^2)$. Therefore, we conclude that the overall time complexity and space complexity for the state and covariance updates through the Modified Householder QR factorization are $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$, respectively, which has *one order of magnitude decrease* than that of using the Standard Householder QR. Additionally, it achieves the same computational complexity as of using (9)-(10).

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 (6)]. In particular, suppose that the dimensions of $\mathbf{H}$ are $m \times n$ $(m \geq n)$, and denote $\tau = \max(\mathbf{nnz}(\mathbf{h}_1), \ldots, \mathbf{nnz}(\mathbf{h}_n))$. Following the analysis conducted in Sections V-B1 and V-B2, it can be shown that the space complexity of the Modified Householder QR algorithm is of $\mathcal{O}(\max\{\min\{\tau n, m\}, n^2\})$, and its associated time complexity is of $\mathcal{O}(\max\{\tau n^2, n^3\})$, in contrast to $\mathcal{O}(mn^2)$ of the Standard Householder QR algorithm [24].

VI. Simulation Results

In the previous section, we have shown that the worst-case time complexity and space complexity of the Modified Householder QR algorithm, when applied to the sparse measurement Jacobian matrix $\mathbf{H}$ in CL [see (6)], is of $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$, respectively. 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 generated the robots' poses and assumed 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 $2(N-1)N \times 3N$.

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 [see (11)] when employing the Modified Householder QR algorithm (MH-QR). The average running times are summarized in Table II, as well as in Figure 1. Furthermore, we compared our results with the CPU running time when employing SuiteSparseQR (SS-QR) [27], [28], the current state-of-the-art QR decomposition package for sparse matrices, which is an implementation of the *multi-frontal sparse* QR *factorization* algorithm. In addition, SuiteSparseQR has been integrated into MATLAB® (version 7.9 and later), and can be invoked through MATLAB built-in function "**qr**" [29]. All simulations were run under MATLAB 7.12 on a Linux (kernel 2.6.32) desktop computer with a 2.66 GHz Intel Core-i5 Quadcore CPU and 4 GB of RAM.

TABLE I
CPU RUNTIME (SEC)

N	SS-QR	MH-QR
6	7.5378×10^{-5}	4.9683×10^{-5}
11	2.0238×10^{-4}	2.1658×10^{-4}
16	6.3311×10^{-4}	6.4813×10^{-4}
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}
76	1.1370×10^{-1}	7.5767×10^{-2}
101	3.2935×10^{-1}	1.8164×10^{-1}
151	1.4685×10^{0}	6.3158×10^{-1}
201	4.3048×10^{0}	1.5436×10^{0}
251	9.9297×10^{0}	3.5646×10^{0}
301	1.9804×10^{1}	6.7462×10^{0}
351	3.6083×10^{1}	1.1572×10^{1}
401	5.7972×10^{1}	1.8523×10^{1}
451	1.9673×10^{2}	2.7400×10^{1}
501	1.5869×10^{3}	3.8437×10^{1}
551	N/A	5.3501×10^{1}
601	N/A	7.0800×10^{1}
651	N/A	9.3076×10^{1}
701	N/A	1.1808×10^{2}
751	N/A	1.4926×10^{2}
801	N/A	1.8626×10^{2}
851	N/A	2.2381×10^{2}
901	N/A	2.6653×10^{2}
951	N/A	3.1863×10^{2}
1001	N/A	3.7570×10^{2}

The results presented in Table I and Figure 1 illustrate that when the number of robots is small ($N \leq 26$), both SS-QR and MH-QR achieve indistinguishable performances, with SS-QR slightly faster as compared to MH-QR. However, as N increases ($N \geq 26$), MH-QR significantly outperforms SS-QR. Additionally, we were unable to perform QR decomposition using SS-QR when $N \geq 501$, due to memory shortage. On the other hand, MH-QR is applicable even when the number of robots increases to 1001, and it successfully performs QR factorization on $\mathbf{H}$, whose dimensions are 2 million by 3 thousand, in about 375 seconds. Additionally, after performing

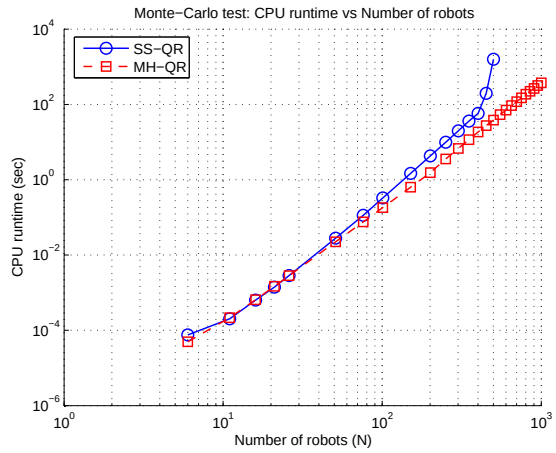


Fig. 1. [Monte Carlo simulations] Average CPU runtime of QR decomposition of $\mathbf{H}$ in 120 trials. Comparison between SuiteSparseQR (SS-QR) and Modified Householder QR (MH-QR).

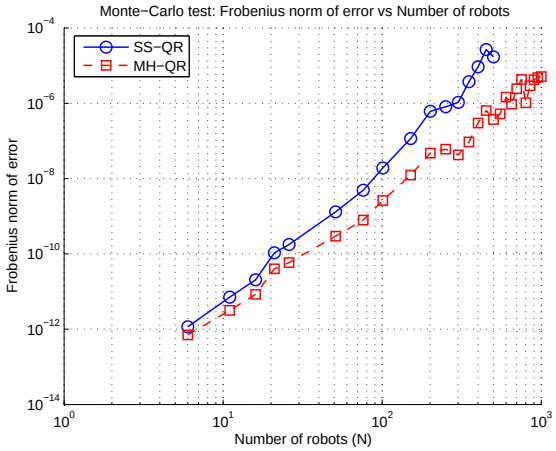


Fig. 2. [Monte Carlo simulations] Average Frobenius norm of $\mathbf{H}^{\mathrm{T}}\mathbf{H}-\mathbf{R}^{\mathrm{T}}\mathbf{R}$ in 120 trials. Comparison between SuiteSparseQR (SS-QR) and Modified Householder QR (MH-QR).

linear regression on the *logarithm* of the CPU running time and the number of robots N, we have determined that the logarithm of the average time required by each of these algorithms is

$$\log(t_{\text{SS-QR}}) = 3.6067 \times \log(N) - 17.2798$$
$$\log(t_{\text{MH-QR}}) = 3.1802 \times \log(N) - 16.1481\ . \qquad (40)$$

Both Figure 1 and (40) confirm that the time complexity of MH-QR is *cubic* in N. Finally, we should note that the main reason for the linear coefficient in (40) (i.e., 3.1802) deviating from its ideal value 3 is because as N increases, a significant portion of CPU resources is devoted to memory accessing, reading, and writing. One of our future research directions is to optimize the implementation of MH-QR and improve its performance in terms of CPU running time.

Furthermore, we have examined the accuracy of MH-QR. In particular, we computed the Frobenius norm of $\mathbf{H}^{\mathrm{T}}\mathbf{H} - \mathbf{R}_{\mathbf{H}}^{\mathrm{T}}\mathbf{R}_{\mathbf{H}}$, which is 0 in the ideal case. We have compared the Frobenius norms of SS-QR and MH-QR, averaged over 120 simulations for each N. As evident from Table II and Figure 2, MH-QR attains higher numerical accuracy than SS-QR, which is attributed to column pivoting implemented in MH-QR.

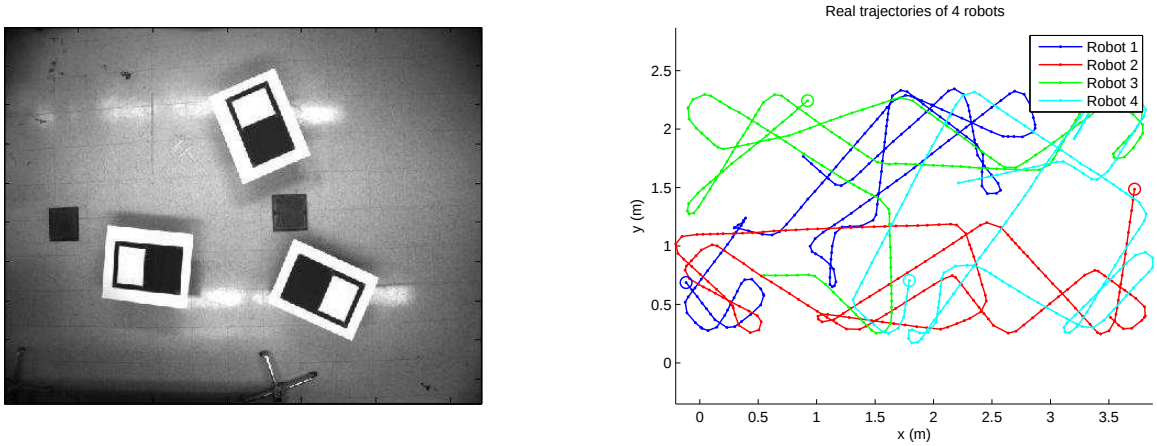
TABLE II
$\left\|\mathbf{H}^{\mathrm{T}}\mathbf{H} - \mathbf{R}_{\mathbf{H}}^{\mathrm{T}}\mathbf{R}_{\mathbf{H}}\right\|_F$

N	SS-QR	MH-QR
6	1.1529×10^{-12}	7.0749×10^{-13}
11	7.1060×10^{-12}	3.1729×10^{-12}
16	2.0568×10^{-11}	8.4163×10^{-12}
21	1.0719×10^{-10}	4.0289×10^{-11}
26	1.7801×10^{-10}	5.8771×10^{-11}
51	1.3050×10^{-9}	2.9588×10^{-10}
76	4.9221×10^{-9}	7.9515×10^{-10}
101	1.9087×10^{-8}	2.6235×10^{-9}
151	1.1631×10^{-7}	1.2418×10^{-8}
201	6.1207×10^{-7}	4.7511×10^{-8}
251	8.0832×10^{-7}	6.0513×10^{-8}
301	1.0625×10^{-6}	4.2499×10^{-8}
351	3.7326×10^{-6}	9.3947×10^{-8}
401	9.3274×10^{-6}	3.0049×10^{-7}
451	2.6734×10^{-5}	6.4118×10^{-7}
501	1.7036×10^{-5}	3.7327×10^{-7}
551	N/A	5.2642×10^{-7}
601	N/A	1.4607×10^{-6}
651	N/A	9.4386×10^{-7}
701	N/A	2.4464×10^{-6}
751	N/A	4.3013×10^{-6}
801	N/A	1.0321×10^{-6}
851	N/A	2.9335×10^{-6}
901	N/A	4.2254×10^{-6}
951	N/A	4.8307×10^{-6}
1001	N/A	5.1291×10^{-6}

VII. Experimental Results

We further validate the performance of our proposed Modified Householder QR algorithm through experiments. Our experimental setup is shown in Figure 3(a), where a team of four Pioneer II robots are deployed in a rectangular region of size approximately 4.5 m $\times$ 2.5 m. A calibrated overhead camera is employed to provide ground truth for evaluating the estimator's performance. For this purpose, rectangular boards with specific patterns (see Figure 3(a)) are mounted on top of the Pioneers, and the pose (position and orientation) of each robot, with respect to a global frame of reference, is computed from the captured images. These measurements are corrupted by zero-mean noise with standard deviation of 0.0087 rad for orientation and 0.01 m along each axis for position. The real trajectories of four robots are shown in Figure 3(b). For the sake of clarity, only partial trajectories, corresponding to the first 200 time steps, are plotted in Figure 3(b).

In the experiment, each robot moves at approximately constant speed of 0.1 m/sec, while avoiding collisions between each other and the boundary of the rectangular arena. The robots record linear and acceleration velocities at a frequency



(a) (b)

Fig. 3. [Experimental setup] (a) Four Pioneer robots perform CL in a rectangular region of size approximately 4.5 m $\times$ 2.5 m, each with a pattern board attached on its top. (b) Real trajectories of four robots. For clarity, only partial trajectories, corresponding to the first 200 time steps, are plotted. The robots' starting points are marked by $\circ$.

of 1 Hz. These odometry measurements are corrupted by zero-mean process noise. Due to manufacture imperfections, the accuracy of the odometry measurements is *not identical* for all four robots. In particular, the noise in the rotational velocity measurements follows Gaussian distribution with standard deviation of 0.0239 rad/sec, 0.095 rad/sec, 0.0121 rad/sec, 0.094 rad/sec for robots $1, 2, 3, 4$ respectively. Similarly, the linear velocity measurement noise is well modeled as Gaussian noise with standard deviation of 0.0037 m/sec, 0.0063 m/sec, 0.0027 m/sec, 0.0035 m/sec for robots $1, 2, 3, 4$, respectively. The experimental data from robots' sensors is collected and processed off-line.

We consider the scenario where each robot can measure both relative distance and bearing to the remaining three robots, resulting in the measurement Jacobian matrix $\mathbf{H}$ of dimensions 24×12 at every time step. These relative observations are generated synthetically by adding noise to the relative distance and bearing calculated from the robots' pose estimates using the overhead camera. In our experiment, the standard deviations of the relative distance and bearing measurement noise are set to $\sigma_d = 0.05$ m and $\sigma_\theta = 0.035$ rad, respectively.

The duration of the experiment is 1000 sec (i.e., 1000 time steps). At every time step, the state estimate and its associated covariance are updated based upon (12) and (13), with QR decomposition of $\mathbf{H}$ [see (11)] implemented by using the Modified Householder QR algorithm.

Figures 4(a)–4(d) depict the time evolutions of the robots' pose estimation errors and corresponding 3σ-bounds along x, y, and ϕ-axis, respectively. As evident from Figures 4(a)–4(d), the EKF based on the Modified Householder QR algorithm performs well in a real-world CL experiment. Furthermore, the EKF produces consistent pose estimates for all four robots, i.e., the real robots' poses is within the 3σ bound centered at the robots' estimated pose.

Finally, we plot the real and estimated trajectories of the robots in Figures 5(a)–5(d), while Table III shows the averaged root-mean-square (RMS) errors of the robots' position and orientation. From these experimental results it is clear that the EKF based on the Modified Householder QR algorithm is robust and applicable to real systems.

TABLE III
RMS ERROR IN THE ROBOTS' POSE ESTIMATES

	Position Err. RMS (m)	Orient. Err. RMS (rad)
R1	0.2085	0.0801
R2	0.2250	0.0775
R3	0.2255	0.0811
R4	0.2224	0.0795

VIII. 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 $\mathcal{O}(N^3)$ in time complexity and $\mathcal{O}(N^2)$ in space complexity, 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. Furthermore, we performed experiments using a team of four mobile robots that demonstrate the applicability of the EKF update using the Modified Householder QR to real systems.

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

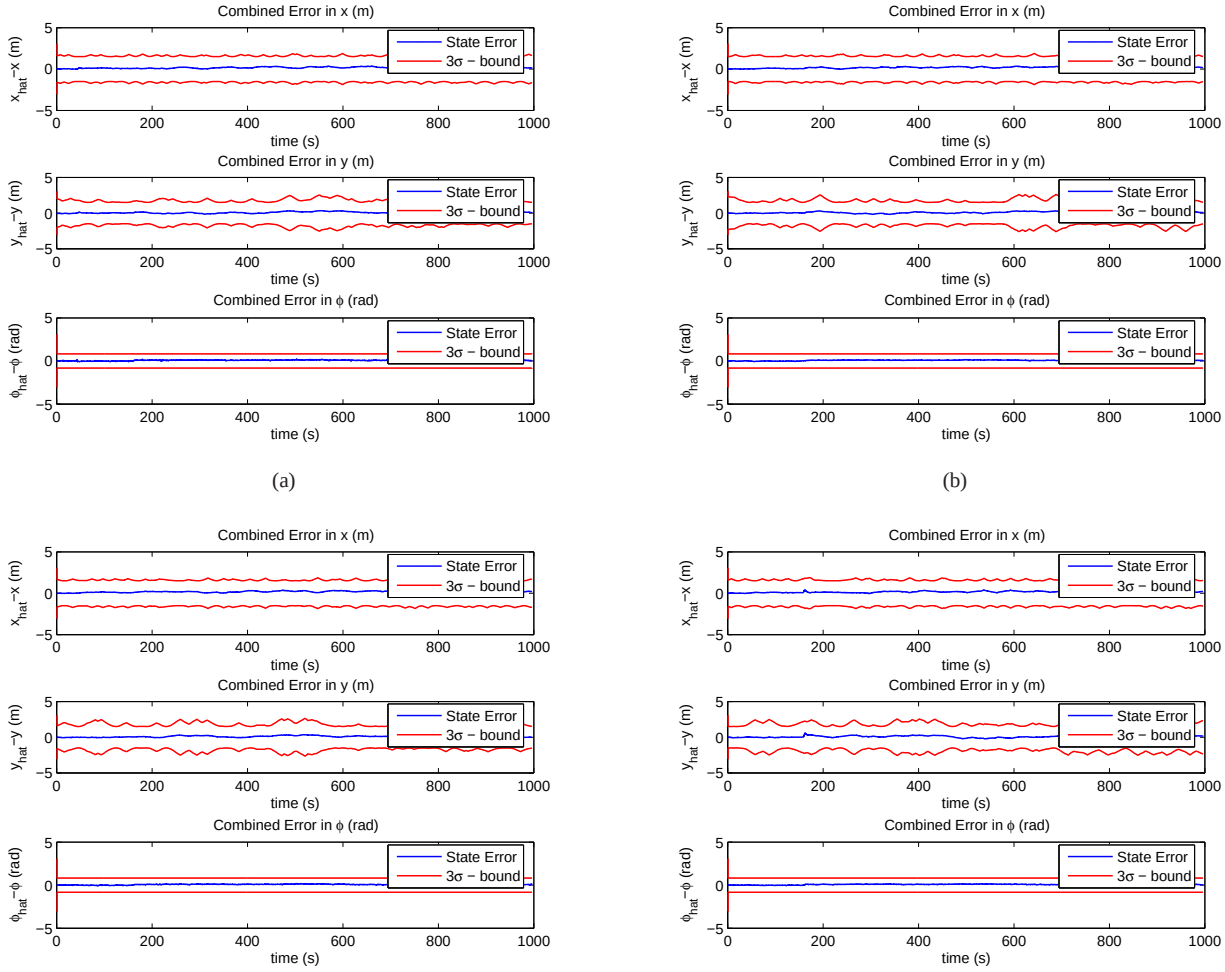


Fig. 4. [Experimental results] Pose estimation errors and corresponding 3σ-bounds for (a) Robot 1, (b) Robot 2, (c) Robot 3, and (d) Robot 4.

team [16], as well as to account for limitations on the robots' communication bandwidth and range [14], [30].

Appendix A
Computational Complexity of Information Filter

We show that due to the sparse structure of $\mathbf{H}_k$ [see (6)], computing $\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k$ and $\mathbf{P}_{k|k}\mathbf{H}_k^{\mathrm{T}}\tilde{\mathbf{z}}_{k|k-1}$ requires $\mathcal{O}(N^2)$ basic steps. To proceed, let us first partition $\mathbf{H}_k = \begin{bmatrix}\mathbf{H}_{k_1} & \dots & \mathbf{H}_{k_N}\end{bmatrix}$, where $\mathbf{H}_{k_i}$ is a $2(N-1)N \times 3$ sub-matrix consisting of the $(3i-2)$th, $(3i-1)$th, and $3i$th columns of $\mathbf{H}_k$, for $i = 1, \dots, N$. Following this partition, $\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k = \left[\mathbf{H}_{k_i}^{\mathrm{T}}\mathbf{H}_{k_j}\right]_{i,j}$, where $\mathbf{H}_{k_i}^{\mathrm{T}}\mathbf{H}_{k_j}$ $(i, j = 1, \dots, N)$ are 3×3 block matrices.

Next we observe that the sparse structure of $\mathbf{H}_k$ implies that (for $1 \leq i \neq j \leq N$)

$$\mathbf{H}_{k_i}^{\mathrm{T}}\mathbf{H}_{k_i} = \sum_{j=1,\ j\neq i}^{N} \left((\mathbf{\Psi}_k^{i,j})^{\mathrm{T}}\mathbf{\Psi}_k^{i,j} + (\mathbf{\Upsilon}_k^{j,i})^{\mathrm{T}}\mathbf{\Upsilon}_k^{j,i}\right), \quad (41)$$

$$\mathbf{H}_{k_i}^{\mathrm{T}}\mathbf{H}_{k_j} = (\mathbf{\Psi}_k^{i,j})^{\mathrm{T}}\mathbf{\Upsilon}_k^{i,j} + (\mathbf{\Upsilon}_k^{j,i})^{\mathrm{T}}\mathbf{\Psi}_k^{j,i}. \quad (42)$$

Since the right-hand-side (RHS) of (41)-(42) involves multiplications and additions of matrices $(\mathbf{\Psi}_k^{i,j}, \mathbf{\Upsilon}_k^{i,j}, \mathbf{\Psi}_k^{j,i}, \mathbf{\Upsilon}_k^{j,i})$ of *constant* dimensions 2×3 [see (4)-(5)], the time complexity for calculating each term of (41) and (42) are $\mathcal{O}(N)$ and $\mathcal{O}(1)$, respectively. Therefore the overall time complexity for evaluating $\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k$ is quadratic in N.

Similarly, it can be shown that calculating $\mathbf{P}_{k|k}\mathbf{H}_k^{\mathrm{T}}\tilde{\mathbf{z}}_{k|k-1}$ has time complexity $\mathcal{O}(N^2)$. To proceed, we first compute the $3N$ dimensional vector $\tilde{\mathbf{y}} = \mathbf{H}_k^{\mathrm{T}}\tilde{\mathbf{z}}_{k|k-1}$. Notice that the jth element of $\tilde{\mathbf{y}}$, $\tilde{y}_j$, is the *inner product* of the vector $\tilde{\mathbf{z}}_{k|k-1}$ and the jth column of $\mathbf{H}_k$. Recall that each column of $\mathbf{H}_k$ has at most $4(N-1)$ nonzero elements (see Remark 1), therefore the processing requirement of computing $\tilde{y}_j$ has time complexity $\mathcal{O}(N)$, despite the $2(N-1)N$ dimensional vector $\tilde{\mathbf{z}}_{k|k-1}$. Thus, the time complexity of calculating $\tilde{\mathbf{y}}$ is $\mathcal{O}(N^2)$. Next, we compute $\mathbf{P}_{k|k}\tilde{\mathbf{y}}$, a matrix (of dimensions $3N \times 3N$) and vector (of dimension $3N$) multiplication, with $\mathcal{O}(N^2)$ basic operations. Hence, the overall time complexity of computing $\mathbf{P}_{k|k}\mathbf{H}_k^{\mathrm{T}}\tilde{\mathbf{z}}_{k|k-1}$ is again quadratic in N.

Appendix B
Computational Complexity of Alternative QR Decomposition Algorithms

In this appendix, we analyze the computational complexity of QR decomposition of $\mathbf{\Lambda}$ [see (24)] by employing (1)

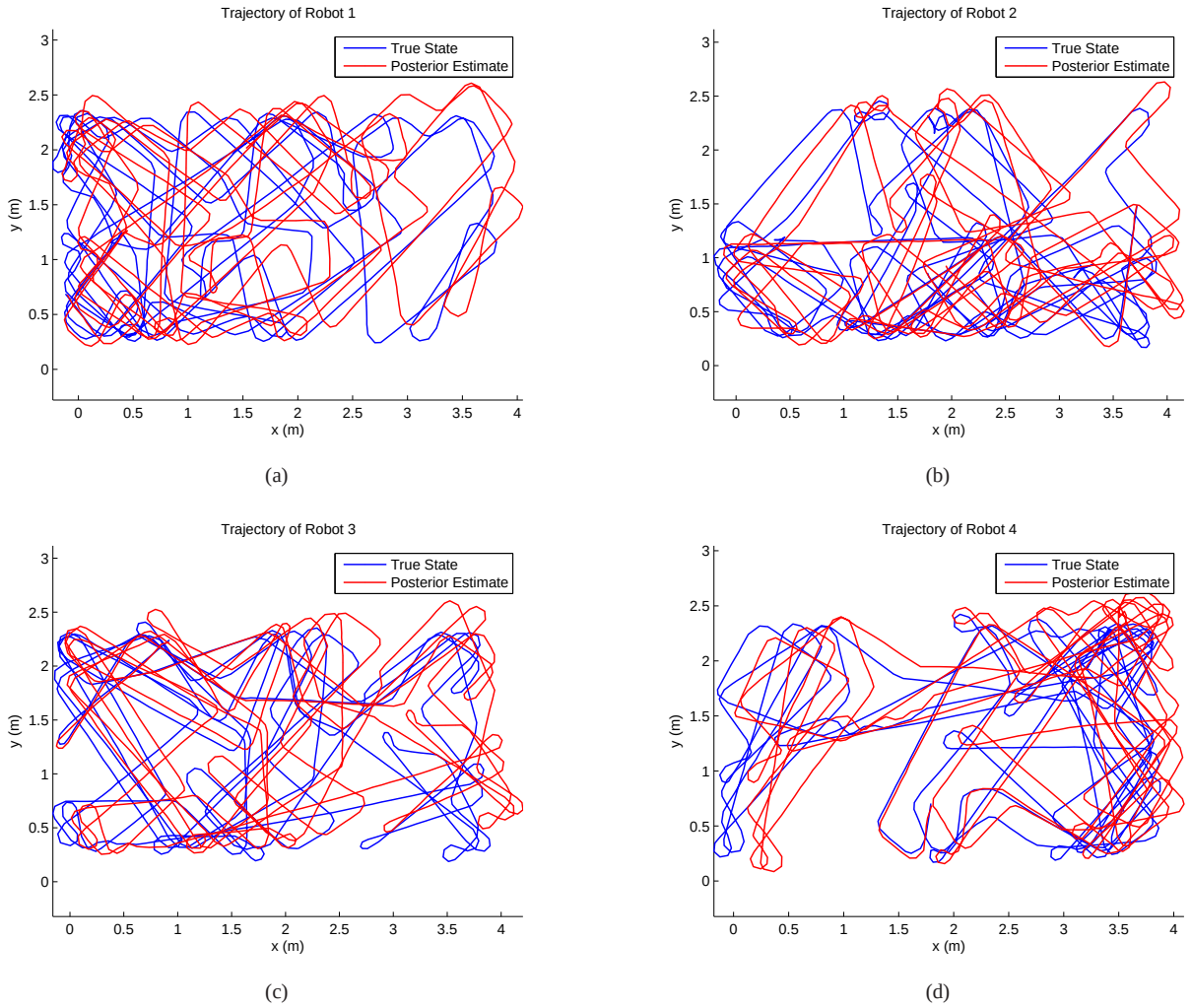


Fig. 5. [Experimental results] Real and estimated trajectories of (a) Robot 1, (b) Robot 2, (c) Robot 3, and (d) Robot 4, when employing the EKF based on the Modified Householder QR algorithm.

the Cholesky decomposition, (2) the modified Gram-Schmidt process, and (3) the Givens rotations [24, Section 5.2]. Similar to the discussion in Section IV-C, we assume $\mathbf{\Pi}_\ell = \mathbf{I}_{n_{\mathbf{\Lambda}}}, \ell = 1, \ldots, n_{\mathbf{\Lambda}}$, for simplicity. Furthermore, we suppose the QR decomposition of $\mathbf{\Lambda}$ takes the form $\mathbf{\Lambda} = \mathbf{Q}_{\mathbf{\Lambda}}\mathbf{R}_{\mathbf{\Lambda}}$, where $\mathbf{Q}_{\mathbf{\Lambda}}$ is orthonormal and $\mathbf{R}_{\mathbf{\Lambda}}$ is upper triangular.

A. Cholesky Decomposition [24, Section 4.2.3]

Since the columns of $\mathbf{Q}_{\mathbf{\Lambda}}$ are orthonormal, we have $\mathbf{\Lambda}^{\mathrm{T}}\mathbf{\Lambda} = \mathbf{R}_{\mathbf{\Lambda}}^{\mathrm{T}}\mathbf{R}_{\mathbf{\Lambda}}$. Therefore, we can apply a Cholesky decomposition of $\mathbf{\Lambda}^{\mathrm{T}}\mathbf{\Lambda}$ to compute the upper triangular matrix $\mathbf{R}_{\mathbf{\Lambda}}$.

Analogy to the analysis shown in Section III-C, the process of evaluating $\mathbf{\Lambda}^{\mathrm{T}}\mathbf{\Lambda}$ is only quadratic in N due to the sparse structure of $\mathbf{\Lambda}$. In addition, since Cholesky decomposition of a square matrix of dimension n requires $\mathcal{O}(n^3)$ operations [24, Section 4.2.3], we conclude that the time complexity of QR factorization of $\mathbf{\Lambda}$ using Cholesky decomposition is $\mathcal{O}(N^3)$ (note that $\mathbf{Q}_{\mathbf{\Lambda}}$ is not computed when using Cholesky decomposition). On the other hand, since $\mathbf{nnz}(\mathbf{\Lambda}) \sim \mathcal{O}(N^2)$ and both matrices $\mathbf{\Lambda}^{\mathrm{T}}\mathbf{\Lambda}$ and $\mathbf{R}_{\mathbf{\Lambda}}$ are of dimensions $N \times N$, the space complexity is $\mathcal{O}(N^2)$.

Although QR factorization of $\mathbf{\Lambda}$ using Cholesky decomposition can achieve desired computational complexity, it has two drawbacks. Firstly, the matrix product $\mathbf{\Lambda}^{\mathrm{T}}\mathbf{\Lambda}$ squares the condition number of $\mathbf{\Lambda}$, i.e., $\kappa(\mathbf{\Lambda}^{\mathrm{T}}\mathbf{\Lambda}) = \kappa^2(\mathbf{\Lambda})$, which can introduce numerical instability to Cholesky decomposition algorithm. Secondly, to use Cholesky decomposition, the matrix product $\mathbf{\Lambda}^{\mathrm{T}}\mathbf{\Lambda}$ is *required* to be *strictly positive definite*. Unfortunately, in our case $\mathbf{H}$ is rank-deficient, rendering $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ *positive semi-definite*. Hence, Cholesky decomposition is *not* applicable for QR factorization of $\mathbf{H}$ addressed in this paper.

B. Modified Gram-Schmidt [24, Section 5.2.8]

For clarity, we use $\boldsymbol{\lambda}_i$ and $\mathbf{q}_i$ to denote the ith columns of $\mathbf{\Lambda}$ and $\mathbf{Q}_{\mathbf{\Lambda}}$, respectively. Furthermore, we use $r_{i,j}$ to denote the (i,j)th element of $\mathbf{R}_{\mathbf{\Lambda}}$. For the purpose of complexity analysis, we outline the algorithmic flow chart of the Modified Gram-Schmidt process in Algorithm 3.

From Algorithm 3, it is clear that the vector $\mathbf{q}_\ell$ is a linear combination of $\boldsymbol{\lambda}_\ell$ and $\{\mathbf{q}_j, j = 1, \ldots, \ell - 1\}$. Since every $\mathbf{q}_j, j = 2, \ldots, \ell - 1$, can be expressed as a linear combination of $\boldsymbol{\lambda}_j$ and $\{\mathbf{q}_i, i = 1, \ldots, j - 1\}$, and $\mathbf{q}_1$ is a

Algorithm 3 Modified Gram-Schmidt [24, Algorithm 5.2.5]

Require: $\mathbf{\Lambda}$ of dimensions $m_{\mathbf{\Lambda}} \times n_{\mathbf{\Lambda}}$ ($m_{\mathbf{\Lambda}} \geq n_{\mathbf{\Lambda}}$).
Ensure: $\mathbf{R}_{\mathbf{\Lambda}}$ of dimensions $n_{\mathbf{\Lambda}} \times n_{\mathbf{\Lambda}}$.
1: **for** $\ell = 1$ **to** n, **do**
2: Set $\bar{\mathbf{q}} = \boldsymbol{\lambda}_\ell$.
3: **for** $j = 1$ **to** $\ell - 1$, **do**
4: Compute $r_{j,\ell} = \mathbf{q}_j^{\mathrm{T}} \bar{\mathbf{q}}$.
5: Update $\bar{\mathbf{q}} \Leftarrow \bar{\mathbf{q}} - r_{j,\ell} \mathbf{q}_j$.
6: **end for**
7: Calculate $r_{\ell,\ell} = \|\bar{\mathbf{q}}\|_2$.
8: **if** $r_{\ell,\ell} = 0$, **then**
9: STOP.
10: **else**
11: $\mathbf{q}_\ell = \bar{\mathbf{q}} / r_{\ell,\ell}$.
12: **end if**
13: **end for**
14: **return** $\mathbf{R}_{\mathbf{\Lambda}}$.

scalar multiplication of $\boldsymbol{\lambda}_1$, we conclude by recursion that the vector $\mathbf{q}_\ell$ is a linear combination of $\{\boldsymbol{\lambda}_j, j = 1, \ldots, \ell\}$. Exploiting the structure of $\boldsymbol{\lambda}_j$ ($j = 1, \ldots, \ell$), we have $\mathbf{nnz}(\mathbf{q}_\ell) = \sum_{j=1}^{\ell}(N-j)$ for $\ell = 1, \ldots, N$. Hence, updating $\bar{\mathbf{q}}$ at the jth iteration inside the inner "for-loop" (see Algorithm 3, Line 5), which is the most dominant computational process, requires approximately $\sum_{i=1}^{j}(N-i)$ steps. Therefore, the overall time complexity of QR decomposition of $\mathbf{\Lambda}$ using the Modified Gram-Schmidt QR algorithm is in the order of

$$\sum_{\ell=1}^{N} \left\{ \sum_{j=1}^{\ell-1} \Big[\sum_{i=1}^{j} (N-i) \Big] \right\} \sim \mathcal{O}(N^4) \ .$$

Furthermore, for space complexity, we focus on the storage of $\{\mathbf{q}_\ell, \ell = 1, \ldots, n_{\mathbf{\Lambda}}\}$, which is the most demanding source in terms of memory usage. Since $\mathbf{nnz}(\mathbf{q}_\ell) = \sum_{j=1}^{\ell}(N-j)$, we conclude that the space complexity is in the order of

$$\sum_{\ell=1}^{N} \left[\sum_{j=1}^{\ell} (N-j) \right] \sim \mathcal{O}(N^3) \ .$$

An alternative method to the Modified Gram-Schmidt QR algorithm is to apply the Classical Gram-Schmidt process [24, Section 5.2.7]. However, the Classical Gram-Schmidt process only differs from the Modified Gram-Schmidt in the formula of computing $r_{j,\ell}$, while the update of $\bar{\mathbf{q}}$ is *exactly the same* for both methods. Hence, we conclude that the computational complexity of QR decomposition of $\mathbf{\Lambda}$ using the Classical Gram-Schmidt is exactly the same as that of the Modified Gram-Schmidt. Furthermore, the Classical Gram-Schmidt suffers from numerical instability in practice [24, Section 5.2.8].

C. Givens QR [24, Section 5.2.3]

The Givens QR algorithm is one of the most widely adopted technique for implementing QR factorization [24, Section 5.2.3]. To understand the Givens QR algorithm, we firstly introduce Givens rotation $\mathbf{G}^\phi = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$, a 2-D rotational matrix. Given a 2×1 vector $\mathbf{x} = [x_1 \ x_2]^{\mathrm{T}}$ with $x_2 \neq 0$, by selecting $\phi = \arctan(-x_2/x_1)$, it is straightforward to verify $\mathbf{G}^\phi \mathbf{x} = [\|\mathbf{x}\|_2 \ 0]^{\mathrm{T}}$ [24, Section 5.1.8]. In other words, left-multiplying $\mathbf{x}$ by $\mathbf{G}^\phi$ eliminates x_2. The essence of the Givens QR is to successively invoke Givens rotations to transform a matrix into upper triangular form by eliminating non-zeros below its diagonal [24, Algorithm 5.2.2].

The time complexity of the Givens QR process for decomposing an arbitrary matrix $\mathbf{\Lambda}$ is in the order of $m_{\mathbf{\Lambda}} n_{\mathbf{\Lambda}}^2$, where $m_{\mathbf{\Lambda}}$ and $n_{\mathbf{\Lambda}}$ are the number of rows and columns of $\mathbf{\Lambda}$, respectively [24, Section 5.2.3]. For $\mathbf{\Lambda}$ addressed here [see (24)], substituting $m_{\mathbf{\Lambda}} = (N-1)N/2$ and $n_{\mathbf{\Lambda}} = N$ results in the time complexity $\mathcal{O}(N^4)$. However, this does not take into account of the structure of $\mathbf{\Lambda}$, and it is natural to conjecture that the overall time complexity using the Givens QR can be reduced by exploiting the sparsity of $\mathbf{\Lambda}$. Unfortunately, this conjecture is false. In what follows, we examine both time complexity and space complexity of applying the Givens QR algorithm to decompose $\mathbf{\Lambda}$ in (24). For clarity, we use $\lambda_{i,j}$ to denote the (i,j)th element of $\mathbf{\Lambda}$. Furthermore, we partition $\mathbf{\Lambda}$ by rows into $\mathbf{\Lambda} = [\mathbf{\Lambda}_1^{\mathrm{T}} \ \ldots \ \mathbf{\Lambda}_{N-1}^{\mathrm{T}}]^{\mathrm{T}}$, where the rows of the sub-matrix $\mathbf{\Lambda}_i$ correspond to the $(J_{i-1}+1)$th through J_ith rows of $\mathbf{\Lambda}$, with $J_0 = 0$ and $J_i = J_{i-1} + (N-i) = \sum_{j=1}^{i}(N-j)$ for $i = 1, \ldots, N-1$. To facilitate the discussion, we provide the pseudo-code of the Givens QR in Algorithm 4. It is worth noting that each Givens rotation effects only two rows of $\mathbf{\Lambda}$ (see Algorithm 4, Lines 5-9), whereas the processing cost of Line 7 is constant (4 multiplications and 2 additions).

Algorithm 4 Givens QR [24, Algorithm 5.2.2]

Require: $\mathbf{\Lambda}$ of dimensions $m_{\mathbf{\Lambda}} \times n_{\mathbf{\Lambda}}$ ($m_{\mathbf{\Lambda}} \geq n_{\mathbf{\Lambda}}$).
Ensure: $\mathbf{R}_{\mathbf{\Lambda}}$ of dimensions $n_{\mathbf{\Lambda}} \times n_{\mathbf{\Lambda}}$.
1: **for** $\ell = 1$ **to** $n_{\mathbf{\Lambda}}$, **do**
2: Set $i_0 = \ell$, and seek all indices $\{i_1, \ldots, i_{F_\ell}\}$ such that $i_0 < i_1 < \cdots < i_{F_\ell} \leq m_{\mathbf{\Lambda}}$ and $\lambda_{i_\eta,\ell} \neq 0, \forall \eta = 1, \ldots, F_\ell$.
3: **for** $\eta = F_\ell$ **to** 1, **do**
4: Compute $\phi = \arctan\left(-\frac{\lambda_{i_\eta,\ell}}{\lambda_{i_{\eta-1},\ell}}\right)$, update $\lambda_{i_{\eta-1},\ell} \Leftarrow (\lambda_{i_{\eta-1},\ell}^2 + \lambda_{i_\eta,\ell}^2)^{1/2}$, $\lambda_{i_\eta,\ell} \Leftarrow 0$.
5: **for** $j = \ell + 1$ **to** $n_{\mathbf{\Lambda}}$, **do**
6: **if** $[\lambda_{i_{\eta-1},j} \ \lambda_{i_\eta,j}]^{\mathrm{T}} \neq \mathbf{0}_{2\times 1}$, **then**
7: $\begin{bmatrix} \lambda_{i_{\eta-1},j} \\ \lambda_{i_\eta,j} \end{bmatrix} \Leftarrow \mathbf{G}^\phi \begin{bmatrix} \lambda_{i_{\eta-1},j} \\ \lambda_{i_\eta,j} \end{bmatrix}$
8: **end if**
9: **end for**
10: **end for**
11: **end for**
12: **return** $\mathbf{R}_{\mathbf{\Lambda}}$, the first $n_{\mathbf{\Lambda}}$ rows of $\mathbf{\Lambda}$.

At the 1st iteration (i.e., $\ell = 1$), a sequence of $N-2$ Givens rotations is applied to $\mathbf{\Lambda}$ (more specifically, $\mathbf{\Lambda}_1$) so that its first column has zeros below the 1st component. In particular, the 1st Givens rotation $\mathbf{G}^{\phi_1}$ with $\phi_1 = \arctan\left(-\frac{\psi^{1,N}}{\psi^{1,N-1}}\right)$ eliminates $\psi^{1,N}$, while introducing 2 nonzero fillings $\lambda_{N-1,N-1}, \lambda_{N-2,N}$ to $\mathbf{\Lambda}_1$. Similar in spirit, after the 2nd Givens rotation $\mathbf{G}^{\phi_2}$ [$\phi_2 = \arctan\left(-\frac{\lambda_{N-2,1}}{\psi^{1,N-2}}\right)$], $\lambda_{N-2,1}$ vanishes and 3 extra non-zeros $\lambda_{N-2,N-2}, \lambda_{N-3,N-1}, \lambda_{N-3,N}$ are added to $\mathbf{\Lambda}_1$. In particular, $\lambda_{N-3,N}$ is introduced to $\mathbf{\Lambda}_1$ due to $\lambda_{N-2,N}$, which is gener-

ated previously by $\mathbf{G}^{\phi_1}$. The above process is repeated until all non-zeros below the 1st component in the first column of $\mathbf{\Lambda}$ are eliminated, and the resulting matrix $\mathbf{\Lambda}$, at the end of this process, takes the form

$$\mathbf{\Lambda}=\begin{bmatrix} \times & \times & \boxtimes & \boxtimes & \cdots & \boxtimes \\ & \boxtimes & \times & \boxtimes & \cdots & \boxtimes \\ & & \ddots & \ddots & \ddots & \vdots \\ & & & & & \times \\ & \psi^{2,3} & \upsilon^{2,3} & & & \\ & \psi^{2,4} & & \upsilon^{2,4} & & \\ & \vdots & & & \ddots & \\ & \psi^{2,N} & & & & \upsilon^{2,N} \\ & & \vdots & \cdots & & \vdots \\ & & & & \psi^{N-1,N} & \upsilon^{N-1,N} \end{bmatrix} . \quad (43)$$

In particular, $\mathbf{\Lambda}_1$ is transformed into an upper triangular form, while $\mathbf{\Lambda}_j, j=2,\ldots,N-1$, remain intact. The overall number of basic steps during the 1st iteration is in the order of $\sum_{i=2}^{N-1} i$. Furthermore, from (43), the total number of non-zeros of $\mathbf{\Lambda}_1$ is $\frac{(N-1)N}{2}-1$, and $\mathbf{nnz}([\mathbf{\Lambda}_2^{\mathrm{T}} \ \ldots \ \mathbf{\Lambda}_{N-1}^{\mathrm{T}}]^{\mathrm{T}}) = (N-2)(N-1)$. Hence $\mathbf{nnz}(\mathbf{\Lambda}) = \frac{(N-1)N}{2} + (N-2)(N-1) - 1$.

Similarly, at the 2nd iteration (i.e., $\ell = 2$), a sequence of $N-3$ Givens rotations is firstly applied to transform $\mathbf{\Lambda}_2$ into upper triangular with upper bandwidth $q = 1$. This process requires the number of arithmetic operations proportional to $\sum_{i=2}^{N-2} i$, and the resulting $\mathbf{\Lambda}$ has $\mathbf{nnz}(\mathbf{\Lambda}) = \frac{1}{2}\sum_{i=1}^{2}(N-i)(N-i+1) + (N-3)(N-2) - 2$. However, note that after this process $\lambda_{J_1+1,2} \neq 0$. Therefore, an extra Givens rotation is required to eliminate $\lambda_{J_1+1,2}$, which effects the 2nd and (J_1+1)th row of $\mathbf{\Lambda}$. Since the (J_1+1)th row of $\mathbf{\Lambda}$ has $N-1$ nonzero elements, this extra Givens rotation process invokes $\mathcal{O}(N-1)$ basic steps. Thus the total time complexity during the 2nd iteration is in the order of $(N-1) + \sum_{i=2}^{N-2} i$.

To seek the pattern of the Givens QR elimination process, as well as the evolution of the matrix $\mathbf{\Lambda}$, let us further examine the 3rd iteration (i.e., $\ell = 3$). Similarly, we firstly apply a sequence of $N-4$ Givens rotations so that the resulting $\mathbf{\Lambda}_3$ has upper bandwidth $q = 2$, and at the end of this process $\mathbf{nnz}(\mathbf{\Lambda}) = \frac{1}{2}\sum_{i=1}^{3}(N-i)(N-i+1) - 1 + (N-4)(N-3) - 3$, where the subtraction of 1 is due to the vanish of $\lambda_{J_1+1,2}$. After this process, we observe that $\lambda_{J_1+1,3}, \lambda_{J_1+2,3}$, and $\lambda_{J_2+1,3}$ are non-zeros, which requires $2+1=3$ extra Givens rotations to eliminate them, with every Givens rotation invoking a cost proportional to $N-2$ (due to the fact that each row where $\lambda_{J_i+j,\ell}$, $i=1,\ldots,2; j=1,\ldots,3-i$, belongs to has $N-2$ non-zeros), thus the total time complexity during the 3rd iteration is in the order of $(N-2)\sum_{i=1}^{2} i + \sum_{i=2}^{N-3} i$. After these extra Givens rotations have been applied, the total number of non-zeros in $\mathbf{\Lambda}$ is reduced to $\mathbf{nnz}(\mathbf{\Lambda}) = \frac{1}{2}\sum_{i=1}^{3}(N-i)(N-i+1) - \sum_{j=1}^{2}\sum_{i=1}^{j} i + (N-4)(N-3) - 3$.

Now we are ready to infer the time complexity and space requirement at the ℓth iteration by induction. In particular, a sequence of $N-\ell-1$ Givens rotations is firstly employed and the resulting $\mathbf{\Lambda}_\ell$ has upper bandwidth $q = \ell - 1$, and after this process $\mathbf{nnz}(\mathbf{\Lambda}) = \frac{1}{2}\sum_{i=1}^{\ell}(N-i)(N-i+1) - \sum_{j=1}^{\ell-2}\sum_{i=1}^{j} i + (N-\ell-1)(N-\ell) - \ell$. Secondly, a total number of $\sum_{i=1}^{\ell-1} i$ Givens rotations are applied to eliminate the non-zeros $\{\lambda_{J_i+j,\ell} \mid \ i=1,\ldots,\ell-1; j=1,\ldots,\ell-i\}$, with each Givens rotation process of cost proportional to $N-\ell+1$, because of $N-\ell+1$ nonzero elements in every row where $\lambda_{J_i+j,\ell}$, $i=1,\ldots,\ell-1; j=1,\ldots,\ell-i$, is located. Hence, the total time complexity during the ℓth iteration is in the order of $(N-\ell+1)\sum_{i=1}^{\ell-1} i + \sum_{i=2}^{N-\ell} i$. After the elimination of $\sum_{i=1}^{\ell-1} i$ non-zeros $\{\lambda_{J_i+j,\ell} \mid \ i=1,\ldots,\ell-1; j=1,\ldots,\ell-i\}$, the overall number of non-zeros in $\mathbf{\Lambda}$ is reduced to $\mathbf{nnz}(\mathbf{\Lambda}) = \frac{1}{2}\sum_{i=1}^{\ell}(N-i)(N-i+1) - \sum_{j=1}^{\ell-1}\sum_{i=1}^{j} i + (N-\ell-1)(N-\ell) - \ell$.

In summary, the overall time complexity of QR decomposition of $\mathbf{\Lambda}$ using the Givens QR is proportional to

$$\sum_{\ell=1}^{N}\left[(N-\ell+1)\sum_{i=1}^{\ell-1} i + \sum_{i=2}^{N-\ell} i\right] \sim \mathcal{O}(N^4) .$$

On the other hand, since the most dominant source in terms of memory usage is the storage of $\mathbf{\Lambda}$, the space complexity of QR decomposition of $\mathbf{\Lambda}$ using the Givens QR algorithm is

$$\max_{1\leq\ell\leq N}\left[\mathbf{nnz}(\mathbf{\Lambda})\right] \sim \mathcal{O}(N^3) ,$$

where $\mathbf{nnz}(\mathbf{\Lambda}) = \frac{1}{2}\sum_{i=1}^{\ell}(N-i)(N-i+1) - \sum_{j=1}^{\ell-2}\sum_{i=1}^{j} i + (N-\ell-1)(N-\ell) - \ell$.

Appendix C
Proof of Proposition 2

Proof: The proof is straightforward. Firstly, by the definition of $\mathbf{v}_\ell$, we have $\mathbf{Q}_\ell = \mathbf{diag}(\mathbf{I}_{\ell-1}, \mathbf{Q})$, where $\mathbf{Q} = \mathbf{I}_{m_{\mathbf{H}}-\ell+1} - \beta_\ell \mathbf{v}\mathbf{v}^{\mathrm{T}}$. Thus, assuming $\mathbf{\Pi}_\ell = \mathbf{I}_{n_{\mathbf{H}}}$, we have

$$\mathbf{H}^{(\ell)} = \mathbf{Q}_\ell \mathbf{H}^{(\ell-1)}\mathbf{\Pi}_\ell = \begin{bmatrix} \mathbf{H}_{1,1}^{(\ell-1)} & \mathbf{H}_{1,2}^{(\ell-1)} \\ \mathbf{0} & \mathbf{Q}\mathbf{H}_{2,2}^{(\ell-1)} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{1,1}^{(\ell-1)} & \mathbf{t} & \overline{\mathbf{H}}_{1,2}^{(\ell-1)} \\ \mathbf{0} & \mathbf{Q}\mathbf{u} & \mathbf{Q}\overline{\mathbf{H}}_{2,2}^{(\ell-1)} \end{bmatrix} . \quad (44)$$

Since $\mathbf{v}$ is chosen to zero all but the first component of $\mathbf{u}$ [see (19)], we have $\mathbf{Q}\mathbf{u} = r_\ell \mathbf{e}_1$ with the expression of r_ℓ provided by (21). On the other hand [see (14)],

$$\mathbf{Q}\overline{\mathbf{H}}_{2,2}^{(\ell-1)} = \overline{\mathbf{H}}_{2,2}^{(\ell-1)} - \mathbf{v}\Big(\beta_\ell\big(\overline{\mathbf{H}}_{2,2}^{(\ell-1)}\big)^{\mathrm{T}}\mathbf{v}\Big)^{\mathrm{T}} \quad (45)$$
$$= \begin{bmatrix} \bar{\mathbf{s}}^{\mathrm{T}} \\ \underline{\mathbf{H}}_{2,2}^{(\ell-1)} \end{bmatrix} - \begin{bmatrix} v_1 \\ \mathbf{v}_{-1} \end{bmatrix}\boldsymbol{\delta}^{\mathrm{T}} = \begin{bmatrix} \bar{\mathbf{s}}^{\mathrm{T}} - v_1\boldsymbol{\delta}^{\mathrm{T}} \\ \underline{\mathbf{H}}_{2,2}^{(\ell-1)} - \mathbf{u}_{-1}\boldsymbol{\delta}^{\mathrm{T}} \end{bmatrix} ,$$

where the last equality derives from $\mathbf{v}_{-1} = \mathbf{u}_{-1}$ [see Remark 2]. Comparing (44)-(45) with (20) and matching corresponding terms, we obtain (22) and (23). ■

Appendix D
Complexity Analysis of Column Pivoting

In this appendix, we address the computational complexity of column pivoting, which corresponds to Lines 1-4, 6, 10-13 in Algorithm 1.

Firstly, we investigate space complexity. Since the squared-norm vector $[c_1 \ \ldots \ c_{n_{\mathbf{H}}}]$ and permutation vector $[\pi_1 \ \ldots \ \pi_{n_{\mathbf{H}}}]$ are required for performing column pivoting, we conclude that the space complexity is $\mathcal{O}(n_{\mathbf{H}})$, or equivalently, $\mathcal{O}(N)$.

Secondly, for time complexity, we break down the overall arithmetic operations into two phases, namely (1) initial generation of $c_j, j = 1, \ldots, n_{\mathbf{H}}$, in Lines 1-4, and (2) sequential update of $c_j, j = \ell+1, \ldots, n_{\mathbf{H}}$, corresponding to Lines 6, 10-13 during the ℓth iteration of the while-loop.

During the initialization stage, calculating every $c_j, j = 1, \ldots, n_{\mathbf{H}}$, involves $\mathcal{O}(N)$ operations, since each column of $\mathbf{H}$ has at most $4(N-1)$ non-zeros (see Remark 1). Furthermore, finding the maximum element among $n_{\mathbf{H}} = 3N$ scalars can be achieved in $n_{\mathbf{H}} - 1$ basic steps. Thus, executing Lines 1-4 can be fulfilled in $\mathcal{O}(N^2)$ basic steps.

To analyze the complexity associated with the update stage, let us focus on the ℓth iteration of the while-loop. We first note that exchange of the ℓth and ℓ^*th columns of $\mathbf{H}^{(\ell-1)}$, as well as c_ℓ and c_{ℓ^*} in Line 6 can be tracked and effectively implemented by pointers in practice, without physically swapping real data in memory. On the other hand, the total number of operations for updating $c_j, j = \ell+1, \ldots, n_{\mathbf{H}}$ (see Lines 10-12 is $n_{\mathbf{H}} - \ell$. Additionally, the maximum operation in Line 13, similar to that of Line 4, can be achieved in $n_{\mathbf{H}} - \ell$ basic steps. Hence, the number of basic operations for the update phase at the ℓth iteration is of $\mathcal{O}(n_{\mathbf{H}} - \ell)$, and therefore, the total time complexity associated with sequential update is $\mathcal{O}(N^2)$.

In summary, we conclude the overall time complexity of column pivoting is $\mathcal{O}(N^2)$.

Appendix E
Proof of Proposition 3

Proof: Firstly, we notice that $c^* = \|\mathbf{u}\|_2^2$ is readily available from the column pivoting stage (see Line 4 for $\ell = 1$ and Line 13 for $\ell > 1$ in Algorithm 1), and needs *not* to be recomputed. From (19), (21) and Remark 2, it is clear that calculation of $\mathbf{v}, \beta_\ell, r_\ell$ in Line 7 requires a *constant* number of basic steps. Furthermore, $\mathbf{u}$ and $\mathbf{v}$ share *exactly the same* sparsity pattern, thus $\mathbf{nnz}(\mathbf{v}) = \tau_{\ell-1}$.

Secondly, we examine the number of basic steps for computing the matrix-vector product $\boldsymbol{\delta} = \beta_\ell \big(\overline{\mathbf{\Lambda}}_{2,2}^{(\ell-1)}\big)^{\mathrm{T}} \mathbf{v}$, whose element δ_j is the inner product of $\beta_\ell \mathbf{v}$ and the jth column of $\overline{\mathbf{\Lambda}}_{2,2}^{(\ell-1)}, j = 1, \ldots, n_{\mathbf{\Lambda}} - \ell$. Since $\mathbf{nnz}(\mathbf{v}) = \tau_{\ell-1}$, computing δ_j only requires $\tau_{\ell-1}$ scalar multiplications and $\tau_{\ell-1} - 1$ scalar additions. Therefore, computing $\boldsymbol{\delta}$ involves $\mathcal{O}((n_{\mathbf{\Lambda}} - \ell)\tau_{\ell-1})$ operations. Once $\boldsymbol{\delta}$ is computed, updating $\mathbf{s} = [s_1 \ \ldots \ s_{n_{\mathbf{\Lambda}}-\ell}]^{\mathrm{T}}$ is trivial with $n_{\mathbf{\Lambda}} - \ell$ basic steps [see (22)]. Thus, the overall time complexity of Line 8 in Algorithm 1 is $\mathcal{O}((n_{\mathbf{\Lambda}} - \ell)\tau_{\ell-1})$.

Thirdly, we note that (23) is a rank-one modification, where $\mathbf{\Lambda}_{2,2}^{(\ell)}$ is obtained by subtracting an outer product $\mathbf{u}_{-1}\boldsymbol{\delta}^{\mathrm{T}}$ from $\underline{\mathbf{\Lambda}}_{2,2}^{(\ell-1)}$. This rank-one update process has a cost of $(n_{\mathbf{\Lambda}} - \ell)(\tau_{\ell-1} - 1)$. In particular, we only need to modify those rows of $\underline{\mathbf{\Lambda}}_{2,2}^{(\ell-1)}$ whose indices correspond to the linear indices of the nonzero elements of $\mathbf{u}_{-1}$.

In summary, the time complexity of Householder update at the ℓth iteration, dominated by the computation of $\boldsymbol{\delta}$ and the update of $\mathbf{\Lambda}_{2,2}^{(\ell)}$, is $\mathcal{O}((n_{\mathbf{\Lambda}} - \ell)\tau_{\ell-1})$. ■

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 *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 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," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 271–281, Mar. 2003.

[4] G. M. Siouris, G. Chen, and J. Wang, "Tracking an incoming ballistic missile using an extended interval Kalman filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 1, pp. 232–240, Jan. 1997.

[5] K. Zhou and S. I. Roumeliotis, "Optimal motion strategies for range-only constrained multisensor target tracking," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1168–1185, Oct. 2008.

[6] ——, "Multi-robot active target tracking with combinations of relative observations," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 678–695, Aug. 2011.

[7] S. I. Roumeliotis and G. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, Oct. 2002.

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

[9] R. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, May 8-13 1994, pp. 1250–1257.

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

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

[12] N. Trawny and S. I. Roumeliotis, "Cooperative multi-robot localization under communication constraints," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 12-17 2009, pp. 4394–4400.

[13] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Localization for mobile robot teams using maximum likelihood estimation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, Sep. 30 - Oct. 5 2002, pp. 434–459.

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

[15] 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.

[16] E. D. Nerurkar, S. I. Roumeliotis, and A. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 12-17 2009, pp. 1402–1409.

[17] F. Dellaert, F. Alegre, and E. B. Martinson, "Intrinsic localization and mapping with 2 applications: Diffusion mapping and Marco Polo localization," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, Sep. 14-19 2003, pp. 2344–2349.

[18] S. Panzieri, F. Pascucci, and R. Setola, "Multirobot localisation using Interlaced Extended Kalman Filter," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 9-15 2006, pp. 2816 – 2821.

[19] L. Glielmo, R. Setola, and F. Vasca, "An interlaced extended kalman filter," *IEEE Transactions on Automatic Control*, vol. 44, no. 8, pp. 1546–1549, Aug. 1999.

[20] N. Karam, F. Chausse, R. Aufrere, and R. Chapuis, "Localization of a group of communicating vehicles by state exchange," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 9-15 2006, pp. 519–524.

[21] A. Martinelli, "Improving the precision on multi robot localization by using a series of filters hierarchically distributed," in *Proceedings of the*

IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, Oct. 29 - Nov. 2 2007, pp. 1053–1058.

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

[23] D. S. Bayard and P. B. Brugarolas, "On-board vision-based spacecraft estimation algorithm for small body exploration," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 1, pp. 243–260, Jan. 2008.

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

[25] P. A. Businger and G. H. Golub, "Linear least squares solutions by Householder transformations," *Numerische Mathematik*, vol. 7, pp. 269–276, 1965.

[26] L. Kaufman, "Application of dense Householder transformations to a sparse matrix," *ACM Transactions on Mathematical Software*, vol. 5, no. 4, pp. 442–450, Dec. 1979.

[27] T. A. Davis, "Algorithm 915: Multifrontal multithreaded rank-revealing sparse QR factorization," *ACM Transactions on Mathematical Software*, vol. 38, no. 1, Nov. 2011.

[28] ——, *Direct Methods for Sparse Linear Systems*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2006.

[29] "Matlab version 7.12," Natick, MA, 2011, http://www.mathworks.com/.

[30] E. D. Nerurkar and S. I. Roumeliotis, "Asynchronous multi-centralized cooperative localization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 18-22 2010, pp. 4352–4359.