2022 American Control Conference (ACC)
Atlanta, USA, June 8-10, 2022

Distributed Invariant Extended Kalman Filter for 3-D Dynamic State Estimation Using Lie Groups

Jie Xu, Pengxiang Zhu and Wei Ren

Abstract— **Distributed Kalman filters have been widely studied in vector space and been applied to 2-D target state estimation using sensor networks. In this paper, we introduce a novel *distributed invariant extended Kalman filer* (DIEKF) that exploits matrix Lie groups and is suitable to track the target's 6-DOF motion in a 3-D environment. The DIEKF is based on the proposed extended Covariance Intersection (CI) algorithm that guarantees consistency in matrix Lie groups. The DIEKF is fully distributed as each agent only uses the information from itself and the one-hop communication neighbors, and it is robust to a time-varying communication topology and changing blind agents. To evaluate the performance, we apply the algorithm in a camera network to track a target pose. Extensive Monte-Carlo simulations have been performed to analyze the performance. Overall, the proposed algorithm is more accurate and more consistent in comparison with our recent work on the quaternion-based distributed EKF (QDEKF).**

I. INTRODUCTION

Sensor networks have been widely used in many applications. Today, distributed algorithms that use only each agent's own and communication neighbors' information draw more attention in both control and robotics society. In distributed algorithms, each agent maintains an estimator of the *same* target. To fuse the information from neighbors, there is a need to handle the unknown cross-covariances between different estimators on the agents. Naively fusing these estimators yields an inconsistent estimator that will diverge. The consensus [1] and the Covariance Intersection (CI) [2] algorithms have been widely used to design a consistent distributed extended Kalman filter (EKF) in the existing works. The consensus algorithm as a tool of information distributed averaging has been applied to the information pairs (i.e., information vectors and matrices) [3], the measurements [3] and the hybrid of the two in [4]. These approaches require multiple communication iterations at each timestamp. To be more efficient, the CI algorithm that computes a convex combination of the local information pairs from one-hop communication is used to design the DEKF. In [5], each agent first updates the estimator using its own measurements and then the resulting information pairs are fused with the pairs from neighbors in CI. In [6], CI is first used to fuse the prior information pairs among neighborhood and then the improved prior information is updated with the local and neighboring measurements. Note that all these algorithms work on *vector space* that has additive errors. Besides, the effectiveness is only evaluated on the tracking problem in 2D cases. Although, one can naively extend the vector space algorithm to the 3D case by using the Euler angle representation for rotations, it suffers the well-known Gimbal lock problem.

This work was supported by National Science Foundation under Grant CMMI-2027139.

Jie Xu, Pengxiang Zhu and Wei Ren are with the Department of Electrical and Computer Engineering, University of California, Riverside, CA 92521, USA(jxu150@ucr.edu, pzhu008@ucr.edu, ren@ee.ucr.edu).

To address this issue, our recent work [7] introduces a quaternion-based distributed EKF (QDEFK) algorithm where the 3D orientation is represented as a unit quaternion. CI is for the first time extended to the 3D space using "quaternion average" [8]. Good performance is shown by tracking a drone using a camera network. However, the filer is built upon the error-state EKF where the position, velocity and orientation errors are decoupled [9], and the linearized error dynamics Jacobian and the measurement Jacobian are still functions of the estimated states. Then the unobservable states can gain spurious information and become observable by the filter. This hurts the consistency and then the accuracy. Besides, like the vector space DEKF, we only study the case where the target motion model is known.

Recently, a new type of EKF is designed based on the invariant observer theory [10]. The estimation error is *invariant* under the action of matrix Lie groups and satisfies a log-linear autonomous differential equation with nice properties [11]. In particular, in the case of $\mathrm{SE}_K(3)$, the position, velocity and orientation errors are coupled. This invariant EKF (IEKF) has been successfully applied to leg robot state estimation [12] and SLAM [13], where the IEKF achieves promising performance especially with poor initialization. It is proved in these papers that the observability of the linearized system is coincident with the original nonlinear system.

However, the IEKF has not been applied to multi-agent cases. In this paper, we design a distributed invariant EKF (DIEKF) for solving the problem of distributed state estimation using sensor networks in a 3D environment. To design DIEKF, we extend the well-known CI to matrix Lie groups for the first time. The proposed algorithm is fully distributed that each agent only estimates its own state and covariance by using the information among the neighborhood. The algorithm is applied to track target 3D motion in a camera network. The accuracy and consistency in both position and orientation are improved as compared against the QDEKF [7].

II. PRELIMINARIES

A. Notation and Definitions

We denote $\mathbf{0}_{m\times n}$ as a $m \times n$ zero matrix, and $\mathbf{I}_n$ ($\mathbf{0}_n$) as $n \times n$ square identity (zero) matrix. Given a 3×1 vector $\mathbf{q} = [q_1, q_2, q_3]^\top$, its skew symmetric matrix is defined as

$$(\mathbf{q})_\times = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix},$$

and its projection function is defined as $\Pi(\mathbf{q}) = \frac{1}{q_3}[q_1, q_2]$. The Jacobian of the projection function is computed as

$$\mathbf{H}_p(\mathbf{q}) = \frac{1}{q_3}\begin{bmatrix} 1 & 0 & -\frac{q_1}{q_3} \\ 0 & 1 & -\frac{q_2}{q_3} \end{bmatrix}. \tag{1}$$

In a network of $\mathcal{M}$ agents, we define a directed graph $\mathfrak{G} = (\mathcal{V}, \mathcal{E})$ to represent the communication topology among agents, where $\mathcal{V}$ indicates the set of all the agents, and $\mathcal{E}$ stands for the set of communication links defined as $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$. Specifically, if $(j, i) \in \mathcal{E}$, agent j is a neighbor of agent i, and agent i can receive information from agent j. We assume that self communication always exists, i.e., $(i, i) \in \mathcal{E}$, $\forall i \in \mathcal{V}$. The set of all the communicating neighbors of agent i is defined as $\mathcal{N}_i = \{j | (j, i) \in \mathcal{E},\ j \in \mathcal{V}\}$.

B. Problem Formulation

Consider a network of agents in the 3-D environment with fixed and known positions aiming to cooperatively track a moving target's state. Each agent can communicate with its neighbors and is equipped with an on-board camera. Denote G, T, C_i as the global frame, the target frame, and ith agent's camera frame, respectively. Let ${}^G_T R$ be the rotation matrix that describes the rotation from T to G. Let ${}^G v$ and ${}^G p$ be the target's velocity and position in the global frame. Let ${}^G p_{C_i}$ be the position of agent i's camera in the global frame. The target has additional non-representative feature points, whose relative positions are unknown but fixed in the target frame. For convenience, we assume that there is only one non-representative feature point. However, the state can be augmented to include multiple non-representative points. Let ${}^T p_f$ be the position of the non-representative feature point in the target frame.

The state of the target is represented as

$$\mathsf{x} = ({}^G_T R \;\; {}^G v \;\; {}^G p \;\; {}^T p_f), \tag{2}$$

which includes the target's 6-DoF pose ${}^G_T R$ and ${}^G p$, the linear velocity ${}^G v$ in the global frame, and the 3-D position of a non-representative point in the target frame ${}^T p_f$. The individual dynamics of the state is given as

$$\begin{aligned} {}^G_T \dot{R} &= {}^G_T R(\omega - n_\omega)_\times, \\ {}^G \dot{v} &= {}^G_T R(a - n_a) + g, \\ {}^G \dot{p} &= {}^G v, \\ {}^T \dot{p}_f &= \mathbf{0}_{3\times 1}, \end{aligned} \tag{3}$$

where ω and a are, respectively, the angular velocity and the linear acceleration of the target in the target frame, the corresponding n_ω and n_a are white Gaussian noises, and g is the gravity vector.

The measurements of the representative feature at time t_k (specifying the target's position) and the non-representative feature obtained by agent i's camera are given by, respectively,

$$\begin{aligned} z_i^k &= \Pi({}^{C_i} p^k) + n_i^k, \\ z_{f_i}^k &= \Pi({}^{C_i} p_f^k) + n_{f_i}^k, \end{aligned} \tag{4}$$

where n_i^k and $n_{f_i}^k$ are the measurement noises of agent i's camera at time t_k, assumed to be white Gaussian, and ${}^{C_i} p^k$ and ${}^{C_i} p_f^k$ denote, respectively, the representative feature's position (target's position) and the non-representative feature's position in agent i's camera frame at time t_k. The objective of our work is to let each agent compute an accurate estimate of the target's state.

C. Lie Group and Lie Algebra

Here we briefly introduce the matrix Lie group theory that we will use to derive our algorithm. The material is adopted from [10]. A matrix Lie group $\mathcal{G}$ is a subset of square invertible $N \times N$ matrices satisfying $\mathbf{I}_N \in \mathcal{G}$, $\forall a \in \mathcal{G},\ a^{-1} \in \mathcal{G}$, and $\forall a, b \in \mathcal{G},\ ab \in \mathcal{G}$.

Its Lie algebra is denoted as $\mathfrak{g}$, which is a vector space with the same dimension as $\mathcal{G}$. For convenience, let $(\cdot)^\wedge : \mathbb{R}^{\dim \mathfrak{g}} \to \mathfrak{g}$ be the linear map that transforms the elements in the Lie algebra to the corresponding matrix representation. The exponential map is further defined as $\exp(\xi) = \exp_m(\xi^\wedge) \in \mathcal{G}$, where $\xi \in \mathbb{R}^{\dim \mathfrak{g}}$ is an element in $\mathfrak{g}$, and $\exp_m$ is the matrix exponential. The logarithm map, which is the inverse function of the exponential map, is denoted by $\log(\cdot)$, and satisfies $\log(\cdot) = (\log_m(\cdot))^\vee : \mathcal{G} \to \mathbb{R}^{\dim \mathfrak{g}}$, where $\log_m$ is the matrix logarithm, and $(\cdot)^\vee$ is the inverse operator of $(\cdot)^\wedge$.

Let $\mathbf{X}_t \in \mathcal{G}$ be the state of a system at time t. The dynamics of the system is denoted as $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{X}_t = f_{u_t}(\mathbf{X}_t)$, where u_t is the input. We denote $\mathbf{X}_t$ as the true state, and $\bar{\mathbf{X}}_t$ as the estimate of the state. The right invariant estimation error is then defined as

$$\eta_t = \mathbf{X}_t(\bar{\mathbf{X}}_t)^{-1}. \tag{5}$$

The error (5) is invariant to right multiplication of any element $\Upsilon \in \mathcal{G}$.

Let $\mathbf{I}_d \in \mathcal{G}$ be the identity element of $\mathcal{G}$. If the dynamics of the system satisfy $f_{u_t}(\mathbf{X}_t\bar{\mathbf{X}}_t) = f_{u_t}(\mathbf{X}_t)\bar{\mathbf{X}}_t + \mathbf{X}_t f_{u_t}(\bar{\mathbf{X}}_t) - \mathbf{X}_t f_{u_t}(\mathbf{I}_d)\bar{\mathbf{X}}_t$, the system is group affine. Then the right invariant error dynamics satisfy $\frac{\mathrm{d}}{\mathrm{d}t}\eta_t = g_{u_t}(\eta_t)$, where $g_{u_t}(\eta_t) = f_{u_t}(\eta_t) - \eta_t f_{u_t}(\mathbf{I}_d)$. Define A_t as a matrix satisfying $g_{u_t}(\exp(\xi_t)) \triangleq (A_t\xi_t)^\wedge + \mathcal{O}(||\xi_t||^2)$, and let ξ_t be the solution of $\frac{\mathrm{d}}{\mathrm{d}t}\xi_t = A_t\xi_t$. From the log-linear property of the error η_t, given $\eta_0 = \exp(\xi_0)$, for $t \geq 0$, the error η_t can be computed from ξ_t by $\eta_t = \exp(\xi_t)$.

D. Matrix Lie Group Representation

As shown in [10], [12], the state collection shown in (2) forms a matrix Lie group $\mathrm{SE}_3(3)$ represented as

$$\mathsf{X} = \begin{bmatrix} {}^G_T R & {}^G v & {}^G p & {}^T p_f \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{6\times 6}.$$

Let X_t be the state representation at time t, and $\bar{\mathsf{X}}_t$ be the state estimate. Define the right invariant estimation error η_t given as

$$\eta_t = \mathsf{X}_t(\bar{\mathsf{X}}_t)^{-1} = \begin{bmatrix} \theta_{11}, & \theta_{12}, & \theta_{13}, & \theta_{14} \\ \mathbf{0}_{1\times 3}, & 1, & 0, & 0 \\ \mathbf{0}_{1\times 3}, & 0, & 1, & 0 \\ \mathbf{0}_{1\times 3}, & 0, & 0, & 1 \end{bmatrix}$$

where the individual terms are calculated as

$$\begin{aligned} \theta_{11} &= {}^G_T R_t ({}^G_T \bar{R}_t)^\top, \\ \theta_{12} &= {}^G v_t - {}^G_T R_t ({}^G_T \bar{R}_t)^\top ({}^G \bar{v}_t) \\ \theta_{13} &= {}^G p_t - {}^G_T R_t ({}^G_T \bar{R}_t)^\top ({}^G \bar{p}_t) \\ \theta_{14} &= {}^T p_{f_t} - {}^G_T R_t ({}^G_T \bar{R}_t)^\top ({}^T \bar{p}_{f_t}) \end{aligned}$$

The error vector ξ_t, defined in the Lie algebra of $\mathrm{SE}_3(3)$, denoted by $\mathrm{se}_3(3)$, is given by

$$\xi_t = [(\xi_{R_t})^\top \ (\xi_{v_t})^\top \ (\xi_{p_t})^\top \ (\xi_{p_{f_t}})^\top]^\top \in \mathbb{R}^{12},$$

where ξ_{R_t}, ξ_{v_t}, ξ_{p_t}, and $\xi_{p_{f_t}} \in \mathbb{R}^3$. Here, $\eta_t = \exp(\xi_t) = \exp_m(\xi_t^\wedge)$, with $\xi_t^\wedge$ given as

$$\xi_t^\wedge = \begin{bmatrix} (\xi_{R_t})_\times & \xi_{v_t} & \xi_{p_t} & \xi_{p_{f_t}} \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{6\times 6}. \tag{6}$$

It is shown in [12] that (3) without noise is group affine, and the dynamics of ξ_t are given as

$$\frac{\mathrm{d}}{\mathrm{d}t}\xi_t = A_t \xi_t - \mathrm{Ad}_{\bar{\mathsf{X}}_t} \mathsf{U}_t, \tag{7}$$

where $A_t = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ (g)_\times & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}$, and $\mathrm{Ad}_{\bar{\mathsf{X}}_t}$ denotes the adjoint of $\mathrm{SE}_3(3)$ at $\bar{\mathsf{X}}_t$, and $\mathsf{U}_t = [n_{\omega_t}^\top, n_{a_t}^\top, \mathbf{0}_{1\times 6}]^\top$.

III. Proposed Algorithm

For notation simplicity, let X^k denote the target state at time t_k. Also let $\bar{\mathsf{X}}_i^k$ and $\hat{\mathsf{X}}_i^k$ denote, respectively, agent i's prior and posterior estimates of the target state at time t_k. Let $\eta_i^{k|k-1} = \mathsf{X}^k(\bar{\mathsf{X}}_i^k)^{-1} = \exp(\xi_i^{k|k-1})$, and $\eta_i^{k|k} = \mathsf{X}^k(\hat{\mathsf{X}}_i^k)^{-1} = \exp(\xi_i^{k|k})$ denote, respectively, agent i's prior and posterior estimation errors in $\mathrm{SE}_3(3)$. Here $\xi_i^{k|k-1}$ and $\xi_i^{k|k}$ denote, respectively, the prior and posterior estimation errors in $\mathrm{se}_3(3)$. The covariances associated with $\xi_i^{k|k-1}$ and $\xi_i^{k|k}$ are denoted, repectively, as $\bar{P}_i^k$ and $\hat{P}_i^k$.

The first step is to propagate the posterior estimation pair $(\hat{\mathsf{X}}_i^{k-1}, \hat{P}_i^{k-1})$ to obtain the prior estimation pair $(\bar{\mathsf{X}}_i^k, \bar{P}_i^k)$. We can follow Appendix A is [12] to discretize (3) without process noises to obtain $\bar{\mathsf{X}}_i^k$ from $\hat{\mathsf{X}}_i^{k-1}$, ω^{k-1} and a^{k-1} (the target's angular velocity and acceleration at time t_{k-1}), and $\Delta t = t_k - t_{k-1}$. To propagate the covariance from $\hat{P}_i^{k-1}$ to $\bar{P}_i^k$, we use

$$\begin{aligned} Q_i^{k-1} &= \mathrm{Ad}_{\hat{\mathsf{X}}_i^{k-1}} \mathrm{cov}(\mathsf{U}^{k-1}) (\mathrm{Ad}_{\hat{\mathsf{X}}_i^{k-1}})^\top, \\ Q_d^{k-1} &= \Phi(t_k, t_{k-1}) Q_i^{k-1} \Phi(t_k, t_{k-1}) \Delta t, \\ \bar{P}_i^k &= \Phi(t_k, t_{k-1}) \hat{P}_i^{k-1} \Phi(t_k, t_{k-1}) + Q_d^{k-1}, \end{aligned} \tag{8}$$

where the state transition matrix $\Phi(t_k, t_{k-1})$ associated with A_t defined in (7) is computed as

$$\Phi(t_k, t_{k-1}) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ (g)_\times \Delta t & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \Delta t & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix},$$

U^{k-1} denotes U_t defined after (7) at time t_{k-1}, and $\mathrm{cov}(\cdot)$ denotes the covariance.

In the second step, agent i aims to update its local estimate by communicating with its neighbors. The goal is to fuse all the prior estimates among the neighbors, i.e., $(\bar{\mathsf{X}}_j^k, \bar{P}_j^k)$, $j \in \mathcal{N}_i^k$, to obtain an intermediate estimation pair $(\check{\mathsf{X}}_i^k, \check{P}_i^k)$. By doing so, the agents with better estimates would help those with poor estimates. For example, blind agents, which refer to the agents that themselves and their one-hop communication neighbors cannot see the target, would have poor estimates. Because of the communication from previous timesteps, and the fact that the agents are estimating the same target, $\bar{\mathsf{X}}_i^k$ and $\bar{\mathsf{X}}_j^k$ would be correlated. However, in the distributed setting, it is not possible to keep tracking the cross-correlations and hence they are unknown. The CI algorithm [14] can be used to fuse estimates with unknown correlations. However, the CI algorithm is applicable in the vector space and it is not clear how to fuse the Lie group elements $\bar{\mathsf{X}}_i^k$ and $\bar{\mathsf{X}}_j^k$, which are in $\mathrm{SE}_3(3)$. Next, we extend the CI algorithm in the context of error-state EKF in Lie groups to estimate the error states represented in $\mathrm{se}_3(3)$ and map back to $\mathrm{SE}_3(3)$.

Recall that for each $j \in \mathcal{N}_i^k$, $\eta_j^{k|k-1} = \mathsf{X}^k(\bar{\mathsf{X}}_j^k)^{-1}$ is the prior estimation error in $\mathrm{SE}_3(3)$, and $\xi_j^{k|k-1} = \log(\eta_j^{k|k-1})$ is the corresponding prior estimation error in $\mathrm{se}_3(3)$. We will make use of $\bar{\mathsf{X}}_j^k$, $j \in \mathcal{N}_i^k$, to identify the estimates of $\xi_i^{k|k-1}$, and then fuse these estimates. Note that for each $j \in \mathcal{N}_i^k$,

$$\begin{aligned} \exp(\xi_j^{k|k-1}) &= \mathsf{X}^k(\bar{\mathsf{X}}_j^k)^{-1} = \mathsf{X}^k(\bar{\mathsf{X}}_i^k)^{-1}(\bar{\mathsf{X}}_i^k)(\bar{\mathsf{X}}_j^k)^{-1} \\ &= \exp(\xi_i^{k|k-1})(\bar{\mathsf{X}}_i^k)(\bar{\mathsf{X}}_j^k)^{-1}. \end{aligned}$$

It follows that $\bar{\mathsf{X}}_j^k(\mathsf{X}_i^k)^{-1} = \exp(-\xi_j^{k|k-1})\exp(\xi_i^{k|k-1})$. According to the Baker–Campbell–Hausdorff (BCH) formula, the exponential map satisfies the property $\exp(\xi_1)\exp(\xi_2) \approx \exp(\xi_1 + \xi_2)$, if ξ_1 and ξ_2 are small. Then (5) can be rewritten as $\bar{\mathsf{X}}_j^k(\bar{\mathsf{X}}_i^k)^{-1} \approx \exp(\xi_i^{k|k-1} - \xi_j^{k|k-1})$, which implies that $\xi_i^{k|k-1} - \xi_j^{k|k-1} \approx \log(\bar{\mathsf{X}}_j^k(\bar{\mathsf{X}}_i^k)^{-1})$, or equivalently $\xi_i^{k|k-1} - \log(\bar{\mathsf{X}}_j^k(\bar{\mathsf{X}}_i^k)^{-1}) \approx \xi_j^{k|k-1}$, for each $j \in \mathcal{N}_i^k$. As a result, each $\log(\bar{\mathsf{X}}_j^k(\bar{\mathsf{X}}_i^k)^{-1})$, $j \in \mathcal{N}_i^k$, can be treated as one prior estimate of $\xi_i^{k|k-1}$, and the corresponding estimation error is given by $\xi_j^{k|k-1}$ with covariance $\bar{P}_j^k$. We can use the CI algorithm to fuse all estimation pairs $\left(\log(\bar{\mathsf{X}}_j^k(\bar{\mathsf{X}}_i^k)^{-1}), \bar{P}_j^k\right)$ to get an intermediate estimation pair $(\breve{\xi}_i^{k|k-1}, \breve{P}_i^k)$ for $\xi_i^{k|k-1}$ by

$$\breve{P}_i^k = \left[\sum_{j\in\mathcal{N}_i^k} \pi_j^k (\bar{P}_j^k)^{-1}\right]^{-1},$$

$$\breve{\xi}_i^{k|k-1} = \breve{P}_i^k \left[\sum_{j\in\mathcal{N}_i^k} \pi_j^k (\bar{P}_j^k)^{-1} \log((\bar{\mathsf{X}}_j^k)(\bar{\mathsf{X}}_i^k)^{-1})\right],$$

where $\pi_j^k \in [0,1]$ and $\sum_{j\in\mathcal{N}_i} \pi_j^k = 1$. Note that as $i \in \mathcal{N}_i^k$, the prior estimate of $\xi_i^{k|k-1}$ by agent i is simply $\log(\bar{\mathsf{X}}_i^k(\bar{\mathsf{X}}_i^k)^{-1}) = \mathbf{0}_{12\times 1}$ with covariance $\bar{P}_i^k$, which is consistent with the definition of $\xi_i^{k|k-1}$. While π_j^k can be solved from an optimization problem, a simplified algorithm can be used to compute π_j^k according to [14]:

$$\pi_j^k = \frac{1/\mathrm{Tr}\{\bar{P}_j^k\}}{\sum_{j\in\mathcal{N}_i} 1/\mathrm{Tr}\{\bar{P}_j^k\}}.$$

The intermediate state estimate of X^k in $\mathrm{SE}_3(3)$, denoted by $\breve{\mathsf{X}}_i^k$, can be recovered from $\breve{\xi}_i^{k|k-1}$ by

$$\breve{\mathsf{X}}_i^k = \exp(\breve{\xi}_i^{k|k-1})\bar{\mathsf{X}}_i^k. \tag{9}$$

Now define $\varepsilon_i^{k|k-1}$ as the new error vector in $\mathrm{se}_3(3)$ with $\breve{\mathsf{X}}_i^k$ being the prior estimate of X^k satisfying

$$\exp(\varepsilon_i^{k|k-1}) = \mathsf{X}^k(\breve{\mathsf{X}}_i^k)^{-1}. \tag{10}$$

Because we are going to apply the error-state EKF on the intermediate estimation error $\varepsilon_i^{k|k-1}$, we need the corresponding covariance. Note that

$$\begin{aligned}
\mathsf{X}^k = \exp(\varepsilon_i^{k|k-1})\breve{\mathsf{X}}_i^k &= \exp(\xi_i^{k|k-1})\bar{\mathsf{X}}_i^k \\
\Rightarrow \exp(\varepsilon_i^{k|k-1})\exp(\breve{\xi}_i^{k|k-1})\bar{\mathsf{X}}_i^k &= \exp(\xi_i^{k|k-1})\bar{\mathsf{X}}_i^k \\
\Rightarrow \exp(\varepsilon_i^{k|k-1})\exp(\breve{\xi}_i^{k|k-1}) &= \exp(\xi_i^{k|k-1}) \\
\Rightarrow \exp(\varepsilon_i^{k|k-1}) &= \exp(\xi_i^{k|k-1})\exp(-\breve{\xi}_i^{k|k-1}) \\
\Rightarrow \exp(\varepsilon_i^{k|k-1}) &\approx \exp(\xi_i^{k|k-1} - \breve{\xi}_i^{k|k-1}) \\
\Rightarrow \varepsilon_i^{k|k-1} &\approx \xi_i^{k|k-1} - \breve{\xi}_i^{k|k-1},
\end{aligned} \tag{11}$$

and $\breve{P}_i^k$ is the estimated covariance for $\xi_i^{k|k-1} - \breve{\xi}_i^{k|k-1}$. As a result, $\breve{P}_i^k$ can be directly used as the estimated covariance for the new error $\varepsilon_i^{k|k-1}$.

The third step is to fuse agent i's intermediate estimation pair $(\breve{\mathsf{X}}_i^k, \breve{P}_i^k)$ with all the measurements from itself and its neighbors, i.e., representative feature measurements z_j^k, $\forall j \in \mathcal{N}_i^k$, and non-representative feature measurements $z_{f_j}^k$, $\forall j \in \mathcal{N}_i^k$, defined by (4). To calculate the Jacobian associated with z_j^k, denoted by H_j^k, first define

$${}^{C_j}\breve{p}_i^k = {}^{C_j}_G R\left({}^G\breve{p}_i^k - {}^G p_{C_j}\right), \tag{12}$$

as the linearization point in agent j's camera frame, where ${}^{C_j}_G R$ and ${}^G p_{C_j}$ together denote the 3-D pose of agent j's camera that is fixed and known. Then the difference between the true target's position and the linearization point in agent j's camera frame is calculated as

$$\begin{aligned}
{}^{C_j}p_i^k - {}^{C_j}\breve{p}_i^k &= {}^{C_j}_G R\left({}^G p^k - {}^G p_{C_j}\right) - {}^{C_j}_G R\left({}^G\breve{p}_i^k - {}^G p_{C_j}\right) \\
&= {}^{C_j}_G R\left({}^G p^k - {}^G\breve{p}_i^k\right).
\end{aligned}$$

Note that $\varepsilon_i^{k|k-1}$ is a column stack vector of $\varepsilon_{R_i}^{k|k-1}$, $\varepsilon_{v_i}^{k|k-1}$, $\varepsilon_{p_i}^{k|k-1}$, and $\varepsilon_{p_{f_i}}^{k|k-1}$. Recall from (10) that $\mathsf{X}^k = \exp(\varepsilon_i^{k|k-1})\breve{\mathsf{X}}_i^k \approx (\mathbf{I}_6 + (\varepsilon_i^{k|k-1})^\wedge)\breve{\mathsf{X}}_i^k$, where $(\cdot)^\wedge$ is defined by (6). The position of the target can be calculated as ${}^G p^k \approx [\mathbf{I}_3 + (\varepsilon_{R_i}^{k|k-1})_\times]{}^G\breve{p}_i^k + \varepsilon_{p_i}^{k|k-1} = -({}^G\breve{p}_i^k)_\times \varepsilon_{R_i}^{k|k-1} + \varepsilon_{p_i}^{k|k-1} + {}^G\breve{p}_i^k$. Hence the Jacobian H_j^k is calculated as

$$H_j^k = \mathbf{H}_p({}^{C_j}\breve{p}_i^k)({}^{C_j}_G R)\left[-({}^G\breve{p}_i^k)_\times,\ \mathbf{0}_3,\ \mathbf{I}_3,\ \mathbf{0}_3\right],$$

where $\mathbf{H}_p(\cdot)$ is defined by (1). To calculate the Jacobian associated with the non-representative feature, denoted by $H_{f_j}^k$, define

$${}^{C_j}\breve{p}_{f_i}^k = {}^{C_j}_G R\left({}^G_T\breve{R}_i^k({}^T\breve{p}_{f_i}^k) + {}^G\breve{p}_i^k - {}^G p_{C_j}\right) \tag{13}$$

as its linearization point. Then the difference between the true position of the non-representative feature and the linearization point is given by ${}^{C_j}p_{f_i}^k - {}^{C_j}\breve{p}_{f_i}^k = M_i^k \varepsilon_{R_i}^{k|k-1} + \varepsilon_{p_i}^{k|k-1} + {}^G_T\breve{R}_i^k \varepsilon_{p_{f_i}}^{k|k-1}$, where M_i^k is given by

$$M_i^k = -{}^G_T\breve{R}_i^k({}^T\breve{p}_{f_i}^k)_\times - ({}^G_T\breve{R}_i^{k\,T}\breve{p}_{f_i}^k)_\times - ({}^G\breve{p}_i^k)_\times. \tag{14}$$

Hence the Jacobian $H_{f_j}^k$ is calculated as $H_{f_j}^k = \mathbf{H}_p({}^{C_j}\breve{p}_{f_i}^k)({}^{C_j}_G R)\left[M_i^k,\ \mathbf{0}_3,\ \mathbf{I}_3,\ {}^G_T\breve{R}_i^k\right]$. Note that at time t_k, it is possible that some or even all neighbors do not see the representative or non-representative feature. Let $\bar{\mathcal{N}}_i^k \subset \mathcal{N}_i^k$ (respectively, $\bar{\mathcal{N}}_{f_i}^k \subset \mathcal{N}_i^k$) be the subset of agent i's neighbors that can see the representative feature (respectively, non-representative feature) at time t_k. Define $\tilde{H}_i^k$ by stacking all H_j^k, $j \in \bar{\mathcal{N}}_i^k$, and $H_{f_j}^k$, $j \in \bar{\mathcal{N}}_{f_i}^k$. Let $\tilde{C}_i^k$ be a block diagonal matrix with the measurement noise covariances C_j^k, $j \in \bar{\mathcal{N}}_i^k$, and $C_{f_j}^k$, $j \in \bar{\mathcal{N}}_{f_i}^k$. Let $S_i^k = \tilde{H}_i^k\breve{P}_i^k(\tilde{H}_i^k)^\top + \tilde{C}_i^k$. The Kalman gain K_i^k is calculated as

$$K_i^k = \breve{P}_i^k(\tilde{H}_i^k)^\top(S_i^k)^{-1}, \tag{15}$$

and the measurement residual y_i^k is given by stacking $z_j^k - \Pi({}^{C_j}\breve{p}_i^k)$, $j \in \bar{\mathcal{N}}_i^k$, and $z_{f_j}^k - \Pi({}^{C_j}\breve{p}_{f_i}^k)$, $j \in \bar{\mathcal{N}}_{f_i}^k$. Then

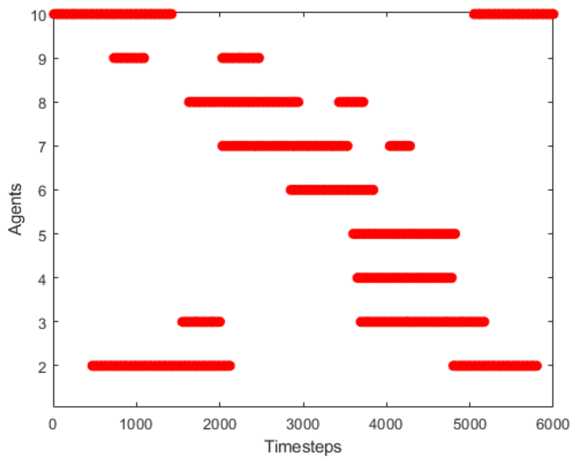

Fig. 1: Target's visibility to the agents. The red lines indicate the timesteps when the agent can directly observe the target's representative and non-representative features.

the posterior estimate of $\varepsilon_i^{k|k-1}$, denoted as $\hat{\varepsilon}_i^k$, and its covariance $\hat{P}_i^k$ are calculated as

$$\begin{aligned} \hat{\varepsilon}_i^k &= K_i^k y_i^k, \\ \hat{P}_i^k &= (\mathbf{I}_{12} - K_i^k \tilde{H}_i^k) \check{P}_i^k. \end{aligned} \tag{16}$$

In the last step, we recover the estimated state $\hat{\mathsf{X}}_i^k$ from $\hat{\varepsilon}_i^k$ and $\check{\mathsf{X}}_i^k$ by (9) as

$$\hat{\mathsf{X}}_i^k = \exp(\hat{\varepsilon}_i^k)\check{\mathsf{X}}_i^k. \tag{17}$$

Note that a similar procedure to (11) can be used to show that $\xi_i^{k|k} \approx \varepsilon_i^{k|k-1} - \hat{\varepsilon}_i^k$. Hence $\hat{P}_i^k$ can be used as the estimated covariance for $\xi_i^{k|k}$.

IV. Simulation Results

We apply the DIEKF algorithm to solve the distributed state estimation problem where 10 fixed agents equipped with cameras are employed to track the 3-D motion of a drone. The positions of the agents are known and the target's state is to be estimated. A non-representative point is created in addition to the center feature point of the target. The position of the non-representative point in the target frame is unknown but is fixed. Based on the positions of the cameras and the drone, the status of which agent can sense the drone is shown in Figure 1. It is clear that each one of the agents is not able to see the target for a long period of time. Extensive Monte Carlo simulations are performed to validate the algorithm. The results are quantified by rooted mean square error (RMSE) that evaluates the accuracy, and normalized estimation error squared (NEES) which evaluates the consistency. The comparison to the results of our previous QDEKF algorithm [7] is also included.

We set the noise of the linear acceleration ω and angular velocity a to be white Gaussian noise with standard deviations of 6.6968×10^{-3} rad$/(\mathrm{s}\sqrt{\mathrm{Hz}})$ and 2×10^{-2} m$/(\mathrm{s}^2\sqrt{\mathrm{Hz}})$ respectively. To show the result of cooperative tracking, we define a communication rate, which means that each agent has a certain probability to communicate with other agents. For instance, 20% communication means that each agent has 20% probability to communicate with each one of the other agents. Hence the set of communication neighbors are randomly determined at every time step.

In the first test, we assume that the agents have the knowledge of the target's ground truth state at the first timestep. So we initialize our estimator with the ground truth, and give it a very small initial covariance. We conduct 50 Monte Carlo simulations and calculate the average result. Figure 2 shows the averaged position RMSE (PRMSE) and averaged orientation RMSE (ORMSE) for different percentages of the communication. A comparison with the QDEKF algorithm

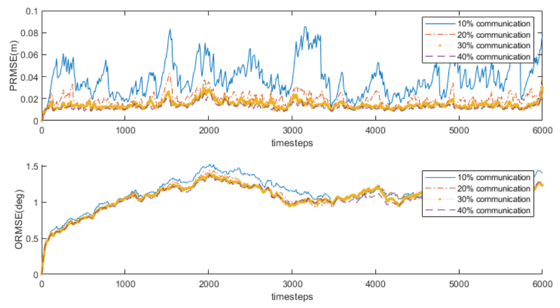

Fig. 2: Averaged PRMSE and ORMSE using DIEKF at different communication rates without initialization errors

in our previous work [7] is shown in Table I. It is clear

TABLE I: Averaged RMSE for the estimated target pose over 50 Monte-Carlo runs and all timesteps.

communication rate		10 %	20 %	30 %	40%
QDEKF	PRMSE (m)	0.0884	0.0352	0.0206	0.0158
	ORMSE (deg)	1.4098	1.2291	1.1678	1.1311
DIEKF	PRMSE (m)	0.0387	0.0195	0.0144	0.0124
	ORMSE (deg)	1.1518	1.0773	1.0795	1.0611

that DIEKF outperforms the QDEKF in estimating both position and orientation. In addition, the estimation accuracy improves as communication rate increases.

We further show the NEES result at 40% communication rate. As shown in Figure 3, while the orientation NEES (ONEES) appears to be similar, the position NEES (PNEES) for DIEKF is closer to 3 as compared to the QDEKF algorithm. This indicates the improvement of the consistency [15].

In the second test, we assume that the agents do not have perfect knowledge of the target's ground truth state at the first timestep. We initialize our state estimator to a value near the ground truth, and give it a larger covariance. Similarly, we conduct 50 Monte Carlo simulations on the DIEFK and QDEKF with the same initialized state estimate and equivalent covariance. Figures 4 and 5 show the result for both DIEKF and QDEKF using the same initialization, measurements and noise. Compared with the results of QDEKF, our DIEKF algorithm obviously converges faster in the position estimation, and is more accurate in general.

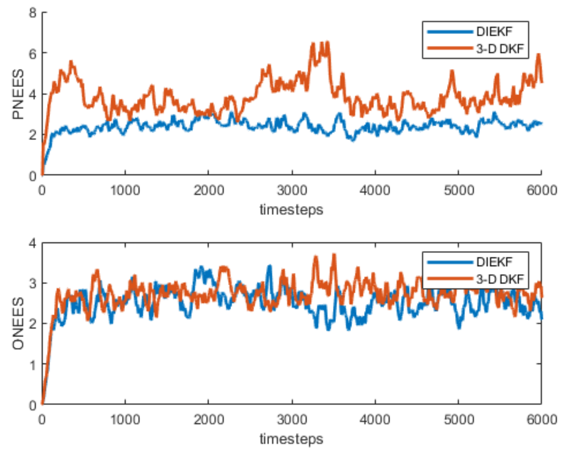

Fig. 3: PNEES and ONEES for DIEKF at 40% communication rate without initialization errors

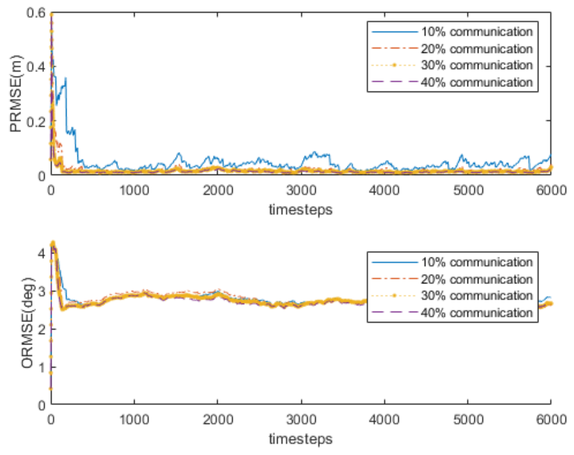

Fig. 4: PRMSE and ORMSE for DIEKF at different communication rates with initialization errors

Overall, our algorithm can accurately track the trajectory of the target in the 3-D space, and can maintain consistency.

V. Conclusion

In this paper, by using the matrix Lie group representation of the state, we introduced a new DIEKF algorithm that yields consistent and accurate estimates of the target in the 3-D space over the sensor networks. The proposed algorithm requires only one communication iteration with its communication neighbors at every time instant. Further the algorithm is shown to be robust to changing communication topology and blind agents. These properties ensure that our approach can have a wide application in multi-agent scenarios. The performance of the proposed algorithm is tested via Monte Carlo simulations. Some key performance are compared with the algorithm that did not use the Lie group representation for the state.

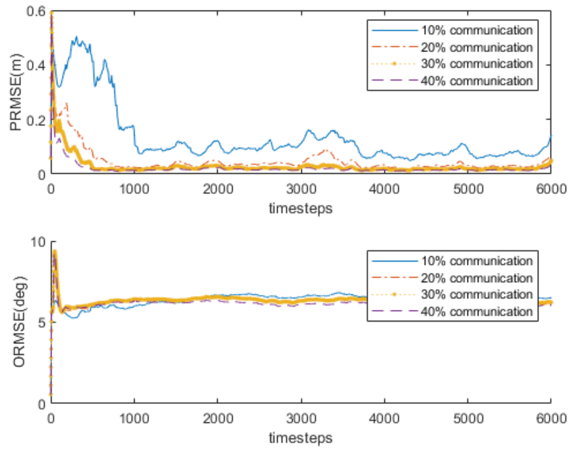

Fig. 5: PRMSE and ORMSE for of QDEKF at different communication rates with initialization errors

References

[1] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control systems magazine*, vol. 27, no. 2, pp. 71–82, 2007.

[2] S. J. Julier and J. K. Uhlmann, "A non-divergent estimation algorithm in the presence of unknown correlations," in *Proceedings of the American Control Conference*, vol. 4, 1997, pp. 2369–2373.

[3] G. Battistelli, L. Chisci, G. Mugnai, A. Farina, and A. Graziano, "Consensus-based linear and nonlinear filtering," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1410–1415, 2014.

[4] G. Battistelli and L. Chisci, "Kullback–leibler average, consensus on probability densities, and distributed state estimation with guaranteed stability," *Automatica*, vol. 50, no. 3, pp. 707–718, 2014.

[5] X. He, W. Xue, and H. Fang, "Consistent distributed state estimation with global observability over sensor network," *Automatica*, vol. 92, pp. 162–172, 2018.

[6] S. Wang and W. Ren, "On the convergence conditions of distributed dynamic state estimation using sensor networks: A unified framework," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1300–1316, 2017.

[7] P. Zhu and W. Ren, "Distributed kalman filter for 3-d moving object tracking over sensor networks," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 2418–2423.

[8] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.

[9] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3d attitude estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, vol. 2, p. 2005, 2005.

[10] A. Barrau and S. Bonnabel, "The invariant extended kalman filter as a stable observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2016.

[11] S. Bonnabel, P. Martin, and P. Rouchon, "Non-linear symmetry-preserving observers on lie groups," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1709–1713, 2009.

[12] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended kalman filtering for robot state estimation," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 402–430, 2020.

[13] T. Zhang, K. Wu, J. Song, S. Huang, and G. Dissanayake, "Convergence and consistency analysis for a 3-d invariant-ekf slam," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 733–740, 2017.

[14] W. Niehsen, "Information fusion based on fast covariance intersection filtering," in *Proceedings of the IEEE International Conference on Information Fusion*, vol. 2, 2002, pp. 901–904.

[15] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.