2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Relative Position Estimation Between Two UWB Devices with IMUs

Charles Champagne Cossette, Mohammed Shalaby, David Saussié, James Richard Forbes, Jérôme Le Ny

***Abstract*— For a team of robots to work collaboratively, it is crucial that each robot have the ability to determine the position of their neighbors, relative to themselves, in order to execute tasks autonomously. This letter presents an algorithm for determining the three-dimensional relative position between two mobile robots, each using nothing more than a single ultra-wideband transceiver, an accelerometer, a rate gyro, and a magnetometer. A sliding window filter estimates the relative position at selected keypoints by combining the distance measurements with acceleration estimates, which each agent computes using an on-board attitude estimator. The algorithm is appropriate for real-time implementation, and has been tested in simulation and experiment, where it comfortably outperforms standard estimators. A positioning accuracy of less than 1 meter is achieved with inexpensive sensors.**

I. Introduction

The ability to determine the relative position between two robots, or *agents*, is needed in many applications. For instance, multi-robot tasks such as collaborative exploration and mapping, search and rescue, and formation control, each require relative position information. Access to inter-robot distance measurements is becoming increasingly accessible due to technologies such as ultra-wideband radio (UWB). UWB in particular is attractive due to its low cost, low power, high accuracy, and ability to function in GPS-denied environments. Such advantages have even warranted the presence of UWB transceivers in smartphones and smartwatches, which could be used to provide position information of one user relative to another or, more generally, any other UWB transceiver. However, determining a full three-dimensional relative position estimate from a single distance measurement is impossible. There is an infinite set of relative positions that will produce the same distance measurement. Hence, to realize an observable relative positioning solution, more information is required.

Distance-based positioning is very commonly achieved by measuring distances to several landmarks with known positions, referred to as *anchors* when using UWB [1, 2]. However, the problem of estimating the relative position between moving robots, without any external reference, is much more difficult. Nevertheless, there are several papers that achieve relative positioning from distance measurements, and do so by using additional sensors. For example, cameras are often used in addition to UWB [3], but vision requires substantial computational capabilities, which imposes a lower bound on the size of the agent. In [4], the presence of some agents with two UWB tags makes the relative positions observable. The observability analysis provided by [5] concludes that when velocity information is available in addition to single-distance measurements, the relative position is observable provided that a *persistency of excitation* condition is satisfied, meaning that the agents must be in continuous relative motion. This is simulated in [6], where velocity measurements are assumed to be available, and experimented in [7], however GPS is used to provide displacement information for the agents. Cameras with optical flow are used to measure agent motion in [8–10], and [11] proposes a particle filter to estimate the relative positions of a group of mobile UWB devices, which also have IMUs. However, their solution requires a central server to perform the estimation task. The literature lacks three-dimensional solutions that use only UWB and IMU measurements.

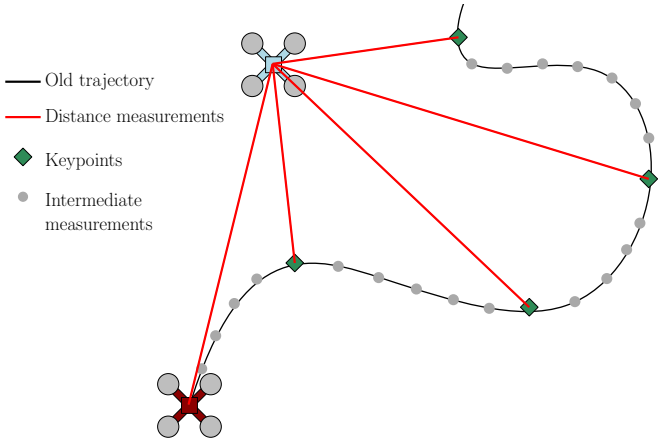

Fig. 1. An example scenario where two quadrotors possess UWB modules, providing distance measurements between them. Under sufficient motion, it is possible to determine the relative position between them in a common reference frame.

The main contribution of this letter is a three-dimensional relative position estimation algorithm for two mobile agents, each using nothing more than a single sensor measuring the distance between them, and a 9-DOF IMU. No fixed infrastructure, GPS, cameras, or heavy computing is

*This work was supported by the FRQNT under grant 2018-PR-253646, with funding also acknowledged from CFI JELF, the William Dawson Scholar Program, and the NSERC Discovery Grant Program.

C. C. Cossette, M. Shalaby, J.R. Forbes are with the Department of Mech. Engineering, McGill University. charles.cossette@mail.mcgill.ca, mohammed.shalaby@mail.mcgill.ca, james.richard.forbes@mcgill.ca

D. Saussié, J. Le Ny, are with the Department of Electrical Engineering, Polytechnique Montréal. d.saussie@polymtl.ca jerome.le-ny@polymtl.ca

required, and the position is resolved in a known local frame, such as an East-North-Up reference frame. The proposed solution is easily decentralizable, and achieves sub-meter level accuracy in experiment. The algorithm requires persistency of excitation, meaning that the agents must be moving in a non-planar trajectory.

The remainder of this letter is as follows. Section II introduces the notation and probabilistic estimation tools used in this letter. Section III outlines the proposed algorithm, with simulation and experimental results presented in Sections IV and V, respectively.

II. Preliminaries

A. Notation

In this letter, $p(\mathbf{x})$ denotes the joint probability density function (PDF) of the continuous random variable $\mathbf{x} \in \mathbb{R}^n$. If $\mathbf{x}$ is normally distributed with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, it is written as $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The compressed notation $\mathbf{x}_{0:K} = [\mathbf{x}_0^\mathsf{T} \dots \mathbf{x}_K^\mathsf{T}]^\mathsf{T}$ is used for brevity. A bolded $\mathbf{1}$ indicates an appropriately-sized identity matrix.

An arbitrary reference frame 'a' is denoted $\mathcal{F}_a$. Physical vectors resolved in $\mathcal{F}_a$ are denoted with a subscript $\mathbf{v}_a$. The same physical vector resolved in a different frame $\mathcal{F}_b$ can be related by a direction cosine matrix (rotation matrix), denoted $\mathbf{C}_{ab} \in SO(3)$, such that $\mathbf{v}_a = \mathbf{C}_{ab}\mathbf{v}_b$ [12]. A direction cosine matrix (DCM) $\mathbf{C}_{ab}$ can be parameterized using a rotation vector, denoted $\boldsymbol{\phi} \in \mathbb{R}^3$, using $\mathbf{C}_{ab} = \exp\left(\boldsymbol{\phi}^\times\right)$, where $(\cdot)^\times$ denotes the skew-symmetric cross-product matrix operator. In the context of this letter, $\mathcal{F}_b$ is associated with an agent's body frame, and $\mathcal{F}_a$ refers to some local common frame, such as an East-North-Up frame.

B. Maximum A Posteriori Estimation

Consider generic, nonlinear process and measurement models, given respectively by

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}, \qquad k = 1, \dots, K, \quad (1)$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k) + \mathbf{v}_k, \qquad k = 0, \dots, K, \quad (2)$$

where $\mathbf{x}_k \in \mathbb{R}^n$ denotes the system state at time step k, $\mathbf{u}_{k-1}$ is the system input, and $\mathbf{y}_k$ is the output. The terms $\mathbf{w}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ and $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ represent random process and measurement noise, respectively.

The *maximum a posteriori* framework aims to find the states $\mathbf{x}_{0:K}$ which maximize the posterior probability density function, given a history of input measurements $\mathbf{u}_{0:K-1}$, output measurements $\mathbf{y}_{0:K}$, and a prior distribution of the initial state, $\mathbf{x}_0 \sim \mathcal{N}(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0)$. That is, the estimate $\hat{\mathbf{x}}_{0:K}$ is given by

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x} | \check{\mathbf{x}}_0, \mathbf{u}_{0:K-1}, \mathbf{y}_{0:K}), \quad (3)$$

where $\mathbf{x} = \mathbf{x}_{0:K}$ has been written without subscripts to reduce notation. It is well known that (3) is equivalently posed as a nonlinear least-squares problem [13, Ch. 3.1] given by

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\mathbf{e}(\mathbf{x})^\mathsf{T}\mathbf{W}\mathbf{e}(\mathbf{x}), \quad (4)$$

where

$$\mathbf{e}(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_0(\mathbf{x}) \\ \mathbf{e}_{u,1}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{u,K}(\mathbf{x}) \\ \mathbf{e}_{y,0}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{y,K}(\mathbf{x}) \end{bmatrix}, \quad \begin{aligned} \mathbf{e}_0(\mathbf{x}) &= \mathbf{x}_0 - \check{\mathbf{x}}_0, \\ \mathbf{e}_{u,k}(\mathbf{x}) &= \mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}), \\ \mathbf{e}_{y,k}(\mathbf{x}) &= \mathbf{y}_k - \mathbf{g}(\mathbf{x}_k), \\ \mathbf{W} &= \operatorname{diag}(\check{\mathbf{P}}_0^{-1}, \mathbf{Q}_0^{-1}, \dots, \mathbf{Q}_{K-1}^{-1}, \\ &\qquad \mathbf{R}_0^{-1}, \dots, \mathbf{R}_K^{-1}). \end{aligned} \quad (5)$$

This is known as the full *batch nonlinear least-squares* estimator. The optimization problem (4) can be solved iteratively using the *Gauss-Newton* algorithm, which starts with an initial state estimate $\mathbf{x}^{(0)}$, and computes an update $\delta\mathbf{x}^{(\ell)}$ to the ℓ^{th} iteration by solving

$$(\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H})\delta\mathbf{x}^{(\ell)} = -\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{e}(\mathbf{x}^{(\ell)}), \quad (6)$$

where

$$\mathbf{H} = \left.\frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^{(\ell)}} = \begin{bmatrix} \mathbf{1} & & & & \\ -\mathbf{A}_0 & \mathbf{1} & & & \\ & \ddots & \ddots & & \\ & & -\mathbf{A}_{K-1} & \mathbf{1} & \\ -\mathbf{C}_0 & & & & \\ & -\mathbf{C}_1 & & & \\ & & \ddots & & \\ & & & & -\mathbf{C}_K \end{bmatrix},$$

where $\mathbf{A}_k$ and $\mathbf{C}_k$ are the Jacobians of the process and measurement models, respectively, with respect to the state $\mathbf{x}_k$. The state estimate is updated with $\mathbf{x}^{(\ell+1)} = \mathbf{x}^{(\ell)} + \delta\mathbf{x}^{(\ell)}$, and the process is repeated until convergence. In practice, variants such as the *Levenberg-Marquart* algorithm, are used to improve convergence.

In this batch framework, local observability of the system defined by (1)-(2) amounts to showing that the matrix $\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H}$ is full rank when the prior is ignored. In [13, Ch. 3.1], it is shown that this culminates in the equivalent requirement that

$$\operatorname{rank}\begin{bmatrix} \mathbf{C}_0^\mathsf{T} & \mathbf{A}_0^\mathsf{T}\mathbf{C}_1^\mathsf{T} & \mathbf{A}_0^\mathsf{T}\mathbf{A}_1^\mathsf{T}\mathbf{C}_2^\mathsf{T} & \dots & \mathbf{A}_0^\mathsf{T}\dots\mathbf{A}_{K-1}^\mathsf{T}\mathbf{C}_K^\mathsf{T} \end{bmatrix} = n. \quad (7)$$

C. Sliding Window Filtering

The *sliding window filter* is a batch estimation framework with constant time and memory requirements, achieving so by *marginalizing out* older states [14, 15]. Consider a scenario where a robot travels until time t_{k_1}, at which point it performs a full batch estimate of its state history, to produce $\hat{\mathbf{x}}_{0:k_1}$. It then continues to travel until time t_{k_2}, adding the new states $\mathbf{x}_{k_1+1:k_2}$ to its state history. The m oldest states $\mathbf{x}_{0:m-1}$ are then marginalized out, thus removing them from the optimization problem being solved at t_{k_2}. The remaining states from the previous window's estimate are $\mathbf{x}_{m:k_1}$.

$$\underbrace{\underbrace{\mathbf{x}_0 \;\; \mathbf{x}_1 \;\; \dots \;\; \mathbf{x}_{m-1}}_{\text{marginalized out}} \;\; \underbrace{\mathbf{x}_m \;\; \dots \;\; \mathbf{x}_{k_1}}_{\text{old window of length } K=k_1+1}}_{\text{new window of length } K=k_2-m+1} \;\; \mathbf{x}_{k_1+1} \;\; \dots \;\; \mathbf{x}_{k_2}$$

The joint PDF of the new window, given the entire history of measurements until t_{k_2}, can be written as

$$p(\mathbf{x}_{m:k_2}|\check{\mathbf{x}}_0, \mathbf{u}_{0:k_2-1}, \mathbf{y}_{0:k_2}) = \eta p(\mathbf{x}_{m+1:k_2}|\mathbf{u}_{m:k_2-1}, \mathbf{y}_{m:k_2}, \mathbf{x}_m) p(\mathbf{x}_m|\check{\mathbf{x}}_0, \mathbf{u}_{0:m-1}, \mathbf{y}_{0:m-1}). \tag{8}$$

The crux of the marginalization process is to determine an expression for $p(\mathbf{x}_m|\check{\mathbf{x}}_0, \mathbf{u}_{0:m}, \mathbf{y}_{0:k_{m-1}})$, which takes the role of the new "prior" distribution of the new window. Performing marginalization properly, as opposed to naively removing the oldest states from the current estimation problem, is critical to maintaining an accurate, consistent estimator. To do this, consider instead the following PDF, for which an analytical expression is available,

$$p(\mathbf{x}_{0:m}|\check{\mathbf{x}}_0, \mathbf{u}_{0:m}, \mathbf{y}_{0:m-1}) = \beta \exp(-\frac{1}{2}\mathbf{e}_m(\mathbf{x}_{0:m})^\mathsf{T}\mathbf{W}_m\mathbf{e}_m(\mathbf{x}_{0:m})), \tag{9}$$

where

$$\mathbf{e}_m(\mathbf{x}_{0:m}) = \begin{bmatrix} \mathbf{e}_0^\mathsf{T} & \mathbf{e}_{u,1}^\mathsf{T} & \cdots & \mathbf{e}_{u,m}^\mathsf{T} & \mathbf{e}_{y,0}^\mathsf{T} & \cdots & \mathbf{e}_{y,m-1}^\mathsf{T} \end{bmatrix}^\mathsf{T},$$
$$\mathbf{W}_m = \mathrm{diag}(\check{\mathbf{P}}_0^{-1}, \mathbf{Q}_0^{-1}, \ldots, \mathbf{Q}_{m-1}^{-1}, \mathbf{R}_0^{-1}, \ldots, \mathbf{R}_{m-1}^{-1}),$$

β is a normalization constant, and the $\mathbf{e}_0, \mathbf{e}_{u,i}, \mathbf{e}_{y,i}$ terms are defined in (5), but written without arguments. This is, in general, not a Gaussian distribution due to the nonlinear nature of $\mathbf{e}_m(\cdot)$, but it can be approximated as one by linearizing $\mathbf{e}_m(\cdot)$ about a subset of the state estimates obtained from the previous window, being $\hat{\mathbf{x}}_{0:m}$. Substituting a first-order approximation of $\mathbf{e}_m(\cdot)$ into (9), and after some manipulation, the mean and covariance of a Gaussian approximation to $p(\mathbf{x}_{0:m}|\check{\mathbf{x}}_0, \mathbf{u}_{0:m}, \mathbf{y}_{0:m-1}) \approx \mathcal{N}(\boldsymbol{\mu}_{0:m}, \boldsymbol{\Sigma}_{0:m})$ can be shown to be

$$\boldsymbol{\mu}_{0:m} = \begin{bmatrix} \boldsymbol{\mu}_{0:m-1} \\ \boldsymbol{\mu}_m \end{bmatrix} = \hat{\mathbf{x}}_{0:m} - (\mathbf{H}_m^\mathsf{T}\mathbf{W}_m\mathbf{H}_m)^{-1}\mathbf{H}_m^\mathsf{T}\mathbf{W}_m\mathbf{e}_m(\hat{\mathbf{x}}_{0:m}), \tag{10}$$

$$\boldsymbol{\Sigma}_{0:m} = \begin{bmatrix} \boldsymbol{\Sigma}_{0:m-1} & \boldsymbol{\Sigma}_{0:m-1,m} \\ \boldsymbol{\Sigma}_{m,0:m-1} & \boldsymbol{\Sigma}_m \end{bmatrix} = (\mathbf{H}_m^\mathsf{T}\mathbf{W}_m\mathbf{H}_m)^{-1}, \tag{11}$$

respectively, where

$$\mathbf{H}_m = \left.\frac{\partial \mathbf{e}_m(\mathbf{x})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{0:m}}. \tag{12}$$

It finally follows that

$$p(\mathbf{x}_m|\check{\mathbf{x}}_0, \mathbf{u}_{0:m}, \mathbf{y}_{0:m-1}) \approx \mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m).$$

The specific step in which the marginalization occurred is when $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ are extracted from (10) and (11), respectively. With an expression for the new prior now identified, one may return to (8) to construct a nonlinear least-squares problem in the exact same manner as the full batch problem, and solve it with the Gauss-Newton or Levenberg-Marquart algorithm.

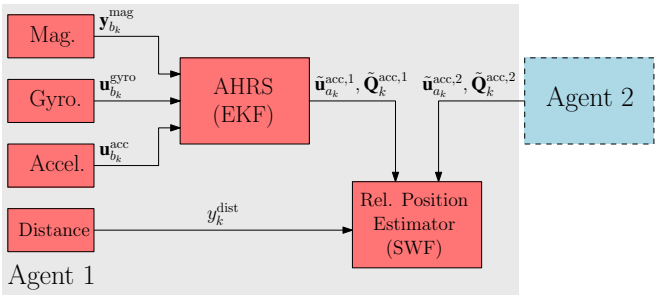

Fig. 2. High-level diagram of the architecture of the proposed algorithm. Each agent contains an AHRS that uses on-board accelerometer, rate gyro, and magnetometer measurements.

III. Proposed Algorithm

A. High-level architecture

The proposed algorithm consists of two main components, with the architecture displayed graphically in Figure 2. Although each agent can execute the following algorithm, it will be explained from the perspective of Agent 1.

1) **Attitude & Heading Reference System (AHRS)** At each IMU measurement, the agents calculate their own attitudes relative to a common local frame, such as an East-North-Up reference frame, denoted $\mathcal{F}_a$. Using the attitude estimate, Agent 2 communicates an estimate of their translational acceleration, resolved in $\mathcal{F}_a$, along with a corresponding covariance.
2) **Relative Position Estimator (RPE)** Using the translational acceleration information of Agent 2 communicated to Agent 1, as well as a set of previous distance measurements, a sliding window filter determines the position and velocity of Agent 1 relative to Agent 2 in $\mathcal{F}_a$. The high-frequency acceleration information of both agents is integrated between a set of optimized *keypoints*, which will be explained in Section III-C.

The primary contribution of this letter is the development and testing of the RPE. The choice of using a sliding window filter instead of a one-step-ahead filter, such as an extended Kalman filter (EKF), stems from the fact that position and velocity states are instantaneously unobservable given one distance measurement. However, as will be shown in Section III-D, the system is observable when multiple distance measurements are used for state estimation, which is what the sliding window filter does.

The cascaded architecture involving the AHRS and the RPE is *loosely-coupled* due to the separation of attitude and translational state estimators. This choice comes with several advantages and disadvantages, when compared to a *tightly-coupled* framework that would estimate the attitude and translational states in one estimator. The main disadvantage is that any coupling information between the AHRS and RPE states is lost. This point represents the largest possible source of inaccuracy, as it results in the RPE performance being highly dependent on the accuracy of the attitude estimates. However, this cost is justified by the following advantages.

- The corrective step of the AHRS can be executed at an arbitrarily high frequency, thus enabling the incorporation of all possible magnetometer and accelerometer measurements for attitude correction.
- The processing can easily be distributed among the two agents, and communication requirements are heavily reduced as gyroscope and magnetometer measurements do not need to be shared.
- The relative position dynamics reduce to a linear model, which is convenient from both mathematical and computational points of view.

B. Attitude & Heading Reference System

The AHRS used in the presented simulations and experiments is an extended Kalman filter (EKF) very similar to [12, Ch. 10], but with some modifications based on [16]. The AHRS estimates the agent attitude at time step k, denoted as $\hat{\mathbf{C}}_{ab_k}$. The estimate also has a corresponding covariance $\mathbf{P}_k^{\text{ahrs}}$, defined such that $\mathbf{C}_{ab_k} = \hat{\mathbf{C}}_{ab_k} \exp(\delta\boldsymbol{\phi}^\times)$, where $\delta\boldsymbol{\phi} \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_k^{\text{ahrs}})$ and $\mathbf{C}_{ab_k}$ is the true agent attitude. The translational acceleration vector $\tilde{\mathbf{u}}_{a_k}^{\text{acc}}$, resolved in $\mathcal{F}_a$, can be calculated using the vehicle attitude and the raw accelerometer measurements $\mathbf{u}_{b_k}^{\text{acc}}$ using

$$\tilde{\mathbf{u}}_{a_k}^{\text{acc}} = \mathbf{C}_{ab_k}\mathbf{u}_{b_k}^{\text{acc}} + \mathbf{g}_a, \tag{13}$$

where $\mathbf{g}_a$ is the gravity vector resolved in $\mathcal{F}_a$. However, a covariance associated with $\tilde{\mathbf{u}}_{a_k}^{\text{acc}}$ is also required. Equation (13) is a nonlinear function of two random variables, $\mathbf{C}_{ab_k}$ and $\mathbf{u}_{b_k}^{\text{acc}}$, and as such the translational acceleration distribution can be approximated as $\tilde{\mathbf{u}}_{a_k}^{\text{acc}} \sim \mathcal{N}(\hat{\mathbf{C}}_{ab_k}\mathbf{u}_{b_k}^{\text{acc}} + \mathbf{g}_a, \tilde{\mathbf{Q}}_k^{\text{acc}})$ where

$$\tilde{\mathbf{Q}}_k^{\text{acc}} = \mathbf{M}\mathbf{Q}_k^{\text{acc}}\mathbf{M}^\mathsf{T} + \mathbf{G}\mathbf{P}_k^{\text{ahrs}}\mathbf{G}^\mathsf{T},$$

and $\mathbf{Q}_k^{\text{acc}}$ is the covariance associated with the raw accelerometer measurements. The matrices $\mathbf{M} = \hat{\mathbf{C}}_{ab_k}$ and $\mathbf{G} = -\hat{\mathbf{C}}_{ab_k}\mathbf{u}_{b_k}^{\text{acc}^\times}$ are the Jacobians of (13) with respect to $\mathbf{u}_{b_k}^{\text{acc}}$ and $\delta\boldsymbol{\phi}$, respectively. Although not specified in the notation used in this section, each of the two agents computes their own, independent estimates of $\tilde{\mathbf{u}}_{a_k}^{\text{acc}}$ and $\tilde{\mathbf{Q}}_k^{\text{acc}}$. They are both communicated to the relative position estimator, consisting of 3 numbers for the $\tilde{\mathbf{u}}_{a_k}^{\text{acc}}$ and 6 numbers for the symmetric covariance matrix $\tilde{\mathbf{Q}}_k^{\text{acc}}$.

C. Relative Position Estimator

Let $\mathbf{r}_{a_k}^{12}$ be the position of Agent 1, relative to Agent 2, resolved in $\mathcal{F}_a$. Let $\mathbf{v}_{a_k}^{12}$ be the velocity of Agent 1, relative to Agent 2, with respect to $\mathcal{F}_a$. The relative position estimator, which will be on-board Agent 1, estimates the state $\mathbf{x}_k = [\mathbf{r}_{a_k}^{12^\mathsf{T}} \quad \mathbf{v}_{a_k}^{12^\mathsf{T}}]^\mathsf{T}$. The translational acceleration and corresponding covariances of Agents 1 and 2, being $(\tilde{\mathbf{u}}_{a_k}^{\text{acc},1}, \tilde{\mathbf{Q}}_k^{\text{acc},1})$ and $(\tilde{\mathbf{u}}_{a_k}^{\text{acc},2}, \tilde{\mathbf{Q}}_k^{\text{acc},2})$, are combined to produce a relative acceleration estimate, $\mathbf{u}_k \triangleq \tilde{\mathbf{u}}_{a_k}^{\text{acc},1} - \tilde{\mathbf{u}}_{a_k}^{\text{acc},2}$, with covariance $\tilde{\mathbf{Q}}_k = \tilde{\mathbf{Q}}_k^{\text{acc},1} + \tilde{\mathbf{Q}}_k^{\text{acc},2}$. This allows the relative position and velocity process model to be written as

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{14}$$

where $\Delta t_k = t_k - t_{k-1}$, $\mathbf{w}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$,

$$\mathbf{A}_{k-1} = \begin{bmatrix} \mathbf{1} & \Delta t_k\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad \mathbf{B}_{k-1} \begin{bmatrix} (\Delta t_k^2/2)\mathbf{1} \\ \Delta t_k\mathbf{1} \end{bmatrix},$$

$$\mathbf{Q}_{k-1} = \begin{bmatrix} \frac{1}{3}\Delta t_k^3\tilde{\mathbf{Q}}_{k-1} & \frac{1}{2}\Delta t_k^2\tilde{\mathbf{Q}}_{k-1} \\ \frac{1}{2}\Delta t_k^2\tilde{\mathbf{Q}}_{k-1} & \Delta t_k\tilde{\mathbf{Q}}_{k-1} \end{bmatrix}.$$

Furthermore, the RPE uses distance measurements, which have a measurement model of the form

$$y_k^{\text{dist}} = \left\|\mathbf{r}_{a_k}^{12}\right\| + v_k, \tag{15}$$

where $v_k \sim \mathcal{N}(0, R_k)$. The Jacobian of (15) with respect to $\mathbf{x}_k$ is given by $\mathbf{C}_k = \begin{bmatrix} \boldsymbol{\rho}_k^\mathsf{T} & \mathbf{0} \end{bmatrix}$, where $\boldsymbol{\rho}_k = \mathbf{r}_{a_k}^{12} / \left\|\mathbf{r}_{a_k}^{12}\right\|$.

A naive implementation of the sliding window filter would result in including a state at every single measurement. With modern IMUs, this typically occurs at a frequency of 100 Hz - 1000 Hz, which can result in an enormous amount of states in the window if it were to span only a few seconds. The number of states in the window can be reduced by omitting all except a select few called *keypoints* [17]. The keypoints are identified as a set of K distinct time indices $\mathcal{K} = \{p_0, p_1, \ldots, p_{K-1}\}$ such that the states $\mathbf{x}_{p_0}, \mathbf{x}_{p_1}, \ldots, \mathbf{x}_{p_{K-1}}$ are the only states estimated inside the window. To do this, a process model of some sort is required that relates successive keypoint states.

A technique called *pre-integration* expresses the process model between two non-adjacent states $\mathbf{x}_i$ and $\mathbf{x}_j$ by a single relation. The use of the linear model in (14) makes pre-integration trivial, as direct iteration of (14) leads to

$$\mathbf{x}_j = \left(\prod_{k=i}^{j-1}\mathbf{A}_k\right)\mathbf{x}_i + \sum_{k=i}^{j-1}\left(\prod_{\ell=k+1}^{j-1}\mathbf{A}_\ell\right)\mathbf{B}_k\mathbf{u}_k, \tag{16}$$

$$\triangleq \mathbf{A}_{ji}\mathbf{x}_i + \mathbf{b}_{ji}. \tag{17}$$

Importantly, the terms $\mathbf{A}_{ji}$ and $\mathbf{b}_{ji}$ do not need to be recomputed at each Gauss-Newton iteration of the sliding window filter, since they are independent of the state estimate. This yields substantial computational savings compared to a case where the process model is too complicated to easily pre-integrate. With this pre-integration ability, states in the sliding window filter can be placed arbitrarily far apart in time with a very minor increase in computation time.

As such, this letter constructs a sliding window filter where each state is a keypoint, successive keypoint states are related by the pre-integrated process model (17), and the measurement model at each keypoint is given by (15). The ability to place keypoints at arbitrary locations in an agent's state history now gives rise to an entire design problem, which is to design an appropriate keypoint selection strategy. This letter will propose one of many possible strategies.

D. Observability Analysis

Evaluated at arbitrary keypoints, the observability rank condition given in (7) requires that

$$\text{rank}(\mathcal{O}) = 6, \tag{18}$$

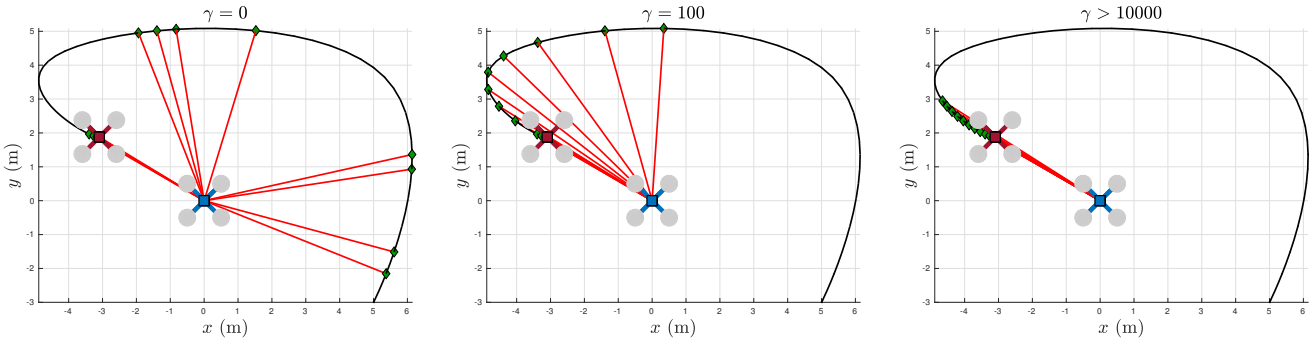

Fig. 3. Effect of the parameter γ on the keypoint selection with $K = 12$, and the old trajectory shown as the black line. (left) With $\gamma = 0$, the algorithm greedily minimizes the GDOP, and results in keypoints placed far apart in time. (middle) With a moderate value for γ, the algorithm obtains a balance between pure GDOP minimization, and making all the keypoints as close as possible. (right) With a large value for γ, the minimal cost is the solution with the keypoints as close as possible, and the "plain vanilla" sliding window filter is recovered.

where

$$\mathcal{O} = \begin{bmatrix} \boldsymbol{\rho}_{p_0} & \boldsymbol{\rho}_{p_1} & \boldsymbol{\rho}_{p_2} & \cdots & \boldsymbol{\rho}_{p_{K-1}} \\ \mathbf{0} & \Delta t_{p_1 p_0} \boldsymbol{\rho}_{p_1} & \Delta t_{p_2 p_0} \boldsymbol{\rho}_{p_2} & \cdots & \Delta t_{p_{K-1} p_0} \boldsymbol{\rho}_{p_{K-1}} \end{bmatrix},$$

$\boldsymbol{\rho}_k = \mathbf{r}^{12}_{a_k} / \left\| \mathbf{r}^{12}_{a_k} \right\|$, and $\Delta t_{ji} = t_j - t_i$. To ensure observability, the keypoints must be chosen to satisfy (18).

Theorem 3.1: Sufficient conditions to satisfy (18) are

1) $K \geq 6$,
2) $\Delta t_{p_1 p_0} \neq \Delta t_{p_2 p_0} \neq \ldots \Delta t_{p_K p_0} \neq 0$,
3) and that there exists two non-intersecting subsets of $\{\boldsymbol{\rho}_{p_0}, \boldsymbol{\rho}_{p_1}, \ldots, \boldsymbol{\rho}_{p_{K-1}}\}$, each composed of 3 linearly independent elements.

Proof: See Appendix A. ■

These conditions result in the requirement that the agents' relative motion be non-planar, and that keypoints are at distinct times.

E. Keypoint Selection Strategy

The keypoint selection strategy aims to find a suitable keypoint index set $\mathcal{K}$. The proposed strategy is a greedy algorithm, which places keypoints one-by-one, searching linearly through all possible candidate keypoint locations, and selecting the location with the lowest cost. The chosen cost function is based on the *geometric dilution of precision* (GDOP) [12, Ch. 8.5], and is given by

$$J(\mathcal{K}) = \operatorname{tr}\left((\mathbf{D}(\mathcal{K})^\mathsf{T} \mathbf{D}(\mathcal{K}))^{-1} \right) + \gamma \sum_{i=1}^{|\mathcal{K}|} \Delta t_{p_i p_{i-1}}, \qquad (19)$$

where $\mathbf{D}(\mathcal{K}) = \begin{bmatrix} \boldsymbol{\rho}_{p_0} & \boldsymbol{\rho}_{p_1} & \boldsymbol{\rho}_{p_2} & \cdots & \boldsymbol{\rho}_{p_{|\mathcal{K}|}} \end{bmatrix}^\mathsf{T}$, $|\mathcal{K}|$ denotes the number of elements in $\mathcal{K}$, and γ is a tuning parameter. The first term in (19) is the square of the GDOP, also noting that the form of the $\mathbf{D}(\mathcal{K})$ matrix shows a natural similarity to the observability matrix shown in (18). Minimizing this first term avoids a keypoint selection that will make the observability matrix close to rank deficient. The second term in (19) penalizes the keypoints for being placed too far apart in time. Excessively old keypoints will require long pre-integration between their adjacent states, which increases the covariance of the estimate. The tuning parameter γ is used to balance the two terms, and its effect is illustrated in Figure 3.

Algorithm 1 Greedy Keypoint Selection. Given the current time step k, the total number of keypoints K, and the keypoint index set of the previous window $\mathcal{K}_{\text{old}}$, calculate the new keypoint index set $\mathcal{K}$. A set of candidate indices $\mathcal{C}$ is maintained.

```
1:  function GetKeypoints(k, K, K_old)
2:      K ← {k − 3, k − 2, k − 1, k}
3:      C ← K_old ∪ {max(K_old) + 1, ..., k − 4}
4:      for i = 1, ..., K − 4 do
5:          J_best ← ∞
6:          for all p ∈ C do
7:              K_temp ← K ∪ {p}
8:              if J(K_temp) ≤ J_best then
9:                  J_best ← J(K_temp)
10:                 p_best ← p
11:             end if
12:         end for
13:         K ← K ∪ {p_best}
14:         C ← C \ {p_best}
15:     end for
16:     return K
17: end function
```

Finally, it is also possible to recursively compute the matrix $\boldsymbol{\Lambda} = (\mathbf{D}(\mathcal{K})^\mathsf{T} \mathbf{D}(\mathcal{K}))^{-1}$, which avoids recomputing the inverse when a new keypoint is added to $\mathcal{K}$. Defining $\tilde{\mathcal{K}} = \mathcal{K} \cup \{p\}$, it can be shown that

$$\left(\mathbf{D}(\tilde{\mathcal{K}})^\mathsf{T} \mathbf{D}(\tilde{\mathcal{K}}) \right)^{-1} = \left(\mathbf{D}(\mathcal{K})^\mathsf{T} \mathbf{D}(\mathcal{K}) + \boldsymbol{\rho}_p \boldsymbol{\rho}_p^\mathsf{T} \right)^{-1} \qquad (20)$$

$$= \boldsymbol{\Lambda} - \frac{\boldsymbol{\Lambda} \boldsymbol{\rho}_p \boldsymbol{\rho}_p^\mathsf{T} \boldsymbol{\Lambda}}{1 + \boldsymbol{\rho}_p^\mathsf{T} \boldsymbol{\Lambda} \boldsymbol{\rho}_p}, \qquad (21)$$

where, in (21), the Woodbury matrix identity is used. Equation (21) is particularly useful when evaluating the **if** statement in line 8 of Algorithm 1, which shows the details of the proposed keypoint placement strategy. The algorithm initializes $\mathcal{K}$ with the 4 most recent indices in order to

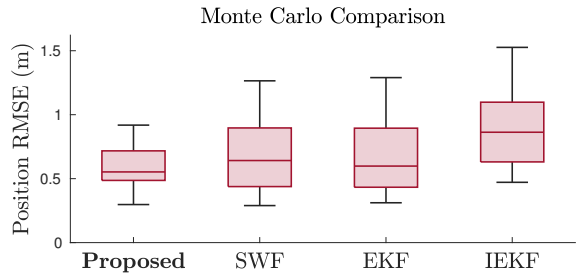

Fig. 4. Results of 200 Monte Carlo trials. The proposed method offers an enhancement over the "plain vanilla" implementation of the sliding window filter (SWF), which simply places the keypoints at the most recent range measurements, as well as a standard extended Kalman filter (EKF), and an Iterated EKF (IEKF).

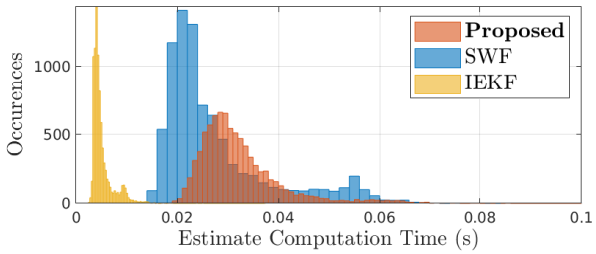

Fig. 5. Histogram of the computation time of the different algorithms, when simulated on a laptop with an Intel Core i7-9750H CPU.

produce an invertible $(\mathbf{D}(\mathcal{K})^\mathsf{T}\mathbf{D}(\mathcal{K}))^{-1}$ matrix, after which the cost function naturally leads to a selection of older keypoints.

F. Multi-robot Scenario

Although this paper focuses on a scenario with two agents, one way to extend the proposed algorithm to multiple agents is to naively copy the estimator for each pair of agents. In any case, it is possible to leverage the proposed framework to drastically simplify the inter-agent communication. By recalling that $\mathbf{u}_k = \tilde{\mathbf{u}}_{a_k}^{\text{acc},1} - \tilde{\mathbf{u}}_{a_k}^{\text{acc},2}$, it is straightforward to show that $\mathbf{b}_{ji}$ in (17) can be written as

$$\mathbf{b}_{ji} = \mathbf{b}_{ji}^{\text{acc},1} - \mathbf{b}_{ji}^{\text{acc},2},$$

where $\mathbf{b}_{ji}^{\text{acc},1}$ and $\mathbf{b}_{ji}^{\text{acc},2}$ only depend on measurements from Agents 1 and 2, respectively. This has a significant implication: the agents do not need to communicate their acceleration measurements at high frequency, but can instead preintegrate them in advance to produce $\mathbf{b}_{ji}^{\text{acc},1}$ and $\mathbf{b}_{ji}^{\text{acc},2}$, then communicate these quantities at an arbitrarily lower frequency. This is of substantial practical interest, as communicating all high-frequency IMU measurements could quickly exceed the inter-agent communication capacity, especially for multi-robot scenarios.

IV. Simulation Results

The proposed algorithm is tested in simulation, and compared against a "plain vanilla" sliding window filter (*Vanilla SWF* or *SWF* in the figures), which simply uses the K most recent range measurements as the keypoint times, as shown on the right of Figure 3. A comparison is also made to using a standard EKF as the relative position estimator. The simulation is repeated for 200 Monte Carlo trials, where each trial uses a different, randomized trajectory. Table I shows the values used when generating zero-mean Gaussian noise on each sensor.

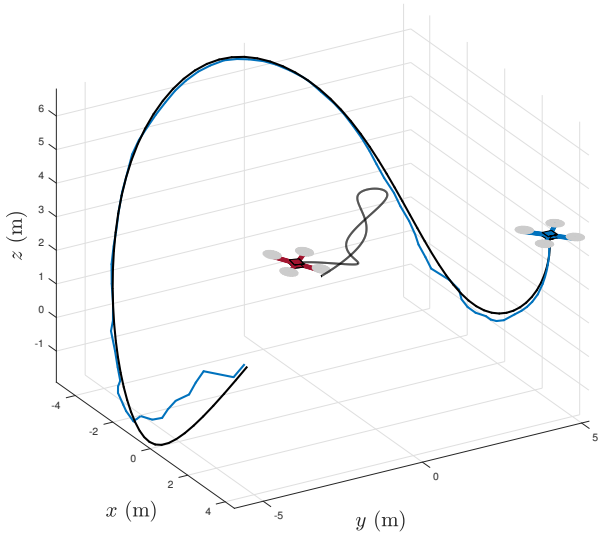

Fig. 6. Simulation of the proposed estimation algorithm. The ground truth trajectory is shown in black, and the estimated trajectory in blue.

Figure 4 shows a box plot of the position root-mean-squared error (RMSE) for the 200 Monte Carlo trials, where the $\text{RMSE} = \sqrt{(1/N)\sum_{k=1}^{N} \mathbf{e}_{r_k}^\mathsf{T}\mathbf{e}_{r_k}}$, $\mathbf{e}_{r_k} = \hat{\mathbf{r}}_{a_k}^{12} - \mathbf{r}_{a_k}^{12}$.

Using the greedy keypoint placement scheme shows a 9% reduction in average position RMSE when compared to the Vanilla SWF, 7% compared to the EKF, and 32% compared to the iterated EKF [13, Ch. 4.2.5]. An interesting observation is that EKF can perform as well, if not better than the sliding window filter and the iterated EKF, which both iterate the state estimate until a locally minimal least-squares cost is found. This implies that there exists a state with lower overall process and measurement error, yet higher true estimation error. Many instances have been observed where the least-squares cost decreases while true estimation error increases, a possible consequence of the unobservable nature of this problem.

The proposed algorithm takes an average of 0.035 s per estimate computation, whereas the Vanilla SWF takes 0.027 s, the IEKF takes 0.005 s, and the EKF takes 0.0006 s. A histogram of the computation time is shown in Figure 5.

V. Experimental Results

The proposed algorithm is also tested in a real experiment, with two different hardware setups, both shown in Figure V. The first consists of a Raspberry Pi 4B, with an LSM9DS1 9-DOF IMU and a Pozyx UWB Developer Tag, providing distance measurements to an identical device. Although the cost of the actual sensors embedded in one of these prototypes is $15 USD, each cost a total of

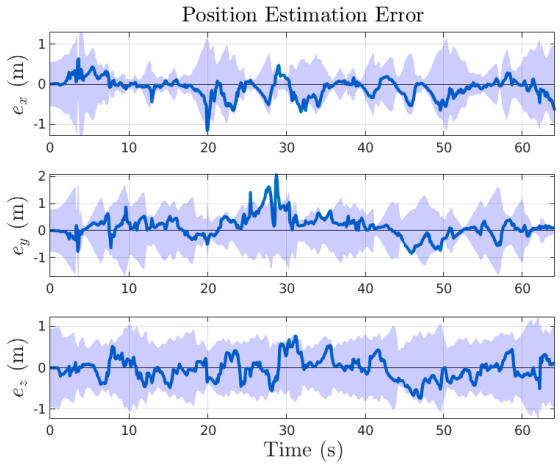

Fig. 7. Components of the position estimation error during an experimental trial, using the greedy keypoint placement method. The error compared to ground truth is shown as the blue line, with the shaded area representing the $\pm 3\sigma$ confidence bounds.

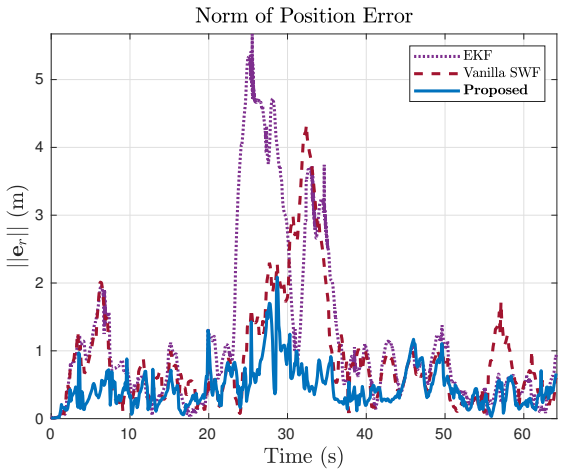

Fig. 8. Comparison of the norm of the position estimation error $\mathbf{e}_r$, during an experimental trial.

TABLE I

SIMULATION NOISE PROPERTIES

Specification	Value	Units
Magnetometer std. dev.	1	μF
Gyroscope std. dev.	0.001	rad/s
Accelerometer std. dev.	0.01	m/s^2
Distance std. dev	0.1	m
Initial position std. dev.	0.8	m
Initial velocity std. dev.	0.1	m/s
Initial attitude std. dev.	0.001	red
Accel/Gyro/Mag frequency	100	Hz
Distance frequency	10	Hz
RPE Window Size K	20	-
RPE keypoint selection parameter γ	100	-
RPE frequency	10	Hz

approximately $270 USD. However, this is mainly due to the cost of the Raspberry Pi 4B, the battery pack, and the Pozyx Developer Tag. The second setup consists of a Pozyx Developer tag mounted to quadrotors possessing a Pixhawk 4, which provides IMU and magnetometer data. In both setups, two identical devices are randomly moved around a room by hand, in an overall volume of roughly $5\text{ m} \times 4\text{ m} \times 2\text{ m}$. When using the quadrotors, the motors are spinning without propellors to examine the effect of vibrations and magnetic interference. Ground truth position and attitude measurements are collected using an OptiTrack optical motion capture system.

Table II shows the position RMSE of the different algorithms, for each trial. The proposed method outperforms "plain vanilla" implementations of the SWF and the EKF by a large margin, in all of the trials, showing the value of the greedy keypoint selection strategy. Figure 7 shows the position estimation error of the proposed method, and that the estimate stays within the $\pm 3\sigma$ confidence bounds, except for two very brief moments. Finally, Figure 8 shows the norm of the position estimation error. During the periods of large error, between 20 and 40 seconds in Figure 8, the least squares cost for all three algorithms did not dramatically increase relative to other periods. This implies that the EKF and the Vanilla SWF have converged to ambiguous states which also have low process and measurement error, again a possible consequence of unobservability.

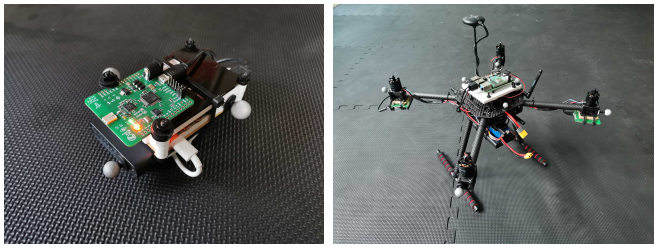

Fig. 9. (left) Pozyx UWB Developer Tag mounted to a Raspberry Pi 4B, in a case. (right) Pozyx UWB Developer Tag mounted to a quadrotor with a Pixhawk 4.

TABLE II

POSITION RMSE OF EXPERIMENTAL TRIALS

	Proposed	Vanilla SWF	EKF
Trial 1 (one static agent)	**0.68 m**	1.41 m	2.11 m
Trial 2 (both moving)	**0.55 m**	1.22 m	1.71 m
Trial 3 (both moving)	**0.86 m**	1.56 m	2.69 m
Trial 4 (spinning motors)	**0.87 m**	1.34 m	1.12 m

VI. CONCLUSION

This letter shows that it is possible to estimate the relative position of one UWB-equipped agent relative to another, provided that there is sufficient motion between them, and that they also possess 9-DOF IMUs. The proposed algorithm with strategically placed keypoints shows ubiquitous improvement over standard estimators, both in simulation and experiment, where there is 2 to 3-fold reduction in positioning error. Future work should consider switching the loosely-coupled architecture for a tightly-coupled one, as the current scheme is heavily dependent on accurate attitude estimates. In environments with large magnetic perturbations, this can deteriorate performance. Furthermore, this algorithm should be tested on actual flying

quadrotors, where vibrations can be more significant than those experienced in these experiments.

APPENDIX

A. Proof of Theorem 3.1

Proof: Satisfying the rank condition in (18) amounts to showing that there are 6 linearly independent columns in the matrix $\mathcal{O}$. Since there are K columns in $\mathcal{O}$, $K \geq 6$, which is Condition 1, is immediately required so that there are at least 6 columns. If the theorem is proven for $K = 6$, then it also holds for $K > 6$, as the addition of columns will not affect the linear independence of the first six. As such, using a change in notation for brevity, the rank condition in (18) requires that

$$\mathcal{O} = \begin{bmatrix} \boldsymbol{\rho}_0 & \boldsymbol{\rho}_1 & \boldsymbol{\rho}_2 & \boldsymbol{\rho}_3 & \boldsymbol{\rho}_4 & \boldsymbol{\rho}_5 \\ \mathbf{0} & \Delta t_1 \boldsymbol{\rho}_1 & \Delta t_2 \boldsymbol{\rho}_2 & \Delta t_3 \boldsymbol{\rho}_3 & \Delta t_4 \boldsymbol{\rho}_4 & \Delta t_5 \boldsymbol{\rho}_5 \end{bmatrix}$$

have rank 6. Without loss of generality, assume $\{\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2\}$ are linearly independent. It immediately follows that the first three columns of $\mathcal{O}$ are linearly independent. Consider now $\boldsymbol{\rho}_3$, again without loss of generality. Since $\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2 \in \mathbb{R}^3$, $\boldsymbol{\rho}_3$ is linearly dependent on $\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2$ and can be written as

$$\boldsymbol{\rho}_3 = \alpha_0 \boldsymbol{\rho}_0 + \alpha_1 \boldsymbol{\rho}_1 + \alpha_2 \boldsymbol{\rho}_2. \tag{22}$$

If the fourth column of $\mathcal{O}$ were linearly dependent on the first three, then there would exist $\beta_0, \beta_1, \beta_2$ such that

$$\begin{bmatrix} \boldsymbol{\rho}_3 \\ \Delta t_3 \boldsymbol{\rho}_3 \end{bmatrix} = \beta_0 \begin{bmatrix} \boldsymbol{\rho}_0 \\ \mathbf{0} \end{bmatrix} + \beta_1 \begin{bmatrix} \boldsymbol{\rho}_1 \\ \Delta t_1 \boldsymbol{\rho}_1 \end{bmatrix} + \beta_2 \begin{bmatrix} \boldsymbol{\rho}_2 \\ \Delta t_2 \boldsymbol{\rho}_2 \end{bmatrix}. \tag{23}$$

Substituting (22) into (23) yields

$$\begin{bmatrix} \alpha_0 \boldsymbol{\rho}_0 + \alpha_1 \boldsymbol{\rho}_1 + \alpha_2 \boldsymbol{\rho}_2 \\ \Delta t_3 (\alpha_0 \boldsymbol{\rho}_0 + \alpha_1 \boldsymbol{\rho}_1 + \alpha_2 \boldsymbol{\rho}_2) \end{bmatrix} = \beta_0 \begin{bmatrix} \boldsymbol{\rho}_0 \\ \mathbf{0} \end{bmatrix} + \beta_1 \begin{bmatrix} \boldsymbol{\rho}_1 \\ \Delta t_1 \boldsymbol{\rho}_1 \end{bmatrix} + \beta_2 \begin{bmatrix} \boldsymbol{\rho}_2 \\ \Delta t_2 \boldsymbol{\rho}_2 \end{bmatrix}. \tag{24}$$

The first three lines of (24) immediately require that $\alpha_0 = \beta_0$, $\alpha_1 = \beta_1$, $\alpha_2 = \beta_2$, which reduces the last three lines of (24) to

$$\Delta t_3 \alpha_0 \boldsymbol{\rho}_0 + \Delta t_3 \alpha_1 \boldsymbol{\rho}_1 + \Delta t_3 \alpha_2 \boldsymbol{\rho}_2 = \alpha_1 \Delta t_1 \boldsymbol{\rho}_1 + \alpha_2 \Delta t_2 \boldsymbol{\rho}_2. \tag{25}$$

Hence, the fourth column of $\mathcal{O}$ is linearly dependent on the first three if and only if $\alpha_0 = 0$, $\Delta t_3 = \Delta t_2$, $\Delta t_3 = \Delta t_1$. Condition 2 in Theorem 3.1 means $\Delta t_3 \neq \Delta t_2 \neq \Delta t_1$, hence proving the linear independence of the fourth column of $\mathcal{O}$ from the first three. Identical proofs show that the fifth and sixth columns of $\mathcal{O}$ are also linearly independent from the first three. If additionally $\{\boldsymbol{\rho}_3, \boldsymbol{\rho}_4, \boldsymbol{\rho}_5\}$ are linearly independent, then the fourth, fifth, and sixth columns of $\mathcal{O}$ will be linearly independent from each other, in addition to each being linearly independent from the first three. Hence, all 6 columns of $\mathcal{O}$ will be linearly independent, giving it a rank of 6. Since this proof was done for a completely arbitrary choice of linearly independent sets $\{\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2\}$ and $\{\boldsymbol{\rho}_3, \boldsymbol{\rho}_4, \boldsymbol{\rho}_5\}$, it is only required that there exists two distinct non-intersecting subsets of $\{\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_K\}$, each containing 3 linearly independent elements, which is Condition 3. ■

REFERENCES

[1] A. Ledergerber, M. Hamer, and R. D'Andrea, "A Robot Self-Localization System using One-Way Ultra-Wideband Communication," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, Hamburg, 2015, pp. 3131–3137.

[2] J. Cano, S. Chidami, and J. Le Ny, "A Kalman Filter-Based Algorithm for Simultaneous Time Synchronization and Localization in UWB Networks," in *IEEE Intl. Conf. on Robotics and Automation*, Montreal, 2019, pp. 1431–1437.

[3] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, "Decentralized Visual-Inertial-UWB Fusion for Relative State Estimation of Aerial Swarm," in *IEEE Intl. Conf. on Robotics and Automation*, Paris, 2020, pp. 8776–8782.

[4] M. Shalaby, C. C. Cossette, J. R. Forbes, and J. Le Ny, "Relative Position Estimation in Multi-Agent Systems Using Attitude-Coupled Range Measurements (under review)," *IEEE Robotics and Automation Letters*, 2021.

[5] P. Batista, C. Silvestre, and P. Oliveira, "Single range aided navigation and source localization: Observability and filter design," *Systems and Control Letters*, vol. 60, no. 8, pp. 665–673, 2011.

[6] I. Sarras, J. Marzat, S. Bertrand, and H. Piet-Lahanier, "Collaborative multiple micro air vehicles' localization and target tracking in GPS-denied environment from range–velocity measurements," *International Journal of Micro Air Vehicles*, vol. 10, no. 2, pp. 225–239, 2018.

[7] K. Guo, Z. Qiu, W. Meng, L. Xie, and R. Teo, "Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in GPS denied environments," *Intl. Jrnl. of Micro Air Vehicles*, vol. 9, no. 3, pp. 169–186, 2017.

[8] S. van der Helm, K. N. McGuire, M. Coppola, and G. de Croon, "On-board Range-based Relative Localization for Micro Aerial Vehicles in Indoor Leader-Follower Flight," *Autonomous Robots*, vol. 44, pp. 415–441, 2020.

[9] T. M. Nguyen, Z. Qiu, T. H. Nguyen, M. Cao, and L. Xie, "Distance-Based Cooperative Relative Localization for Leader-Following Control of MAVs," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3641–3648, 2019.

[10] ——, "Persistently Excited Adaptive Relative Localization and Time-Varying Formation of Robot Swarms," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 553–560, 2020.

[11] R. Liu, C. Yuen, T. N. Do, D. Jiao, X. Liu, and U. X. Tan, "Cooperative relative positioning of mobile users by fusing IMU inertial and UWB ranging information," in *IEEE Intl. Conf. on Robotics and Automation*, Singapore, 2017, pp. 5623–5629.

[12] J. A. Farrell, "Aided Navigation: GPS with High Rate Sensors," in *Applied Mathematics in Integrated Navigation Systems, Third Edition*, The McGraw-Hill Companies, 2008.

[13] T. Barfoot, *State Estimation for Robotics*. Toronto, ON: Cambridge University Press, 2019.

[14] G. Sibley, "A Sliding Window Filter for SLAM," University of Southern California, Tech. Rep., 2006.

[15] T. C. Dong-Si and A. I. Mourikis, "Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5655–5662, 2011.

[16] A. Barrau and S. Bonnabel, "Three examples of the stability properties of the invariant extended Kalman filter," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 431–437, 2017.

[17] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *Intl. Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.