Relative Position Estimation Between Two UWB Devices With IMUs

Charles Champagne Cossette, Mohammed Shalaby, David Saussié, James Richard Forbes, and 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 m is achieved with inexpensive sensors.

Index Terms—Localization, multi-robot systems, range sensing.

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.

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 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, \( p(x) \) denotes the joint probability density function (PDF) of the continuous random variable \( x \in \mathbb{R}^n \). If \( x \) is normally distributed with mean \( \mu \) and covariance \( \Sigma \), it is written as \( x \sim \mathcal{N}(\mu, \Sigma) \). The compressed notation \( x_{0:K} = [x_0^T \ldots x_K^T]^T \) is used for brevity. A bolded 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 \( v_a \). The same physical vector resolved in a different frame \( \mathcal{F}_b \) can be related by a direction cosine matrix (rotation matrix), denoted \( C_{ab} \in SO(3) \), such that \( v_a = C_{ab}v_b \) [12]. A direction cosine matrix (DCM) \( C_{ab} \) can be parameterized using a rotation vector, denoted \( \phi \in \mathbb{R}^3 \), using \( C_{ab} = \exp(\phi^a) \), where \( (\cdot)^a \) 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

\[
\begin{align*}
    x_k &= f(x_{k-1}, u_{k-1}) + w_{k-1}, \quad k = 1, \ldots, K, \quad (1) \\
    y_k &= g(x_k) + v_k, \quad k = 0, \ldots, K, \quad (2)
\end{align*}
\]

where \( x_k \in \mathbb{R}^n \) denotes the system state at time step \( k \), \( u_{k-1} \) is the system input, and \( y_k \) is the output. The terms \( w_{k-1} \sim \mathcal{N}(0, Q_{k-1}) \) and \( v_k \sim \mathcal{N}(0, R_k) \) represent process and measurement noise, respectively.

The maximum a posteriori framework aims to find the states \( x_{0:K} \) which maximize the posterior probability density function, given a history of input measurements \( u_{0:K-1} \), output measurements \( y_{0:K} \), and a prior distribution of the initial state, \( x_0 \sim \mathcal{N}(\bar{x}_0, \bar{P}_0) \). That is, the estimate \( \hat{x}_{0:K} \) is given by

\[
\hat{x} = \text{arg max}_x p(x|x_0, u_{0:K-1}, y_{0:K}), \quad (3)
\]

where \( x = x_{0:K} \) 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

\[
\hat{x} = \text{arg min}_x \frac{1}{2} e(x)^T W e(x), \quad (4)
\]

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 \( x^{(0)} \), and computes an update \( \delta x^{(t)} \) to the \( t^{th} \) iteration by solving

\[
(H^T WH)\delta x^{(t)} = -H^T W e(x^{(t)}), \quad (5)
\]

where

\[
H = \frac{\partial e(x)}{\partial x}\bigg|_{x^{(t)}} = \begin{bmatrix}
1 & -A_0 & 1 & \cdots \\
& \ddots & \ddots & \ddots \\
& & -A_{K-1} & 1 \\
& & & -C_0 \\
& & & & -C_1 \\
& & & & \ddots \\
& & & & & -C_K
\end{bmatrix},
\]

where \( A_k \) and \( C_k \) are the Jacobians of the process and measurement models, respectively, with respect to the state \( x_k \). The state estimate is updated with \( x^{(t+1)} = x^{(t)} + \delta x^{(t)} \), and the process is repeated until convergence. In practice, variants such as the Levenberg-Marquardt 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 \( H^T WH \) is full rank when the prior is ignored. In [13, Ch. 3.1], it is shown that this culminates in the equivalent requirement that

\[
\text{rank} \begin{bmatrix}
C_0^T & A_0^T C_1^T & A_0^T A_1^T C_2^T & \cdots & A_0^T \cdots A_{K-1}^T C_K^T
\end{bmatrix} = n. \quad (7)
\]

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 $x_{0:k_1}$. It then continues to travel until time $t_{k_2}$, adding the new states $x_{k_1+1:k_2}$ to its state history. The $m$ oldest states $x_{0:m-1}$ 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 $x_{m:k_1}$.

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

$$p(x_{m:k_2}|x_0, u_{0:k_2-1}, y_{0:k_2})$$

and

$$= \eta p(x_{m+1:k_2}|u_{m:k_2-1}, y_{m:k_2}, x_m) \times p(x_m|x_0, u_{0:m-1}, y_{0:m-1}).$$

The specific step in which the marginalization occurred is when $p(x_m|x_0, u_{0:m-1}, y_{0:m-1})$ is 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.

### III. PROPOSED ALGORITHM

#### A. High-Level Architecture

The proposed algorithm consists of two main components, with the architecture displayed graphically in Fig. 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 $F_o$. Using the attitude estimate, Agent 2 communicates an estimate of their translational acceleration, resolved in $F_o$, 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 $F_o$. 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.

![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.](image-url)
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, and as such the translational acceleration distribution can be computed as gyroscope and magnetometer measurements do not need to be recomputed.

\[ \tilde{\delta} = v \delta \Delta \] and \( \delta = \hat{\delta} \) expresses the process model at each keypoint \( j \), denoted as \( C_{a_{jk}} \). The estimate also has a corresponding covariance \( P_{a_{jk}} \), defined such that \( \tilde{C}_{a_{jk}} = C_{a_{jk}} \exp(\delta \hat{\delta}) \), where \( \delta \hat{\delta} \sim \mathcal{N}(0, P_{a_{jk}}) \) and \( C_{a_{jk}} \) is the true agent attitude. The translational acceleration vector \( \tilde{u}_{a_{jk}} \), resolved in \( F_a \), can be calculated using the vehicle attitude and the raw accelerometer measurements \( u_{b_{jk}} \) using

\[ \tilde{u}_{a_{jk}} = C_{a_{jk}} u_{b_{jk}} + g_a, \quad (13) \]

where \( g_a \) is the gravity vector resolved in \( F_a \). However, a covariance associated with \( \tilde{u}_{a_{jk}} \) is also required. Equation (13) is a nonlinear function of two random variables, \( C_{a_{jk}} \) and \( u_{b_{jk}} \), and as such the translational acceleration distribution can be approximated as \( \tilde{u}_{a_{jk}} \sim \mathcal{N}(C_{a_{jk}} u_{b_{jk}} + g_a, Q_{a_{jk}}) \) where

\[ Q_{a_{jk}} = \text{MQ}_{a_{jk}} \text{M}^T + \text{GP}_{a_{jk}} \text{G}^T, \]

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

C. Relative Position Estimator

Let \( x_{a_{1j}} \) be the position of Agent 1, relative to Agent 2, resolved in \( F_a \). Let \( v_{a_{1j}} \) be the velocity of Agent 1, relative to Agent 2, with respect to \( F_a \). The relative position estimator, which will be on-board Agent 1, estimates the state \( x_k = [x_{a_{1k}}^T \ y_{a_{1k}}^T]^T \). The translational acceleration and corresponding covariances of Agents 1 and 2, being \( (\tilde{u}_{a_{1j}}^{\text{acc}}, \ Q_{a_{1j}}^{\text{acc}}) \) and \( (\tilde{u}_{a_{2j}}^{\text{acc}}, \ Q_{a_{2j}}^{\text{acc}}) \), are combined to produce a relative acceleration estimate, \( u_k \triangleq \tilde{u}_{a_{1j}}^{\text{acc}} - \tilde{u}_{a_{2j}}^{\text{acc}}, \) with covariance \( Q_k = Q_{a_{1j}}^{\text{acc}} + Q_{a_{2j}}^{\text{acc}} \). This allows the relative position and velocity process model to be written as

\[ x_k = A_{k-1} x_{k-1} + B_{k-1} u_{k-1} + w_{k-1}, \quad (14) \]

where \( \Delta t_k = t_k - t_{k-1} \), \( w_{k-1} \sim \mathcal{N}(0, Q_{k-1}) \),

\[ A_{k-1} = \begin{bmatrix} 1 & \Delta t_k^1 \\ 0 & 1 \end{bmatrix}, \quad B_{k-1} = \begin{bmatrix} (\Delta t_k^2)^1/2 \\ \Delta t_k \end{bmatrix}, \]

\[ Q_{k-1} = \begin{bmatrix} \frac{1}{3} \Delta t_k^3 Q_{k-1}^1 & \frac{1}{2} \Delta t_k^2 Q_{k-1}^1 \\ \frac{1}{2} \Delta t_k^2 Q_{k-1}^1 & \frac{1}{3} \Delta t_k^3 Q_{k-1}^1 \end{bmatrix}. \]

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

\[ y_k^{\text{dist}} = \|x_{a_{1k}}^2\| + v_k, \quad (15) \]

where \( v_k \sim \mathcal{N}(0, R_k) \). The Jacobian of (15) with respect to \( x_k \) is given by \( C_k = [p_k^T \ 0] \), where \( p_k = x_{a_{1k}}^2/\|x_{a_{1k}}^2\| \).

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 \( K = \{p_1, p_2, \ldots, p_{K-1}\} \) such that the states \( x_{p_1}, x_{p_2}, \ldots, x_{p_{K-1}} \) 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 \( x_i \) and \( x_j \), by a single relation. The use of the linear model in (14) makes pre-integration trivial, as direct iteration of (14) leads to

\[ x_j = \left( \prod_{k=i}^{j-1} A_k \right) x_i + \sum_{k=i}^{j-1} \left( \prod_{\ell=k+1}^{j-1} A_{\ell} \right) B_k u_k, \quad (16) \]

\[ \triangleq A_{ji} x_i + b_{ji}. \quad (17) \]

Importantly, the terms \( A_{ji} \) and \( b_{ji} \) 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

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

where

$$O = \begin{bmatrix} \rho_{p_0} & \rho_{p_1} & \rho_{p_2} & \cdots & \rho_{p_{K-1}} \\ 0 & \Delta t_{p_1p_0} & \Delta t_{p_2p_0} & \cdots & \Delta t_{p_{K-1}p_0} \end{bmatrix},$$

$$\rho_k = r_{k,0}^2/||r_{k,0}||,$$

and $$\Delta t_{ji} = t_j - t_i.$$ To ensure observability, the keypoints must be chosen to satisfy (18).

**Theorem III.1:** Sufficient conditions to satisfy (18) are

1. $$K \geq 6,$$
2. $$\Delta t_{p_1p_0} \neq \Delta t_{p_2p_0} \neq \cdots \neq \Delta t_{p_{K-1}p_0} \neq 0,$$
3. and that there exists two non-intersecting subsets of

$$\{\rho_{p_0}, \rho_{p_1}, \cdots, \rho_{p_{K-1}}\},$$

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 $$K.$$ 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

$$J(K) = \text{tr}\left( (D(K)^T D(K))^{-1} \right) + \gamma \sum_{i=1}^{\lvert K \rvert} \Delta t_{p_ip_{i-1}},$$

where $$D(K) = [\rho_{p_0 \rho_{p_1 \rho_{p_2} \cdots \rho_{p_{K-1}}}]}^T,$$ $$\lvert K \rvert$$ denotes the number of elements in $$K,$$ and $$\gamma$$ is a tuning parameter. The first term in (19) is the square of the GDOP, also noting that the form of the $$D(K)$$ 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 $$\gamma$$ is used to balance the two terms, and its effect is illustrated in Figure 3. Finally, it is also possible to recursively compute the matrix $$\Lambda = (D(K)^T D(K))^{-1},$$ which avoids recomputing the inverse when a new keypoint is added to $$K.$$ Defining $$\tilde{K} = K \cup \{p\},$$ it can be shown that

$$(D(\tilde{K})^T D(\tilde{K}))^{-1} = (D(K)^T D(K) + \rho_p^T \rho_p)^{-1}$$

(20)

$$\Lambda = \frac{\Lambda \rho_p \rho_p^T \Lambda}{1 + \rho_p^T \Lambda \rho_p},$$

(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 $$K$$ with the 4 most recent indices in order to produce an invertible $$(D(K)^T D(K))^{-1}$$ 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 $$u_k = \tilde{u}_{a_k}^{acc,1} - \tilde{u}_{a_k}^{acc,2},$$ it is straightforward to show that $$b_{ji}$$ in (17) can be written as

$$b_{ji} = b_{ji}^{acc,1} - b_{ji}^{acc,2},$$

where $$b_{ji}^{acc,1}$$ and $$b_{ji}^{acc,2}$$ only depend on measurements from Agents 1 and 2, respectively. This has a significant implication:
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 \textsc{GetKeypoints} ($k, K, \mathcal{K}_{\text{old}}$)
2: \hspace{1em} $\mathcal{K} \leftarrow \{k - 3, k - 2, k - 1, k\}$
3: \hspace{1em} $\mathcal{C} \leftarrow \mathcal{K}_{\text{old}} \cup \{\max(\mathcal{K}_{\text{old}}) + 1, \ldots, k - 4\}$
4: \hspace{1em} for $i = 1, \ldots, K - 4$ do
5: \hspace{2em} $J_{\text{best}} \leftarrow \infty$
6: \hspace{2em} for all $p \in \mathcal{C}$ do
7: \hspace{3em} $\mathcal{K}_{\text{temp}} \leftarrow \mathcal{K} \cup \{p\}$
8: \hspace{3em} if $J(\mathcal{K}_{\text{temp}}) \leq J_{\text{best}}$ then
9: \hspace{4em} $J_{\text{best}} \leftarrow J(\mathcal{K}_{\text{temp}})$
10: \hspace{4em} $p_{\text{best}} \leftarrow p$
11: \hspace{2em} end if
12: \hspace{1em} end for
13: \hspace{1em} $\mathcal{K} \leftarrow \mathcal{K} \cup \{p_{\text{best}}\}$
14: \hspace{1em} $\mathcal{C} \leftarrow \mathcal{C} \setminus \{p_{\text{best}}\}$
15: end for
16: return $\mathcal{K}$

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.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magnetometer std. dev.</td>
<td>1 $\mu K$</td>
<td></td>
</tr>
<tr>
<td>Gyroscope std. dev.</td>
<td>0.001 rad/s</td>
<td></td>
</tr>
<tr>
<td>Accelerometer std. dev.</td>
<td>0.01 m/s²</td>
<td></td>
</tr>
<tr>
<td>Distance std. dev.</td>
<td>0.1 m</td>
<td></td>
</tr>
<tr>
<td>Initial position std. dev.</td>
<td>0.8 m</td>
<td></td>
</tr>
<tr>
<td>Initial velocity std. dev.</td>
<td>0.1 m/s</td>
<td></td>
</tr>
<tr>
<td>Initial attitude std. dev.</td>
<td>0.001 red</td>
<td></td>
</tr>
<tr>
<td>Accl/Gyro/Mag frequency</td>
<td>100 Hz</td>
<td></td>
</tr>
<tr>
<td>Distance frequency</td>
<td>10 Hz</td>
<td></td>
</tr>
<tr>
<td>RPE window size $K$</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>RPE keypoint selection parameter $\gamma$</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

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 9. 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,
TABLE II
POSITION RMSE OF EXPERIMENTAL TRIALS

<table>
<thead>
<tr>
<th></th>
<th>Proposed</th>
<th>Vanilla SWF</th>
<th>EKF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trial 1 (one static agent)</td>
<td>0.68 m</td>
<td>1.41 m</td>
<td>2.11 m</td>
</tr>
<tr>
<td>Trial 2 (both moving)</td>
<td>0.55 m</td>
<td>1.22 m</td>
<td>1.71 m</td>
</tr>
<tr>
<td>Trial 3 (both moving)</td>
<td>0.86 m</td>
<td>1.56 m</td>
<td>2.69 m</td>
</tr>
<tr>
<td>Trial 4 (spinning motors)</td>
<td>0.87 m</td>
<td>1.34 m</td>
<td>1.12 m</td>
</tr>
</tbody>
</table>

each cost a total of 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 $5 \times 4 \times 2$ m.

When using the quadrotors, the motors are spinning without propellers 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 in one of the trials. 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.

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

PROOF OF THEOREM III.1

**Proof:** Satisfying the rank condition in (18) amounts to showing that there are 6 linearly independent columns in the matrix \( O \). Since there are \( K \) columns in \( O \), \( K \geq 6 \), 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 \( \alpha \) 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

\[
\begin{bmatrix}
\rho_0 & \rho_1 & \rho_2 & \rho_3 & \rho_4 & \rho_5 \\
0 & \Delta t_1 \rho_1 & \Delta t_2 \rho_2 & \Delta t_3 \rho_3 & \Delta t_4 \rho_4 & \Delta t_5 \rho_5
\end{bmatrix}
\]

have rank 6. Without loss of generality, assume \{\rho_0, \rho_1, \rho_2\} are linearly independent. It immediately follows that the first three columns of \( O \) are linearly independent. Consider now \( \rho_3 \), again without loss of generality. Since \( \rho_0, \rho_1, \rho_2 \in \mathbb{R}^3 \), \( \rho_3 \) is linearly dependent on \( \rho_0, \rho_1, \rho_2 \) and can be written as

\[
\rho_3 = \alpha_0 \rho_0 + \alpha_1 \rho_1 + \alpha_2 \rho_2.
\]  

If the fourth column of \( O \) were linearly dependent on the first three, then there would exist \( \beta_0, \beta_1, \beta_2 \) such that

\[
\begin{bmatrix}
\rho_3 \\
\Delta t_3 \rho_3
\end{bmatrix}
= \beta_0 \begin{bmatrix}
\rho_0 \\
0
\end{bmatrix}
+ \beta_1 \begin{bmatrix}
\rho_1 \\
\Delta t_1 \rho_1
\end{bmatrix}
+ \beta_2 \begin{bmatrix}
\rho_2 \\
\Delta t_2 \rho_2
\end{bmatrix}.
\]  

Substituting (22) into (23) yields

\[
\begin{bmatrix}
\alpha_0 \rho_0 + \alpha_1 \rho_1 + \alpha_2 \rho_2 \\
\Delta t_3 (\alpha_0 \rho_0 + \alpha_1 \rho_1 + \alpha_2 \rho_2)
\end{bmatrix}
= \beta_0 \begin{bmatrix}
\rho_0 \\
0
\end{bmatrix}
+ \beta_1 \begin{bmatrix}
\rho_1 \\
\Delta t_1 \rho_1
\end{bmatrix}
+ \beta_2 \begin{bmatrix}
\rho_2 \\
\Delta t_2 \rho_2
\end{bmatrix}.
\]  

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

\[
\Delta t_3 \alpha_0 \rho_0 + \Delta t_3 \alpha_1 \rho_1 + \Delta t_3 \alpha_2 \rho_2 = \alpha_1 \Delta t_1 \rho_1 + \alpha_2 \Delta t_2 \rho_2.
\]  

Hence, the fourth column of \( O \) 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 III.1 means \( \Delta t_3 \neq \Delta t_2, \Delta t_3 \neq \Delta t_1 \), hence proving the linear independence of the fourth column of \( O \) from the first three. Identical proofs show that the fifth and sixth columns of \( O \) are also linearly independent from the first three. If additionally \{\rho_3, \rho_4, \rho_5\} are linearly independent, then the fourth, fifth, and sixth columns of \( O \) will be linearly independent from each other, in addition to each being linearly independent from the first three. Hence, all 6 columns of \( O \) will be linearly independent, giving it a rank of 6. Since this proof was done for a completely arbitrary choice of linearly independent sets \{\rho_0, \rho_1, \rho_2\} and \{\rho_3, \rho_4, \rho_5\}, it is only required that there exists two distinct non-intersecting subsets of \{\rho_0, \rho_1, \ldots, \rho_K\}, each containing 3 linearly independent elements, which is Condition 3.

REFERENCES


