
جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Peer-to-Peer Relative Localization of Aerial Robots With Ultrawideband Sensors

Item Type	Article
Authors	Güler, Samet; Abdelkader, Mohamed; Shamma, Jeff S.
Citation	Guler, S., Abdelkader, M., & Shamma, J. S. (2020). Peer-to-Peer Relative Localization of Aerial Robots With Ultrawideband Sensors. IEEE Transactions on Control Systems Technology, 1–16. doi:10.1109/tcst.2020.3027627
Eprint version	Post-print
DOI	10.1109/TCST.2020.3027627
Publisher	Institute of Electrical and Electronics Engineers (IEEE)
Journal	IEEE Transactions on Control Systems Technology
Rights	(c) 2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.
Download date	07/02/2022 10:31:03
Link to Item	http://hdl.handle.net/10754/665552

Peer-to-Peer Relative Localization of Aerial Robots with Ultrawideband Sensors

Samet Güler, Mohamed Abdelkader, and Jeff S. Shamma, *Fellow, IEEE*

***Abstract*— Robots in swarms take advantage of localization infrastructure such as a motion capture system or GPS sensors to obtain their global position, which can then be communicated to other robots for swarm coordination. However, the availability of localization infrastructure need not be guaranteed, e.g., in GPS denied environments. Likewise, the communication overhead associated with broadcasting locations may be undesirable. For reliable and versatile operation in a swarm, robots must sense each other and interact locally. Motivated by this requirement, we propose an onboard relative localization framework for multi-robot systems. The setup consists of an anchor robot with three onboard ultrawideband (UWB) sensors and a tag robot with a single onboard UWB sensor. The anchor robot utilizes the three UWB sensors to estimate the tag robot's location by using its onboard sensing and computational capabilities solely, without explicit inter-robot communication. Because the anchor UWB sensors lack the physical separation that is typical in fixed UWB localization systems, we introduce filtering methods to improve estimation of the tag's location. In particular, we utilize a mixture Monte-Carlo Localization (MCL) approach to capture maneuvers of the tag robot with acceptable precision. We validate the effectiveness of our algorithm with simulations as well as indoor and outdoor field experiments on a two-drone setup. The proposed mixture MCL algorithm yields highly accurate estimates for various speed profiles of the tag robot and demonstrates superior performance over the standard particle filter and the extended Kalman filter.**

***Index Terms*— Multi-robot localization, Ultrawideband (UWB) sensor, Monte-Carlo Localization, Formation control**

I. Introduction

Autonomous mobile robots have been deployed in various civil and military applications such as goods delivery in urban areas, service industry, manufacturing, and border security. A mobile robot's decision mechanism can function reliably only with a high-performance localization framework. The mobile robot localization problem is defined as developing a hypothesis about a robot's location in a given environment which is usually represented by a set of landmarks or a detailed map.

This work was supported by King Abdullah University of Science and Technology (KAUST). (Corresponding Author: Samet Güler.)

S. Güler is with the Electrical and Electronics Engineering, Abdullah Gül University, Kayseri 38080, Turkey. (e-mail: samet.guler@agu.edu.tr).

M. Abdelkader is with Algorythma, Abu Dhabi, United Arab Emirates. (e-mail: mohamedashraf123@gmail.com).

J. S. Shamma is with the Computer, Electrical and Mathematical Sciences and Engineering, KAUST, Thuwal 23955–6900, Saudi Arabia. (e-mail: jeff.shamma@kaust.edu.sa).

The standard methods for mobile robot localization include geometric, optimization, and filtering methods. The geometric and optimization based approaches take a set of anchor-sensor distance measurements and produce solutions for the possible sensor locations based on distance geometry. The accuracy of both methods is impacted by measurement noises and motion of the localized sensors. If the distance measurements are constantly available, then various filtering approaches can be used. Bayesian approaches, particularly the extended Kalman filter (EKF), are commonly employed for localization. Filtering based methods first predict the robot motion with inertial sensors then update the belief with exteroceptive sensor data. EKF localization yields high performance in many scenarios. However, tuning the EKF parameters requires extensive time and experiments, the initial condition significantly affects the EKF performance, and EKF localization usually does not suffice to track agile robots.

One can obtain sensor data for mobile robot localization in two ways. In the first method, sensor data related to the robot position are obtained from a fixed infrastructure in a well-designed environment. A conventional indoor setup for this approach comprises at least three anchors, a ground station, and sensors mounted on robots (Fig. 1a). The anchors are installed in a room at certain positions separated at the maximum distances from each other so that they form a large convex hull. Therefore, the mobile robot always moves inside the convex hull of the anchors. The ground station estimates the positions of the mobile robots and transmits the estimates to the robots continuously. A common example of these localization setups is motion capture (mocap) systems. Generally, this method yields highly accurate location estimate with high data rate. However, this framework entirely depends on the environment, i.e., localization can be performed only in that particular environment. In the second method, the robot implements a localization algorithm with its onboard sensing and computational capabilities solely. The robot either measures its distances and bearing angles to specific landmarks or employs vision sensors to identify its location in a given map of the environment.

In a multi-robot system, a mobile robot needs to localize itself with respect to the other robots as well. Recently, several works exploited the advantages of both approaches for multi-robot localization under the framework of onboard anchor configuration (Fig. 1b) [1]–[3]. In this framework, a robot is equipped with a set of anchors onboard, with the inter-anchor distances being limited by the physical characteristics of the robot. Notably, the resulting anchor configuration forms

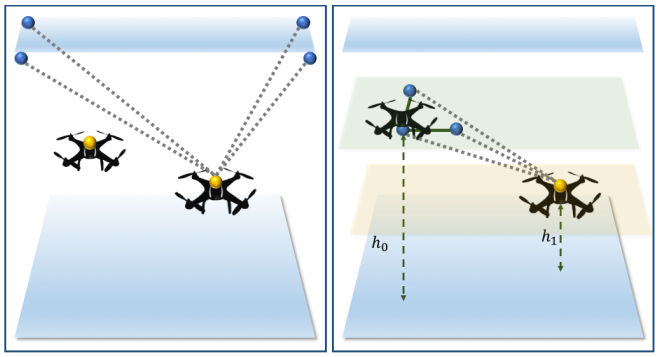


Fig. 1: Left: A conventional localization framework. A set of anchors are mounted at certain locations for high precision positioning. No robot is used as landmark. Right: An onboard localization framework. A robot acts as a set of anchors (three blue sensors) to localize another robot (with the yellow tag sensor). The horizontal planes on which robot $\mathsf{R}_0, \mathsf{R}_1$ fly are shown with the transparent green and orange planes, respectively.

a smaller convex hull than in the conventional case, and the localized robot always lies outside of the anchors' convex hull. Accordingly, several filtering approaches need to be applied to improve the estimation performance.

We are interested in a completely distributed, real-time, infrastructure-free, and onboard localization algorithm for multi-robot systems. Particularly, we propose a localization framework for a two-robot system that utilizes three UWB anchors on one robot and a single UWB sensor on the other. We use a mixture Monte-Carlo Localization (MCL) algorithm based on a particle filter to estimate the relative position between the two robots. The proposed filter utilizes the most recent sensory information in the prediction phase based on a probabilistic measure. This framework allows the anchor robot to track agile maneuvers of the localized robot with a small number of particles, which would otherwise require a larger number of particles under standard particle filtering. Remarkably, this framework does not utilize an explicit communication structure, i.e., the robots do not communicate with each other or with a ground station. Furthermore, we demonstrate with experiments that the estimation accuracy suffices to implement some formation control objectives such as relative position maintenance when integrated with simple motion control algorithms.

Our contributions are as follows:

- We propose an MCL approach based on an onboard UWB configuration for the multi-robot localization problem that is suited to handle agile robot motions by using a small number of particles.
- We relax the assumptions of a good initial condition and *a priori* information about the velocity profile of the localized robot, without imposing a communication framework.
- We combine the localization algorithm with simple motion control laws to solve a formation control objective and demonstrate its experimental performance on a two-drone system.

The rest of the paper is organized as follows. Section II reviews the literature on mobile robot localization with detailed comparisons. Section III presents the multi-robot localization problem in general terms. Section IV summarizes the standard and dual MCL algorithms. Section V presents the proposed localization algorithm. Sections VI and VII demonstrate the simulation and experimental results. Section VIII gives a discussion on the results. Finally, Section IX contains concluding remarks.

II. Related Works

Mobile robot localization has been studied extensively; see [4]–[9] and the references therein for a detailed survey. In the literature, the localization problem for a multi-robot system was translated in two ways based on the control objectives: Self-localization and relative localization. Self-localization refers to computing a robot's location in a global coordinate frame by a fixed localization system. Accordingly, the multi-robot localization problem can be posed as self-localization of each robot in the system. However, this approach does not lead to distributed applications for several reasons listed below. Relative localization refers to estimating relative quantities between robots in body frames of the robots.

The majority of previous works on indoor localization utilized the conventional anchor configuration [10]–[12] to solve self-localization. Reference [10] combines UWB sensors with a visual inertial system to correct drifts when building maps. The authors in [12] utilize particle filtering to handle the multimodal error behavior of the non-line-of-sight (NLOS) UWB measurements. Furthermore, the conventional configuration is used in [11] to solve the single-robot as well as the multi-robot localization problem. The authors in [13]–[16] exploited inter-robot communication and proposed cooperative EKF localization architectures to improve estimation accuracy in multi-robot systems. The key idea in [13]–[16] is that each robot receives estimation related information from its neighbor robots through communication.

In outdoor environments, several works employed GPS sensors on board the robots to achieve formation control tasks. For instance, reference [17] equipped every robot in a swarm with a GPS receiver and demonstrated a flock behavior with drones by the aid of inter-robot communication. Although the frameworks in [13]–[15], [17] improve the estimation performance for multi-robot systems with undirected graphs, they still depend on a global positioning system (GPS or mocap system) and bring an extra cost for the additional communication layer, which makes their reliability aspect questionable in occluded environments.

Toward the goal of freeing the localization framework from environment completely, recently several works have considered an onboard anchor configuration where a moving vehicle equipped with a set of anchors localizes another robot or human [1]–[3], [18], [19]. In [1], a quadrotor equipped with UWB anchors on board tracks a target with a single UWB sensor by employing an iterated EKF. However, the approach of [1] still depends on infrastructure because the quadrotor control relies on Mocap or GPS data instead of the localization feedback. In [2], a quadrotor searches for safe paths for a

ground vehicle and localizes itself with respect to the vehicle by unscented Kalman filter and optimization techniques.

An alternative method to estimate the tag location is designing custom UWB sensor kits based on the angle-of-arrival (AOA) method [20], [21]. This procedure requires modifying the anchor antenna to measure the angle of the received UWB signals. For instance, the authors in [20] installed two antennas on the anchor sensor and used the phase-difference-of-arrival (PDOA) for precise bearing angle estimation.

Trilateration method takes a set of anchor-sensor distance measurements at a particular time instant and produces a closed-form solution for possible sensor locations based on the distance geometry [22]–[24]. Similarly, optimization methods minimize the additive noise on a set of anchor-sensor distances, subject to equalities obtained from the geometric structure [2], [25]. However, neither approach is preferred for mobile robot localization standalone because they suffer from measurement noises. To obtain reliable estimation results for mobile robots under noisy distance measurements, Bayesian methods are commonly employed. The authors in [26] designed an EKF algorithm with sonar anchors. Reference [27] fused inertial and UWB sensor data to estimate a quadrotor's position. In [12], a particle filter based localization algorithm was applied to UWB distance data for both LOS and NLOS measurement cases. Similarly, in [11], particle filtering was applied to localize single- and multi-robot systems in a well-designed environment. The authors in [16] proposed a simple model that captures the multimodal error behavior of the UWB measurements in NLOS environments and designed a particle filter based localization algorithm for multi-robot systems in an indoor environment. Our framework differs from [2], [11], [12], [16], [22]–[27] in that these works utilized a set of UWB beacons located at known positions in a room to provide the robots with distance data, which makes the algorithms infrastructure-dependent.

MCL algorithms have been developed to track the states of non-linear, non-Gaussian models [5], [28]. Particle filters have been employed in robot localization and mapping problems [12], [16], [29], [30], for object detection in images [31], [32], in wireless communication [33], and in many other applications. The dual MCL approach was initially proposed in [9] to handle the particle depletion issue in cases where the state transition distribution covariance is incomparably higher than the measurement covariance. This technique was used in [34] to solve the grid mapping problem with precise laser range finders. This framework, with suitable modifications, fits well our particular problem setting because the UWB measurements produce a better prediction about an agile robot's current location than the state transition distribution of the robots. Therefore, we opted for a mixture MCL algorithm to solve our particular localization objective.

Preliminary versions of this work were presented in [3], [19]. The main differences between the current work and [3] are twofold. First, we propose an MCL algorithm here whereas [3] proposed EKF based algorithms. Second, while non-holonomic ground robots were considered in [3], here we develop the results for holonomic vehicles as well. In [19], a dual MCL algorithm was utilized to solve the multi-robot localization objective. Here, we extend the result of [19] with a mixture MCL algorithm and demonstrate a comprehensive set of simulation and experimental results.

III. System Definition

Consider a two-robot system $\{\mathsf{R}_0, \mathsf{R}_1\}$ where R_i denotes the ith robot. Robot R_0 constitutes the base for our localization architecture, and thus the derivation of our algorithm varies based on its motion characteristics. Accordingly, we consider two commonly used motion models for robot R_0: Holonomic and non-holonomic. We assume that robots $\mathsf{R}_0, \mathsf{R}_1$ always fly on horizontal planes at operational altitudes h_0, h_1, respectively, from the ground. These planes are parallel to the ground, and the operational altitudes do not need to be the same. We set $h_0 = h_1$ in the design process for ease of calculations and describe the small algorithmic modifications for $h_0 \neq h_1$ in later sections.

Let $\mathcal{F}_\mathrm{G} \subseteq \Re^2$ denote the global frame and $\mathcal{F}_\mathrm{i}$ denote the body frame of R_i. We represent the vectors in the body frame $\mathcal{F}_0$ of robot R_0 unless we denote a vector with superscript v^G which represents frame $\mathcal{F}_\mathrm{G}$. For the case of holonomic R_0, we consider the following kinematics model:

$$\begin{bmatrix} \mathrm{p}_{k+1}^{0\mathrm{G}} \\ \mathrm{v}_{k+1}^{0\mathrm{G}} \end{bmatrix} = \begin{bmatrix} \mathrm{I}_2 & T_s\mathrm{I}_2 \\ 0_2 & \mathrm{I}_2 \end{bmatrix} \begin{bmatrix} \mathrm{p}_{k}^{0\mathrm{G}} \\ \mathrm{v}_{k}^{0\mathrm{G}} \end{bmatrix} + \begin{bmatrix} (T_s^2/2)\mathrm{I}_2 \\ T_s\mathrm{I}_2 \end{bmatrix} \mathrm{a}_k^{0\mathrm{G}} + \mathrm{w}_k^0, \quad (1)$$

where $\mathrm{p}^{0\mathrm{G}} = \left[x^{0\mathrm{G}}, y^{0\mathrm{G}}\right]^\top \in \Re^2$ is the position, $\mathrm{v}^{0\mathrm{G}} \in \Re^2$ is the velocity, and $\mathrm{a}^{0\mathrm{G}} \in \Re^2$ is the acceleration of robot R_0, k is the time step, T_s is the sampling time, and $\mathrm{w}^0 \in \Re^4$ is the process noise. The model (1) approximates the behavior of a second-order mechanical system where the acceleration between two successive time steps is assumed constant. Particularly, the horizontal motion of a quadrotor is well approximated by this model when it flies at a constant altitude and its internal nonlinear dynamics are controlled by an independent, low-level microcontroller.

For the case of nonholonomic R_0, we consider the following kinematics model:

$$\mathrm{p}_{k+1}^{0\mathrm{G}} = \mathrm{p}_k^{0\mathrm{G}} + \begin{bmatrix} \cos(\theta_k^0)T_s & 0 \\ \sin(\theta_k^0)T_s & 0 \\ 0 & T_s \end{bmatrix} \begin{bmatrix} v_k^0 \\ \omega_k^0 \end{bmatrix} + \mathrm{w}_k^0, \quad (2)$$

where $\mathrm{p}^{0\mathrm{G}} = \left[x^{0\mathrm{G}}, y^{0\mathrm{G}}, \theta^{0\mathrm{G}}\right]^\top \in \Re^3$ is the vector of position and the heading angle, v^0 and ω^0 are the linear and angular speeds, respectively, and $\mathrm{w}^0 \in \Re^3$ is the process noise.

We consider a holonomic kinematics model for robot R_1:

$$\begin{bmatrix} \mathrm{p}_{k+1}^{1\mathrm{G}} \\ \mathrm{v}_{k+1}^{1\mathrm{G}} \end{bmatrix} = \begin{bmatrix} \mathrm{I}_2 & T_s\mathrm{I}_2 \\ 0_2 & \mathrm{I}_2 \end{bmatrix} \begin{bmatrix} \mathrm{p}_{k}^{1\mathrm{G}} \\ \mathrm{v}_{k}^{1\mathrm{G}} \end{bmatrix} + \mathrm{w}_k^1, \quad (3)$$

where $\mathrm{p}^{1\mathrm{G}}, \mathrm{v}^{1\mathrm{G}} \in \Re^2$ are the position and velocity vectors, and $\mathrm{w}^1 \in \Re^4$ is the random acceleration input which satisfies

$$\|\mathrm{w}_k^1\| \leq \bar{\mathrm{w}}^1, \quad (4)$$

for all k and for some finite $\bar{\mathrm{w}}^1 > 0$. For convenience, we assume that the following saturation operation is in effect in robot R_1:

$$\mathrm{v}^{1,\min} \leq \mathrm{v}_k^{1\mathrm{G}} \leq \mathrm{v}^{1,\max},$$

where the constant vectors $\mathrm{v}^{1,\min}, \mathrm{v}^{1,\max} \in \Re^2$ are pre-defined based on the physical characteristics of robot R_1.

We denote robot R_0 as the *anchor robot* and mount three UWB anchors on robot R_0 at positions $\mathrm{q}^1, \mathrm{q}^2, \mathrm{q}^3$ with respect to frame $\mathcal{F}_0$ as follows (Fig. 1b):

$$\mathrm{q}^1 = [l, 0]^\top, \ \mathrm{q}^2 = [0, 0]^\top, \ \mathrm{q}^3 = [0, l]^\top, \tag{5}$$

where the design parameter l is determined based on the physical characteristics of robot R_0. Thus, the three anchors are rigidly linked to each other on R_0. We set $l = 1$ in the algorithm design process and discuss the modifications for different values of l in Section VII.

The observation vector consists of three distance measurements between q^i on R_0 and a UWB sensor at p^1, the center of robot R_1, i.e.,

$$\mathrm{z}_k = \left[d_k^1, d_k^2, d_k^3\right]^\top, \tag{6}$$

$$d_k^i = \bar{d}_k^i + \epsilon_k^i, \quad i \in \{1, 2, 3\}, \tag{7}$$

where

$$\bar{d}_k^i = \|\mathrm{p}_k^1 - \mathrm{q}^i\|, \tag{8}$$

denotes the true distance between q^i and p^1 (in frame $\mathcal{F}_0$), and ϵ^i denotes the measurement noise. As anticipated from the geometry of the system, p^1 always remains outside of the convex hull of the triangle $T(\mathrm{q}^1, \mathrm{q}^2, \mathrm{q}^3)$.

We consider the following state vector to represent the relative quantities between robots R_0 and R_1:

$$\mathrm{x}_k = [\mathrm{r}_k^\top, \ (\mathrm{v}_k^{1\mathrm{G}})^\top]^\top, \tag{9}$$

where r_k is the relative position between robots R_0 and R_1 in frame $\mathcal{F}_0$, and $\mathrm{v}^{1\mathrm{G}}$ is the velocity of robot R_1 in frame $\mathcal{F}_\mathrm{G}$.

We assume that frame $\mathcal{F}_0$ coincides with frame $\mathcal{F}_\mathrm{G}$ at $k = 0$. For holonomic R_0, the dynamics of the relative position r is trivially derived as follows:

$$\mathrm{r}_{k+1} = \mathrm{r}_k + (\mathrm{v}_k^{1\mathrm{G}} - \mathrm{v}_k^{0\mathrm{G}})T_s - 0.5\mathrm{a}_k^0 T_s^2 + \tilde{\mathrm{w}}_k, \tag{10}$$

with $\tilde{\mathrm{w}}_k$ being the noise vector. Here, we focus on holonomic robots which can perform agile maneuvers and refer to [3] for derivation of the relative position dynamics for nonholonomic R_0.

We assume that the robots do not have an explicit communication structure, i.e., they do not exchange information with each other. Furthermore, we assume that the setup does not include a ground station which can collect sensory information and implement the estimation algorithm. Hence, the robots are to utilize their onboard sensors solely for the estimation and motion control objectives.

Robot R_0 assumes that robot R_1 moves based on the motion model (3). Also, robot R_0 knows the motion capability of robot R_1 which is encoded in $\mathrm{v}^{1,\min}, \mathrm{v}^{1,\max}, \bar{\mathrm{w}}^1$. However, robot R_0 does not have access to the instant velocity $\mathrm{v}_k^{1\mathrm{G}}$. This assumption reflects a realistic multi-robot scenario where each robot is informed with the motion capabilities of the other robot but cannot access the other robot's states such as position and velocity in real time.

Denoting the estimate of x_k by $\hat{\mathrm{x}}_k$, we define the objective in this paper as follows. Given the system $\{\mathsf{R}_0, \mathsf{R}_1\}$ of robot R_0 with the motion model (1) or (2), and robot R_1 with the motion model (3), the noisy distances d_k^1, d_k^2, d_k^3, a suitable initial condition $\hat{\mathrm{x}}_0$, and the aforementioned assumptions, generate the estimate $\hat{\mathrm{x}}_k$ for $k \geq 1$ so that the error $e_k = \|\hat{\mathrm{x}}_k - \mathrm{x}_k\|$ is minimized.

In the following section, we review particle filters. Then, we present our algorithm in Section V.

IV. Particle Filter Review

Unlike the Kalman filter and its variations, a particle filter does not use a compact state space model to represent the state distribution. Instead, it uses a large number of samples to represent the current belief about the state. Similarly to other Bayesian filters, a particle filter generates the state estimate in two phases: prediction and update. The resampling process in the update phase forms an important part of particle filters, where the samples are rearranged based on the current exteroceptive measurement data. Particle filters can be applied to models where the noise shows a non-Gaussian behavior because inherently particle filters can track any distribution under certain assumptions. We now summarize the standard particle filter algorithm and refer the reader to [29], [33], [35] for detailed descriptions.

Consider a dynamical system with state x, input u, and output z. We denote by $\mathrm{bel}(\mathrm{x}_k)$ the current belief, or the posterior probability, of the state distribution. Ideally, the belief represents the actual state distribution as follows:

$$\mathrm{bel}(\mathrm{x}_k) = \pi(\mathrm{x}_k | \mathrm{z}_{1:k}, \mathrm{u}_{1:k}), \tag{1}$$

where $\mathrm{u}_{1:k}$ and $\mathrm{z}_{1:k}$ denote the inputs and observations up to time step k, respectively. At any $t = kT_s$, where T_s denotes the sample time, a particle filter "approximates" the distribution (1) with the set of samples $\mathrm{S}_k = \{\mathrm{s}_k^1, \cdots, \mathrm{s}_k^N\}$, where $N \in \mathbb{Z}_+$ denotes the number of samples, and $\mathrm{s}_k^i = \{\mathrm{x}_k^i, w_k^i\}$ denotes the ith sample with the state hypothesis x^i and the corresponding importance weight $w^i \in [0, 1)$. At each k, each sample denotes the algorithm's hypothesis on where the system state x may lie. Thus, the combination of the hypothesis x^i with their associated weights w^i constitute $\mathrm{bel}(\mathrm{x}_k)$ to represent an approximation of the posterior distribution $\pi(\mathrm{x}_k | \mathrm{z}_{1:k}, \mathrm{u}_{1:k})$. Conceptually, as N approaches infinity, the belief improves, i.e., $\mathrm{bel}(\mathrm{x}_k)$ approaches $\pi(\mathrm{x}_k | \mathrm{z}_{1:k}, \mathrm{u}_{1:k})$, at the expense of increased computational complexity. Usually, N is chosen large, e.g., $N > 1000$.

For the prediction phase, a common practice in mobile robot localization is to propagate the samples in the set S_{k-1} proportional to the robot's state transition distribution which depends only on the last state and the current input, i.e.,

$$\varphi_k \sim \pi(\mathrm{x}_k | \mathrm{x}_{k-1}, \mathrm{u}_k), \tag{2}$$

where φ_k denotes the "proposal distribution". Subsequently, in the update phase, the weights are calculated based on the observation model as follows:

$$w_k^i = \eta \pi(\mathrm{z}_k | \mathrm{x}_k^i). \tag{3}$$

Next, the new set of samples S_k constructed with a resampling process based on the weights $\{w_k^i\}$ represents the posterior probability $\mathrm{bel}(\mathrm{x}_k)$.

Although the framework (2)-(3) usually yields high performance, it may cause the *particle depletion* issue [5] in some applications including our specific problem. Especially, when the variance of the exteroceptive measurement model is much lower than the variance of the robot's state transition distribution, propagating the particles with the proposal distribution (2) may populate most of the particles in regions that do not align with the exteroceptive measurement model. This misalignment would set the weights of the majority (or all) of the particles to small values and reduce the efficiency of the resampling process. To address this issue, several alternative proposal distributions were proposed. In [9], [30], [36], the authors invert the roles of the prediction and update phases. In [9], the measurement model is used in the proposal distribution:

$$\varphi_k = \frac{\pi(\mathrm{z}_k|\mathrm{x}_k)}{\pi^n_k}, \tag{4}$$

where π^n is a normalizer. Accordingly, the following importance weights are used:

$$w^i_k = \pi(\mathrm{x}_k|\mathrm{u}_{1:k}, \mathrm{z}_{1:k-1}), \tag{5}$$

where $\pi(\mathrm{x}_k|\mathrm{u}_{1:k}, \mathrm{z}_{1:k-1})$ is calculated by an extra sampling process at each time step. In other words, the belief is predicted with the exteroceptive measurements, and the update is performed based on the motion model, in contrast to the standard particle filter. Therefore, this approach populates the particles around the most recent observation and hence solves the particle depletion issue for some scenarios. However, this alternative approach usually yields chattering outcomes because the state hypotheses are driven by measurements which may contain signals with low signal-to-noise ratio.

To overcome the issues above, the authors in [9] mix the two approaches and propose the mixture MCL algorithm. The designer sets a threshold $\phi \in [0, 1]$ and at every step implements the dual PF algorithm with probability ϕ and the standard PF algorithm with probability $1 - \phi$. We adopt this approach to solve our localization problem in the following section. We use the terms MCL algorithm and PF algorithm interchangeably in the rest of the paper.

V. Peer-to-Peer Relative Localization

We aim to design a distributed algorithm where robot R_0 estimates the relative position r in its local frame $\mathcal{F}_0$ in real-time by employing its own computational devices solely. Our design is assumed to include neither a central computational unit, e.g., a ground station, nor an explicit communication layer. However, the robots sense ranges with the onboard UWB sensors by an implicit communication mechanism, which we consider as a ranging mechanism similar to the case of the laser range finder with a receiver. Since the measurement data acquired from the sensors are noisy, the algorithm has to deal with uncertainties. Also, the proposed algorithm's performance should suffice to be used as feedback to further motion control algorithms on robot R_0.

In the remainder of this section, we propose our localization framework for a two-robot system. We describe the details of the algorithm in Section V-A-V-C. Then, we give the pseudo-code of our algorithm in V-D. We give the calibration procedure of the UWB sensors in Section VII-B. We discuss the details of data acquisition and implementation in Section VII.

A. The Block Diagram

The block diagram of the two robot system $\{\mathsf{R}_0, \mathsf{R}_1\}$ is depicted in Fig. 2. Robot R_0 is to generate the estimate vector $\hat{\mathrm{x}}$. The filtering algorithm runs on board the anchor robot R_0. Three raw UWB distance values are acquired from the UWB anchors and passed to the UWB CALIBRATION & SMOOTHING block to eliminate biases based on a calibration procedure and smoothen the chattering measurement signal. Afterwards, the MEASUREMENT MODEL block accepts the three distance measurements (and the IMU measurement if R_0 is nonholonomic) of robot R_0, and outputs the "constructed" measurements. The output signal is passed through another smoothing block. Finally, the FILTERING block generates the state estimate $\hat{\mathrm{x}}$, which is then relayed back to the motion controller of R_0 to close the motion control loop. Also, the state estimate can be transmitted to robot R_1 by communication to allow R_1 to use the estimate for better formation control performance, but we do not consider that case in this paper. We assume that the estimation takes place and is used in robot R_0 solely.

In the remainder of this section, we propose the localization algorithm by assuming that the motion of each robot is controlled by its low-level motion controller which is commanded by exogenous inputs. We study the integration of the localization output with the motion control algorithms in Sections VI and VII. Inspired by the dual MCL approach of [9], we now design the proposal distribution and resampling process of our particle filter. We start by constructing the observation model.

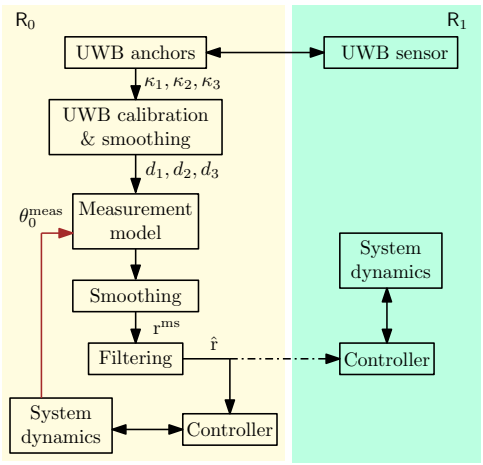


Fig. 2: The block diagram of the proposed framework. Robots R_0 and R_1 are represented with yellow and green backgrounds, respectively. The dashed arrow shows the optional data transmission from R_0 to R_1. The brown arrow denotes the heading angle feedback which is required for nonholonomic robot R_0 case only.

B. Observation Model

In this part, we model the distribution $\pi(\mathrm{z}_k|\mathrm{x}_k)$. The exteroceptive measurement model consists of three independent UWB measurements $\kappa^1, \kappa^2, \kappa^3$. First, we calibrate these measurements to generate $\tilde{\kappa}^i,\ i = \{1,2,3\}$. Details of the calibration procedure is given in Section VII-B. Next, we extract the height difference between the sensor's antenna and the anchors' antennas from $\tilde{\kappa}^i$ as follows:

$$\bar{\kappa}_k^i = \left(\left(\tilde{\kappa}_k^i\right)^2 - (h_0 - h_1)^2\right)^{1/2}, \quad i \in \{1,2,3\}.$$

Then, we filter the calibrated measurements $\bar{\kappa}^i$ with the first-order exponential smoothing algorithm as follows:

$$d_k^i = \alpha\bar{\kappa}_k^i + (1-\alpha)d_{k-1}^i, \quad i \in \{1,3\}, \tag{1}$$

where d_k^i denotes the smoothed distance measurement and $0 < \alpha \leq 1$ is a design parameter.

We use the constructed measurement model of [3] to map the distance measurements to a location estimate as follows:

$$\mathrm{r}_k^{\mathrm{fs}} = \lambda_k^1\mathrm{q}^1 + \lambda_k^3\mathrm{q}^3 = \left[\lambda_k^1, \lambda_k^3\right]^\top, \tag{2}$$

where r^{fs} denotes the measured position of R_1 in frame $\mathcal{F}_0$, $\mathrm{q}^1, \mathrm{q}^3$ are the anchor locations, and $\lambda^i = \mathrm{sgn}(\lambda^i)|\lambda^i|$ are the coordinates of p^1 in $\mathcal{F}_0$ (Fig. 5a). The function $\mathrm{sgn}(\cdot)$ is defined as $\mathrm{sgn}(x) = -x$ for $x < 0$, $\mathrm{sgn}(x) = x$ for $x > 0$, and $\mathrm{sgn}(x) = 0$ for $x = 0$. Notably, the line segments $L(\mathrm{q}^2, \mathrm{q}^1), L(\mathrm{q}^2, \mathrm{q}^3)$ form the virtual x-y axes of $\mathcal{F}_0$. We have the following geometric relations [25], [37]:

$$|\lambda_k^1| = \frac{|\mathcal{A}(\mathrm{p}_k^1, \mathrm{q}^2, \mathrm{q}^3)|}{|\mathcal{A}(\mathrm{q}^1, \mathrm{q}^2, \mathrm{q}^3)|}, \quad |\lambda_k^3| = \frac{|\mathcal{A}(\mathrm{p}_k^1, \mathrm{q}^1, \mathrm{q}^2)|}{|\mathcal{A}(\mathrm{q}^1, \mathrm{q}^2, \mathrm{q}^3)|}, \tag{3}$$

$$\mathrm{sgn}(\lambda_k^i) = \mathrm{sgn}((d_k^2)^2 + 1 - (d_k^i)^2), \tag{4}$$

where $\mathcal{A}(\mathrm{q}^1, \mathrm{q}^2, \mathrm{q}^3) = 0.5l^2$ denotes the area of the triangle formed by the three anchors on robot R_0. Particularly, if $l = 1\mathrm{m}$, we have that

$$|\mathcal{A}(\mathrm{p}_k^1, \mathrm{q}^2, \mathrm{q}^3)| = \frac{1}{4}\left(-\left((d_k^2)^2 - (d_k^3)^2\right)^2 + 2\left((d_k^2)^2 + (d_k^3)^2\right) - 1\right)^{1/2},$$

$$|\mathcal{A}(\mathrm{p}_k^1, \mathrm{q}^1, \mathrm{q}^2)| = \frac{1}{4}\left(-\left((d_k^1)^2 - (d_k^2)^2\right)^2 + 2\left((d_k^1)^2 + (d_k^2)^2\right) - 1\right)^{1/2}.$$

Therefore, we decompose the calculation of r^{fs} into two parts: λ^1 and λ^3. Further, we decompose the calculation of each λ^i into two parts, its magnitude and sign. To solve for $|\lambda_k^i|,\ i = (1,3)$, we calculate the possible intersections of the circles $\mathcal{C}(\mathrm{q}^j, d_k^j)$ and $\mathcal{C}(\mathrm{q}^2, d_k^2)$ for $j \in \{1,3\},\ j \neq i$. Then, we incorporate the distance d_k^i to calculate the half plane in which λ_k^i resides.

Consider the following inequalities:

$$d_k^i + d_k^2 > l, \quad |d_k^i - d_k^2| < l, \quad i \in \{1,3\}. \tag{5}$$

If condition (5) is satisfied, the method (3)-(4) eliminates the singularities which arise due to noisy distance measurements. We illustrate two cases for the construction of r^{fs} in Fig. 3. In Fig. 3a, we represent a case where the noisy measurements satisfy (5). Due to the distance measurements noises, the circles $\mathcal{C}(\mathrm{q}^i, d^i)$ intersect at multiple locations (green squares) instead of the correct target location. In this particular case, the trilateration algorithm does not yield a unique solution whereas our algorithm yields a unique location estimate (brown point).

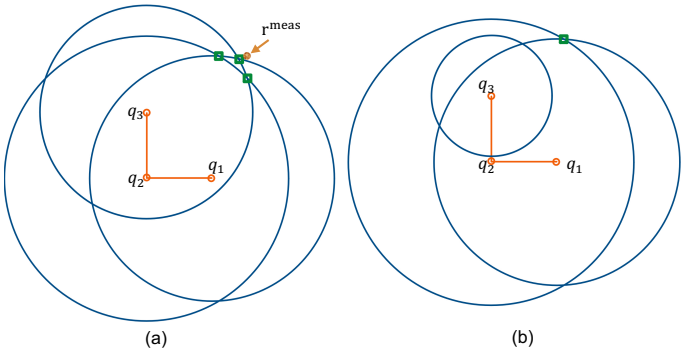


Fig. 3: (a) Construction of r^{fs} under noisy distance measurements: The proposed algorithm yields a unique point (brown), while the intersection of the circles corresponds to three points near the actual solution; (b) A singularity case where the noise ϵ^3 is high.

If condition (5) is not satisfied, at least two of the three circles $\mathcal{C}(\mathrm{q}^i, d_k^i),\ i \in \{1,2,3\}$, do not intersect. For instance, the measurements d^2 and d^3 do not satisfy (5) in Fig. 3b, which results in an infeasible measurement set. As a result, neither trilateration nor the method (2)-(4) can yield a solution. To overcome this issue, we modify the construction method (2)-(4) such that once the condition (5) is not satisfied, we use the last feasible observation vector as the current observation. We formalize this approach in Algorithm 1. By initializing $\mathrm{z}_0^{\mathrm{feas}}$ with a feasible $\mathrm{r}_k^{\mathrm{fs}}$, we guarantee to produce a unique $\mathrm{r}_k^{\mathrm{fs}}$ for all k. Also, with Algorithm 1, we can terminate the operation as a safety measure if the number of infeasible observations exceeds a certain threshold $\mathrm{ctr_thres}$. When $l \neq 1$, we scale the distances d^i and the constructed vector $\mathrm{r}_k^{\mathrm{fs}}$ accordingly. We note that if $l \leq 1$, the distance measurements scale up, and as a result, the measurement noises scale up as well, which may cause performance degradation. The smoothing and filtering algorithms are to suppress the adverse affects of this scaling. Indeed, the proposed algorithm performed well for $l = 0.44$ in our simulations and experiments (see Sections VI and VII).

Proposition 5.1: Consider $\{\mathsf{R}_0, \mathsf{R}_1\}$ and the observation vector z. A sufficient condition for $\mathrm{r}_k^{\mathrm{fs}}$ to be well-defined is condition (5). Also, any $\mathrm{r}_k^{\mathrm{fs}}$ satisfying (5) is unique. Furthermore, if the initial condition $\mathrm{z}_0^{\mathrm{feas}}$ is feasible, Algorithm 1 generates unique $\mathrm{r}_k^{\mathrm{fs}}$ for all k.

Proof: Consider the triangle $T(\mathrm{p}_k^1, \mathrm{q}^2, \mathrm{q}^i)$ for $i \in \{1,3\}$ with the edge lengths $\{d^2, d^j, l\}$ where $j \in \{1,3\}, j \neq i$. Then, the condition (5) should be satisfied to have a valid $|\lambda^i|$. Furthermore, since the set of d^is which satisfy (5) yield unique $\mathcal{A}(\mathrm{p}_k^1, \mathrm{q}^2, \mathrm{q}^3), \mathcal{A}(\mathrm{p}_k^1, \mathrm{q}^1, \mathrm{q}^2)$ based on (3), one has unique $|\lambda_k^1|, |\lambda_k^3|$. Since the right hand side of equation (4) is also unique for each $i \in \{1,3\}$, the resulting vector $\mathrm{r}_k^{\mathrm{fs}}$ is unique.

It is easy to see the last claim since Algorithm 1 yields either the vector $\mathrm{r}_k^{\mathrm{fs}}$ constructed with (2)-(4) or the last feasible observation vector $\mathrm{z}_{k-1}^{\mathrm{feas}}$ if the operation is not terminated due to large number of infeasible measurements. ■

Algorithm 1 Observation Vector Construction

Require: $\mathrm{z}_{k-1}^{\mathrm{feas}}, \mathrm{z}_k, \mathrm{ctr}, \mathrm{ctr_thres}$
Ensure: $\mathrm{r}_k^{\mathrm{fs}}, \mathrm{z}_k^{\mathrm{feas}}, \mathrm{ctr}$
if $l \neq 1$ **then**
  $d_k^i \leftarrow (1/l) d_k^i$
end if
if Condition (5) is satisfied **then**
  Calculate $\mathrm{r}_k^{\mathrm{fs}}$ from z_k by (2)-(4)
  if $l \neq 1$ **then**
    $\mathrm{r}_k^{\mathrm{fs}} \leftarrow l\, \mathrm{r}_k^{\mathrm{fs}}$
  end if
  $\mathrm{z}_k^{\mathrm{feas}} \leftarrow \mathrm{r}_k^{\mathrm{fs}}$
else
  $\mathrm{z}_k^{\mathrm{feas}}, \mathrm{r}_k^{\mathrm{fs}} \leftarrow \mathrm{z}_{k-1}^{\mathrm{feas}}$
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$
  if $\mathrm{ctr} = \mathrm{ctr_thres}$ **then**
    TERMINATE
  end if
end if

Finally, we smoothen the relative position measurement vector $\mathrm{r}_k^{\mathrm{fs}}$ with a first-order exponential smoothing algorithm as in (1) and denote the resulting signal by $\mathrm{r}_k^{\mathrm{ms}}$.

C. Proposed Mixture MCL Algorithm

References [1], [3] study the same estimation problem with the one stated above. They assume that the velocity $\mathrm{v}^{1\mathrm{G}}$ is a slightly varying state and use EKF to estimate the relative position r. Similarly to [1], [3], here we assume that robot R_0 does not have access to the instant velocity of robot R_1. However, unlike [1], [3], we assume that robot R_1 can be a slowly moving ground robot or an aerial vehicle with agile motion behavior. Our design aims at yielding good estimation performance for a broad spectrum of motion characteristics for robot R_1 including aggressive maneuvers. We exploit the non-parametric nature of particle filters to estimate r for different motion behaviors of robot R_1.

We argue that implementation of the standard MCL algorithm standalone may yield poor performance for our particular problem. In Fig. 4, we illustrate a reason why the estimation

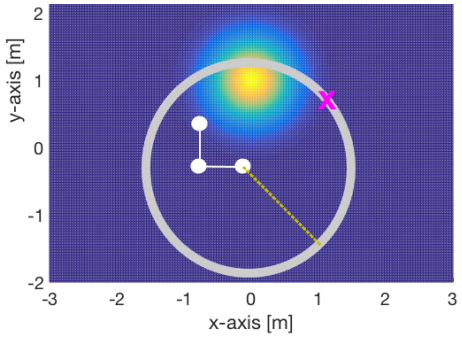


Fig. 4: The likelihood of the motion model (demonstrated as a Gaussian distribution originated at $\mathrm{p} = [0, 1]^\top$ m), the likelihood of the first sensor's observation model (gray ring), three onboard UWB anchors (white dots), and the true location of robot R_1 (magenta cross). If only the motion model is used for the proposal distribution, the particles would condense at the peak of the Gaussian distribution (yellow region) and likely miss robot R_1's true location.

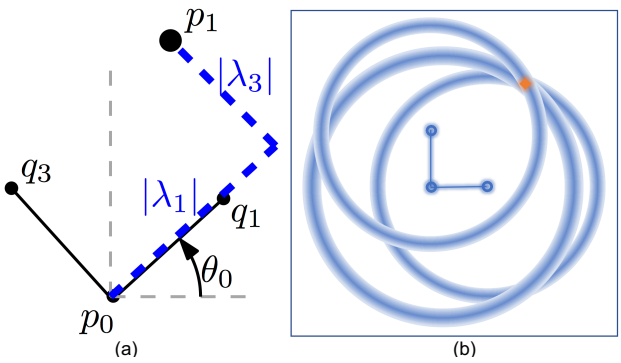


Fig. 5: (a) Construction of r^{fs}; (b) A representation of the uncertainty model of r^{fs}. The three rings depict the uncertainty regions of the distance measurements within a certain σ bound (the darker the color the more likely the target may be). The target is shown in orange.

performance may degrade if the state transition distribution is used as the proposal distribution. Consider the problem definition in Section III. Assume that the entries of $\mathrm{v}^{1,\max}$, the maximum velocity of robot R_1, are high. The standard MCL algorithm would propagate the particles of the previous time step with the motion model (depicted in yellow-orange). However, robot R_1 may reside far away from the center of the proposal distribution, e.g., at the magenta cross. In such a case, only a few particles, if not none, would survive in the resampling process in which the measurement model is evaluated. The repetition of this process would likely cause the particle depletion issue. A solution to the particle depletion requires to use a large number of particles, e.g., more than 1000, which can be computationally inefficient for onboard applications.

We propose to use a mixture MCL algorithm motivated by the necessity of incorporating measurements in the prediction phase. The mixture MCL algorithm combines the standard and dual MCL algorithms based on a probabilistic measure [34]. Particularly, the designer chooses the mixing parameter $\phi \in [0, 1]$ and implements at each step the standard MCL with probability ϕ and the dual MCL with probability $1 - \phi$. Accordingly, in the prediction phase, the particles are propagated effectively based on either the state transition model or the measurement model.

1) The Proposal Distribution: In the standard MCL algorithm, the particles are propagated based on the proposal distribution (2). We use the motion model (10) for this distribution.

In the dual MCL algorithm, the measurement model is incorporated into the proposal distribution as in (4). We now model the distribution $\pi(\mathrm{z}_k|\mathrm{x}_k)$ by using the constructed vector $\mathrm{r}_k^{\mathrm{ms}}$. If the objective was to localize the robot in a given map, then a common method would be to take a large number of sensor measurements, build the joint distribution $\pi(\mathrm{z}_k|\mathrm{x}_k)$, and form a grid map by kd-trees conditioned on some functions of features [9]. Since our problem statement does not include a map, we use a direct approach to obtain the distribution $\pi(\mathrm{z}_k|\mathrm{x}_k)$.

In Section V-B, we constructed the vector r^{ms} from the smoothed distance measurements d^1, d^2, d^3 by algebraic op-

erations. In Section VII-B, we present a procedure to characterize the uncertainty of the distance measurements d^i for calibration purposes, which can be well-modeled by Gaussian distribution. However, the uncertainty characteristics of r^{ms} greatly differs from that of the measurements d^i. Therefore, we use an approximation of the uncertainty characteristics of r^{ms}. Each distance measurement produces a circular likelihood region as hypothesis for the true relative position r_k. Notably, in the absence of noise, the true distance measurements $\bar{d}^1, \bar{d}^2, \bar{d}^3$ intersect at the true relative position r_k (Fig. 5b). Thus, the resulting configuration attains the highest probability at r_k and diminishing probability as it moves away from r_k. Assuming that the calibrated distance measurements are unbiased, we approximate this uncertainty model as a Gaussian distribution centered at r^{ms}. Accordingly, we generate N relative position hypothesis around r^{ms} as follows:

$$\mathrm{r}_k^i \sim \mathcal{N}\left(\mathrm{r}_k^{\mathrm{ms}}, \mathrm{Q}_{\mathrm{obs}}\right), \tag{6}$$

where $\mathrm{Q}_{\mathrm{obs}} = \mathrm{diag}\ (\sigma^2_{\mathrm{x,obs}}, \sigma^2_{\mathrm{y,obs}})$ denotes the measurement covariance matrix and is a design parameter, and $\sigma_{\mathrm{x,obs}}$ and $\sigma_{\mathrm{y,obs}}$ denote the standard deviations in the x and y axes in frame $\mathcal{F}_0$, respectively. Similarly, we generate N velocity estimates as follows:

$$\mathrm{v}_k^i = \mathrm{sat}(\tilde{\mathrm{v}}_k^i, \mathrm{v}^{1,\min}, \mathrm{v}^{1,\max}), \quad i \in \{1, \dots, N\}, \tag{7}$$

$$\tilde{\mathrm{v}}_k^i \sim \mathcal{N}\left(\mathrm{v}_k^{1,\mathrm{ms}}, \mathrm{Q}_{\mathrm{obs}}\right), \tag{8}$$

$$\mathrm{v}_k^{1,\mathrm{ms}} = (\mathrm{r}_k^{\mathrm{ms}} - \hat{\mathrm{r}}_{k-1})/T_s + \mathrm{v}_k^{0\mathrm{G}}, \tag{9}$$

where $\mathrm{sat}(\mathrm{x}, a, b)$ is a saturation function which limits every entry of the vector x within the bounds a, b, and $\mathrm{v}_k^{1,\mathrm{ms}}$ denotes a measure of $\mathrm{v}_k^{1\mathrm{G}}$ seen by robot R_0. In summary, we generate N particles around the constructed measurement vectors $\mathrm{r}_k^{\mathrm{ms}}, \mathrm{v}_k^{1,\mathrm{ms}}$ to represent the state hypothesis.

The design parameters $\sigma_{\mathrm{x,obs}}, \sigma_{\mathrm{y,obs}}$ can be found empirically with numerical simulations. Evidently, a set of distance measurements with high noise variances will result in high $\sigma_{\mathrm{x,obs}}, \sigma_{\mathrm{y,obs}}$. We suggest using the values that yield the best observed performance. Notably, as $\sigma_{\mathrm{x,obs}}, \sigma_{\mathrm{y,obs}}$ increase, the number of particles required for high performance estimation would increase. We emphasize that the approximations for these parameters are expected to perform well because the particle filter does not require a perfect measurement. In our experiments, we obtained sufficient performance with a small number of particles and with a set of parameter values found empirically.

An alternative approach is to propagate the particles based on the three distance measurements directly without constructing $\mathrm{r}_k^{\mathrm{ms}}$, i.e., by distributing the particles in the rings that represent the uncertainty regions of the distance measurements (Fig. 5b). However, one would need a redundantly huge number of particles to cover all the three rings. Therefore, this method would yield a computationally inefficient algorithm.

2) Resampling: To calculate the importance weights w^i in the resampling phase of the standard MCL algorithm, we use (3) with

$$\mathrm{z}_k = \begin{bmatrix} \mathrm{r}_k^{\mathrm{ms}} \\ \mathrm{v}_k^{1,\mathrm{ms}} \end{bmatrix}, \tag{10}$$

where $\mathrm{r}_k^{\mathrm{ms}}$ is generated by Algorithm 1 and $\mathrm{v}_k^{1,\mathrm{ms}}$ is as in (9).

We now model the distribution $\pi(\mathrm{x}_k|\mathrm{z}_{1:k-1}, \mathrm{u}_{1:k})$ to calculate the importance weights w^i of the dual MCL algorithm. The distribution $\pi(\mathrm{x}_k|\mathrm{z}_{1:k-1}, \mathrm{u}_{1:k})$ is analogous to the predicted belief distribution of Kalman filtering where the belief is calculated by processing the last action u_k before utilizing the last observation z_k. In [9], the authors propose to use the kernel density estimation method to construct the distribution $\pi(\mathrm{x}_k|\mathrm{z}_{1:k-1}, \mathrm{u}_{1:k})$. In that method, every particle in $\mathrm{bel}(\mathrm{x}_{k-1})$ is propagated based on the motion model $\pi(\mathrm{x}_k|\mathrm{u}_k, \mathrm{x}_{k-1})$. As a result, the new particles construct the kd-tree which represents the likelihood of the particles.

In our framework, the distribution $\pi(\mathrm{x}_k|\mathrm{u}_k, \mathrm{x}_{k-1})$ stands for the dynamics (10). We assumed that robot R_0 does not have access to the instant velocity of robot R_1 but has a rough knowledge about its state transition distribution. This uncertainty can be modeled with any distribution scheme including Gaussian distribution, multimodal Gaussian distribution, and beta distribution, based on the *a priori* knowledge on the motion behavior of robot R_1. Notably, the Gaussian and uniform distributions are good candidates to approximate the state transition of robots with agile maneuver capabilities. Here, we model the velocity of robot R_1 as a normal distribution centered at the previous estimate vector $\hat{\mathrm{v}}_{k-1}^1$. Therefore, we derive the weights as follows:

$$w_k^i \sim \pi(\mu_k^r|\mathrm{r}_k^i)\pi(\mu_k^v|\mathrm{v}_k^i), \tag{11}$$

where $\pi(\mu_k^r|\mathrm{r}_k^i)$ and $\pi(\mu_k^v|\mathrm{v}_k^i)$ are Gaussian distributions conditioned on μ_k^r and μ_k^v, respectively, with covariances $\mathrm{Q}_{\mathrm{mot}}$, and

$$\mu_k^r = \hat{\mathrm{r}}_{k-1} + (\hat{\mathrm{v}}_{k-1}^{1\mathrm{G}} - \mathrm{v}_{k-1}^{0\mathrm{G}})T_s, \tag{12}$$

$$\mu_k^v = \hat{\mathrm{v}}_{k-1}^{1\mathrm{G}}, \tag{13}$$

$$\mathrm{Q}_{\mathrm{mot}} = \mathrm{diag}\ \left(\sigma^2_{\mathrm{x,mot}}, \sigma^2_{\mathrm{y,mot}}\right). \tag{14}$$

Here, $\mathrm{Q}_{\mathrm{mot}}$ is a critical design parameter which can be tuned based on the application. For instance, small values for $\sigma_{\mathrm{x,mot}}, \sigma_{\mathrm{y,mot}}$ can be used for non-holonomic vehicles with slow angular velocities while relatively higher values can be used for holonomic vehicles with aggressive maneuvers. The velocity model (13) assumes no *a priori* information of motion behavior of robot R_1. If further information about instant velocity of robot R_1 through another measurement (e.g. a camera image) is available, it can be integrated to (13).

D. The Algorithm

We propose our localization algorithm in Algorithm 2. Algorithm 2 requires a two-robot system $\{\mathsf{R}_0, \mathsf{R}_1\}$ with the onboard UWB sensor configuration given in Section III. At any time step k, the algorithm receives the previous particle set S_{k-1}, the smoothed three distance measurements d_k^i, and the control input v_k^0 of robot R_0 and generates the new particle set S_k. The algorithm also requires the design parameters $\mathrm{Q}_{\mathrm{mot}}, \mathrm{Q}_{\mathrm{obs}}, \alpha$ inherently, but they are not explicitly presented here for clarity.

First, the relative position estimate $\mathrm{r}_k^{\mathrm{ms}}$ is constructed from d_k^i (lines 3-4), and the velocity estimate $\mathrm{v}_k^{1,\mathrm{ms}}$ is calculated

Algorithm 2 Proposed Mixture MCL Algorithm

Require: $\mathrm{x}_0, \mathrm{v}_k^0, d_k^1, d_k^2, d_k^3, \phi$
Ensure: $\chi_k, \hat{\mathrm{r}}_k$
1: Initialize $k=1,\ \bar{\chi}_0=\emptyset,\ \chi_0 \sim \pi(\mathrm{x}_0)$
2: **while** not TERMINATE **do**
3: $\quad \mathrm{r}_k^{\mathrm{fs}} \leftarrow$ ALGORITHM1(d_k^1, d_k^2, d_k^3)
4: $\quad \mathrm{r}_k^{\mathrm{ms}} \leftarrow$ SMOOTHEN$(\mathrm{r}_k^{\mathrm{fs}})$
5: $\quad$ Calculate $\mathrm{v}_k^{1,\mathrm{ms}}$
6: $\quad$ **if** $\phi \geq$ UNIFORM$(0,1)$ **then**
7: $\quad\quad$ **for** $i=1:N$ **do**
8: $\quad\quad\quad$ Generate $\mathrm{x}_k^i \sim \pi(\mathrm{x}_k|\mathrm{z}_{k-1}, \mathrm{u}_k)$
9: $\quad\quad$ **end for**
10: $\quad\quad$ **for** $i=1:N$ **do**
11: $\quad\quad\quad$ $w_k^i \leftarrow \eta\pi(\mathrm{z}_k|\mathrm{x}_k^i)$
12: $\quad\quad\quad$ $\bar{\chi}_k \leftarrow \bar{\chi}_k + \{\mathrm{x}_k^i, w_k^i\}$
13: $\quad\quad$ **end for**
14: $\quad$ **else**
15: $\quad\quad$ **for** $i=1:N$ **do**
16: $\quad\quad\quad$ Generate r_k^i by (6)
17: $\quad\quad\quad$ Generate v_k^i by (7)
18: $\quad\quad$ **end for**
19: $\quad\quad$ **for** $i=1\ldots N$ **do**
20: $\quad\quad\quad$ Calculate μ_k^r, μ_k^v by (12)-(13)
21: $\quad\quad\quad$ Calculate w_k^i by (11)
22: $\quad\quad\quad$ $\bar{\chi}_k \leftarrow \bar{\chi}_k + \{\mathrm{x}_k^i, w_k^i\}$
23: $\quad\quad$ **end for**
24: $\quad$ **end if**
25: $\quad \chi_k \leftarrow$ RESAMPLE$(\bar{\chi}_k)$
26: $\quad \bar{\chi}_k \leftarrow \emptyset$
27: $\quad k \leftarrow k+1$
28: **end while**

(line 5). Then, the algorithm chooses the filtering method at the current time step based on ϕ generated with uniform random number generator (line 6). Afterward, either the standard MCL algorithm (lines 7-13) or the dual MCL algorithm (lines 15-23) runs. Finally, the belief is updated by resampling the particle set based on the importance weights w^i (line 25). The designer can freely choose the resampling method.

VI. SIMULATIONS

We examined performance of the proposed mixture MCL algorithm through extensive simulations. We aimed to answer the following questions:

Q1. Does the proposed algorithm provide sufficient localization accuracy?
Q2. In what aspects does the mixture MCL algorithm differ from the standard MCL and EKF?

Although question Q1 could be answered subjectively, we focused on consistency and reliability of the algorithm's outcome. We observed that our algorithm does not yield centimeter-level precision in most scenarios; however, the obtained precision sufficed to implement formation control algorithms. We used the root mean square of the relative position error as the performance measure:

$$e_{\mathrm{est}} = \left(\frac{1}{K} \sum_{k=1}^{K} \|\mathrm{r}_k - \hat{\mathrm{r}}_k\|^2 \right)^{1/2}, \qquad (1)$$

where K is the termination step.

TABLE I: Parameter values in simulations

Parameter	Value
l	0.44m
f	8Hz
$\mathrm{Q}_{\mathrm{mot}}$	$100\,\mathrm{diag}(2,2,5,5)$
σ_{dist}	0.05m
$\mathrm{v}^{\mathrm{i,min}}$	$[-4,-4]$m/s
$\mathrm{v}^{\mathrm{i,max}}$	$[4,4]$m/s
$\mathrm{r}(0)$	$[-2,2]^\top$m
$\hat{\mathrm{r}}_{\mathrm{x}}(0)$	UNIFORM$(N,-10,10)$
$\hat{\mathrm{r}}_{\mathrm{y}}(0)$	UNIFORM$(N,-10,10)$
$\mathrm{v}^{0\mathrm{G}}$	$[0,0.2]^\top$m/s
$\mathrm{v}^{1\mathrm{G}}$	$[4s,0.3]^\top$m/s

We simulated our framework on a two-drone system with the robot operating system (ROS) Gazebo software. The framework consisted of an anchor robot R_0 and a tag robot R_1. We assumed that robots $\mathsf{R}_0, \mathsf{R}_1$ always move at constant altitudes h_0, h_1, respectively. We used the Pixhawk controller tools in Gazebo to maintain the velocities and headings at given set-points by controlling the internal dynamics of the quadrotors. We considered two cases to evaluate the localization performance: Externally actuated robots case and localization-based formation control case. We used the "low-variance resampling" method [5] in the filtering algorithm.

We followed a two-phase procedure in simulations. In the first phase, the quadrotors hovered and were stabilized at the desired altitudes $h_0 = h_1 = 2$m. The quadrotors were driven to desired locations. In the second phase, we sent the velocity commands to the quadrotors, which were either externally entered or produced by the localization feedback, and started the filtering algorithm concurrently.

A. Case 1: Externally Actuated Robots

In this part, we examine the localization performance on externally actuated quadrotors. Accordingly, we sent the velocity set-points to the quadrotors externally and analyzed the localization error for various parameter values. In other words, the estimated state $\hat{\mathrm{x}}$ was not fed to the robot controllers, which led to an open-loop system in each quadrotor (see Fig. 2).

We used the parameter values given in Table I where UNIFORM(N,a,b) denotes N random numbers generated with respect to uniform distribution within the boundary values $[a,b]$, and $s = \mathrm{sgn}(\sin(\pi k/20))$. Notably, robot R_1 shows agile motion behavior with this velocity profile. We perturbed the distance readings with additive white noise $\epsilon^i \sim \mathcal{N}(0, \sigma_{\mathrm{dist}}^2)$. We selected σ_{dist} inspired by the real-time characteristics of the UWB sensors used in the experiments (see Section VII).

In the first set of simulations, we analyzed the effect of the parameter ϕ for agile robot R_1 (Fig. 6a). Recall that ϕ tunes the mixing ratio of the standard and dual MCL algorithms: $\phi = 0$ corresponds to a standard MCL whereas $\phi = 1$ corresponds to a dual MCL. In these simulations, we used $N = 20$ particles which can be considered a small number compared to many conventional localization settings, yielding a computationally effective structure [6]. We performed 42 simulation runs for

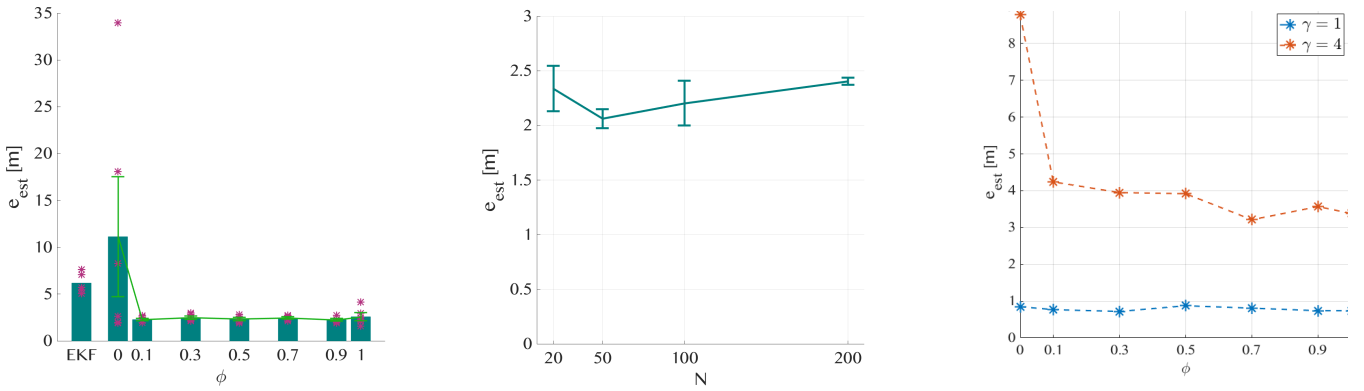


Fig. 6: Simulation results: (a) RMSE of an EKF implementation and of the mixture MCL algorithm for various ϕ values. The data points for the mixture MCL algorithm represent e_{est} for different $\alpha, \text{Q}_{\text{mot}}$ values. The data points for the EKF algorithm represent e_{est} for different motion and observation covariance matrices. The rectangular bars represent the average e_{est} over six simulations; (b) RMSE of the mixture MCL algorithm for $\phi = 0.5$ and $N = \{20, 50, 100, 200\}$; (c) RMSE versus ϕ in the square path experiments for two speed profiles of robot R_1.

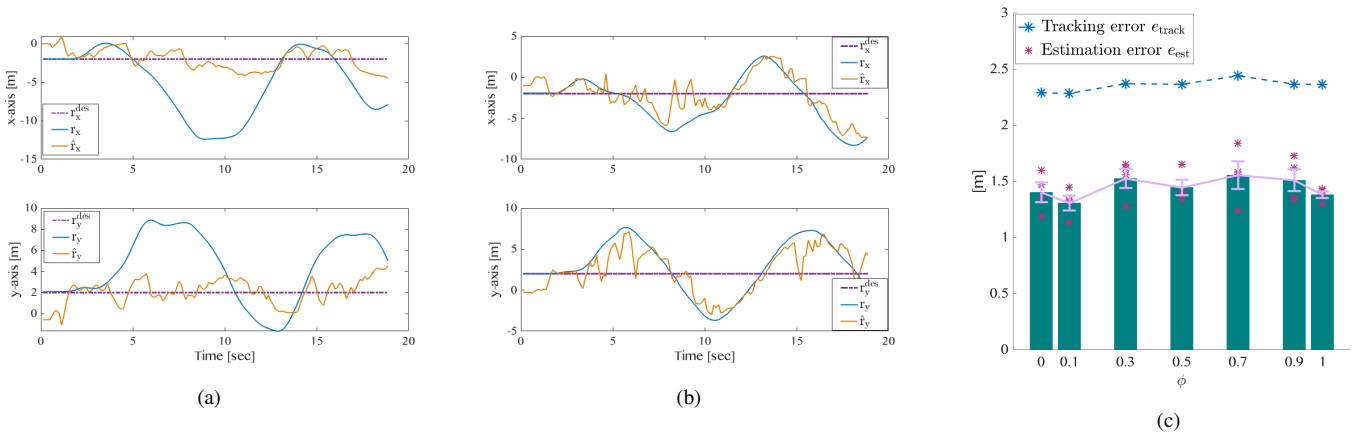


Fig. 7: Simulation results: Relative position estimates in a square path experiment for (a) $\phi = 0$ and (b) $\phi = 0.7$; (c) Tracking and estimation RMSE for seven ϕ values for a square wave velocity profile.

six sets of parameter values of $\alpha, \text{Q}_{\text{mot}}$ evaluated at seven ϕ values. For $\phi = 0$, the algorithm could not track robot R_1 in three runs and resulted in high errors. We argue that the particle degeneracy issue mainly caused the high level of error, i.e., a small number of particles were distributed on a large area in the plane, resulting in a false belief of robot R_1 motion. However, the algorithm estimated r with reasonable accuracy for $0 < \phi \leq 1$. Furthermore, we observed similar levels of errors for $\phi > 0$ irrespective of the values of the other parameters, $\alpha, \text{Q}_{\text{mot}}$. Therefore, we argue that incorporating the distance measurements in the prediction phase led to reasonable precision even with a small number of particles. As a result, the mixture MCL algorithm can be considered more reliable than the standard MCL algorithm. Moreover, we demonstrate the performance of the EKF algorithm of [3] for five sets of parameter values in Fig. 6a. We used $\hat{\text{r}}_0$ close to the true value r_0 and used high motion and observation covariance matrices. Nevertheless, the EKF yielded $e_{\text{est}} = 6\text{m}$ on average over five runs. Notably, the EKF performance highly depended on the initial conditions and agility level of robot R_1. Also, we observed that the standard MCL and EKF algorithms resulted in time-delayed estimates. These results demonstrate the necessity of a good initial condition and a sufficient number of particles for the standard MCL and EKF algorithms for a reliable estimation performance.

Furthermore, we set $\phi = 0.5$ and compared the estimation performance for $N = \{20, 50, 100, 200\}$ in terms of RMSE (Fig. 6b). We simulated the algorithm with $\alpha = \{0.1, 0.5, 1\}$ for each N. We did not observe a significant difference between $N = \{20, 50, 100\}$. However, the setting $N = 200$ resulted in high RMSE for higher ϕ values because the time required to run a simulation step exceeded the step time 0.125sec, and as a result, the algorithm ran slower than 8Hz. In our simulations, a dual MCL epoch was computationally heavier than a standard MCL epoch.

B. Case 2: Localization-based Formation Control

We tested the localization performance in a feedback control system on robot R_0. Robot R_1 was externally actuated with time-based velocity set-points while robot R_0 was to maintain the relative position r at a desired constant value r^{des} by utilizing the estimate $\hat{\text{r}}$. We used the parameter values given

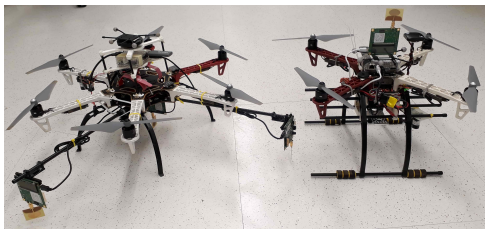


Fig. 8: The hexacopter and the quadrotor used in the experiments

in Table I. To compare the tracking performance in various scenarios, we use the following measure together with e_{est}:

$$e_{\text{track}} = \left(\frac{1}{K} \sum_{k=1}^{K} \|\mathrm{r}_k - \mathrm{r}^{\text{des}}\|^2 \right)^{1/2}, \tag{2}$$

where r^{des} is the constant desired relative position and K is the termination step. We used a simple proportional controller in robot R_0 for relative position keeping as follows:

$$\mathrm{v}_{0\mathrm{x}}^{\text{des}} = \begin{cases} K_v e_x & \text{if } e_x \geq \bar{e} \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

$$\mathrm{v}_{0\mathrm{y}}^{\text{des}} = \begin{cases} K_v e_y & \text{if } e_y \geq \bar{e} \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $e_x = \mathrm{r}_{\mathrm{x}}^{\text{des}} - \hat{\mathrm{r}}_{\mathrm{x}}$, $e_y = \mathrm{r}_{\mathrm{y}}^{\text{des}} - \hat{\mathrm{r}}_{\mathrm{y}}$, and $\bar{e}$ is a small threshold. We found the best N, K_v and $\bar{e}$ values empirically as $K_v = -2$, $\bar{e} = 0.1$m, and $N = 50$.

In the first set of simulations, we analyzed the effects of robot R_1's velocity profile on the estimation performance (Fig. 6c). We moved robot R_1 on a square path with $\mathrm{v}^{\mathrm{1G}} = \gamma[1+\text{sgn}(\cos(k\pi/40)), 1+\text{sgn}(\sin(k\pi/40))]$m/s, where $\gamma = 1$ and $\gamma = 4$ for the first and second experiment, respectively. For $\gamma = 1$, the proposed algorithm yielded similar e_{est} for all ϕ values, including the standard MCL algorithm, with low tracking errors. On the other hand, for $\gamma = 4$, the standard MCL algorithm yielded high e_{est}; as a result, robot R_0 lost track of robot R_1. We demonstrate the paths of the robots for $\phi = 0$ (Fig. 7a) and $\phi = 0.7$ (Fig. 7b). For $\phi = 0$, robot R_0 produced a smoother estimation but lost track of the agile maneuver of robot R_1, yielding $e_{\text{track}} = 6.99$m (Fig. 7a). For $\phi = 0.7$, robot R_0 estimated the relative position within a reasonable bound and showed an acceptable tracking performance (Fig. 7b) at the expense of a chattering estimate signal.

We note that (3) may not be the best controller for the particular dynamical system, and a better tracking performance can be obtained by implementing a more sophisticated control algorithm instead of (3). We aim to demonstrate the sufficiency of the proposed estimation algorithms in closed-loop systems. Therefore, we leave comparison of different controllers to future works.

Moreover, we compared the estimation and tracking performance of the proposed algorithm for a square-wave velocity profile in robot R_1 for $\phi = \{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1\}$ (Fig. 7c). We set $\mathrm{v}^{\mathrm{1G}} = [4s, 0.3]$m/s, which resulted in an almost sinusoidal path on the x-axis. We performed four simulations for each ϕ with parameters $\alpha = \{0.75, 1\}, \sigma_{\text{obs}} = \{1, 2\}$. We observed slight variations in the tracking and estimation performance for different ϕ values. Unlike the square path case, the standard MCL algorithm ($\phi = 0$) was able to track the sinusoidal path of robot R_1 with 50 particles. We argue that the high performance of the standard MCL algorithm stemmed from the smoothness of the path of robot R_1.

TABLE II: Drone Components

Component	Description
Airframe	DJI F550 & DJI F450
Flight controller	Pixhawk with PX4 autopilot firmware
Range finder	LiDAR Lite v3 for precision altitude measurement
Optical flow sensor	PX4Flow for hover stabilization and velocity feedback
Onboard computer	Odroid XU4 installed with Ubuntu 16 and ROS Kinetic for high-level computations
UWB sensors	Decawave TREK1000 for distance measurements

VII. Experiments

A. Experimental Setup

We performed experiments with two drones, a hexacopter equipped with three UWB anchors and a quadrotor equipped with a single UWB sensor (Fig. 8). The hexacopter was to estimate the relative position to the quadrotor. Each drone used a Pixhawk flight controller[1] running a PX4 open-source autopilot firmware to provide attitude stability and velocity tracking. The flight controller used the PX4Flow optical flow sensor [38] to provide accurate velocity feedback. We used an onboard Odroid XU4 computer[2] as a high-level controller to send the velocity set-points to the flight controller and to execute the localization algorithms. We used a laser range finder on each drone for precision altitude control in outdoor experiments. Specifications of the components are given in Table II. Videos of some experiments are available online[3].

Since we assumed that the headings of the drones remained constant during the entire operation, we set the attitude controllers to maintain the yaw angles of the drones at their initial configuration. Notably, this approach does not restrict the motion capabilities of drones because a holonomic air vehicle can reach the entire plane with constant heading.

The test procedure consisted of two stages. In the first stage, both drones were driven to certain locations and altitudes manually. Then, we switched to the autonomous mode and ran the localization algorithm simultaneously. In the autonomous mode, the quadrotor (robot R_1) moved based on pre-defined velocity set-points, and the hexacopter (robot R_0) moved based on either external inputs or the localization algorithm feedback. To avoid occlusions between the UWB anchors and the sensor, the drones flew at different altitudes. In all

[1] https://pixhawk.org
[2] https://www.hardkernel.com/main/products
[3] https://youtu.be/M9BXGaib0aU

experiments, we used $N = 50$ particles for a fair comparison between different parameter sets.

B. UWB Sensor Calibration

We followed a standard procedure explained in [3] to calibrate the DecaWave UWB sensors. The procedure requires to record a set of distance measurements and find the bias and noise variance for each anchor. In [3], the UWB sensors of the same type were calibrated for low data rate setting. Here, we calibrate the sensors for a higher (6.8Mbps) data rate. We calibrate each sensor separately because the sensors can show slightly different characteristics due to several reasons. We assume that the robots fly close to each other, e.g., $\mathrm{r} < 5$m, without any obstacles in between and focus on the LOS cases in calibrating the UWB sensors. References [1], [11], [12], [16] provide tools and algorithms to handle NLOS cases and multi-path effects which can improve the estimation for general scenarios. We leave the detailed analysis of such cases to future works.

Similar to [3], we assume that distance reading can be formulated as follows:

$$d^i = \bar{d}^i + \beta^i(\bar{d}^i) + \zeta^i(\bar{d}^i), \tag{1}$$

where β^i, ζ^i denote ith anchor's bias and zero-mean additive noise, respectively. Notably, both parameters can be dependent on the true distance value $\bar{d}^i$.

The dataset contained a certain number of distance measurements collected at several anchor-sensor distances for each anchor. Particularly, we took 700 measurements for each of ten distance values ranging between $\bar{d}^i = 0.5$m and $\bar{d}^i = 5$m. Let

$$\mathrm{D}_{\bar{d}^i} = \left\{d^i_1, \ldots, d^i_K\right\} \tag{2}$$

denote the sequence of measurements at the distance $\bar{d}^i$ for anchor i, and let

$$\mathrm{D}_{i,j} = \left\{d^i_1 - \bar{d}^i, \ldots, d^i_K - \bar{d}^i\right\} \tag{3}$$

denote the sequence of additive noise for $\bar{d}^i$, where j denotes the index corresponding to $\bar{d}^i$. For all datasets, we observed histograms for $\mathrm{D}_{i,j}$ similar to the one in Fig. 9a. This histogram can be well approximated by a Gaussian distribution with certain mean and variance for each $\bar{d}^i$. Accordingly, we set $\beta^i(\bar{d}^i) = \mathrm{mean}(\mathrm{D}_{i,j})$. The bias values of the three anchors at ten distance values and their first-order linear approximations are given in Fig. 9b. Particularly, we found

$$\begin{aligned}
\beta^{0,\mathrm{fit}} &= 0.04478\bar{d}^0 - 0.09589 \\
\beta^{1,\mathrm{fit}} &= 0.03227\bar{d}^1 + 0.04895 \\
\beta^{2,\mathrm{fit}} &= 0.03705\bar{d}^2 - 0.08832.
\end{aligned}$$

Next, we assumed that $\zeta^i(\bar{d}^i)$ are zero-mean Gaussian noises with variance equal to the variance of the Gaussian fits, i.e.,

$$\zeta^i(\bar{d}^i) \sim \mathcal{N}\left(0, \mathrm{var}(\mathrm{D}_{i,j})\right). \tag{4}$$

We calculated the variance of the anchors for each dataset as in Fig. 9c. Although first-order polynomials could yield better approximations, we fitted a constant value for each anchor's variance for ease of implementation.

The measurement noises ζ^i might show non-Gaussian behavior depending on the sensor and environment conditions. References [39]–[41] aim at modeling the noise behavior with maximum likelihood approach and Gaussian processes. We note that an advantage of our mixture MCL approach is the use of particles to represent the tag sensor's location. The distribution of the particles scattered around the constructed measurement z_k can be adjusted with the parameters $\mathrm{Q}_{\mathrm{mot}}$ and $\mathrm{Q}_{\mathrm{obs}}$ for the best observed performance. Therefore, zero-mean Gaussian model for ζ^i is expected to yield sufficient performance in our framework. We leave the more detailed analysis of the UWB antenna characteristics and angle-dependent offsets as a future work, which can be useful for enhancing the estimation performance.

Although the UWB sensors generally produced reliable data measurements at 10Hz, they yielded zero readings at times. To tackle the zero measurement issue, we used two methods. First, if a set of measurements contained zero reading, we used the last non-zero reading at the current time step. The second measure was the two-step smoothing procedure explained in Section V.

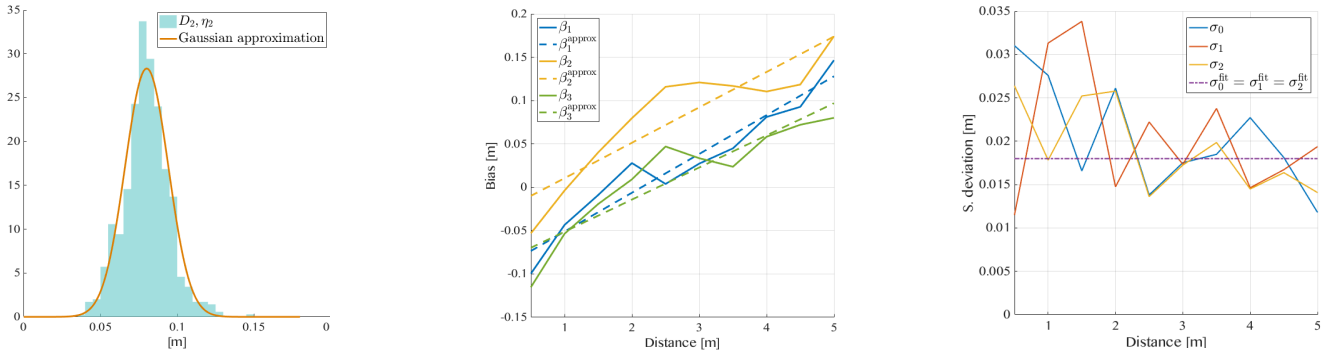
C. Indoor Experiments

We performed several tests in a cage with a motion capture system for externally actuated agile robots. Due to space constraint of the motion capture arena, we did not perform formation control experiments indoor. For high precision flights, the control algorithms maintained the headings and altitudes of the drones constant by the aid of the motion capture system.

The hexacopter hovered stationary, and the quadrotor was driven with high speeds in random directions manually. The maximum velocity of robot R_1 was $\mathrm{v}^{1,\max} = [3.65, 3.65]^\top$ m/s. In Fig. 10, we demonstrate the results of three tests for $\phi = \{0, 0.5, 1\}$. We used $\alpha = 0.8$ and $\mathrm{Q}_{\mathrm{mot}} = 20I$. We observed that the algorithm could not capture the agile motion of robot R_1 for $\phi = 0$ and yielded $e_{\mathrm{est}} = 3.56$m. Furthermore, the algorithm lost track of robot R_1 after $t = 40$ seconds. However, the algorithm yielded reasonable performance for $\phi = 0.5$ ($e_{\mathrm{est}} = 1.87$m) and $\phi = 1$ ($e_{\mathrm{est}} = 1.69$m). Notably, this result is in line with our simulation results where the setting $\phi = 0$ yielded unreliable performance for agile robot R_1 case. In Fig. 10b and 10c, there are time lags between the estimations and the ground truth values. We argue that these lags occur mainly because the speed of robot R_1 was too high to capture for the algorithm with $\sim$ 10Hz distance measurements.

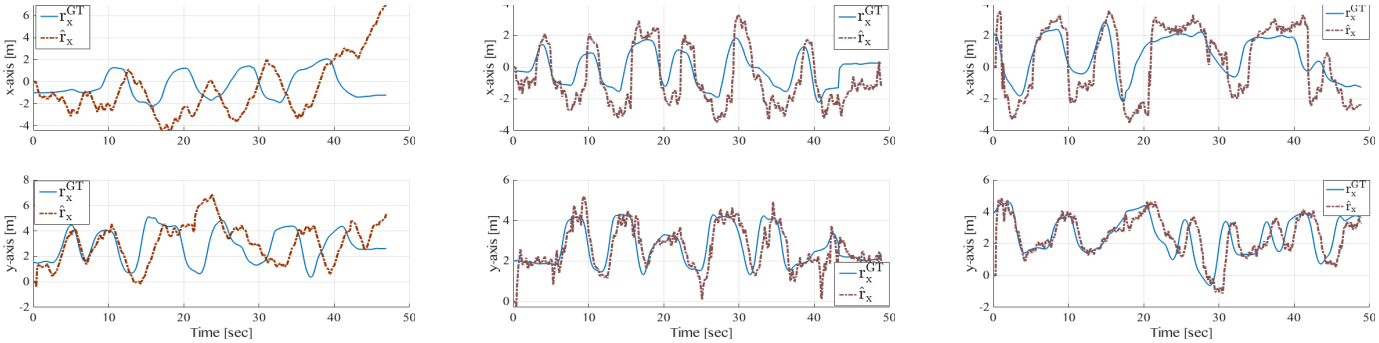
D. Outdoor Experiments

We conducted outdoor experiments for both the externally actuated robots case and the formation control case. We set the desired altitudes $h_0 = 2$m, $h_1 = 1.3$m and used a laser range sensor to stabilize the altitudes of the drones. We acquired the ground truth data from GPS sensors onboard. We used an optical flow sensor on each drone for hovering and planar motion control, i.e., moving the drone with the given velocity set-points. Since optical flow sensors can drift in the absence of a global positioning system, we also utilized the GPS data in sensor fusion wherever they are available. We followed the



(a) (b) (c)

Fig. 9: UWB calibration results: (a) A sample histogram of distance measurement noise at $\bar{d}^i = 5$m for anchor 2 and its Gaussian approximation; (b) Bias values of the anchors and their linear approximations; (c) Variance values of the anchors and their constant approximations.



(a) (b) (c)

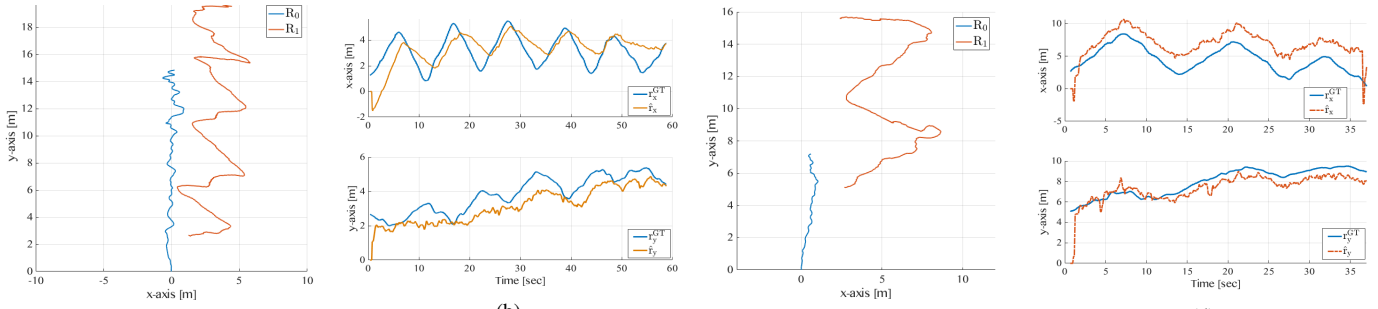
Fig. 10: Indoor experiments, externally actuated robots case results: (a) Standard MCL ($\phi = 0$); (b) mixture MCL ($\phi = 0.5$); (c) dual MCL ($\phi = 1$m).

same test procedure with the indoors case: We first stabilized the drones at the desired altitudes, then we ran the localization algorithm and sent the velocity set-points.

a) Externally actuated robots: The hexacopter moved on a straight path with velocity $\mathrm{v}^{0\mathrm{G}} = [0, 0.2]^\top$ m/s, and the quadrotor moved on an almost sinusoidal path with velocity $\mathrm{v}^{1\mathrm{G}} = [\mathrm{sgn}(\sin(\pi k/20)), 0.3]^\top$ m/s. Trajectories of the drones and the relative position estimates are presented in Fig. 11 for $\phi = 0$ and $\phi = 0.5$. The standard MCL algorithm showed reasonable performance for this smooth trajectory of robot R_1 and yielded $e_{\mathrm{est}} = 1.42$m (Fig. 11a and 11b). However, the x-axis estimation captured the true location (with an offset) after $t = 6$sec. On the other hand, the mixture MCL algorithm with $\phi = 0.5$ captured the true locations in both axes in the first two seconds and tracked the sinusoidal pattern of robot R_1 with an offset yielding $e_{\mathrm{est}} = 2.92$m.

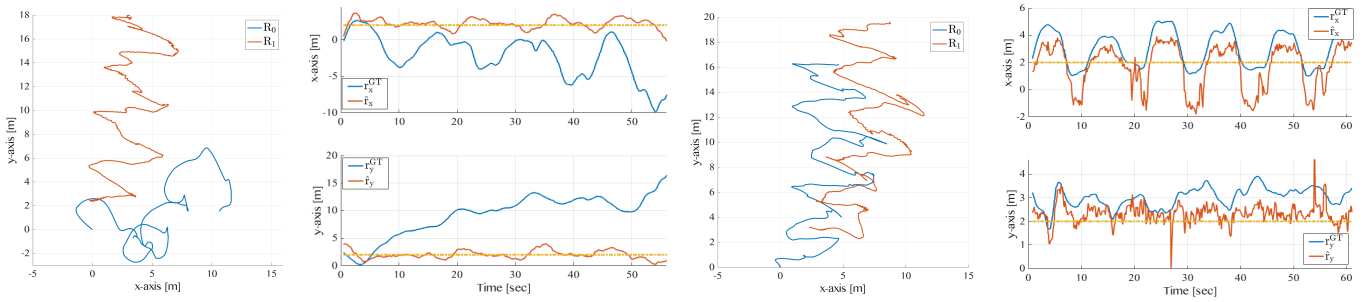
b) Formation control: In this set of experiments, we fed the relative position estimate $\hat{\mathrm{r}}$ back to the control algorithm of robot R_0 while robot R_1 moved with external inputs. In the first scenario, the quadrotor moved on an almost sinusoidal path with velocity $\mathrm{v}^{1\mathrm{G}} = [\mathrm{sgn}(\sin(\pi k/20)), 0.3]^\top$ m/s. The hexacopter was to maintain the relative position at the desired value $\mathrm{r}^{\mathrm{des}} = [2, 2]^\top$m by utilizing the proportional controller (3). The robot trajectories and relative position estimates are presented in Fig. 12. The standard MCL algorithm lost track of robot R_1 after $t = 5$sec and yielded high e_{est} and e_{track} (Table III); as a result, robot R_0 could not capture the sinusoidal pattern of robot R_1 (Fig. 12a and 12b). In contrast, the mixture MCL algorithm with $\phi = 0.5$ showed a reasonable estimation and tracking performance and yielded lower e_{est} and e_{track} compared to the standard MCL algorithm (Fig. 12c and 12d). Notably, the oscillations in Fig. 12d mainly stemmed from the controller choice, and the tracking performance can be improved by utilizing more advanced control algorithms (see Section VIII).

In the second scenario, the quadrotor moved in a random trajectory with high speeds. Particularly, the trajectory consisted of piecewise straight paths which were connected by corners where the quadrotor had sharp turns. A sample trajectory of robot R_1 from one of these tests is demonstrated in Fig. 13a. The maximum speed of robot R_1 in both directions was 4m/s. We used $N = 150$ particles for the standard MCL algorithm (Fig. 13b), $N = 50, \phi = 0.5$ for the mixture MCL algorithm (Fig. 13c), and $N = 50$ for the dual MCL algorithm (Fig. 13d). The standard MCL algorithm could not capture the agile motion of robot R_1 whereas the dual MCL algorithm showed high performance in both estimation and tracking. The mixture MCL algorithm showed a reasonable estimation performance.



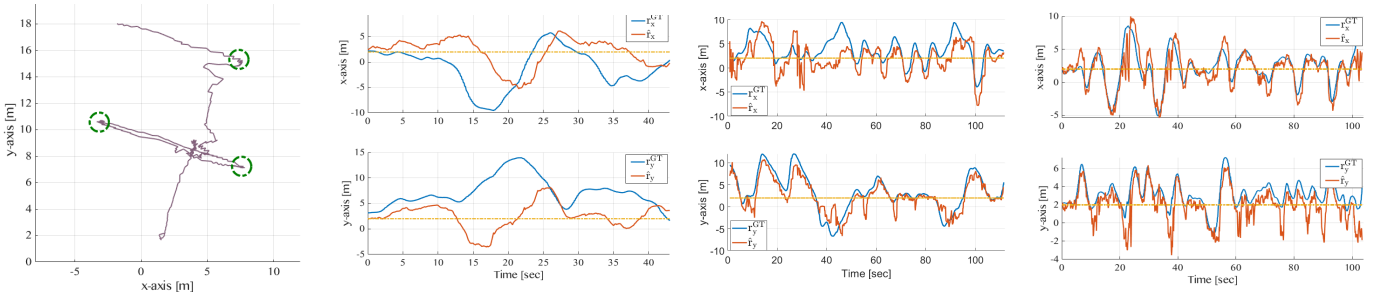
(a) (b) (c) (d)

Fig. 11: Outdoor experiments, externally actuated robots case results: (a)-(b) The paths of the robots and the relative position estimates with standard MCL algorithm ($\phi = 0$); (c)-(d) the paths of the robots and the relative position estimates with mixture MCL algorithm ($\phi = 0.5$).



(a) (b) (c) (d)

Fig. 12: Outdoor experiments, formation control, square-wave velocity case results: (a)-(b) The paths of the robots and the relative position estimates with standard MCL algorithm ($\phi = 0$); (c)-(d) the paths of the robots and the relative position estimates with mixture MCL algorithm ($\phi = 0.5$). The desired positions are presented with yellow dashed lines in (b) and (d).



(a) (b) (c) (d)

Fig. 13: Outdoor experiments, formation control, agile robots case results: (a) A sample trajectory of robot R_1 (the corners where the robot had sharp turns are represented with green circles); and the relative position estimates for (b) the standard MCL algorithm ($\phi = 0, N = 150$), (c) the mixture MCL algorithm ($\phi = 0.5$), and (d) the dual MCL algorithm ($\phi = 1$). The desired positions are presented with yellow dashed lines in (b)-(d).

VIII. Discussion on Results

We presented simulation as well as indoor and outdoor experiment results for two typical cases, namely, localization in externally actuated robots and localization-based formation control. Our algorithm yielded sufficient accuracy in most simulation runs. We used less than 200 particles in all simulations. Furthermore, we showed that our algorithm could yield sufficient accuracy even in tracking agile robots by utilizing only 50 particles. We now note some critical practical aspects.

Although we named our particular problem as localization, it dramatically differs from the conventional localization problem in which mobile robots are localized in a well-represented map. We aim at localizing a moving robot on another flying robot by utilizing the onboard computational capabilities solely. Furthermore, the localized robot is allowed to show aggressive behavior, we have not imposed a good initial estimate assumption, and no communication took place in the system. Admittedly, this setting inherently contains much more difficulties compared to the conventional localization problem. The main difficulty stems from the presumed assumption: The state-space for the estimation result is the entire plane instead of a constrained map. We have dealt with this difficulty by exploiting the non-parametric structure of the mixture MCL algorithm. We showed that incorporating exteroceptive measurements in the prediction phase enables the algorithm to capture agile characteristics of the localized robot with reasonable performance.

TABLE III: RMSE Errors in Field Experiments

Experiment	N	ϕ	e_{est} (m)	e_{track} (m)
Externally actuated robots				
R_0: straight, R_1: sinusoidal	50	0	1.42	–
R_0: straight, R_1: sinusoidal	50	0.5	2.91	–
Formation control				
R_1: sinusoidal	50	0	10.07	10.01
R_1: sinusoidal	50	0.5	1.67	1.98
R_1: random	150	0	7.84	7.65
R_1: random	50	0.5	3.42	5.19
R_1: random	50	1	2.24	3.19

Additionally, in the conventional localization setup, a robot in motion localizes itself by taking measurements from stationary landmarks with known positions. In contrast, our setup lacks a reference entity from which robot R_0 could take precise exteroceptive measurements. We emphasize that not only the distance measurement noises but also the motion inaccuracies of the robots have direct effects on the performance. Therefore, it is natural to observe high-level RMSE compared to the conventional localization setups.

Furthermore, our algorithm together with the particular anchor setting alleviates the effects of distance noises and outputs unique estimates even in cases the trilateration method fails.

Our algorithm does not require a communication framework except the internal communication mechanism. Our UWB sensors served as distance sensors only. Although our setup removes the need for another physical layer and risk of communication failures, it also removes a possible coordination structure between the two robots, which results in two separate robots that are unaware of each other's actions. A communication framework can be included in our framework easily for better coordination and formation control performance at the expanse of additional cost and possible communication delays.

Moreover, the loop rate, or the system frequency, has a significant impact on the performance. It is common to use high-frequency sensors in drone applications, such as IMUs with 1000Hz data rate. However, our UWB sensors generated data at around 10Hz. Since we aimed to imitate the real-life scenario in our simulations, we set the loop rate at 8Hz, which corresponds to a 1.25s interval between two successive time steps. We argue that while this data rate setting allows sufficient time for filtering, the drone's control mechanism could yield much better performance in higher frequencies.

We emphasize that we combined solutions for two separate problems in the formation control scenarios, localization and motion control, by feeding the estimated state vector to the motion control algorithm. Notably, since observation of the velocity of robot R_1 was not reliable due to noisy distance measurements, we did not utilize the velocity estimate in the feedback control algorithm. Therefore, we opted for a proportional controller for tracking robot R_1. The oscillations of robot R_0 in the x-axis when tracking robot R_1 stemmed from lack of a velocity term in the controller. We argue that a different formation control algorithm or an additional communication layer to transmit instant velocity information of robot R_1 to robot R_0 may yield better tracking performance. However, improvement of the motion control performance based on localization feedback is a separate task and beyond the scope of this work.

Unlike the standard particle filter and EKF, our algorithm does not require an initial guess for high-performance estimation if $\phi > 0$. Therefore, the robots may start or end the localization at any time in an operation. This feature provides great flexibility in some applications such as the kidnapped robot problem [5]. However, due to the structure of the dual MCL and the non-linear transformation from raw distances to r^{ms}, our algorithm is prone to high noise in distance measurements if $\phi \approxeq 1$. Therefore, the designer has a trade-off between the smoothness of the outcome and the ability to obtain high-performance without a good initial condition. The performance of our framework may vary based on the experimental testbed, vehicle type, or even the wind condition on the test day. For instance, lower ϕ values may yield higher performance with ground robots or with different types of drones. In this sense, the parameter ϕ provides flexibility to the designer. We suggest to use low $\phi > 0$ values for low $\mathrm{v}^{1,\max}$ cases and to use ϕ values close to 1 for agile robot R_1 cases. Similarly, the signal smoothing term α had a significant effect on the performance. Although low α values, e.g., $\alpha = 0.1$, mitigated the chattering issue, they caused lags in estimating the smoothed signal and adversely affected the tracking performance for agile robots. In summary, we have provided a versatile estimation algorithm with few adjustable parameters, and the designer is suggested to tune these parameters based on the particular application.

IX. Conclusion

Motivated by the need for a reliable and versatile multi-robot localization solution, we have designed an onboard UWB localization framework for a two-robot system. Our framework utilizes UWB distance measurements and motion models of the robots to generate an estimate of the inter-robot relative position in real-time. We have exploited the non-linear structure of the dual MCL algorithm to generate accurate estimates for a broad class of tag robot velocity profiles including agile maneuvers. Remarkably, our framework runs on board the anchor robot in real time without any need for a central computational unit such as a ground station. Also, our framework does not employ an explicit communication structure. Therefore, our framework provides a flexible multi-robot localization solution for both indoor and field operations. Extensive simulation and experimental studies have proved the reliability and repeatability of our framework. We have demonstrated the trade-offs between the standard and mixture MCL algorithms regarding estimation performance in the case of agile robots. To the best of our knowledge, this paper is the first to represent a real-time, onboard multi-robot localization framework tested on a two-drone setup both in indoor and field experiments.

In the future, we plan to extend our framework to three-dimensional scenarios by adding an extra UWB sensor to the anchor drone. Furthermore, we plan to study various advanced control techniques to improve the formation control

performance. Notably, we believe that model predictive or robust control algorithms can improve the tracking accuracy.

ACKNOWLEDGMENT

The authors thank Kuat Telegenov for his support in experiments.

REFERENCES

[1] B. Hepp, T. Nägeli, and O. Hilliges, "Omni-directional person tracking on a flying robot using occlusion-robust ultra-wideband signals," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 189–194.

[2] A. Wallar, B. Araki, R. Chang, J. Alonso-Mora, and D. Rus, "Foresight: Remote sensing for autonomous vehicles using a small unmanned aerial vehicle," in *Field and Service Robotics*, M. Hutter and R. Siegwart, Eds. Cham: Springer International Publishing, 2018, pp. 591–604.

[3] S. Güler, J. Jiang, A. A. Alghamdi, R. I. Masoud, and J. S. Shamma, "Real time onboard ultrawideband localization scheme for an autonomous two-robot system," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*, 2018, pp. 1151–1158.

[4] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. MIT press, 2011.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT press, 2005.

[6] H. M. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT press, 2005.

[7] G. Mao and B. Fidan, *Localization Algorithms and Strategies for Wireless Sensor Networks: Monitoring and Surveillance Techniques for Target Tracking*. IGI Global, 2009.

[8] S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 70–84, 2005.

[9] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1, pp. 99–141, 2001.

[10] C. Wang, H. Zhang, T. M. Nguyen, and L. Xie, "Ultra-wideband aided fast localization and mapping system," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1602–1609.

[11] A. Prorok, P. Tomé, and A. Martinoli, "Accommodation of nlos for ultra-wideband tdoa localization in single- and multi-robot systems," in *2011 International Conference on Indoor Positioning and Indoor Navigation*, 2011, pp. 1–9.

[12] J. González, J. L. Blanco, C. Galindo, A. Ortiz-de Galisteo, J. A. Fernández-Madrigal, F. A. Moreno, and J. L. Martínez, "Mobile robot localization based on ultra-wide-band ranging: A particle filter approach," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 496–507, 2009.

[13] S. S. Kia and S. Martinez, "A partially decentralized ekf scheme for cooperative localization over unreliable communication links," *arXiv:1608.00609*, 2016.

[14] S. S. Kia, S. Rounds, and S. Martinez, "Cooperative localization for mobile agents: A recursive decentralized algorithm based on kalman-filter decoupling," *IEEE Control Systems*, vol. 36, no. 2, pp. 86–101, 2016.

[15] I. Rekleitis, G. Dudek, and E. Milios, "Multi-robot collaboration for robust exploration," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 7–40, 2001.

[16] A. Prorok and A. Martinoli, "Accurate indoor localization with ultra-wideband using spatial models and collaboration," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 547–568, 2018/04/15 2013.

[17] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3866–3873.

[18] F. Lazzari, A. Buffi, P. Nepa, and S. Lazzari, "Numerical investigation of an uwb localization technique for unmanned aerial vehicles in outdoor scenarios," *IEEE Sensors Journal*, vol. 17, no. 9, pp. 2896–2903, 2017.

[19] S. Güler, M. Abdelkader, and J. S. Shamma, "Infrastructure-free multi-robot localization with ultrawideband sensors," in *2019 American Control Conference (ACC)*, 2019, pp. 13–18.

[20] I. Dotlic, A. Connell, H. Ma, J. Clancy, and M. McLaughlin, "Angle of arrival estimation using decawave dw1000 integrated circuits," in *2017 14th Workshop on Positioning, Navigation and Communications (WPNC)*, 2017, pp. 1–6.

[21] A. Ledergerber, M. Hamer, and R. D'Andrea, "Angle of arrival estimation based on channel impulse response measurements," in *2019 IEEE/RJS International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1–6.

[22] D. E. Manolakis, "Efficient solution and performance analysis of 3-d position estimation by trilateration," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 4, pp. 1239–1248, 1996.

[23] F. Thomas and L. Ros, "Revisiting trilateration for robot localization," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 93–101, 2005.

[24] E. Doukhnitch, M. Salamah, and E. Ozen, "An efficient approach for trilateration in 3d positioning," *Computer Communications*, vol. 31, no. 17, pp. 4124–4129, 2008.

[25] M. Cao, B. D. O. Anderson, and A. S. Morse, "Sensor network localization with imprecise distances," *Systems & Control Letters*, vol. 55, no. 11, pp. 887–893, 2006.

[26] S. J. Kim and B. K. Kim, "Dynamic ultrasonic hybrid localization system for indoor mobile robots," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 10, pp. 4562–4573, 2013.

[27] M. W. Mueller, M. Hamer, and R. D'Andrea, "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1730–1736.

[28] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[29] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[30] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[31] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "An adaptive color-based particle filter," *Image and Vision Computing*, vol. 21, no. 1, pp. 99–110, 2003.

[32] A. Del Bimbo and F. Dini, "Particle filter-based visual tracking with a first order dynamic model and uncertainty adaptation," *Computer Vision and Image Understanding*, vol. 115, no. 6, pp. 771–786, 2011.

[33] P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, 2003.

[34] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[35] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[36] J.-L. Blanco, J. González, and J.-A. Fernández-Madrigal, "Optimal filtering for non-parametric observation models: Applications to localization and slam," *The International Journal of Robotics Research*, vol. 29, no. 14, pp. 1726–1742, 2018/05/04 2010.

[37] Y. Diao, Z. Lin, and M. Fu, "A barycentric coordinate based distributed localization algorithm for sensor networks," *IEEE Transactions on Signal Processing*, vol. 62, no. 18, pp. 4760–4771, 2014.

[38] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1736–1741.

[39] A. Ledergerber and R. D'Andrea, "Ultra-wideband range measurement model with gaussian processes," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, 2017, pp. 1929–1934.

[40] A. Ledergerber and R. D'andrea, "Calibrating away inaccuracies in ultra wideband range measurements: A maximum likelihood approach," *IEEE Access*, vol. 6, pp. 78 719–78 730, 2018.

[41] J. Tiemann, J. Pillmann, and C. Wietfeld, "Ultra-wideband antenna-induced error prediction using deep learning on channel response data," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, 2017, pp. 1–5.