

Maximum likelihood localization of a network of moving agents from ranges, bearings and velocity measurements

Filipa Valdeira^{a,c,*}, Cláudia Soares^a, João Gomes^b

^a Computer Science Department, NOVA LINES, NOVA School of Science and Technology, Universidade NOVA de Lisboa, Portugal

^b Institute for Systems and Robotics (ISR/IST), LARSyS, Instituto Superior Técnico, Universidade de Lisboa, Portugal

^c Center for Mathematics and Applications (CMA), NOVA MATH, NOVA School of Science and Technology, Universidade NOVA de Lisboa, Portugal

ARTICLE INFO

Keywords:

Maximum likelihood estimation
Dynamic network localization
Hybrid measurements
Convex optimization
Distributed optimization

ABSTRACT

Localization is a fundamental enabler technology for many applications, like vehicular networks, IoT, and even medicine. While Global Navigation Satellite Systems solutions offer great performance, they are unavailable in scenarios like indoor or underwater environments, and, for large networks, the instrumentation cost is prohibitive. We develop a localization algorithm from ranges and bearings, suitable for generic mobile networks. Our algorithm is built on a tight convex relaxation of the Maximum Likelihood position estimator. To serve positioning to mobile agents, a horizon-based version is developed accounting for velocity measurements at each agent. To solve the convex problem, a distributed gradient-based method is provided. This constitutes an advantage over centralized approaches, which usually exhibit high latency for large networks and present a single point of failure. Additionally, the algorithm estimates all required parameters and effectively becomes parameter-free. Our solution to the dynamic network localization problem is theoretically well-founded and still easy to understand. We obtain a parameter-free, outlier-robust and trajectory-agnostic algorithm, with nearly constant positioning error regardless of the trajectories of agents and anchors, achieving better or comparable performance to state-of-the-art methods, as our simulations show. Furthermore, the method is distributed, convex and does not require any particular anchor configuration.

1. Introduction

While multi-agent systems often rely on Global Navigation Satellite Systems (GNSS) to accomplish their tasks, some environmental constraints prevent their use. For example, for Autonomous Underwater Vehicles (AUVs) [1,2], where the high conductivity in water decreases radio wave propagation, hence preventing the use of satellite positioning. This paper addresses the network localization problem for GNSS-denied environments from ranges, bearings, and velocity noisy measurements, with emphasis on AUV applications. Advancements in technology have made communication between these vehicles possible, so that agents may share information to obtain better localization accuracy. This is called a *cooperative* approach, which is also a trend in other GNSS-deprived settings, such as indoor environments [3,4]. Besides, as networks grow in size, *distributed* approaches (as opposed to centralized) are often desirable. We define distributed operation of a network of sensing and computing nodes as a mode where each agent performs the same computations and only communicates with one-hop neighbors on the communication graph. As the distributed system does not centralize on a single processor all available information, it avoids the

communication strain near the central node and on the bottleneck hubs (observed in centralized approaches), thus imposing a small latency. Importantly, the distributed paradigm does not present a single point of failure, traditionally found on federated star topologies. Therefore, we target the case of generic agent networks, able to cooperate with each other to accomplish self-localization tasks in a distributed manner, while performing their own trajectories, i.e., in a dynamical setting.

Problem formulation. A common approach for position estimation in dynamic settings is the use of Bayesian Filters, such as the Extended Kalman Filter (EKF), which is very often employed for localization tasks [1,2,5–7]. In spite of its popularity, EKF introduces linearization errors (as nonlinear systems must be linearized) and requires correct initialization to avoid convergence problems, which may not be available. In an attempt to overcome these challenges, we rely instead on the Maximum Likelihood Estimator (MLE) which is, for large sets of data and high signal to noise ratio, asymptotically unbiased and asymptotically efficient [8]. Therefore, the advantage over competing formulations is that if the error does follow the considered distribution, MLE will be the optimal estimator.

* Corresponding author.

E-mail addresses: f.valdeira@fct.unl.pt (F. Valdeira), claudia.soares@fct.unl.pt (C. Soares), jpg@isr.tecnico.ulisboa.pt (J. Gomes).

Nonconvexity and relaxations. Regardless of the formulation, the localization problem typically leads to a *nonconvex* cost function. While it is possible to preserve the nonconvex function [9–11], this requires a correct initialization of the method, which can be difficult to obtain. In order to avoid this limitation, we aim at a convex formulation, which under the MLE formulation calls for a relaxation of the original problem. Popular relaxation strategies for the nonconvex MLE include Semidefinite Programming (SDP) [12,13] and Second-Order Cone Programming (SOCP) [14]. Both approaches have strong requirements on the network topologies, in particular regarding anchor placement. For example, SOCP relaxations require the nodes to remain in the convex hull of the anchors, which can be a limitation when the area of actuation is unknown a priori. Other proposals include a Sum of Squares relaxation after problem reformulation [15], which is a computationally expensive alternative. Our proposed inclusion of angle measurements in the position estimation proves helpful in removing these constraints on network configuration and consequently allows more flexibility in trajectories, while remaining suitable for large networks.

Measurement availability. While the previous methods consider only pairwise range measurements between agents [12,16–18], pairwise bearings may also be available, leading to bearing-only [19] or hybrid methods [20–24]. Here, we consider the hybrid setting, which has been shown to be beneficial when compared to the single measurement case [25]. We consider that all nodes have access to range measurements with respect to their neighbors and a subset of them are able to additionally retrieve bearings from a different device. This differs, for instance, from [26] where each node has access to a single type of measurement, which can be of a different nature. We note that one important line of work takes into account Radio Signal Strength (RSS) and direction measurements [27,28], which calls for considerably different range models, than those assumed here.

ML hybrid localization. Similarly to range-only, we find ML [20,24] and non-ML [21–23] derived cost functions for the hybrid setting. In [20] the authors compare different distributions of angle measurements, where ranges are often available. The formulation is closely related to ours, although derived for a 2D case and for single target location. The nonconvexity is not relaxed and a Least Squares approach is used for initialization, after which the optimization is carried out with gradient descent. Consequently, this is subject to local minima and under higher noise levels some points are poorly estimated. In [24,29], the authors follow a ML formulation where directional data is modeled with the von Mises–Fisher distribution. Our proposed method uses the relaxation in [29] that is found to outperform the SDP relaxation [24] in positioning accuracy and is more amenable to larger networks [29].

Mobile networks. While methods devised for static networks can be applied for position estimation in mobile networks, it is beneficial to look at different time instants to retrieve additional information. An example is the bearing-only approach proposed in [19] where velocity is estimated alongside position. Other settings assume that relative velocity measurements between nodes are available [30,31] and extend existing range-only approaches. Instead, we assume that each node has access to its own velocity measurements (in addition to range and bearing measurements). Thus, we extend the hybrid ML formulation to account for individual velocity measurements, adding a time-window to account for different time instants. We show that this formulation is still amenable to the relaxation proposed in [29], thus overcoming the limitations in alternative relaxations to the hybrid ML problem. However, we emphasize that we assume no dynamical model for the movement, unlike the standard EKF approach.

Distributed setting. Given their centralized nature, most of the previous approaches are not suitable for networks with a large number of nodes, which are often found in current applications. Consequently, there has been a growing interest in distributed methods [6,16,17,19,32–34]. In particular, distributed versions of the SOCP relaxation [16] and of the

edge-based SDP relaxation [17] have been proposed for the range-only setting, but they still imply strong assumptions on network configuration. We show that our dynamic formulation for the hybrid case upon convex relaxation can be solved in a distributed manner. Thus, we reach a convex method (not dependent on initialization) which benefits from hybrid pairwise measurements (improving positioning accuracy) and takes into account dynamic information (without specifying an underlying dynamical model), while being scalable to large networks of agents (due to its distributed implementation).

Noise parameters. We note that, while the convex relaxation of the MLE does not need a precise initialization, it does require knowledge of noise parameters beforehand. This aspect is often overlooked and it is assumed that these values are known or previously obtained by a different method [21,23,24], which limits straightforward applicability to new scenarios. In contrast, our approach includes the estimation of noise distribution parameters, making it parameter-free and more versatile.

1.1. Contributions

We extend the previous work through three main contributions:

- We propose a horizon-based version of the hybrid MLE formulation, accounting for velocity measurements at each agent, which is better suited for mobile agents but still amenable to a tight convex relaxation (Section 3). The final algorithm allows for a more robust estimation of position compared to existing approaches (Section 6).
- We obtain a distributed algorithm that makes the algorithm scalable for large networks and does not lead to a single point of failure (Section 4).
- We derive a parameter-free version of the method with a minimal decrease in positioning accuracy (Section 5), that can be easily implemented without *a priori* knowledge.

2. Problem formulation and background

In light of previous considerations, we present a localization algorithm for a network of a variable and undefined number of moving autonomous agents. Anchors are assumed to be vehicles with access to an accurate positioning system, either by benefiting from GNSS signals or by remaining fixed at a known position. Besides, each vehicle is assumed to have a ranging device providing noisy distance measurements between itself and other vehicles within a certain range. Some vehicles are also equipped with a vector sensor producing bearing measurements. This angular information is assumed to be already in a common frame of reference, which could be obtained resorting to compass measurements, so that all angles are given with respect to the North direction, for instance. It is further considered that if vehicle “A” has measurements with respect to vehicle “B”, then the reverse is also true. All vehicles are potentially mobile and can follow any trajectory, including anchors.

The main goal is, then, to determine the position of each autonomous agent on a network, during a certain time interval. For this purpose, the network is represented as an undirected connected graph $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t))$, where each different graph corresponds to a given time instant. The vertices, also called nodes, $\mathcal{V} = \{1, \dots, n\}$ correspond to the vehicles, while the edges $i \sim j \in \mathcal{E}^x(t)$ indicate the existence of a measurement of type x between node i and j , at time t . In particular, $\mathcal{E}^d(t)$ refers to distance measurements and $\mathcal{E}^a(t)$ to angle measurements between two nodes. The known vehicle positions required for localization (anchors) are indicated by the set $\mathcal{A} = \{1, \dots, m\}$. Furthermore, the subset $\mathcal{A}_i^x(t) \subset \mathcal{A}(t)$ contains the anchors relative to which node i has an available measurement of type x at time t . Specifically, $\mathcal{A}_i^d$ refers to distance measurements and $\mathcal{A}_i^a$ to angle measurements to anchors.

We formulate our localization problem in $\mathbb{R}^p$ space. Practical relevant cases are $p = 2$ for a bidimensional problem and $p = 3$ for a

tridimensional one. The position of node i at time t is indicated as $x_i(t) \in \mathbb{R}^p$, while anchor positions are designated as $a_k(t) \in \mathbb{R}^p$. Each node collects measurements with respect to its neighboring nodes or nearby anchors within a certain distance. Noisy distance measurements at time t are represented as $d_{ij}(t) = d_{ji}(t)$ if they exist between two nodes (i and j) or as $r_{ik}(t)$ if they occur between a node i and anchor k . Noisy angle measurements between node i and j are represented as a unit-norm vector in the correspondent direction $u_{ij}(t)$. Similarly, noisy angle measurements between node i and anchor k are taken as the unit-norm vector $q_{ik}(t)$.

The problem is, thus, to estimate the unknown vehicle positions $x(t) = \{x_i(t) : i \in \mathcal{V}\}$ given a dataset of measurements

$$D(t) = \underbrace{\{d_{ij}(t) : i \sim j \in \mathcal{E}^d(t)\}}_{\text{node-node ranges}} \cup \underbrace{\{r_{ik}(t) : i \in \mathcal{V}, k \in \mathcal{A}_i^d(t)\}}_{\text{node-anchor ranges}} \\ \cup \underbrace{\{u_{ij}(t) : i \sim j \in \mathcal{E}^u(t)\}}_{\text{node-node angles}} \cup \underbrace{\{q_{ik}(t) : i \in \mathcal{V}, k \in \mathcal{A}_i^u(t)\}}_{\text{node-anchor angles}}.$$

A common assumption is to model distance noise with a Gaussian distribution of zero mean and variances σ_{ij}^2 and ζ_{ik}^2 , for node-node and node-anchor edges, respectively. Bearing noise is better represented by a von Mises–Fisher distribution, specifically developed for directional data, with mean direction zero and concentration parameter κ_{ij} and λ_{ik} , for node-node and node-anchor edges, respectively. Since we assume that nodes have two distinct devices for bearing and range measurements, we assume that the noise is uncorrelated between them. Therefore, with the additional assumption of independent and identically distributed noise, the maximum likelihood estimator is given by the solution of the optimization problem

$$\underset{x}{\text{minimize}} \quad f_{\text{dist}}(x(t)) + f_{\text{ang}}(x(t)) \quad (1)$$

where the functions are given as

$$f_{\text{dist}}(x(t)) = \sum_{i \sim j \in \mathcal{E}^d(t)} \frac{1}{2\sigma_{ij}^2} (\|x_i(t) - x_j(t)\| - d_{ij}(t))^2 + \\ \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{A}_i^d(t)} \frac{1}{2\zeta_{ik}^2} (\|x_i(t) - a_k(t)\| - r_{ik}(t))^2 \quad (2)$$

for range measurement terms, and for the angular terms

$$f_{\text{ang}}(x(t)) = - \sum_{i \sim j \in \mathcal{E}^u(t)} \left(\kappa_{ij} u_{ij}^T(t) \frac{x_i(t) - x_j(t)}{\|x_i(t) - x_j(t)\|} \right) - \\ \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{A}_i^u(t)} \left(\lambda_{ik} q_{ik}^T(t) \frac{x_i(t) - a_k(t)}{\|x_i(t) - a_k(t)\|} \right) \quad (3)$$

Solving (1) is difficult because the problem is nonconvex and nondifferentiable, especially for a large-scale network of agents. Problem (1) is nonconvex in both the distance and angle terms. Nonconvexity in distance terms arises from the negative argument of the square for position differences smaller than d_{ij} , i.e., $\|x_i - x_j\| < d_{ij}$, while the angle terms include a non-linear denominator $\|x_i - x_j\|$. For further insights on the relaxation employed we refer the reader to [29].

2.1. Convex relaxation

We follow the relaxation proposed in [29] to deal with the nonconvexity of Problem (1). We start by summarizing this approach, before extending it with velocity measurements in Section 3. It should be noted that, throughout this section, we drop the time dependency notation for clarity. First, a new variable y_{ij} is introduced to obtain the following equivalent formulation

$$(\|x_i - x_j\| - d_{ij})^2 = \inf_{\|y_{ij}\|=d_{ij}} \|x_i - x_j - y_{ij}\|^2. \quad (4)$$

The constraint is then relaxed as $\|y_{ij}\| \leq d_{ij}$, with the same reasoning applied to node-anchor edges.

However, the problem is still not convex due to the angular terms. Consequently, they are relaxed with their proxies from the distance term minimization. Therefore, the denominator $\|x_i - x_j\|$ is approximated by d_{ij} because this is exactly the purpose of the distance terms in the cost function. With similar reasoning, $x_i - x_j$ is approximated by y_{ij} . Summing up, the nodes' angular terms become $\frac{x_i - x_j}{\|x_i - x_j\|} \approx \frac{y_{ij}}{d_{ij}}$ and respectively for anchors $\frac{x_i - a_k}{\|x_i - a_k\|} \approx \frac{w_{ik}}{r_{ik}}$. The final expression for this convex approximation is then

$$\underset{x, y, w}{\text{minimize}} \quad f_{\text{dist}}(x, y, w) + f_{\text{ang}}(y, w) \quad (5)$$

subject to $\|y_{ij}\| \leq d_{ij}, \|w_{ik}\| \leq r_{ik}$

where

$$f_{\text{dist}}(x, y, w) = \sum_{i \sim j \in \mathcal{E}^d} \frac{1}{2\sigma_{ij}^2} \|x_i - x_j - y_{ij}\|^2 + \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{A}_i^d} \frac{1}{2\zeta_{ik}^2} \|x_i - a_k - w_{ik}\|^2$$

and

$$f_{\text{ang}}(y, w) = - \sum_{i \sim j \in \mathcal{E}^u} \left(\kappa_{ij} u_{ij}^T \frac{y_{ij}}{d_{ij}} \right) - \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{A}_i^u} \left(\lambda_{ik} q_{ik}^T \frac{w_{ik}}{r_{ik}} \right).$$

This is a tight relaxation with respect to the original problem that outperforms comparable SDP state-of-the-art relaxations, as shown in [29].

3. Dynamic formulation

While the previous formulation is adequate for static networks, when vehicles perform a trajectory, it is beneficial to take into account different consecutive positions to obtain a better estimate. Therefore, we introduce a time window and consider that each vehicle i has access to its own velocity $\beta_i(t)$ at time t .

In order to model velocity measurements, we decompose them into a unit vector $v_i(t)$ and magnitude $V_i(t)$. Under this format, we can look at $V_i(t)\Delta T$, the distance traveled by one node during a time interval ΔT , as the distance between two points $x_i(\tau)$ and $x_i(\tau - 1)$, where time has been discretized and is now indexed by an integer τ . Besides, the unit vector $v_i(t)$ can be thought of as the bearing between those same nodes, $x_i(\tau)$ and $x_i(\tau - 1)$, usually denoted as heading. Essentially, this means that we have distance and bearing measurements just as in the previous formulation, except that they exist between the same node at different time instants. So, we can model them in a similar way, with $V_i(t)\Delta T$ following a Gaussian distribution with variance σ_i^2 and $v_i(t)$ following a von Mises–Fisher distribution with concentration parameter κ_i .

Therefore, we obtain a formulation with similar structure to Problem (1), which can be relaxed in the same manner, by introducing an additional variable s , with the same role as y . We define the concatenation of the optimization variables over nodes or edges at instant τ as $x(\tau) = \{x_i(\tau)\}_{i \in \mathcal{V}}$, $y(\tau) = \{y_{ij}(\tau)\}_{i \sim j}$, $s(\tau) = \{s_i(\tau)\}_{i \in \mathcal{V}}$, $w_i(\tau) = \{w_{ik}(\tau)\}_{k \in \mathcal{A}_i}$, $w(\tau) = \{w_{ik}(\tau)\}_{i \in \mathcal{V}}$. Similarly, the concatenation over a time window of size T_0 at time t is given as $x = \{x(\tau)\}_{t-T_0 \leq \tau \leq t}$, $y = \{y(\tau)\}_{t-T_0 \leq \tau \leq t}$, $s = \{s(\tau)\}_{t-T_0 \leq \tau \leq t}$ and $w = \{w(\tau)\}_{t-T_0 \leq \tau \leq t}$. Note that the dependency on t (the time instant) is omitted from the notation in all variables (x, y, s, w) for compactness. The final problem is given as

$$\underset{x, y, w, s}{\text{minimize}} \quad f_{\text{dist}}(x, y, w) + f_{\text{ang}}(y, w) + f_{\text{vel}}(x, s) \quad (6)$$

subject to $\|y_{ij}(\tau)\| \leq d_{ij}(\tau), \|w_{ik}(\tau)\| \leq r_{ik}(\tau), \|s_i(\tau)\| \leq V_i(\tau)\Delta T,$

where

$$f_{\text{vel}}(x, s) = \sum_{i \in \mathcal{V}} \sum_{\tau=t-T_0+1}^t \frac{1}{2\sigma_i^2} (\|x_i(\tau) - x_i(\tau-1) - s_i(\tau)\|)^2 - \\ \sum_{i \in \mathcal{V}} \sum_{\tau=t-T_0+1}^t \kappa_i \left(v_i(\tau)^T \frac{s_i(\tau)}{V_i(\tau)\Delta T} \right), \quad (7)$$

$$f_{\text{dist}}(x, y, w) = \sum_{i \sim j \in \mathcal{E}^d} \sum_{\tau=t-T_0}^t \frac{1}{2\sigma_{ij}^2} (\|x_i(\tau) - x_j(\tau) - y_{ij}(\tau)\|)^2 + \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{A}_i^d} \sum_{\tau=t-T_0}^t \frac{1}{2\varsigma_{ik}^2} (\|x_i(\tau) - a_k(\tau) - w_{ik}(\tau)\|)^2 \quad (8)$$

and

$$f_{\text{ang}}(y, w) = - \sum_{i \sim j \in \mathcal{E}^u} \sum_{\tau=t-T_0}^t \left(x_{ij} u_{ij}(\tau)^T \frac{y_{ij}(\tau)}{d_{ij}(\tau)} \right) - \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{A}_i^u} \sum_{\tau=t-T_0}^t \left(\lambda_{ik} q_{ik}(\tau)^T \frac{w_{ik}(\tau)}{r_{ik}(\tau)} \right). \quad (9)$$

3.1. Reformulation

We shall now reformulate (6) in matrix notation. Let Σ_N be the diagonal matrix of $\frac{1}{\sigma_{ij}}$. Then, the first term of (8) can be written as

$$f_{\text{dist,nodes}}(x, y) = \frac{1}{2} \|\Sigma_N A x - \Sigma_N y\|^2, \quad (10)$$

where matrix A is the Kronecker product of the identity matrix of dimension T_0 with the result from the Kronecker product of C , the arc-node incidence matrix of the network, with identity matrix of dimension p , that is,

$$A = I_{T_0} \otimes (C \otimes I_p) = (I_{T_0} \otimes C) \otimes I_p.$$

Essentially, this extends matrix C along the dimension of the problem (p) and time window (T_0), assuming the set of edges remains constant over T_0 .

In the same way, taking Σ_V to be the diagonal matrix of $\frac{1}{\sigma_i}$, the first term of (7) may be written as

$$f_{\text{dist,vel}}(x, s) = \frac{1}{2} \|\Sigma_V N x - \Sigma_V s\|^2 \quad (11)$$

where $N = C_{\text{vel}} \otimes I_p$. However, it should be clearly noted that, in this formulation, the set of edges has changed, hence the different designation C_{vel} . Each node at time t has an edge with its position at $t-1$ and $t+1$, except for the first and last instants of the time window. The last distance term, concerning anchor-node measurements, may be reformulated as

$$f_{\text{dist,anchors}}(x, w) = \frac{1}{2} \|\Sigma_A E x - \Sigma_A \alpha - \Sigma_A w\|^2 \quad (12)$$

where vectors for anchor positions are concatenated in the usual way as $\alpha_i(\tau) = \{a_{ik}(\tau)\}_{k \in \mathcal{A}_i}$, $\alpha(\tau) = \{\alpha_i(\tau)\}_{i \in \mathcal{V}}$, $\alpha = \{\alpha(\tau)\}_{t-T_0 \leq \tau \leq t}$. E is a selector matrix, indicating which node has a measurement relative to each anchor and Σ_A the diagonal matrix of $\frac{1}{\varsigma_{ik}}$.

Finally, the angle measurements are grouped as $\tilde{u}_{ij}(\tau) = \frac{x_{ij} u_{ij}(\tau)}{d_{ij}(\tau)}$, $\tilde{q}_{ik}(\tau) = \frac{\lambda_{ik} q_{ik}(\tau)}{r_{ik}(\tau)}$ and $\tilde{v}_i(\tau) = \frac{x_i v_i(\tau)}{V_i(\tau) \Delta T}$. Following the same reasoning as before, $u(\tau) = \{\tilde{u}_{ij}(\tau)\}_{i \sim j}$, $v(\tau) = \{\tilde{v}_i(\tau)\}_{i \in \mathcal{V}}$ and $q_i(\tau) = \{\tilde{q}_{ik}(\tau)\}_{k \in \mathcal{A}_i}$, $q(\tau) = \{q_i(\tau)\}_{i \in \mathcal{V}}$. Then, concatenating along the time window, $u = \{u(\tau)\}_{t-T_0 \leq \tau \leq t}$, $v = \{v(\tau)\}_{t-T_0 \leq \tau \leq t}$ and $q = \{q(\tau)\}_{t-T_0 \leq \tau \leq t}$. It follows that

$$f_{\text{ang}}(y, w, s) = -u^T y - q^T w - v^T s. \quad (13)$$

Considering the definitions in (10), (11), (12) and (13), Problem (6) is reformulated as

$$\underset{x, y, w, s}{\text{minimize}} \quad f_{\text{dist,nodes}}(x, y) + f_{\text{dist,anchors}}(x, w) + f_{\text{dist,vel}}(x, s) + f_{\text{ang}}(y, w, s) \quad (14)$$

subject to $\|y_{ij}(\tau)\| \leq d_{ij}(\tau)$, $\|w_{ik}(\tau)\| \leq r_{ik}(\tau)$, $\|s_i(\tau)\| \leq V_i(\tau) \Delta T$.

Introducing a variable $z = (x, y, w, s)$ and defining $\mathcal{Z} = \{z : \|y_{ij}(t)\| \leq d_{ij}(t), i \sim j \in \mathcal{V}; \|w_{ik}(t)\| \leq r_{ik}(t), i \in \mathcal{V}, k \in \mathcal{A}_i; \|s_i(t)\| \leq V_i(t) \Delta T, i \in \mathcal{V}\}$, Problem (14) can be written in a quadratic form as

$$\underset{z}{\text{minimize}} \quad \frac{1}{2} z^T M z - b^T z \quad (15)$$

subject to $z \in \mathcal{Z}$

where $M = M_1 + M_2 + M_3$, with

$$M_1 = \begin{bmatrix} A^T \Sigma_N \\ -\Sigma_N \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \Sigma_N A & -\Sigma_N & 0 & 0 \end{bmatrix},$$

$$M_2 = \begin{bmatrix} E^T \Sigma_A \\ 0 \\ -\Sigma_A \\ 0 \end{bmatrix} \begin{bmatrix} \Sigma_A E & 0 & -\Sigma_A & 0 \end{bmatrix},$$

$$M_3 = \begin{bmatrix} N^T \Sigma_V \\ 0 \\ 0 \\ -\Sigma_V \end{bmatrix} \begin{bmatrix} \Sigma_V N & 0 & 0 & -\Sigma_V \end{bmatrix},$$

and $b = b_1 + b_2$, with

$$b_1 = \begin{bmatrix} E^T \Sigma_A \\ 0 \\ -\Sigma_A I \\ 0 \end{bmatrix} \Sigma_A \alpha, \quad b_2 = \begin{bmatrix} 0 \\ u \\ q \\ v \end{bmatrix}.$$

4. Distributed implementation

Minimization of the convex problem in (15) can be carried out in multiple ways. However, we strive for a method suitable for distributed settings. Hence, we choose the *Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)* [35], which is an extension of the gradient method and will produce a naturally distributed solution, as detailed below. FISTA handles problems defined as

$$\underset{x}{\text{minimize}} \quad f(x) = g(x) + h(x) \quad (16)$$

where $g(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ is a smooth convex function, continuously differentiable with Lipschitz continuous gradient with constant L , and $h(x)$ is a closed and convex function with an inexpensive proximal operator with parameter t_k , expressed as $\text{prox}_{t_k h}$. Under these assumptions, FISTA's solution for Problem (16) is given iteratively as

$$y = x^{(\kappa-1)} + \frac{\kappa-2}{\kappa+1} (x^{(\kappa-1)} - x^{(\kappa-2)})$$

$$x^{(\kappa)} = \text{prox}_{t_k h}(y - t_k \nabla g(y)). \quad (17)$$

This method has a convergence rate $f(x^{(\kappa)}) - f^*$ (where f^* is the optimal value) of $\mathcal{O}(1/\kappa^2)$, for a fixed step size of $t_k = 1/L$.

4.1. Instantiation

For Problem (15) to be solved with FISTA, it is necessary to define $g(x)$ and $h(x)$, along with $\text{prox}_{t_k h}$ and t_k . In this case, $g(x)$ is the quadratic cost function $\frac{1}{2} z^T M z - b^T z$. It is noted that a function is said to have a Lipschitz Continuous Gradient with Lipschitz constant L if the following condition is true

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\| \quad (18)$$

and quadratic functions have a Lipschitz continuous gradient. Besides, the required gradient $\nabla g(z)$ is straightforward, given as $\nabla g(z) = M z - b$.

Term $h(x)$ corresponds to the indicator function¹ $I_{\mathcal{Z}}(z)$ for set $\mathcal{Z}$. From [36], when $h(x)$ is the indicator function of a closed convex set $\mathcal{Z}$, $\text{prox}_{t_k h}$ is the projection of z on set $\mathcal{Z}$, $P_{\mathcal{Z}}(x)$, defined as

$$P_{\mathcal{Z}}(z) = \underset{x \in \mathcal{Z}}{\text{argmin}} \|x - z\|^2,$$

¹ We define the indicator function as $I_{\mathcal{Z}}(z) = \begin{cases} 0, & \text{if } z \in \mathcal{Z} \\ +\infty, & \text{otherwise} \end{cases}$

which for our constraints with general structure $\{z : \|z\| \leq d\}$ is implemented as

$$P_Z(z) = \begin{cases} z, & \text{if } \|z\| \leq d \\ \frac{z}{\|z\|}d, & \text{if } \|z\| > d. \end{cases}$$

Finally, it is necessary to obtain a value for $1/L$, where L is the Lipschitz constant of function $\frac{1}{2}z^T M z - b^T z$. Resorting to [37], we obtain an upper-bound for L as (see Appendix A for the complete derivation)

$$L \leq \frac{1}{\sigma_N^2} 2\delta_{\max} + \frac{1}{\sigma_A^2} \max_{i \in \mathcal{V}} |\mathcal{A}_i| + \frac{1}{\sigma_V^2} 2\delta_{\max}^V(T) + K, \quad (18)$$

where $\delta_{\max}$ is the maximum node degree of the network and $\delta_{\max}^V$ is either 2 for any $T > 2$, 1 for $T = 2$ and 0 for no time window ($T = 1$). The term $\max_{i \in \mathcal{V}} |\mathcal{A}_i|$ corresponds to the maximum number of anchors connected to a node. Finally, $K = \frac{1}{\sigma_N^2} + \frac{1}{\sigma_A^2} + \frac{1}{\sigma_V^2}$, where σ_N , σ_A and σ_V are the maximum values of matrices Σ_N , Σ_A and Σ_V , respectively. Distributed computation of L can be attained, as suggested in [37], by a diffusion algorithm.

Thus, we obtain a distributed method. The explicit algorithm and the full derivation can be found in Appendix B. Recalling that we stay under a convex formulation, it is noted that the method may be initialized with any feasible value without jeopardizing convergence. We further emphasize that each node on the network computes its own estimate, relying only on distances and angles taken with respect to its neighbors (nodes or anchors) and on its own velocity measurements. Notice that quantities related to other nodes are always within the set $\mathcal{A}_i$ (neighbor anchors of node i) or $\mathcal{E}_i$ (neighbor nodes of node i). At the end of each iteration (in κ), nodes also need to share their current estimates, but since this is only needed amongst neighbors, the method remains distributed.

5. Parameter estimation

Until this point, the parameters σ and κ of error distributions have been assumed to be known a priori. In real applications, these parameters could be estimated from previous experiments. However, this can also be included in the algorithm so that no previous knowledge is necessary. With each new measurement, new estimates for κ and σ are computed, which desirably converge to the true values.

Maximum likelihood estimators for the parameters of each distribution are easily found in the literature. Given a random variable X , normally distributed with unknown mean and variance σ^2 , the MLE for the latter parameter is [38]

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2. \quad (19)$$

For estimation of the concentration parameter κ we have [39]

$$\hat{\kappa} = \frac{\bar{\gamma}(p - \bar{\gamma}^2)}{1 - \bar{\gamma}^2}, \quad (20)$$

where $\bar{\gamma} = \frac{\|\sum_{i=1}^n x_i\|}{n}$, x_i are independent and identically distributed sample unit vectors drawn from a Von Mises–Fisher distribution $vMF(\mu, \kappa)$, and p is the dimension of x_i .

5.1. Instantiation

Evidently, there is no access to the true values of our measurements to accurately estimate κ and σ . However, after each computation of position estimates by the algorithm, it is possible to compute estimated distances, angles and velocities which can be used as “true values”. So, exact values of κ and σ are not to be expected, but if the position estimates are close to the true ones, then reasonable approximations should be obtained. However, note that we rely on our position estimates to estimate the distribution parameters and subsequently rely on the estimated parameters to compute the estimated positions. So, a

poor estimated value on either counterpart could affect both estimation procedures (see Section 6).

Denoting by $\hat{x}$ the value obtained from our position estimation, it is possible to compute the resulting distances and bearings as

$$\hat{d}_{ij}(t) = \|\hat{x}_i(t) - \hat{x}_j(t)\|, \quad \hat{u}_{ij}(t) = \frac{\hat{x}_i(t) - \hat{x}_j(t)}{\|\hat{x}_i(t) - \hat{x}_j(t)\|}.$$

Given the obtained measurements d_{ij} and u_{ij} , (19) and (20) can now be used for variance and concentration parameter estimation, respectively, as

$$\hat{\sigma}_{ij}^2(t) = \frac{1}{t-1} \sum_{\tau=1}^t (d_{ij}(\tau) - \hat{d}_{ij}(\tau))^2, \quad \hat{\kappa}_{ij}(t) = \frac{\bar{\gamma}_{ij}(t)(p - \bar{\gamma}_{ij}^2(t))}{1 - \bar{\gamma}_{ij}^2(t)},$$

where

$$\bar{\gamma}_{ij}(t) = \frac{1}{t} \left\| \sum_{\tau=1}^t \frac{u_{ij}(\tau) - \hat{u}_{ij}(\tau)}{\|u_{ij}(\tau) - \hat{u}_{ij}(\tau)\|} \right\|. \quad (21)$$

The same reasoning can be applied to node-anchor edges, with similar resulting equations.

Unlike $\hat{d}_{ij}$ and $\hat{u}_{ij}$, estimation of velocity measurements $\hat{\beta}_i$ requires a more careful approach. While the former requires only current estimates of the position, the latter requires positions at different time steps. Since velocity measurements can change with a high rate over time, a naive approach like averaging over a sliding window does not produce good estimates. Thus, an approach in that vein would only yield accurate estimates for linear constant trajectories. This problem is considered in [40] and the proposed solution is based on central finite differences with a time lag and rejection of high-frequency noise. The proposed estimation is then given as

$$\hat{\beta}_i(t) \Delta T = \frac{5(x_i(t-3) - x_i(t-5))}{32} + \frac{4(x_i(t-2) - x_i(t-6))}{32} + \frac{(x_i(t-1) - x_i(t-7))}{32}. \quad (22)$$

Recalling that $\hat{\beta}_i(t)$ is decomposed as $\hat{\beta}_i(t) = \hat{v}_i(t)\hat{V}_i(t)$, estimates for variance and concentration are given as

$$\hat{\sigma}_i^2(t) = \frac{1}{t-1} \sum_{\tau=1}^t (V_i(\tau) - \hat{V}_i(\tau))^2, \quad \hat{\kappa}_i(t) = \frac{\bar{\gamma}_i(t)(p - \bar{\gamma}_i^2(t))}{1 - \bar{\gamma}_i^2(t)},$$

with $\bar{\gamma}_i(t) = \frac{\|\sum_{\tau=1}^t v_i(\tau) - \hat{v}_i(\tau)\|}{t}$. However, it should be noted that the introduction of a time lag has practical consequences during implementation that should be taken into account. In particular, this means that velocity estimation is only possible after the 7-th time step and, consequently, $\hat{\sigma}_i^2$ and $\hat{\kappa}_i$ are only available after that moment. Furthermore, some time steps are necessary before achieving sufficiently accurate estimates for any of the parameters. Thus, it is still necessary to define initial values ($\hat{\sigma}^0$ and $\hat{\kappa}^0$) for the first iterations. However, we will show in Section 6 that wrong initial estimates do not increase the positioning error when performing parameter estimation.

The addition of parameter estimation to Algorithm 1 produces a final distributed and parameter-free algorithm, that can be found in Appendix C.

6. Simulation results

We test the performance of our proposed method in different settings for the application of AUV missions, comparing against benchmark approaches.

6.1. Simulations setting

Trajectories. We evaluate the performance of the proposed algorithm over three common trajectories in AUV missions: a 2D lawn-mower (Fig. 1(a)), a 3D helix (Fig. 1(b)) and a 2D lap trajectory (Fig. 3). All nodes and anchors describe the depicted trajectories at a speed of 1 m/s (commonly used in AUV missions [41,42]). The three trajectories encompass different range and bearing measurements

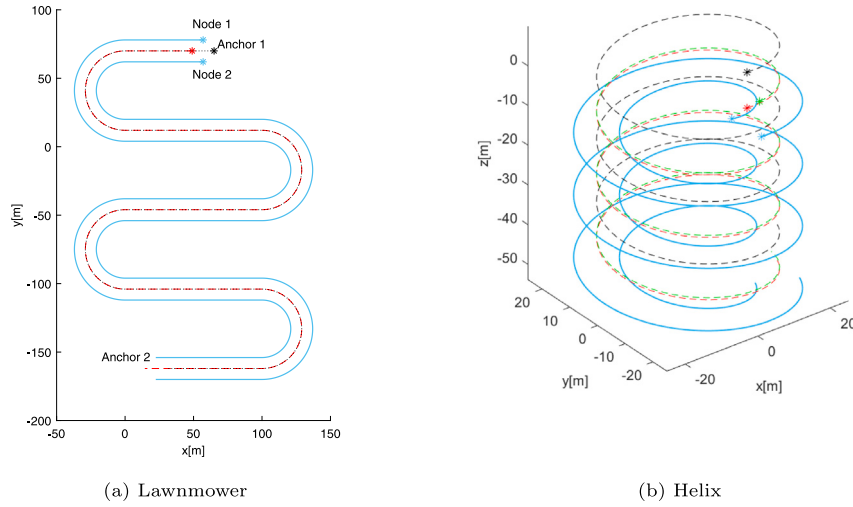


Fig. 1. Lawnmower and helix trajectories. Nodes are depicted in solid blue lines and anchors in dashed lines of different colors. The start point of each trajectory is marked with a star. The lawnmower depicts two nodes, traveling on both sides of two anchors describing the same trajectory with a small lag. The helix includes two nodes, traveling on both sides of three anchors. All paths maintain the same velocity in the x and y axis, progressively decreasing the z component.

amongst themselves. Anchors are assumed to have access to their true positions, either resorting to more accurate equipment or by navigating at the surface and benefiting from GNSS.

Noisy measurements. Considering noise levels obtainable with current instrumentation [43,44], we define $\sigma_d = 0.5$ m (for all distance measurements), $\sigma_v = 0.1$ m/s (for all speed measurements), $\kappa_d = 1000$ (for all bearing measurements) and $\kappa_v = 1000$ (for all heading measurements)². Unless otherwise noted, we always consider 100 Monte Carlo trials for each simulation.

Benchmarks. We want to determine two main points: whether our proposed inclusion of velocity measurements and a time window is relevant and improves the performance with respect to similar static methods; whether our resulting method is effectively competitive with current state-of-the-art solutions. To attest the former we compare with [29], an hybrid convex method for ranges and bearings that does not account for velocity measurements. For the latter, we select the EKF with an equivalent data model, as it remains the state-of-the-art standard for practical AUV applications due to the trade-off between accuracy and computational power [45,46].

Initialization and parameters. Parameters for the EKF are tuned by grid search for each of the considered trajectories and noise levels. EKF also requires a proper initialization to assure convergence. Considering this, the filter is initialized with the true position, contaminated with Gaussian noise of zero mean and standard deviation of 2 m [44]. Our method, being convex, is independent of any initialization and this choice only affects the computation time, not the achieved accuracy (see Section 6.4). The only parameters in our method are the noise distribution parameters, e.g., σ or κ . For comparison with the fine-tuned EKF, we consider the optimal case where we have access to the true parameters. When testing our parameter estimation process we consider a different setup, explained in Section 6.6.

Metrics. In order to evaluate the localization accuracy, we use the Mean Navigation Error (MNE) defined as

$$\text{MNE}(\hat{x}, t) = \frac{1}{MN} \sum_{m=1}^M \sum_{i=1}^N \|\hat{x}_i^m(t) - x_i^m(t)\|_2,$$

where $\hat{x}$ is an estimate of x , M the number of Monte Carlo trials, N the number of unknown nodes and T the time interval. Furthermore, when applicable we consider the Mean Positioning Error (MPE) as the MNE averaged over the full trajectory.

6.2. Comparison with benchmarks

We compare the MNE for each algorithm along the three different trajectories. The advantage introduced by the dynamic formulation with respect to the static one is clear from the decrease in MNE of 0.2m. Comparison with EKF differs depending on the trajectory of the nodes. While EKF outperforms during linear segments, its error overpasses the proposed method during nonlinear motion. Mainly, our method has the advantage of keeping a constant error regardless of the vehicles' movement as it does not rely on a dynamical model of the system (unlike EKF). The corresponding MPE can be found in Table D.1 in appendix (see Fig. 2).

6.3. Robustness to outliers

We consider a more realistic scenario, where outlier distance measurements with value $5d$ are introduced with a probability of 10%, where d is the true distance. Ranges between the outer node and all other vehicles (nodes and anchors) are subject to outliers, evidenced by the peaks in position estimates (Fig. 3). The proposed algorithm presents variations of smaller magnitude when compared to EKF and does not require as much time to recover. The difference between them also seems to accentuate during non-linear trajectory parts with our algorithm attaining superior robustness.

6.4. Performance of our method

For further insight on the performance of the proposed method, we evaluate the computation time and number of iterations per step during a lawnmower trajectory (Fig. 4). During the first steps, while the time window is still not constant, the algorithm is initialized with a random value taking a longer time to converge. After this point, the last estimates are used for initialization and the algorithm is faster to converge (due to convexity the final estimate is independent of initialization choice and this affects only the number of iterations required for convergence). There is also a small increase in convergence time during nonlinear segments of the trajectory.

6.5. Scalability to large networks

We test the scalability of our method by considering networks of 50, 80 and 100 nodes and comparing the centralized and distributed

² Note that $\kappa = 1000$ corresponds to approximately 1.8° .

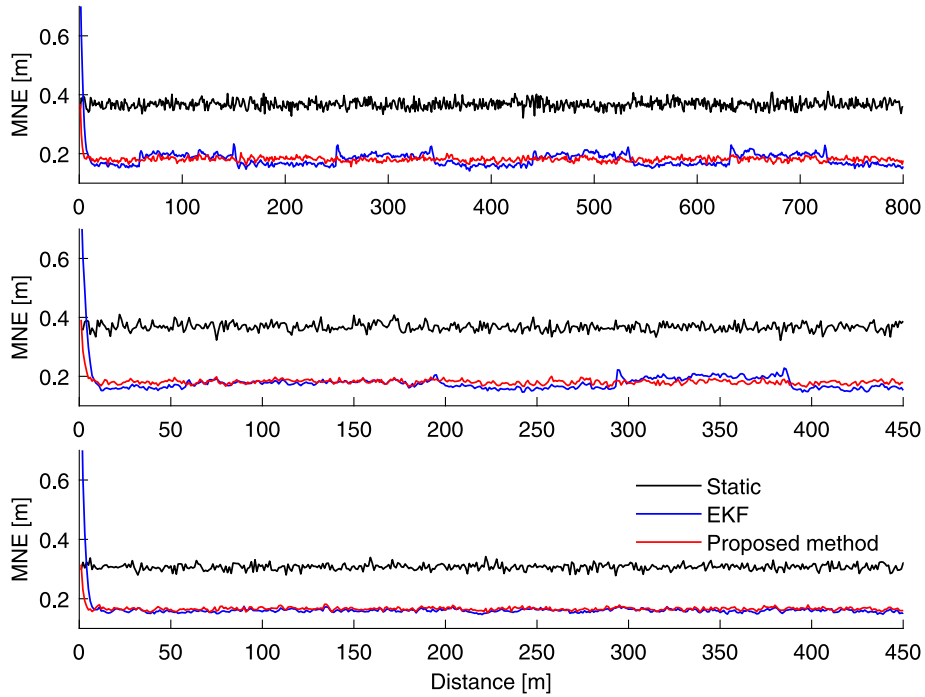


Fig. 2. Comparison of Mean Navigation Error between EKF, a static formulation and our algorithm, during a lawnmower trajectory (top), lap trajectory (middle) and helix trajectory (bottom), averaged over 100 MC trials. The curved segments of the trajectory are evidenced by the higher error in EKF, where our method outperforms it. For the linear parts, EKF shows better accuracy.

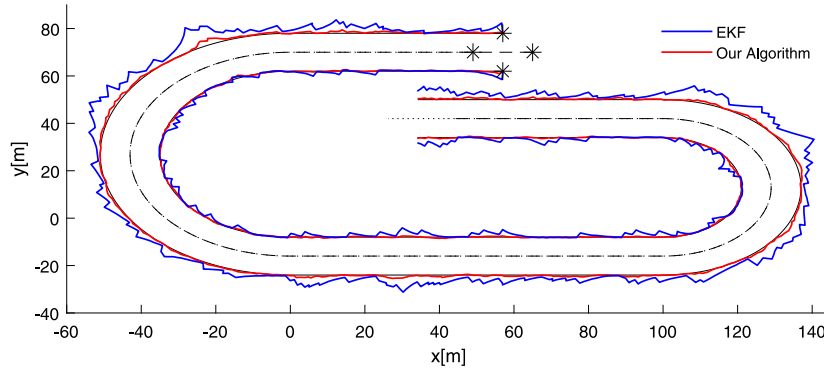


Fig. 3. Outliers during a lap trajectory performed by two anchors, in dashed lines, and two nodes, in black lines, with initial positions marked as a star. Distances between the outer node and all the other vehicles are contaminated with outliers. EKF, in blue, is affected to a higher extent and takes more time to recover than our algorithm, in red.

implementations (Fig. 5)³. The distributed implementation preserves approximately the same computation time per node, regardless of the total number of nodes of the network (if the average node degree is kept constant). Naturally, additional communication time between nodes would have to be considered in order to have an accurate estimate.

6.6. Parameter estimation

We start by examining the evolution of $\hat{\sigma}_d$ for all the distance measurements, when the true σ_d was set to 0.5 m, when performing parameter estimation (Fig. 6). The estimates converge to σ_d , as desired, but it should be noted that a considerable number of measurements is necessary before this happens. Therefore, we require an initial value to be used in the first iterations that will be then corrected. For the next simulations we start parameter estimation at the 20th trajectory step, using σ^0 up to that point.

6.6.1. Impact of initial values

Given the latter observation it is fundamental to assess the impact of the choice of the initial values, or the method does not become effectively parameter-free. We consider as initial value (e.g. σ^0) the true parameter (σ) contaminated with Gaussian noise with zero mean and increasing standard deviation, and compare the version with estimation and without (Fig. 7). With parameter estimation our method uses σ^0 during the first 20 iterations and then uses the estimates according to Section 5. Without estimation the method uses σ^0 throughout the entire trajectory. As expected, when $\sigma^0 \approx \sigma$, the estimation increases the error, as the estimated parameters $\hat{\sigma}$ are less accurate than σ^0 . However, as σ^0 is set further away from the true value, the error without estimation largely increases, while the parameter estimation allows to keep the performance approximately constant. Thus, unless the distribution parameters are known with a large confidence, the estimation process is beneficial.

6.6.2. Impact during trajectories

Finally, we compare the algorithm with and without parameter estimation at level 0.5 of noise in initial parameters (as defined in

³ Details on the large scale setup can be found in [Appendix D](#)

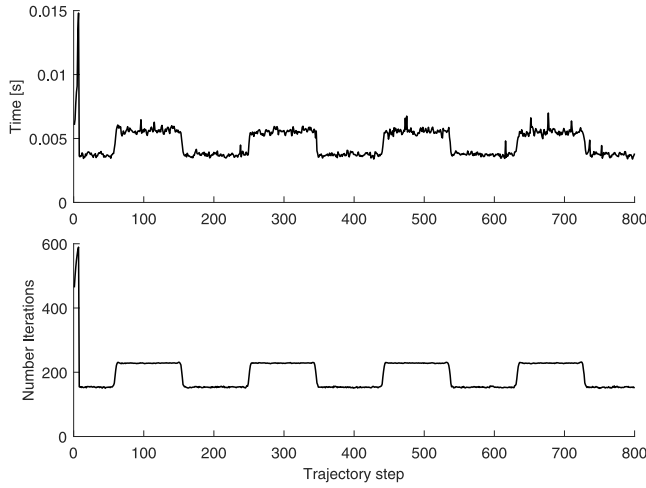


Fig. 4. Time and number of iterations spent to estimate positions, per trajectory step during a lawnmower trajectory. The values are larger during the first iterations when positions are randomly initialized and then suffer small increments during nonlinear motion.

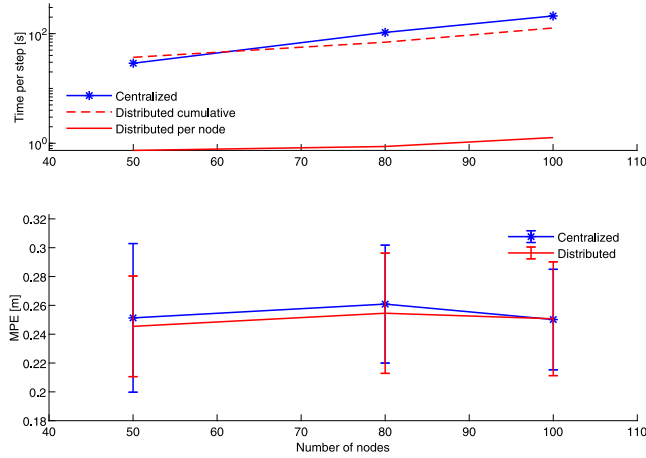


Fig. 5. Variation of the average computation time (per trajectory step) and positioning error with the number of nodes. The average node degree is kept constant over the different simulations. The blue lines refer to the centralized implementation of the proposed method and the red lines to the distributed implementation. In the top plot, the dashed line refers to the total computation time, i.e., the time for all nodes to perform estimates. The solid line is the computation time per node.

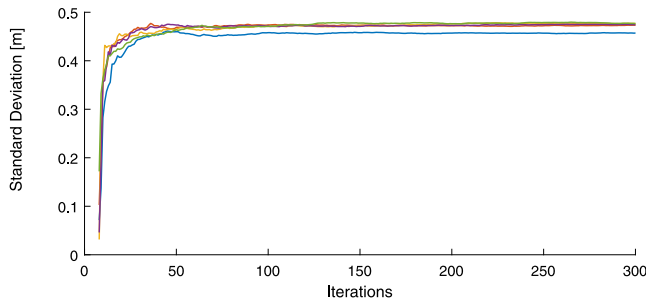


Fig. 6. Evolution of estimates of $\hat{\sigma}_d$ for distance measurements. After some time, the estimates converge to values close to the true $\sigma_d = 0.5$ m. The different lines correspond to σ_{ij} for different edges, all with the same value of STD. Note that parameter estimation does not start at iteration 0 due to the time lag in velocity estimation.

Section 6.6.1) and against the optimal setting where the true noise parameters are known (Fig. 8). During the initial iterations, the noisy parameters are being used in both versions, leading to a higher MNE.

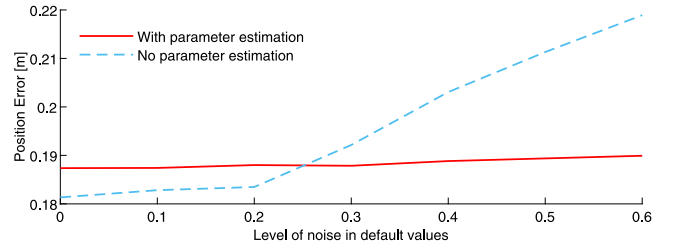


Fig. 7. Variation of MPE for different levels of noise in the initialization of all parameters. The parameters (e.g. σ^0) are initialized with the true value (σ) with additive Gaussian noise of zero mean and standard deviation as a fraction of the true value, that is $\theta\sigma$, where θ corresponds to the levels observed in the x-axis. I.e., for level 0 we are using the true parameters as initial guesses. The algorithm with no parameter estimation uses the initial values (σ^0) throughout the entire trajectory.

After this point, while algorithm with no estimation keeps a similar error, the estimation process allows to achieve an error close to the optimal one. The corresponding MPE can be found in Table D.1 in Appendix D.

7. Concluding remarks

We extend the static MLE formulation for a generic mobile network localization problem from hybrid (range and bearing) measurements to the dynamic setting, in an horizon-based approach. This entails only the additional assumption that each node can measure its own velocity. We show that this extended formulation is still amenable to a tight convex relaxation and can be implemented in a distributed manner.

Comparison with EKF, a centralized state-of-the-art method, shows the superior performance of our method for cases with variable motion patterns, where for mostly linear trajectories the EKF still slightly outperforms our distributed estimator. Nonetheless, unlike EKF, our distributed convex method is proven to converge regardless of initialization and offer nearly constant accuracy throughout the trajectory, thus being more predictable than EKF. More importantly, our algorithm shows more robustness in the presence of outliers in measurements, which is a known issue in real-life scenarios.

Besides, with the proposed parameter estimation process, no parameter tuning is required, so the algorithm can be easily applied to new environments or technologies. However, it should be noted that this is a sensitive step where abnormal values could start a chain of wrong position and parameter estimates. While a formal proof would be desirable in future approaches, our simulations have consistently demonstrated that the method maintains its performance, regardless of the initialization chosen and despite the estimated parameters. Therefore, we have strong empirical evidence of robustness for the proposed method, although additional research is needed to prove its validity formally.

CRedit authorship contribution statement

Filipa Valdeira: Investigation, Writing – original draft, Writing – review & editing, Methodology, Validation, Software, Visualization. **Cláudia Soares:** Conceptualization, Methodology, Supervision, Writing – review & editing, Formal analysis. **João Gomes:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Filipa Valdeira reports financial support was provided by Neuraspaces. Joao Gomes reports financial support was provided by Laboratory for Robotics and Engineering Systems. Claudia Soares reports financial support was provided by NOVA University of Lisbon Laboratory for Computer Science and Informatics.

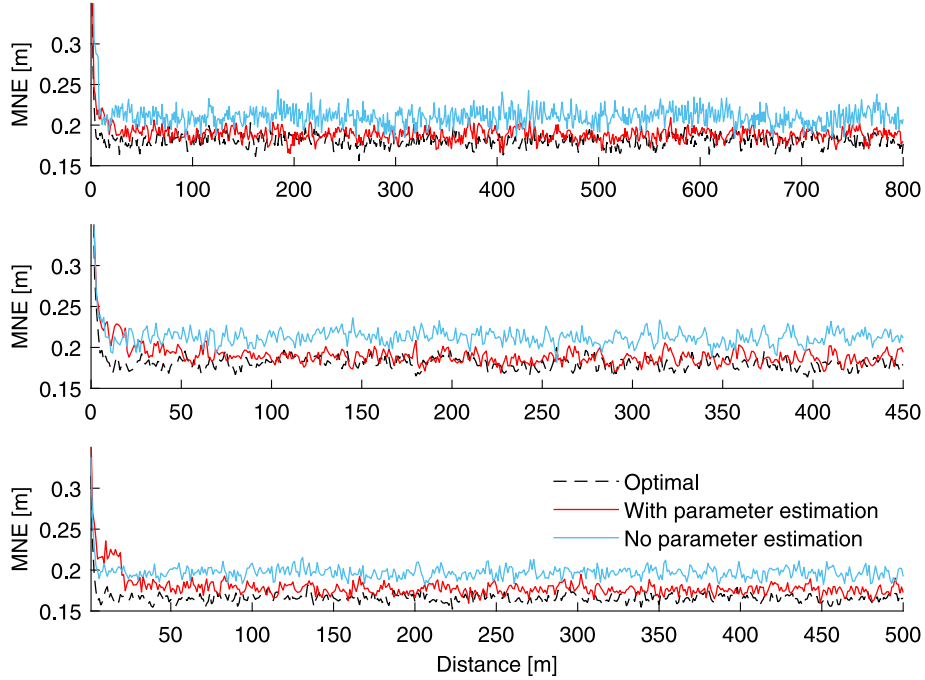


Fig. 8. Comparison of performance between the method with true noise parameters (*Optimal*), with unknown noise parameters throughout the entire trajectory (*No parameter estimation*), with initial unknown parameters and estimation procedure (*With parameter estimation*). Unknown noise parameters correspond to the true ones with additive Gaussian noise with standard deviation of 0.5 of the true value. The results correspond to a lawnmower trajectory (top), lap trajectory (middle) and helix trajectory (bottom).

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by Recovery and Resilience Plan and NextGeneration EU Funds through Project *Artificial Intelligence Fights Space Debris* [C626449889-0046305]; by LARSyS - FCT Project [UIDB/50009/2020]; and by national funds through the FCT – Fundação para a Ciência e a Tecnologia, I.P., under the scope of the projects [UIDB/00297/2020] (<https://doi.org/10.54499/UIDB/00297/2020>) (Center for Mathematics and Applications), [UIDP/00297/2020] (<https://doi.org/10.54499/UIDP/00297/2020>) (Center for Mathematics and Applications) and by [Grant UIDB/04516/2020] (NOVA LINC).

Appendix A. Upper bound on Lipschitz constant

First, the gradient difference is manipulated until an expression with the structure of (18) is obtained as

$$\begin{aligned} \|\nabla f(x) - \nabla f(y)\| &= \|(Mx - b) - (My - b)\| \\ &= \|M(x - y)\| \\ &\leq \|M\| \|x - y\|. \end{aligned} \quad (\text{A.1})$$

The last step of (A.1) corresponds to the definition of Lipschitz continuity with $L = \|M\|$, where $\|M\|$ is the matrix norm of M defined as

$$\|M\| = \sup \left\{ \frac{\|Mx\|}{\|x\|} : x \in \mathbb{R}^n, |x| \neq 0 \right\}.$$

Further manipulations are necessary to obtain a bound for this constant; these are shown in (A.2) and the justification is presented

below.

$$\begin{aligned} L &= \|M\| \\ &\stackrel{1}{\leq} \|M_1\| + \|M_2\| + \|M_3\| \\ &\stackrel{2}{=} \sqrt{\lambda_{\max}(M_1^T M_1)} + \sqrt{\lambda_{\max}(M_2^T M_2)} + \sqrt{\lambda_{\max}(M_3^T M_3)} \\ &\stackrel{3}{=} \lambda_{\max}(M_1) + \lambda_{\max}(M_2) + \lambda_{\max}(M_3) \\ &\stackrel{4}{=} \lambda_{\max}(\Sigma_N^2 A A^T + \Sigma_N^2) + \lambda_{\max}(\Sigma_A^2 E E^T + \Sigma_A^2) \\ &\quad + \lambda_{\max}(\Sigma_V^2 N N^T + \Sigma_V^2) \\ &\stackrel{5}{\leq} \lambda_{\max}(\Sigma_N^2 A A^T) + \lambda_{\max}(\Sigma_A^2 E E^T) + \lambda_{\max}(\Sigma_V^2 N N^T) \\ &\quad + \lambda_{\max}(\Sigma_N^2) + \lambda_{\max}(\Sigma_A^2) + \lambda_{\max}(\Sigma_V^2) \\ &\stackrel{6}{\leq} \frac{1}{\sigma_N^2} \lambda_{\max}(A A^T) + \frac{1}{\sigma_A^2} \lambda_{\max}(E E^T) + \frac{1}{\sigma_V^2} \lambda_{\max}(N N^T) \\ &\quad + \left(\frac{1}{\sigma_N^2} + \frac{1}{\sigma_A^2} + \frac{1}{\sigma_V^2} \right) \end{aligned} \quad (\text{A.2})$$

¹ For 2 matrices A and B , $\|A+B\| \leq \|A\| + \|B\|$ (triangle inequality)

² $\|A\| = \sqrt{\lambda_{\max}(A^T A)}$

³ M_1 , M_2 and M_3 are symmetric ($A^T = A$), thus verifying $\sqrt{\lambda_{\max}(A^T A)} = \lambda_{\max}(A)$

⁴ $\lambda_{\max}(A^T B) = \lambda_{\max}(A B^T)$. For example,

$$\begin{aligned} \lambda_{\max}(M_1) &= \lambda_{\max} \left(\begin{bmatrix} A^T \\ -I \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \Sigma_N^2 A & -\Sigma_N^2 & 0 & 0 \end{bmatrix} \right) \\ &= \lambda_{\max} \left(\begin{bmatrix} \Sigma_N^2 A & -\Sigma_N^2 & 0 & 0 \end{bmatrix} \begin{bmatrix} A^T \\ -I \\ 0 \\ 0 \end{bmatrix} \right) \end{aligned}$$

⁵ Weyl's inequality

⁶ Taking $\frac{1}{\sigma_N^2}, \frac{1}{\sigma_A^2}, \frac{1}{\sigma_V^2}$ as the maximum values of diagonal matrices Σ_N, Σ_A and Σ_V .

The last step is intentionally separated, since it relates with particular characteristics of the network problem. First, the *Laplacian matrix* $L(\mathcal{G})$ of a graph $\mathcal{G}$ with incidence matrix C is defined as $L(\mathcal{G}) = CC^T$. Recalling that matrix A is in fact an arc-node incidence matrix of a virtual network, it is concluded that AA^T is the Laplacian matrix of such network. Additionally, it was proven in [47] that the maximum eigenvalue of $L(\mathcal{G})$ is upper-bounded by twice the maximum node degree of $\mathcal{G}$. Combining these two considerations, it follows that $\lambda_{\max}(AA^T) \leq 2\delta_{\max}^V$, where $\delta_{\max}^V$ denotes the maximum node degree of the virtual network. Finally, under the assumption that the set of edges remains constant under the time window, $\delta_{\max}^V$ is equivalent to $\delta_{\max}$, the maximum node degree of the true network.

The same considerations are applicable to $\lambda_{\max}(NN^T)$, with $\lambda_{\max}(NN^T) \leq 2\delta_{\max}^V$. Here, $\delta_{\max}^V$ is either 2 for any $T > 2$, 1 for $T = 2$, and 0 for no time window ($T = 1$). It will then be designated as $\delta_{\max}^1(T)$, a function of time window. Finally, $\lambda_{\max}(EE^T)$ is noted to be upper bounded by the maximum cardinality over all the sets $\mathcal{A}_i$ [37]. In other words, this value corresponds to the maximum number of anchors connected to a node.

With $K = \frac{1}{\sigma_N^2} + \frac{1}{\sigma_A^2} + \frac{1}{\sigma_V^2}$ and the above considerations, the previous upper bound may be transformed as

$$L \leq \frac{1}{\sigma_N^2} \lambda_{\max}(AA^T) + \frac{1}{\sigma_A^2} \lambda_{\max}(EE^T) + \frac{1}{\sigma_V^2} \lambda_{\max}(NN^T) + K \\ \leq \frac{1}{\sigma_N^2} 2\delta_{\max} + \frac{1}{\sigma_A^2} \max_{i \in \mathcal{V}} |\mathcal{A}_i| + \frac{1}{\sigma_V^2} 2\delta_{\max}^1(T) + K.$$

Appendix B. Distributed algorithm

B.1. Distributed algorithm

We present the algorithm for distributed method in Algorithm 1. Note that the algorithm contains an outer iteration over time t and an inner iteration κ for the solution of problem (15) with FISTA. Finally, iterations in τ correspond to the time window. We define $C_{(i \sim j, i)}$ as the element of C on row referring to edge $i \sim j$ and column referring to node i ; N_i as the sub-matrix of N whose rows correspond to node i ; $\mathbf{s}_i$ and $\mathbf{x}_i$ as the concatenation of $s_i(\tau)$ and $x_i(\tau)$ over the time window; $\mathbf{w}_i$ and $\mathbf{y}_i$ as the concatenation of $w_{ik}(\tau)$ and $y_{ij}(\tau)$ over the respective edges and time window. Besides, the projections for each component are defined as $\mathcal{Y}_{ij}(t) = \{y_{ij}(t) : \|y(t)\| \leq d_{ij}(t)\}$, $\mathcal{W}_{ik}(t) = \{w_{ik}(t) : \|w_{ik}(t)\| \leq r_{ik}(t)\}$ and $\mathcal{S}_i(t) = \{s_i(t) : \|s_i(t)\| \leq v_i(t)\}$. The elements $F_1(\tau)$ through $F_4(\tau)$ for the update of $\mathbf{x}_i^{k+1}(\tau)$ are given as

$$F_1(\tau) = \hat{\mathbf{x}}_i^k(\tau) \left(L - \sum_{i \sim j \in \mathcal{E}_i} \frac{1}{\sigma_{ij}^2} - \sum_{k \in \mathcal{A}_i} \frac{1}{\sigma_{ik}^2} \right),$$

$$F_2(\tau) = \sum_{i \sim j \in \mathcal{E}_i} \frac{1}{\sigma_{ij}^2} (\hat{\mathbf{x}}_j^k(\tau) + C_{(i \sim j, i)}(\tau) \hat{\mathbf{y}}_{ij}^k(\tau)),$$

$$F_3(\tau) = \sum_{k \in \mathcal{A}_i} \frac{1}{\sigma_{ik}^2} (\hat{\mathbf{w}}_{ik}^k(\tau) + \alpha_k(\tau))$$

and

$$F_4(\tau) = [N_i^T \Sigma_V^2 (\hat{\mathbf{s}}_i^k - N_i \hat{\mathbf{x}}_i^k)]_\tau,$$

where $[v]_\tau$ is the element of a vector v , relative to instant τ . The update depends on the previous iteration κ and is split into four terms for convenience of the reader: $F_1(\tau)$ relates to the node position; $F_2(\tau)$ and $F_3(\tau)$ relate to distance edges between the node and its neighbor nodes or anchors (respectively); and $F_4(\tau)$ to the node velocity.

Algorithm 1 Hybrid Convex Localization

Input: L ;

$\{d_{ij}(t), u_{ij}(t) : i \sim j \in \mathcal{E}\}$; $\{r_{ik}(t), q_{ik}(t) : i \in \mathcal{V}, k \in \mathcal{A}\}$; $\{v_i(t) : i \in \mathcal{V}\}$

Output: $\hat{\mathbf{x}} = \{\hat{\mathbf{x}}_i\}$

Choose $\mathbf{z}^0 = (x^0, y^0, w^0, s^0) = 0$;

for all t **do**

$\kappa = 1$; $\mathbf{z}^1 = \mathbf{z}^0$

while some stopping criterion is not met, each node i **do**

$\hat{\mathbf{z}}_i^k = \mathbf{z}_i^k + \frac{\kappa-1}{\kappa} (\mathbf{z}_i^k - \mathbf{z}_i^{k-1})$

for all $t - T_0 \leq \tau \leq t$ **do**

$\mathbf{x}_i^{k+1}(\tau) = \frac{1}{L} [F_1(\tau) + F_2(\tau) + F_3(\tau) + F_4(\tau)]$

end for

for all $i \sim j \in \mathcal{E}_i$ and $t - T_0 \leq \tau \leq t$ **do**

$\mathbf{y}_{ij}^{k+1}(\tau) = P_{\mathcal{Y}_{ij}(\tau)} \left(\frac{L-1/\sigma_{ij}^2}{L} \hat{\mathbf{y}}_{ij}^k(\tau) + \frac{1}{L} \hat{\mathbf{u}}_{ij}(\tau) + \frac{1}{L\sigma_{ij}^2} C_{(i \sim j, i)}(\tau) (\hat{\mathbf{x}}_i^k(\tau) - \hat{\mathbf{x}}_j^k(\tau)) \right)$

end for

for all $k \in \mathcal{A}_i$ and $t - T_0 \leq \tau \leq t$ **do**

$\mathbf{w}_{ik}^{k+1}(\tau) = P_{\mathcal{W}_{ik}(\tau)} \left(\frac{L-1/\sigma_{ik}^2}{L} \hat{\mathbf{w}}_{ik}^k(\tau) + \frac{1}{L\sigma_{ik}^2} (\hat{\mathbf{x}}_i^k(\tau) - \mathbf{a}_k(\tau)) + \frac{\hat{\mathbf{q}}_{ik}(\tau)}{L} \right)$

end for

if $T_0 > 1$ **then**

for all $t - T_0 + 1 \leq \tau \leq t$ **do**

$\mathbf{s}_i^{k+1}(\tau) = P_{\mathcal{S}_i} \left(\frac{L-1/\sigma_i^2}{L} \hat{\mathbf{s}}_i^k(\tau) - \hat{\mathbf{x}}_i^k(\tau - 1) + \frac{\hat{\mathbf{v}}_i(\tau)}{L} \right)$

end for

end if

$\mathbf{z}_i^{k+1} = (\mathbf{x}_i^{k+1}, \mathbf{y}_i^{k+1}, \mathbf{w}_i^{k+1}, \mathbf{s}_i^{k+1})$

$\kappa = \kappa + 1$

broadcast $\mathbf{x}_i$ to all neighbors

end while

return $\hat{\mathbf{x}} = \{\mathbf{x}_i^{k+1}\}$

end for

B.2. Derivation of the distributed algorithm

In this section, we present the full derivation of Algorithm 1. We start by replacing the gradient $\nabla g(\mathbf{z}) = M\mathbf{z} - \mathbf{b}$ in the update equations for FISTA (17), obtaining the two updates as

$$\hat{\mathbf{z}}^k = \mathbf{z}^k + \frac{\kappa-1}{\kappa} (\mathbf{z}^k - \mathbf{z}^{k-1}) \\ \mathbf{z}^{k+1} = P_{\mathcal{Z}} \left(\hat{\mathbf{z}}^k - \frac{1}{L} (M\hat{\mathbf{z}}^k - \mathbf{b}) \right). \quad (\text{B.1})$$

We shall further develop the second equation, in order to retrieve the update for each node i . Considering the update for each component of $\mathbf{z}$ and the definitions of matrices M and $\mathbf{b}$, we get

$$\mathbf{x}^{k+1} = \left[I - \frac{1}{L} (A^T \Sigma_N^2 A + E^T \Sigma_A^2 E + N^T \Sigma_V^2 N) \right] \hat{\mathbf{x}}^k \\ + \frac{1}{L} \Sigma_N^2 A^T \hat{\mathbf{y}}^k + \frac{1}{L} \Sigma_A^2 E^T (\hat{\mathbf{w}}^k + \alpha) + \frac{1}{L} \Sigma_V^2 N^T \hat{\mathbf{s}}^k \\ \mathbf{y}^{k+1} = P_{\mathcal{Y}} \left(\left(I - \frac{1}{L} \Sigma_N^2 \right) \hat{\mathbf{y}}^k + \frac{1}{L} \Sigma_N^2 A \hat{\mathbf{x}}^k + \frac{1}{L} \alpha \right) \\ \mathbf{w}^{k+1} = P_{\mathcal{W}} \left(\left(I - \frac{1}{L} \Sigma_A^2 \right) \hat{\mathbf{w}}^k + \frac{1}{L} \Sigma_A^2 E \hat{\mathbf{x}}^k - \frac{1}{L} \Sigma_A^2 \alpha + \frac{1}{L} q \right) \\ \mathbf{s}^{k+1} = P_{\mathcal{S}} \left(\left(I - \frac{1}{L} \Sigma_V^2 \right) \hat{\mathbf{s}}^k + \frac{1}{L} \Sigma_V^2 N \hat{\mathbf{x}}^k + \frac{1}{L} v \right), \quad (\text{B.2})$$

where $P_{\mathcal{Y}}$, $P_{\mathcal{W}}$, $P_{\mathcal{S}}$ are the projections onto the subsets of $\mathcal{Z}$ of the respective variables y , w , s . We refer the reader to Section 4.1 for the definition of the projection operator and sets $\mathcal{Y}$, $\mathcal{W}$, $\mathcal{Z}$, $\mathcal{Z}$; and to Section 3.1 for the definition of the different matrices. Below, we consider the update for $\mathbf{x}^{k+1}$ separately, as it follows a different structure, while the remaining ones are similar.

B.2.1. Update for the first component

Let us first reformulate the expression for x^{k+1} in (B.2) as

$$x^{k+1} = x^k + \frac{1}{L} \left[A^T \Sigma_N^2 (\hat{y}^k - A \hat{x}^k) + N^T \Sigma_V^2 (\hat{s}^k - N \hat{x}^k) + E^T \Sigma_A^2 (\hat{w}^k + \alpha - E \hat{x}^k) \right], \quad (\text{B.3})$$

where we have joined terms relative to the same edge matrices. We shall go over each of the three terms to obtain an update for $x_i^{k+1}(\tau)$.

Consider the term $A^T \Sigma_N^2 (\hat{y}^k - A \hat{x}^k)$. Only for derivation purposes, we define an auxiliary variable

$$\theta_{ij} = \frac{1}{\sigma_{ij}^2} y_{ij}^k(\tau) - (x_i^k(\tau) - x_j^k(\tau))$$

and we define $\theta(\tau) = \{\theta_{ij}(\tau)\}_{i \sim j}$ and $\theta = \{\theta(\tau)\}_{t-T_0 \leq \tau \leq t}$. It follows that

$$\theta = \Sigma_N^2 (\hat{y}^k - A \hat{x}^k)$$

and, consequently, we consider the term $A^T \theta$. Multiplication on the left by A^T equates to summing all terms in θ related to each node i at time τ . Therefore, we define $C_{(i \sim j, i)}$ as the element of C on row referring to edge $i \sim j$ and column referring to node i and obtain the update relative to $x_i(\tau)$ as

$$\begin{aligned} & \sum_{i \sim j \in \mathcal{E}_i} \frac{1}{\sigma_{ij}^2} (C_{(i \sim j, i)}(\tau) y_{ij}^k(\tau) + x_j^k(\tau) - x_i^k(\tau)) = \\ & - x_i^k(\tau) \sum_{i \sim j \in \mathcal{E}_i} \frac{1}{\sigma_{ij}^2} + \sum_{i \sim j \in \mathcal{E}_i} \frac{1}{\sigma_{ij}^2} (C_{(i \sim j, i)}(\tau) y_{ij}^k(\tau) + x_j^k(\tau)). \end{aligned} \quad (\text{B.4})$$

Now, consider the term $E^T \Sigma_A^2 (\hat{w}^k + \alpha - E \hat{x}^k)$. Following the same reasoning we obtain the update relative to $x_i(\tau)$ as

$$\begin{aligned} & \sum_{k \in \mathcal{A}_i} \frac{1}{\varsigma_{ik}^2} (w_{ik}^k(\tau) + \alpha_k(\tau) - x_i^k(\tau)) = \\ & - x_i^k(\tau) \sum_{k \in \mathcal{A}_i} \frac{1}{\varsigma_{ik}^2} + \sum_{k \in \mathcal{A}_i} \frac{1}{\varsigma_{ik}^2} (w_{ik}^k(\tau) + \alpha_k(\tau)). \end{aligned} \quad (\text{B.5})$$

Finally, consider the term $N^T \Sigma_V^2 (\hat{s}^k - N \hat{x}^k)$. Recall that $N = C_{\text{vel}} \otimes I_p$ and each node has an edge with its position at $t-1$ and $t+1$, except for the first and last instants of the time window. Since the edges do not connect different nodes, we can define N_i as the sub-matrix of N , whose rows correspond to node i . Further defining $\mathbf{s}_i$ and $\mathbf{x}_i$ as the concatenation of $s_i(\tau)$ and $x_i(\tau)$ over the time window, we get $N_i^T \Sigma_V^2 (\mathbf{s}_i^k - N_i \mathbf{x}_i^k)$, as a vector for the entire time window relative to node i . We then define

$$[N_i^T \Sigma_V^2 (\mathbf{s}_i^k - N_i \mathbf{x}_i^k)]_\tau \quad (\text{B.6})$$

as the element relative to instant τ .

Since we have obtained a separate update for $x_i(\tau)$ for each of the terms in (B.3), (and x^k is trivially distributed), we get the complete update for $x_i^{k+1}(\tau)$ from (B.4), (B.5) and (B.6) as

$$\begin{aligned} x_i^{k+1}(\tau) = \frac{1}{L} \left[x_i^k(\tau) \left(L - \sum_{i \sim j \in \mathcal{E}_i} \frac{1}{\sigma_{ij}^2} - \sum_{k \in \mathcal{A}_i} \frac{1}{\varsigma_{ik}^2} \right) + \sum_{i \sim j \in \mathcal{E}_i} \frac{1}{\sigma_{ij}^2} \left(C_{(i \sim j, i)}(\tau) y_{ij}^k(\tau) \right. \right. \\ \left. \left. + x_j^k(\tau) \right) + [N_i^T \Sigma_V^2 (\mathbf{s}_i^k - N_i \mathbf{x}_i^k)]_\tau + \sum_{k \in \mathcal{A}_i} \frac{1}{\varsigma_{ik}^2} (w_{ik}^k(\tau) + \alpha_k(\tau)) \right]. \end{aligned}$$

For convenience, we shall write this update as

$$x_i^{k+1}(\tau) = \frac{1}{L} [F_1(\tau) + F_2(\tau) + F_3(\tau) + F_4(\tau)],$$

where

$$F_1(\tau) = \hat{x}_i^k(\tau) \left(L - \sum_{i \sim j \in \mathcal{E}_i} \frac{1}{\sigma_{ij}^2} - \sum_{k \in \mathcal{A}_i} \frac{1}{\varsigma_{ik}^2} \right),$$

$$F_2(\tau) = \sum_{i \sim j \in \mathcal{E}_i} \frac{1}{\sigma_{ij}^2} \left(\hat{x}_j^k(\tau) + C_{(i \sim j, i)}(\tau) \hat{y}_{ij}^k(\tau) \right),$$

$$F_3(\tau) = \sum_{k \in \mathcal{A}_i} \frac{1}{\varsigma_{ik}^2} (\hat{w}_{ik}^k(\tau) + \alpha_k(\tau))$$

and

$$F_4(\tau) = [N_i^T \Sigma_V^2 (\mathbf{s}_i^k - N_i \hat{x}_i^k)]_\tau.$$

B.2.2. Update for the remaining components

For the update of y, w, z , note that we can reformulate the equations in (B.2) as

$$\begin{aligned} y^{k+1} &= P_y \left(\frac{1}{L} [u + \Sigma_N^2 A \hat{x}^k + (I L - \Sigma_N^2) \hat{y}^k] \right) \\ w^{k+1} &= P_w \left(\frac{1}{L} [q - \Sigma_A^2 \alpha + \Sigma_A^2 E \hat{x}^k + (I L - \Sigma_A^2) \hat{w}^k] \right) \\ s^{k+1} &= P_s \left(\frac{1}{L} [v + \Sigma_V^2 N \hat{x}^k + (I L - \Sigma_V^2) \hat{s}^k] \right). \end{aligned}$$

It follows that the update for $y_{ij}^{k+1}(\tau)$ is given as

$$y_{ij}^{k+1}(\tau) = P_{y_{ij}(\tau)} \left(\frac{1}{L} [\tilde{u}_{ij}(\tau) + \frac{L \sigma_{ij}^2 - 1}{\sigma_{ij}^2} \hat{y}_{ij}^k(\tau) + \frac{C_{(i \sim j, i)}(\tau)}{\sigma_{ij}^2} (\hat{x}_i^k(\tau) - \hat{x}_j^k(\tau))] \right), \quad (\text{B.7})$$

the update for $w_{ik}^{k+1}(\tau)$ as

$$w_{ik}^{k+1}(\tau) = P_{w_{ik}(\tau)} \left(\frac{1}{L} [\tilde{q}_{ik}(\tau) + \frac{L \varsigma_{ik}^2 - 1}{\varsigma_{ik}^2} \hat{w}_{ik}^k(\tau) + \frac{1}{\varsigma_{ik}^2} (\hat{x}_i^k(\tau) - \alpha_k(\tau))] \right) \quad (\text{B.8})$$

and the update for $s_i^{k+1}(\tau)$ as

$$s_i^{k+1}(\tau) = P_{s_i(\tau)} \left(\frac{1}{L} [\tilde{v}_i(\tau) + \frac{L \sigma_i^2 - 1}{\sigma_i^2} \hat{s}_i^k(\tau) + \frac{1}{\sigma_i^2} (\hat{x}_i^k(\tau) - \hat{x}_i^k(\tau - 1))] \right). \quad (\text{B.9})$$

Note that updates in (B.7), (B.8) and (B.9) only require information from node i or its neighbors.

Appendix C. Algorithm with parameter estimation

Algorithm 2 is the extension of Algorithm 1 with parameter estimation. Bold symbols denote the concatenation of respective variables in the following way

$$\begin{aligned} \mathbf{d}(t) &= \{d_{ij}(t) : i \sim j \in \mathcal{E}\} \\ \mathbf{u}(t) &= \{u_{ij}(t) : i \sim j \in \mathcal{E}\} \\ \mathbf{r}(t) &= \{r_{ik}(t) : i \in \mathcal{V}, k \in \mathcal{A}\} \\ \mathbf{q}(t) &= \{q_{ik}(t) : i \in \mathcal{V}, k \in \mathcal{A}\} \\ \boldsymbol{\beta}(t) &= \{\beta_i(t) \in \mathcal{V}\} \\ \boldsymbol{\sigma}(t) &= \{\sigma_{ij}(t) : i \sim j \in \mathcal{E}\} \\ \boldsymbol{\kappa}(t) &= \{\kappa_{ij}(t) : i \sim j \in \mathcal{E}\} \\ \boldsymbol{\varsigma}(t) &= \{\varsigma_{ik}(t) : i \in \mathcal{V}, k \in \mathcal{A}\} \\ \boldsymbol{\lambda}(t) &= \{\lambda_{ij}(t) : i \in \mathcal{V}, k \in \mathcal{A}\} \\ \boldsymbol{\sigma}_v(t) &= \{\sigma_i(t) : i \in \mathcal{V}\} \\ \boldsymbol{\kappa}_v(t) &= \{\kappa_i(t) : i \in \mathcal{V}\}. \end{aligned}$$

Algorithm 2 Parameter Estimation**Input:** L ;

$$z^0 = (x^0, y^0, u^0, s^0), \mathbf{d}(t), \mathbf{u}(t), \mathbf{r}(t), \mathbf{q}(t), \boldsymbol{\beta}(t), \boldsymbol{\sigma}^0, \boldsymbol{\kappa}^0, \boldsymbol{\zeta}^0, \boldsymbol{\lambda}^0, \boldsymbol{\sigma}_v^0, \boldsymbol{\kappa}_v^0;$$

Output: $\hat{\mathbf{x}}(t) = \{\hat{\mathbf{x}}_i(t)\}$ 1: $\sigma'_{ij} = \zeta'_{ik} = \gamma_{ij} = \delta_{ik} = 0$ 2: **for all** t **do**

3: Steps 2 to 26 of Algorithm 1

$$\hat{\mathbf{z}}(t) = \text{Algorithm_1} \left(\hat{\mathbf{z}}(t-1), \mathbf{d}(t), \mathbf{r}(t), \mathbf{u}(t), \mathbf{q}(t), \boldsymbol{\beta}(t), \boldsymbol{\sigma}(t), \boldsymbol{\zeta}(t), \boldsymbol{\kappa}(t), \right. \\ \left. \lambda(t), \boldsymbol{\sigma}_v(t), \boldsymbol{\kappa}_v(t), L(t) \right)$$

4: **for all** node i **do**5: **for all** $i \sim j \in \mathcal{E}_i$ **do**

$$\hat{d}_{ij}(t) = \|\hat{\mathbf{x}}_i(t) - \hat{\mathbf{x}}_j(t)\|, \hat{u}_{ij}(t) = \frac{\hat{\mathbf{x}}_i(t) - \hat{\mathbf{x}}_j(t)}{\|\hat{\mathbf{x}}_i(t) - \hat{\mathbf{x}}_j(t)\|};$$

$$\sigma'_{ij} = \sigma'_{ij} + (d_{ij}(t) - \hat{d}_{ij}(t))^2, \gamma_{ij} = \gamma_{ij} + \frac{u_{ij}(t) - \hat{u}_{ij}(t)}{\|u_{ij}(t) - \hat{u}_{ij}(t)\|};$$

$$\hat{\sigma}_{ij}^2(t+1) = \frac{1}{t} \sigma'_{ij}, \bar{\gamma}_{ij}(t) = \frac{\|\gamma_{ij}\|}{t}$$

$$\hat{\mathbf{x}}_{ij}(t+1) = \frac{\bar{\gamma}_{ij}(t)(p - \bar{\gamma}_{ij}^2(t))}{1 - \bar{\gamma}_{ij}(t)^2}$$

10: **end for**11: **for all** $k \in \mathcal{A}_i$ **do**

$$\hat{r}_{ik}(t) = \|\hat{\mathbf{x}}_i(t) - \mathbf{a}_k(t)\|, \hat{q}_{ik}(t) = \frac{\hat{\mathbf{x}}_i(t) - \mathbf{a}_k(t)}{\|\hat{\mathbf{x}}_i(t) - \mathbf{a}_k(t)\|};$$

$$\zeta'_{ik} = \zeta'_{ik} + (r_{ik}(t) - \hat{r}_{ik}(t))^2, \delta_{ik} = \delta_{ik} + \frac{q_{ik}(t) - \hat{q}_{ik}(t)}{\|q_{ik}(t) - \hat{q}_{ik}(t)\|};$$

$$\hat{\zeta}_{ik}^2(t+1) = \frac{1}{t} \zeta'_{ik}, \bar{\delta}_{ik}(t) = \frac{\|\delta_{ik}\|}{t}$$

$$\hat{\lambda}_{ik}(t+1) = \frac{\bar{\delta}_{ik}(t)(p - \bar{\delta}_{ik}^2(t))}{1 - \bar{\delta}_{ik}(t)^2}$$

16: **end for**17: **if** $t > 7$ **then**

$$\hat{v}_i(t) \hat{V}_i(t) \Delta T = \frac{5(\hat{\mathbf{x}}_i(t-3) - \hat{\mathbf{x}}_i(t-5))}{32} + \frac{4(\hat{\mathbf{x}}_i(t-2) - \hat{\mathbf{x}}_i(t-6))}{32} + \frac{(\hat{\mathbf{x}}_i(t-1) - \hat{\mathbf{x}}_i(t-7))}{32}$$

$$V_i(t) v_i(t) = \boldsymbol{\beta}(t)$$

$$\sigma'_i = \sigma'_i + (V_i(t) - \hat{V}_i(t))^2, \gamma_i = \gamma_i + (v_i(t) - \hat{v}_i(t));$$

$$\hat{\sigma}_i^2(t+1) = \frac{1}{t} \sigma'_i, \bar{\gamma}_i(t) = \frac{\|\gamma_i\|}{t}$$

$$\hat{\mathbf{x}}_i(t+1) = \frac{\bar{\gamma}_i(t)(p - \bar{\gamma}_i^2(t))}{1 - \bar{\gamma}_i(t)^2}$$

24: **end if**25: **end for**26: **return** $\hat{\mathbf{x}}(t)$ 27: **end for****Appendix D. Additional simulation details***D.1. Details on simulation settings*

Trajectories. The lawnmower trajectory starts with all nodes separated by 15 m on both axes and then traveling at a speed of 1 m/s. In the helix trajectory, the two nodes are separated by 10 m and separated from the closest anchors by 8 m and from the furthest by 15 m.

Large scale setting. For large scale experiments a partial lap trajectory is considered (Fig. D.9). The nodes are initially placed in the positions indicated with stars and then three random nodes are selected as anchors. A disk radius of 45m is used to define neighbors for each of the nodes and the network topology is assumed constant throughout the trajectory. All nodes travel at a speed of 1 m/s.

D.2. Mean positioning errors

Table D.1 contains the MPE for the three trajectories and all the considered methods.

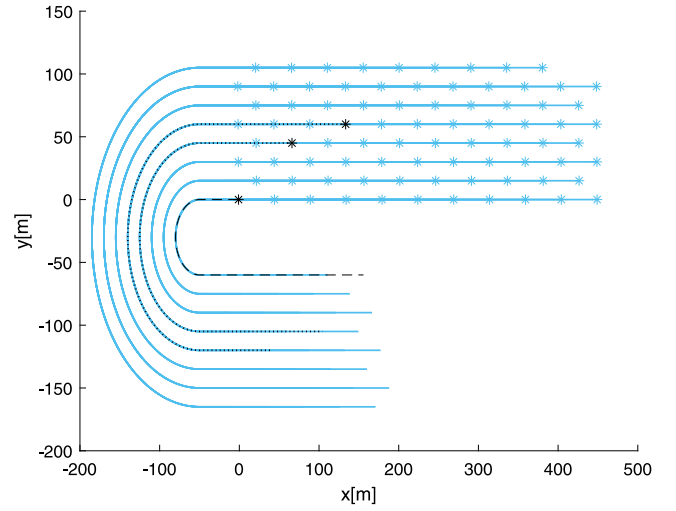


Fig. D.9. Example of a trajectory with 80 nodes used for large scale experiments. Nodes are depicted in blue stars and the corresponding trajectories with blue solid lines. Anchors are depicted with black stars and their trajectories with dashed black lines. Nodes describe a partial lap trajectory.

Table D.1

MPE for each trajectory and all methods.

	EKF	Static	Ours Opt.	Ours Est.	Ours No Est.
Lawnmower	0.1814	0.3675	0.1813	0.1894	0.2113
Lap	0.1792	0.3674	0.1817	0.1903	0.2131
Helix	0.1635	0.3070	0.1665	0.1789	0.1973

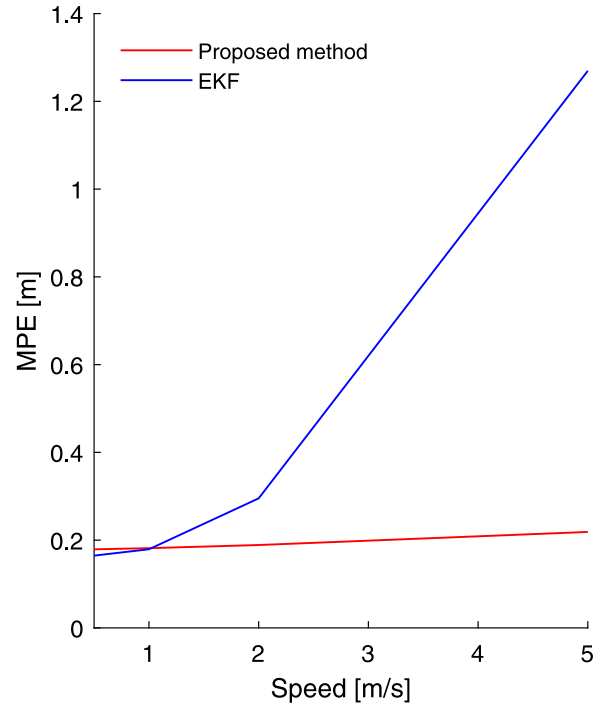


Fig. D.10. MPE for different speed values during a lap trajectory.

D.3. Different speeds

We compare our method and EKF during a lap trajectory for different values of speed (Fig. D.10). However, we note that in this case EKF was not tuned for the different speed measurements, except for

the default value of 1 m/s. While the positioning error increases for both methods as the speed increases, our proposed solution suffers an increase of less than 0.1 m, while EKF increases by one order of magnitude.

References

- [1] P.A. Miller, J.A. Farrell, Y. Zhao, V. Djapic, Autonomous underwater vehicle navigation, *IEEE J. Ocean. Eng.* 35 (3) (2010) 663–678.
- [2] T. Yoo, DVL/RPM based velocity filter aiding in the underwater vehicle integrated inertial navigation system, *J. Sensor Technol.* 4 (2014).
- [3] L. Ruan, G. Li, W. Dai, S. Tian, G. Fan, J. Wang, X. Dai, Cooperative relative localization for UAV swarm in GNSS-denied environment: a coalition formation game approach, *IEEE Internet Things J.* 9 (13) (2022) 11560–11577.
- [4] L. Wielandner, E. Leitinger, K. Witrissal, RSS-based cooperative localization and orientation estimation exploiting antenna directivity, *IEEE Access PP* (2021) 1.
- [5] B.C. Pinheiro, U.F. Moreno, J.T.B. de Sousa, O.C. Rodríguez, Kernel-function-based models for acoustic localization of underwater vehicles, *IEEE J. Ocean. Eng.* 42 (3) (2017) 603–618.
- [6] L. Dong, Cooperative localization and tracking of mobile ad hoc networks, *IEEE Trans. Signal Process.* 60 (7) (2012) 3907–3913.
- [7] G. Sheng, X. Liu, Y. Sheng, X. Cheng, H. Luo, Cooperative navigation algorithm of extended Kalman filter based on combined observation for AUVs, *Remote Sens.* 15 (2) (2023).
- [8] S.M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall, 1997.
- [9] G.C. Calafiore, L. Carlone, M. Wei, Distributed optimization techniques for range localization in networked systems, in: 49th IEEE Conf. on Decision and Control, CDC, 2010, pp. 2221–2226.
- [10] J.A. Costa, N. Patwari, A.O. Hero, Distributed weighted-multidimensional scaling for node localization in sensor networks, *ACM Trans. Sens. Netw.* 2 (1) (2006) 39–64.
- [11] T. Erseghe, A distributed and maximum-likelihood sensor network localization algorithm based upon a nonconvex problem formulation, *IEEE Trans. Signal Inf. Process. Over Netw.* 1 (4) (2015) 247–258.
- [12] P. Biswas, Y. Ye, A distributed method for solving semidefinite programs arising from ad hoc wireless sensor network localization, in: *Multiscale Optimization Methods and Applications*, Springer, 2006, pp. 69–84.
- [13] G. Yang, et al., Improved robust TOA-based source localization with individual constraint of sensor location uncertainty, *Signal Process.* 196 (2022) 108504.
- [14] P. Tseng, Second order cone programming relaxation of sensor network localization, *SIAM J. Optim.* 18 (1) (2007) 156–185.
- [15] J. Nie, Sum of squares method for sensor network localization, *Comput. Optim. Appl.* 43 (2006).
- [16] S. Srirangarajan, A. Tewfik, Z.-Q. Luo, Distributed sensor network localization using SOCP relaxation, *IEEE Trans. Wirel. Commun.* 7 (2009) 4886–4895.
- [17] Q. Shi, C. He, H. Chen, L. Jiang, Distributed wireless sensor network localization via sequential greedy optimization algorithm, *IEEE Trans. Signal Process.* 58 (6) (2010) 3328–3340.
- [18] C. Soares, J. Gomes, STRONG: Synchronous and asynchronous robust network localization, under non-Gaussian noise, *Signal Process.* 185 (2021) 108066.
- [19] Z. Wang, Y. Chang, H. Zhang, H. Yan, Bearing-only distributed localization for multi-agent systems with complex coordinates, *Inform. Sci.* 626 (2023) 837–850.
- [20] D. Crouse, R. Osborne, K. Pattipati, P. Willett, Y. Bar-Shalom, Efficient 2D sensor location estimation using targets of opportunity, *J. Adv. Inf. Fusion* 8 (2013).
- [21] L. Ming-Yong, L. Wen-Bai, P. Xuan, Convex optimization algorithms for cooperative localization in autonomous underwater vehicles, *Acta Automat. Sinica* 36 (5) (2010) 704–710.
- [22] B.Q. Ferreira, J. Gomes, C. Soares, J.P. Costeira, FLORIS and CLORIS: Hybrid source and network localization based on ranges and video, *Signal Process.* 153 (2018) 355–367.
- [23] P. Biswas, H. Aghajan, Y. Ye, Semidefinite programming algorithms for sensor network localization using angle information, in: 39th Asilomar Conference on Signals, Systems and Computers, 2005, pp. 220–224.
- [24] H. Naseri, V. Koivunen, Convex relaxation for maximum-likelihood network localization using distance and direction data, in: 2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications, SPAWC, 2018, pp. 1–5.
- [25] T. Eren, Cooperative localization in wireless ad hoc and sensor networks using hybrid distance and bearing (angle of arrival) measurements, *EURASIP J. Wirel. Commun. Netw.* 2011 (2011).
- [26] Z. Lin, T. Han, R. Zheng, C. Yu, Distributed localization with mixed measurements under switching topologies, *Automatica* 76 (2017) 251–257.
- [27] W. Ding, S. Chang, J. Li, A novel weighted localization method in wireless sensor networks based on hybrid RSS/AoA measurements, *IEEE Access PP* (2021) 1.
- [28] S. Tomic, M. Beko, M. Tuba, A linear estimator for network localization using integrated RSS and AOA measurements, *IEEE Signal Process. Lett.* 26 (3) (2019) 405–409.
- [29] C. Soares, F. Valdeira, J. Gomes, Range and bearing data fusion for precise convex network localization, *IEEE Signal Process. Lett.* 27 (2020) 670–674.
- [30] S. Kumar, R. Kumar, K. Rajawat, Cooperative localization of mobile networks via velocity-assisted multidimensional scaling, *IEEE Trans. Signal Process.* 64 (7) (2016) 1744–1758.
- [31] Y. Fan, K. Ding, X. Qi, L. Liu, Cooperative localization of 3D mobile networks via relative distance and velocity measurement, *IEEE Commun. Lett.* 25 (9) (2021) 2899–2903.
- [32] U. Khan, K. Soumya, J. Moura, DILAND: an algorithm for distributed sensor network localization with noisy distance measurements, *IEEE Trans. Signal Process.* 58 (2010) 1940–1947.
- [33] A. Simonetto, G. Leus, Distributed maximum likelihood sensor network localization, *IEEE Trans. Signal Process.* 62 (6) (2014) 1424–1437.
- [34] S. Salari, I.-M. Kim, F. Chan, Distributed cooperative localization for mobile wireless sensor networks, *IEEE Wirel. Commun. Lett.* 7 (1) (2018) 18–21.
- [35] A. Beck, M. Teboulle, A fast Iterative Shrinkage-Thresholding Algorithm with application to wavelet-based image deblurring, in: 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, 2009, pp. 693–696.
- [36] N. Parikh, S. Boyd, Proximal algorithms, *Found. Trends Optim.* 1 (3) (2014) 127–239.
- [37] C. Soares, J. Xavier, J. Gomes, Simple and fast convex relaxation method for cooperative localization in sensor networks using range measurements, *IEEE Trans. Signal Process.* 63 (17) (2015) 4532–4543.
- [38] D.C. Montgomery, G.C. Runger, *Applied Statistics and Probability for Engineers*, John Wiley and Sons, Inc., 2003.
- [39] A. Banerjee, I.S. Dhillon, J. Ghosh, S. Sra, Clustering on the unit hypersphere using von Mises-Fisher distributions, *J. Mach. Learn. Res.* 6 (2005) 1345–1382.
- [40] C. Soares, J. Gomes, B.Q. Ferreira, J.P. Costeira, LocDyn: robust distributed localization for mobile underwater networks, *IEEE J. Ocean. Eng.* 42 (4) (2017) 1063–1074.
- [41] R.B. Wynn, et al., Autonomous Underwater Vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience, *Mar. Geol.* 352 (2014) 451–468, 50th Anniversary Special Issue.
- [42] E.M. Fischell, N.R. Rypkema, H. Schmidt, Relative autonomy and navigation for command and control of low-cost autonomous underwater vehicles, *IEEE Robot. Autom. Lett.* 4 (2) (2019) 1800–1806.
- [43] e.a. González-García, Autonomous underwater vehicles: Localization, navigation, and communication for collaborative missions, *Appl. Sci.* 10 (4) (2020) 1256.
- [44] C. Soares, P. Ji, J.P. Gomes, A. Pascoal, DiSEL: Distributed SELF-Localization of a network of underwater vehicles, 2017.
- [45] Y. Ben, Y. Sun, Q. Li, X. Zang, A novel cooperative navigation algorithm based on factor graph with cycles for AUVs, *Ocean Eng.* 241 (2021) 110024.
- [46] L. Zhang, S. Wu, C. Tang, Cooperative positioning of underwater unmanned vehicle clusters based on factor graphs, *Ocean Eng.* 287 (2023) 115854.
- [47] W.N. Anderson Jr., T.D. Morley, Eigenvalues of the Laplacian of a graph, *Linear Multilinear Algebra* 18 (2) (1985) 141–145.