ISEE.U: Distributed estimation and control for improved target localization accuracy

Miguel Araújo Vasques

***Abstract*—We propose two distributed algorithms to localize multiple static or moving targets using a network of agents, from range measurements, and to find the control for the network that increases the accuracy of localization of those targets. One of them is based on a tighter approximation, but is biased, while the other is unbiased. We named the latter as ISEE.U.**

In both algorithms, each agent evaluates its position in the network using a local estimate of the Fisher Information Matrix. We note that the FIM is decomposable through agents and, from this key fact, each agent can minimize the overall estimator's error ellipsoid volume and establish its next movement. Target positions estimates are also computed in the same distributed process. We build on two linear approximations to the target localization problem and we get, at each iteration of our consensus-like method, both a FIM estimate and a position estimate, from which we perform estimation and control, even for sparsely connected networks.

Simulation results showed that the ISEE.U estimate could be improved especially in the onset of the operation and, thus, we consider refinement methods to be used on the first few moments of the active localization procedure.

ISEE.U was then compared against a state-of-the-art method in different scenarios showing similar results and sometimes even outperforming the benchmark, and using x100 less computation time, even when ISEE.U is running in one central CPU.

***Index Terms*—Active estimation, distributed estimation, distributed control, target localization, Fisher Information Matrix.**

I. Introduction

The problem considered in this paper is how can a network of randomly deployed mobile agents accurately locate radiating sources and how can we control the movement of that network to better perform its task.

This localization problem can be solved using the *Global Positioning System* (GPS) but this is a restricted solution since GPS-denied environments, especially indoor scenarios, are frequent and not only the average person deals with the need for indoor localization everyday but also several different economic sectors have to deal with this problem at a much larger scale, involving money and time. Examples of these sectors are:

- Logistics;
- Security;
- Minerals and oil exploration;
- Navigation;
- Wireless communications;
- Surveillance.

A. Related work

Different methods found in the literature address this problem, some provide a general solution that can be used in a lot of different situations and others focus in more specific situations.

Most of the methods are based in the Kalman filter [1],[2],[3],[4], and in the Extended Kalman Filter [5],[6], other methods are based in Bayesian estimation [7],[8],[9], and other are based in *Maximum Likelihood* (ML) Estimators [10] that are less common than the previous methods but are more similar to the one presented in this work.

The method that is used as benchmark in this work is presented in [11]. This is a distributed method that uses Bayesian statistics to compute a belief that is used by a *Minimum Mean Square Error* (MMSE) estimator [12]. A particle filter is used to compute a sample-based approximation for sequential state estimation in this problem. The control of the network is computed using an information-seeking controller, which chooses to maximize the negative posterior joint entropy using a gradient ascent.

II. Overview of the Approach and Contributions

The setup of our solution comprises (i) a network of mobile circular agents with a finite *Field of View* (FoV), where the space where each agents can measure the position of the source and communicate with other agents is limited by a circle with finite range, and (ii) radiating sources which we will refer to as targets from now on. A visual representation of this setup is shown in Figure 1.

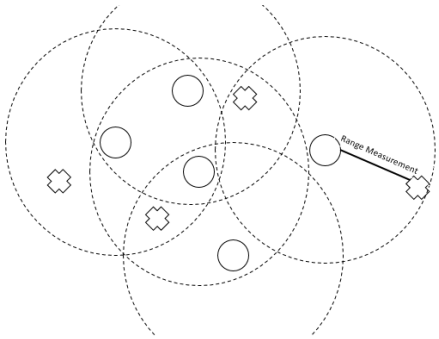


Figure 1: Setup for the considered solution. The circles represent the agents, the crosses represent the targets and the dashed lines are the agents' FoVs.

We will consider the localization of a radiating source to which we call target as an optimization problem. This optimization problem is formulated using a *Squared-Range-Based Least Squares* (SR-LS) approach [13] that, although being nonconvex, can be efficiently and globally solved.

We can write this problem using a linear model and use the distributed linear estimator called ISEE.U, which is a distributed approximation to the *Minimum Variance Unbiased* (MVU) estimator [12], to find an estimate for the position of the targets and the respective covariance matrices of those estimates. The ISEE.U estimator can be used by each agent to compute the position estimates through a consensus + innovations method where each agent exchanges information with its neighbors and uses it to compute the estimator, since this method approaches the centralized architecture of the problem. To the estimation phase of our method we call ISEE.Uest.

The covariance matrices can be used to design the control law for the movement of the network. From the eigenvalues of the covariance matrix, which is the inverse matrix of the *Fisher Information Matrix* (FIM), we can compute the volume of an error ellipsoid for that target. The objective is to minimize the volume of that error ellipsoid in order to improve the accuracy of estimation of the target position. So the next position of the agent will always be the one that minimizes the volume of the error ellipsoid. To the control phase of our method we call ISEE.Uctl.

The main contributions provided by this paper are:

- A distributed method to both estimate the target positions and control of the network agents;
- The distributed estimates and control are computed through a fast consensus + innovations scheme;
- A simple to implement, fast and flexible method demanding only range measurements to the target;
- A method that showed good performance when compared to a state-of-the-art algorithm using a particle filter approach.

Code available at https://github.com/migvas/ISEE.U.

This paper is structured as follows: in Section III a mathematical formulation of the problem is presented; in Section IV a mathematically detailed explanation of our proposed method is given; in Section V we present simulations that demonstrate the performance of our method ; in Section VI we draw conclusions about the work developed and propose some future work that can be done in order to make it more complete.

III. Problem Formulation

We define a network composed by $n(t)$ agents and $K(t)$ targets. The position of agent i at time t is given by $\mathbf{s}_i(t) \in \mathbb{R}^d$, where $d = 2$ or $d = 3$ is the dimension of the space considered, and $\mathbf{p}_k(t) \in \mathbb{R}^d$ represents the position of target $k \in \{1, \ldots, K(t)\}$ at a discrete time instant t.

Consider also a graph given by $G(t) = (V(t), E(t))$ called the communications graph which represents the network established by the communications between agents and where $V(t) = \{1, ..., n(t)\}$ is the graph's set of nodes, where each node represents one agent, and $E(t) = \{i \sim j \ : \ i, j \in V(t)\}$ is the set of edges. Each edge represents a communication link between two agents of the network.

Finally, consider a graph given by $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t))$, the measurements graph, which represents the network of range measurements of the agents relative to the targets. The nodes of the graph are given by $\mathcal{V}(t) = V(t) \cup \mathcal{T}(t)$, where $\mathcal{T}(t) = \{1, ..., K(t)\}$ and where each node represents one agent or target of the network. In this graph the edges are given by $\mathcal{E}(t) = \{i \sim k \ : \ i \in V(t), k \in \mathcal{T}(t)\}$ where each edge is a range measurement between one agent and one target.

The range measurement obtained by agent i to target k at time t is given by

$$r_{ik}(t) = ||\mathbf{s}_i(t) - \mathbf{p}_k(t)|| + w_{ik}(t) \tag{1}$$

where $w_{ik}(t) \sim \mathcal{N}(0, \sigma^2)$ is a term of white Gaussian noise with zero mean and standard deviation σ.

The data model is composed by the positions of the agents $\mathbf{s}_i(t)$ and by the respective range measurements $r_{ik}(t)$. The output is an estimate for the positions of the targets $\hat{\mathbf{p}}_k(t)$ and the control that defines the next positions for the agents $\mathbf{s}_i(t+1)$.

In order to give an easier and clearer mathematical explanation for the solution of the presented problem, a setup consisting of only one target and multiple agents will be considered in Section IV. To scale this method for more targets it is only necessary to change the dimensions of the matrices and repeat their entries.

IV. Proposed Method

The goal is to find the position of the target $\mathbf{p}(t)$ which minimizes the cost function given by the SR-LS as

$$\min_{\mathbf{p}(t) \in \mathbb{R}^d} \sum_{i=1}^{n} (||\mathbf{s}_i(t) - \mathbf{p}(t)||^2 - \mathbf{r}_i^2(t))^2. \tag{2}$$

This problem is nonconvex and so a solution for it is very difficult or even impossible to find. The only option is trying to find a similar convex optimization problem that can provide an accurate approximation to the desired solution of the SR-LS problem.

To clarify the mathematical expressions presented in the rest of the chapter we will not write the time dependence for some variables.

To find this convex optimization problem we manipulated equation (1) and obtained a linear model for it given by

$$\mathbf{y} = \mathbf{A}\mathbf{x} + 2\mathbf{\Xi}\mathbf{w} + (\mathbf{w} \circ \mathbf{w}) \tag{3}$$

where

$$\mathbf{y} = \begin{bmatrix} r_1(t)^2 - ||\mathbf{s}(t)_1||^2 \\ \vdots \\ r_n(t)^2 - ||\mathbf{s}(t)_n||^2 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} 1 & -2\mathbf{s}_1(t)^T \\ \vdots & \vdots \\ 1 & -2\mathbf{s}_n(t)^T \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} ||\mathbf{p}(t)||^2 \\ \mathbf{p}(t) \end{bmatrix}. \tag{4}$$

and $(\mathbf{w} \circ \mathbf{w})$ is the Hadamard product between the same noise vector given by

$$(\mathbf{w} \circ \mathbf{w}) = \begin{bmatrix} w_1^2 \\ \vdots \\ w_n^2 \end{bmatrix} \tag{5}$$

and

$$\mathbf{\Xi} = \begin{bmatrix} ||\mathbf{s}_1 - \mathbf{p}|| & & 0 \\ & \ddots & \\ 0 & & ||\mathbf{s}_n - \mathbf{p}|| \end{bmatrix}. \tag{6}$$

The standard deviation of the noise for each agent was defined considering the influence of the distance in it. So we defined $\sigma_i = \beta||\mathbf{s}_i - \mathbf{p}||\hat{\sigma}$, where $\hat{\sigma}$ is the default standard deviation for the agents, β is a scalar variable that dictates the influence of the distance on the noise and $\beta\hat{\sigma}$ is considered a dimensionless quantity.

The linear model obtained still does not turn this problem into a convex one. As seen in [13] and [14] one possible way to make this problem convex is to neglect the dependence of $x_1 = ||\mathbf{p}||$ on the other elements of the solution vector $\mathbf{x}$. So, considering the mentioned approach, we have a convex problem and there are multiple methods to solve it. We found two methods to compute an estimator $\hat{\mathbf{x}}$ that make equation (3) true.

A. Method 1

The first solution is to see that the term $(\mathbf{w} \circ \mathbf{w})$ considers that the noise term in the measurements is squared and considering that the noise is a small quantity this term will be much smaller than the other noise term that is multiplied by the distance between the agents and targets. So we neglect the squared noise term and obtain the following linear model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + 2\mathbf{\Xi}\mathbf{w}. \tag{7}$$

Now it is possible to apply the MVU linear estimator to solve the problem presented in (7), which is given by

$$\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{C}_\mathbf{y}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{C}_\mathbf{y}^{-1}\mathbf{y} \tag{8}$$

$$\mathbf{C}_{\hat{\mathbf{x}}} = (\mathbf{A}^T\mathbf{C}_\mathbf{y}^{-1}\mathbf{A})^{-1}, \tag{9}$$

where $\mathbf{C}_{\hat{\mathbf{x}}}$ is the covariance matrix of the estimator and considering the inverse of the covariance matrix of the range measurements $\mathbf{C}_\mathbf{y}$ given by

$$\mathbf{C}_\mathbf{y}^{-1} = \begin{bmatrix} \frac{1}{4||\mathbf{s}_1 - \mathbf{p}||^2\sigma_1^2} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{4||\mathbf{s}_n - \mathbf{p}||^2\sigma_n^2}. \end{bmatrix} \tag{10}$$

Since in the considered approximation $\mathbf{y}$ is a normally distributed vector and the estimator is just a linear transformation of it, the statistical performance of $\hat{\mathbf{x}}$ is completely specified and given by

$$\hat{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \mathbf{C}_{\hat{\mathbf{x}}}), \tag{11}$$

where $\hat{\mathbf{x}}$ is a $(d+1) \times 1$ vector and $\mathbf{C}_{\hat{\mathbf{x}}}$ is a $(d+1) \times (d+1)$ matrix. For example, when working in $\mathbb{R}^2$ their sizes become 3×1 and 3×3 respectively.

B. Method 2

The alternative approach is to consider the linear model presented in (3), where the noise term given by $(\mathbf{w} \circ \mathbf{w})$ is not Gaussian, meaning that its expected value is not zero, influencing the problem as a bias, and using a linear estimator to solve it. Since this estimator is biased it does not guarantee that minimum variance is achievable. The covariance matrix $\mathbf{C}_\mathbf{y}$ for this kind of estimator will be slightly different from the one presented in (10).

The inverse of the covariance matrix of the measurements is now given by

$$\mathbf{C}_\mathbf{y}^{-1} = \begin{bmatrix} \frac{1}{4||\mathbf{s}_1 - \mathbf{p}||^2\sigma_1^2 + 2\sigma_1^4} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{4||\mathbf{s}_n - \mathbf{p}||^2\sigma_n^2 + 2\sigma_n^4}. \end{bmatrix} \tag{12}$$

The estimator and its covariance matrix are again given by equations (8) and (9), respectively and the bias is given by

$$E[\hat{\mathbf{x}}] - \mathbf{x} = (\mathbf{A}^T\mathbf{C}_\mathbf{y}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{C}_\mathbf{y}^{-1}\mathbf{\Psi}, \tag{13}$$

where

$$\mathbf{\Psi} = \begin{bmatrix} \sigma_1^2 \\ \vdots \\ \sigma_n^2 \end{bmatrix}. \tag{14}$$

The estimator is given by a distribution which results from a mixture between the Gaussian distribution from the first noise term and a chi-squared distribution from the second, with mean $\mathbf{x} + (\mathbf{A}^T\mathbf{C}_\mathbf{y}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{C}_\mathbf{y}^{-1}\mathbf{\Psi}$ and variance $\mathbf{C}_{\hat{\mathbf{x}}}$.

So it is possible to conclude that the estimates will always be biased but if the bias is small enough they can approximately be centered in the true solution.

Since the first presented method uses a well known estimator and is much simpler, it is easier to find a distributed centered estimator from it and so we chose it to be the main method of this work.

Going back to the first method presented in IV-A, the MVU linear estimator is a centralized method since the estimator is calculated with full knowledge of the measurements and positions of all agents, contained in $\mathbf{y}$ and $\mathbf{A}$ respectively.

In Section II it was said that one of the major properties of this method is distributed computation, so a new distributed approximation to the MVU linear estimator, to which we call ISEE.U estimator, is presented. With this method, each agent will be able to compute an approximation to the MVU linear estimator and the covariance matrix for the estimator by itself using only data from his neighbors.

By inspecting the centralized MVU linear estimator is possible to conclude that the estimator and the covariance matrix can both be computed using a sum of the contribution from every agent as

$$\hat{\mathbf{x}} = \left(\sum_{i=1}^{n} \mathbf{A}_i^T C_{\mathbf{y}_{ii}}^{-1} \mathbf{A}_i \right)^{-1} \sum_{i=1}^{n} \mathbf{A}_i^T C_{\mathbf{y}_{ii}}^{-1} y_i \tag{15}$$

$$\mathbf{C}_{\hat{\mathbf{x}}} = \left(\sum_{i=1}^{n} \mathbf{A}_i^T C_{\mathbf{y}_{ii}}^{-1} \mathbf{A}_i \right)^{-1}, \tag{16}$$

where each agent only has to provide its own line of the matrix $\mathbf{A}$, its own element of the matrix $\mathbf{C_y}$ and its own range measurement from $\mathbf{y}$.

Now, as in [15] we will define the matrix $\boldsymbol{\mathcal{P}}_i(\tau)$ and the vector $\mathbf{z}_i(\tau)$ which are both calculated by agent i at consensus iteration τ and are used to compute the estimator and the covariance matrix for that agent. We have

$$\hat{\mathbf{x}}_i(\tau) = \boldsymbol{\mathcal{P}}_i^{-1}(\tau)\mathbf{z}_i(\tau) \tag{17}$$

$$\mathbf{C}_{\hat{\mathbf{x}}_i(\tau)} = \boldsymbol{\mathcal{P}}_i^{-1}(\tau) \tag{18}$$

where we define

$$\boldsymbol{\mathcal{P}}_i(\tau+1) = \frac{\tau}{\tau+1} \sum_{j \in \mathcal{N}_i} \mathcal{W}_{ij} \boldsymbol{\mathcal{P}}_j(\tau) + \frac{1}{\tau+1} \sum_{j \in \mathcal{N}_i} \mathbf{A}_j^T C_{\mathbf{y}_{jj}}^{-1} \mathbf{A}_j \tag{19}$$

$$\mathbf{z}_i(\tau+1) = \frac{\tau}{\tau+1} \sum_{j \in \mathcal{N}_i} \mathcal{W}_{ij} \mathbf{z}_j(\tau) + \frac{1}{\tau+1} \sum_{j \in \mathcal{N}_i} \mathbf{A}_j^T C_{\mathbf{y}_{jj}}^{-1} y_j(\tau+1) \tag{20}$$

where $\boldsymbol{\mathcal{P}}$ is a $(d+1) \times (d+1)$ matrix and $\mathbf{z}$ is a $(d+1) \times 1$ vector, $\mathcal{N}_i$ represents the set of neighbors of agent i and i itself and where $\boldsymbol{\mathcal{W}}$ is the so called weight matrix. Each agent has only its own line of the matrix, i.e., agent i only knows $\boldsymbol{\mathcal{W}}_i$, and this line contains the weights given by the said agent to the agents with which it communicates, based on the communications graph $G(t)$, and to itself. The entries of the line that each agent has, have to add up to 1. Thus, W is a stochastic matrix. Normally the weights given by one agent to all the other communicating agents and to itself are equal, and so, the agent only has to know how many agents are communicating with it and assign the inverse of that number to each of the weights. In the case where, for example, one of the agents is unreliable it is possible to rebalance the weights according to the information provided by each neighbor.

In equation (20) it is possible to see that $\mathbf{y}$ changes at every iteration in the consensus phase. This happens because at every iteration every agent collects new measurements that are introduced in the consensus to reduce the variance of the measurements and improve estimation. To introduce new data in the consensus process is called consensus + innovations. A key novelty of our iterations is not only to add innovations for the $\mathbf{z}_i$ from measurements collected at i, but also from neighboring nodes. Similarly, the second term of the $\boldsymbol{\mathcal{P}}_i$ recursion is not the traditional consensus nor the traditional consensus + innovations. This, as we will see, will give our method a very clear advantage in mean error of the distributed quantities, for finite time.

By inspecting equations (17) and (18) one may mistakenly think that the second sum in equations (19) and (20) is enough to compute $\boldsymbol{\mathcal{P}}_i(\tau+1)$ and $\mathbf{z}_i(\tau+1)$. This is true if and only if the graph $G(t)$ is fully connected, meaning that every agent of the network communicates with every other, and so the second sum has information from all agents, needed to compute a good approximation for the estimator and its covariance matix. When $G(t)$ is not fully connected each agent has to find a way to get the needed information from the agents with which it does not communicate. This propagation of information via consensus happens in the first sum since both $\boldsymbol{\mathcal{P}}_j(\tau+1)$ and $\mathbf{z}_j(\tau+1)$ have information from agents that do not communicate with agent i.

The main difference between our method and the one presented in [15] lies in the introduction of the information of the neighbors in the second sum of equations (19) and (20). The two fractions before the sums in those equations choose which sum gives the largest contribution to the computation of $\boldsymbol{\mathcal{P}}_i(\tau+1)$ and $\mathbf{z}_i(\tau+1)$. In the initial iterations the second sum has more relevance in order to initialize both quantities using the information that an agent can gather from its neighbors and from itself. When $\tau \gg 0$ the first sum becomes dominant since both quantities already have the information from the agent and its neighbors and we want to diffuse that information to the rest of the network to reach a generalized estimator and covariance matrix.

This estimator is not efficient but preliminary simulations show, for finite consensus rounds, that it is faster than other efficient estimators, like the consensus + innovations one. In conclusion, this section contains one of the major contributions from this work since we presented a distributed method where, at each iteration, each agent computes an unbiased estimate using ISEE.U. Further analysis is required to rebalance the variance of the estimator and better assess the finite and asymptotic behavior of $\tau \mathrm{var}(\mathbf{x}_i(\tau))$.

For the control phase ISEE.Uctl, we choose to move the network so the accuracy of estimation of the target position is maximized. We consider a random deployment of the network's nodes and at each iteration each agent moves to a position where the localization error is minimized. In order to do so we will use the covariance matrix of the estimator for each agent $\mathbf{C}_{\hat{\mathbf{x}}_i}$ defined in (18), also called precision matrix, from which we can formulate a cost function. Many authors choose to use this same approach but using the *Fisher Information Matrix* (FIM), which is basically the inverse matrix of the covariance matrix.

We chose the *D-optimality* [16] criterion to solve the minimization problem since it considers an error ellipsoid, where each of its axes is obtained using the eigenvalues from the covariance matrix and tries to minimize its volume, meaning that it tries to minimize the length of all axes which is the same as minimizing all the eigenvalues of the covariance matrix. To compute the cost function based on this criterion we have to know how to convert the value of the eigenvalues from the covariance matrix into the length of the axes of the ellipsoid. This is done by using

$$l_j = \sqrt{\chi^2_{(d+1),\theta} \lambda_j} \tag{21}$$

where l_j is half of the length of the axis associated with the eigenvalue λ_j. The $\chi^2_{(d+1),\theta}$ is the value for the chi-squared distribution with $d+1$ degrees of freedom and a certain confidence level.

Then, the volume of the ellipsoid is given by

$$\begin{aligned} V_{2\epsilon} &= \frac{\pi^\epsilon}{\epsilon!}\prod_{j=1}^{2\epsilon} l_j \\ V_{2\epsilon+1} &= \frac{2(\epsilon!)(4\pi)^\epsilon}{(2\epsilon+1)!}\prod_{j=1}^{2\epsilon+1} l_j \end{aligned} \tag{22}$$

where 2ϵ or $2\epsilon+1$ is the dimension of the ellipsoid. This dimension should be equal to the dimension of the covariance matrix of the estimator since the number of eigenvalues of this matrix is equal to its dimension.

Considering that the vector of solutions $\mathbf{x}$ has always one more dimension than the space we are working in and so the ellipsoid has that one extra dimension too. So, if our deployment area is in $\mathbb{R}^2$, the ellipsoid will be in $\mathbb{R}^3$.

The procedure used by agent i to compute the motion control is the following:

1) Compute the covariance matrix $\mathbf{C}_{\hat{\mathbf{x}}}$ through consensus;
2) Compute the volume for the current position of the network;
3) Subtract the agent contribution $(\mathbf{A}_i^T C_{\mathbf{y}_{ii}}^{-1}\mathbf{A}_i)^{-1}$ from the covariance matrix and save the resulting matrix;
4) Choose the new position;
5) Compute $(\mathbf{A}_i^T C_{\mathbf{y}_{ii}}^{-1}\mathbf{A}_i)^{-1}$ for the new position;
6) Add its contribution to the resulting matrix from step 3;
7) Compute the new volume;
8) Compare the two volumes, if the new volume is smaller then move the agent to the new position, if it is larger then keep the agent in its old position.

To make this solution realistic we discretize the deployment area. When an agent wakes up and wants to move, it follows the presented procedure from step 4 for all the eight possible positions around its current one. This is a greedy algorithm since each agent makes the locally optimal choice every time it wants to move, hoping that in the end the network can reach a global minimum.

In conclusion we have now presented a way for the agents to move and reconfigure the network, where the control for the movement is computed considering the maximization of the estimation accuracy of the target position.

V. Numerical Results

We show the performance of ISEE.U by comparing it with different methods and also in different scenarios. In Sections IV-A and IV-B we presented the MVU linear estimator and a biased linear estimator, respectively, to solve our problem. We experimentally compared ISEE.U and the biased linear estimator so that we could evaluate the quality of the extra approximation that had to be done so that ISEE.U could be used.

Then, we compared ISEE.U with efficient distributed estimators to demonstrate how this non efficient estimator fares against efficient state-of-the-art ones.

A comparison between ISEE.U and two refinement methods is also presented to evaluate experimentally the performance of our method.

Finally, the performance of our method will be compared with a state-of-the-art method that we chose as benchmark, presented in I, in different scenarios.

In these comparisons, Monte Carlo simulations will be performed, which consists in repeating the same experiment, with random components, multiple times, and the methods will be compared by the *Mean Absolute Error* (MAE) given by

$$\mathrm{MAE}(t) = \frac{\sum_{j=1}^{M} ||\mathbf{p}(t) - \hat{\mathbf{p}}_j(t)||}{M} \tag{23}$$

where $\mathbf{p}(t)$ is the real target position at time t, $\hat{\mathbf{p}}_j(t)$ is the estimate of the target position in run j and also at time t and M is the number of Monte Carlo trials performed.

A. Comparison between the two methods

We present a numerical comparison between the two methods to estimate the target position: the ISEE.U estimator which comes from the method presented in Section IV-A and the biased linear estimator presented in Section IV-B.

To test both method we started with the setup presented in Figure 2. The space used is a 10×10 square and the network is formed by 4 agents and their initial positions was chosen to be far away from a static target so it was possible to see the error of the estimates for different quantities of noise since it depends on the distance between the agents and the target. We ran 100 iterations of our algorithm where, in each iteration, one agent is randomly chosen to wake up and through consensus choose which is its next best position and move (or not) to that chosen position. This means that the minimum number of iterations that the network needs to move is 4 and that the network can move up to 25 times. This is more than enough for the agents to reach their absolute best position given the size of the space. We performed each run of 100 iterations 20 times and computed the MAE. The communication range of each agent was defined as 55% of the length of the diagonal of the square. This means that if two agents are at a distance lower than 55% of the length of the diagonal of the square of each other they communicate. The noise's standard deviation of each agent was $\beta\hat{\sigma} = 0.001$ of the distance between the agent and the target and the number of consensus iterations was defined as $T = 20$.

In each iteration a random agent wakes up and moves to the best position if it is not the one where it is. Then, after the consensus, it computes an estimate for the target position with the ISEE.U estimator and the biased linear estimator. Each estimate is then compared with the real position of the target.

It is possible to see in Figure 3 that the biased estimator outperforms the unbiased. This can be explained by the fact that, although both estimators already consider an approximation of the problem since the dependency of the first position of the solution vector with the two other positions was neglected, ISEE.U considers also another approximation of the problem to remove the bias from the solution, considering the linear

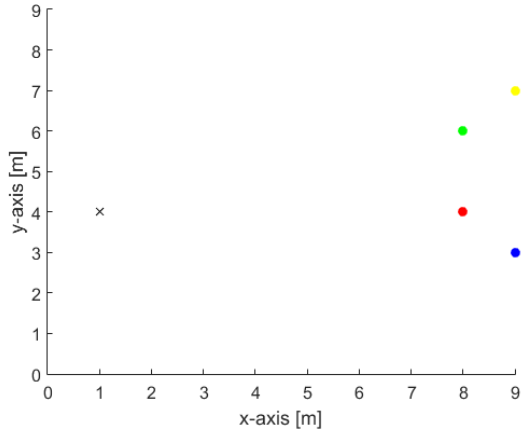


Figure 2: Initial setup. The circles represent the agents, the black cross is the target and the red cross is the estimate for the target position.

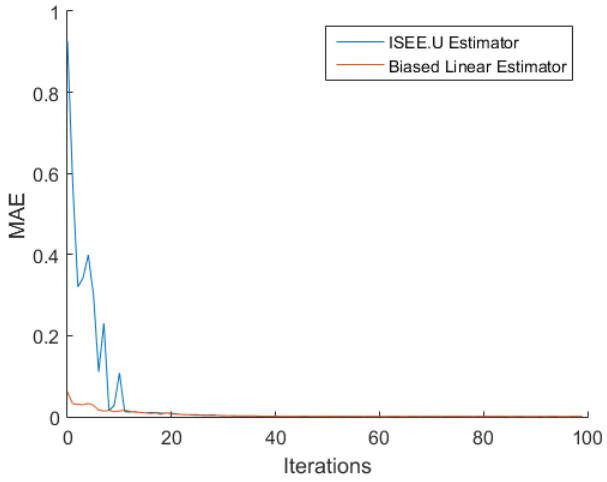


Figure 3: MAE for ISEE.U and for the biased linear estimator for different noise factors. The biased linear estimator outperforms ISEE.U in the first iterations. After 15 iterations their performance is almost equal.

model presented in equation (7). After another approximation is normal that the results turn out to be worse but the ISEE.U estimator has the advantage of being unbiased so it should not be outperformed. However, the bias was found to be very small when compared to the error so its influence in the estimation is very small, which still makes the biased linear estimator produce excellent estimates for the original linear model presented in equation (3).

Other explanation can be the fact that the movement of the agents is based on the covariance matrix of the estimator and the bias does not affect the covariance matrix that results from the linear estimator. This means that the difference between the covariance matrix of both estimators is only based on the covariance matrix of the measurements which is given by 10 for ISEE.U and by 12 for the biased linear estimator. Since the covariance matrix of the measurements used in the biased linear estimator is computed from the model without the extra approximation it describes it better than the other matrix, so the control of the agents with the biased linear estimator is better than the one with ISEE.Uctl and this control can have influence on the position estimates.

In conclusion, we can say that both estimators produce good estimates for the target position but, although being biased, the biased linear estimator outperforms ISEE.U. However, the biased linear estimator is also more influenced by noise so in some cases it may not provide the best estimates.

B. Distributed Estimators

Now we present the performance of our method against efficient state-of-the-art distributed estimators. These methods are the basic consensus method, the traditional consensus + innovations method and a modified consensus + innovations method where the terms of the second sum in equations (19) and (20) were also multiplied by the corresponding weights.

For this experiment we considered a network of 100 agents randomly deployed in a 100×100 square. The communication range was lowered, defined as 30% of the length of the diagonal of the square, the noise's standard deviation of each agent was again $\beta\hat{\sigma} = 0.001$ of the distance and only one Monte Carlo trial was performed. We computed the error between $\boldsymbol{\mathcal{P}}$ obtained for each method and the ML one for each agent of the network, given by $||\boldsymbol{\mathcal{P}}_{\text{method},i} - \boldsymbol{\mathcal{P}}_{ML}||$ and computed the empirical *Cumulative Distribution Function* (CDF). To better understand the results the error was normalized, which is obtained dividing the error by the norm of $\boldsymbol{\mathcal{P}}_{ML}$. The same test was done for $\mathbf{z}$.

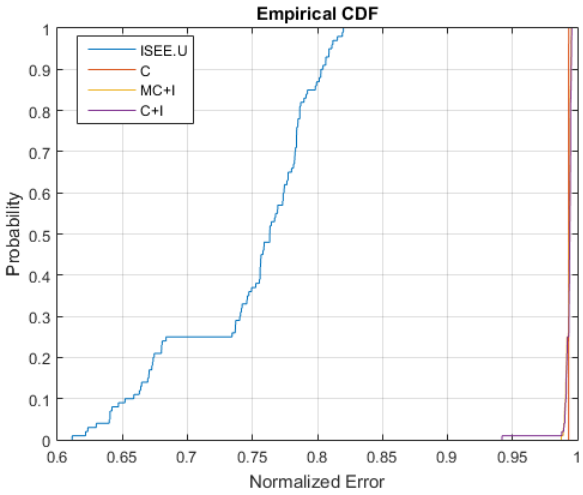


Figure 4: Empirical CDF of the normalized error for $\boldsymbol{\mathcal{P}}$ for ISEE.U and multiple efficient estimators: consensus (C), consensus + innovations (C+I) and modified consensus + innovations (MC+I) for 20 consensus rounds. ISEE.U outperforms the other estimators.

From Figures 4 and 5 it is possible to conclude that ISEE.U clearly outperforms the other state-of-the-art estimators because it is much more probable for ISEE.U to obtain a much smaller normalized error than the other estimators, supporting the claims made in Section IV where it was said that ISEE.U had a very clear advantage in mean error of the distributed quantities.

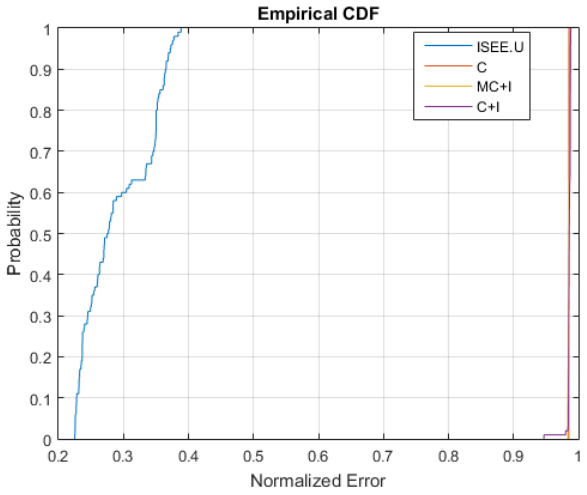


Figure 5: Empirical CDF of the normalized error for **z** for ISEE.U and multiple efficient estimators: consensus (C), consensus + innovations (C+I) and modified consensus + innovations (MC+I) for 20 consensus rounds. ISEE.U outperforms the other estimators.

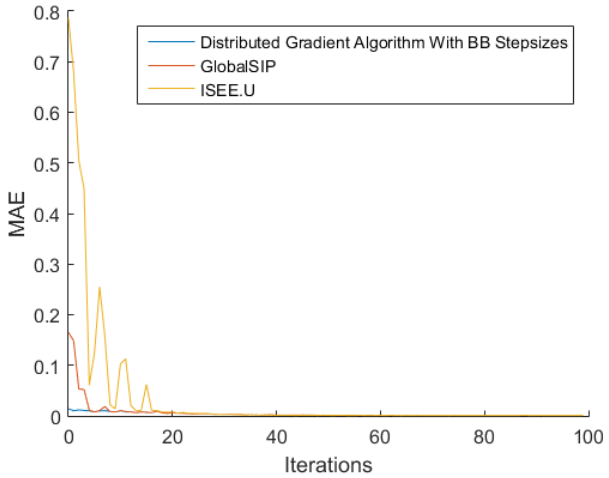


Figure 6: MAE for ISEE.U and for the two refinement methods presented in this section. The refinement methods outperform ISEE.U in the first iterations but the three methods start producing identical estimates after 20 iterations.

C. Refinement

To try to produce better estimates for all positions of the network we implemented a refinement phase after the target position estimation. This phase takes the current position estimate and tries to improve it. To do so, we use adaptations of two state-of-the-art methods: Distributed Gradient Algorithm With BB Stepsizes [17] and Distributed ML agent network localization (GlobalSIP) [18].

Both methods are used for self localization of agents in a network of agents and anchors that have full knowledge of their positions. We applied these methods to perform the self localization of the target using the agents of the network as anchors. So we end up with just one node that has to locate itself. Both methods solve a minimization problem using the gradient descent method [19] with *Barzilai-Borwein* stepsizes [20],[21] and the projected gradient method [22], respectively, using as initial estimate the ISEE.U estimate.

The experiment consists in measuring the accuracy of the estimates of the target position relative to the real one. To do so, the same setup as in Section V-A is used where in each iteration the randomly selected agent moves to the best position and through consensus computes an estimate using ISEE.U. This estimate is then used as initial estimate for the refinement methods. Finally, we compute the MAE for each of the three computed estimates.

The results of this experiment can be observed in Figure 6. In this figure it is possible to see that the refinement process improves the estimation mostly in the first iteration, where the agents are further away from the target and the measures have a bigger noise component. After approximately 20 iterations, the network is already close enough to the target to compute very accurate estimates through the ISEE.U estimator, since the three methods produce almost equal results. One can also notice that for some iterations the refinement even produces worst estimates than the initial one. The cause for this phenomenon is the fact that the cost function is parametrized by the actual noisy range measurements, and the shape and extreme of the ML function will change with different instances of these measurements. Thus, the global minimum of the ML cost will not coincide with the true positions, but will be at a short distance from them. If our initialization is very good we can even see the localization error growing when the cost is declining.

Finally it is also important to compare the behavior of the two refinement methods. In Figure 6 we can see that initially the refinement methods produce different errors from each other being the GlobalSIP the one that produces the worst results. After some iterations it becomes very hard to distinguish the behavior of the two since they produce almost equal estimates. The difference between the MAE of the two methods may be related to the number of communications made between the agents and target in each method since the number of communications performed by the agents in the Distributed Gradient Algorithm With BB Stepsizes method is much bigger than the one for the GlobalSIP method.

Although the refinement process is very useful to improve our estimate in certain cases, the two presented methods cannot be implemented on our method directly. This is because the refinement methods assume that the target is a cooperative agent, that can communicate and exchange information with agents, and it can make computations since these methods assume that the target tries to locate itself. For those reasons the refinement methods cannot be implemented and can only be used to validate the estimate that comes from the ISEE.U estimator. Since we concluded that the refinement methods were only useful for certain positions of the network we can say that our estimator already provides accurate estimates.

D. Simulations with a state-of-the-art method

We will now provide a simulation made with ISEE.U and the benchmark, already presented in Section I, in order to draw some comparisons between the two and to numerically

validate the method presented in this work against a published one.

We will start by defining the different parameters that were used in the simulation. The space that was considered was a 200×200 square and thus our grid now has 40000 possible positions for the agents. We had to consider a bigger space than the one used in the previous experiments because the benchmark method uses samples to represent the belief of the target position and a small grid could not represent a big number of samples. The standard deviation for the noise continues to depend on the distance and $\beta\hat{\sigma} = 0.1$ and was applied to both methods which implied some changes in the benchmark since it only consider a single constant standard deviation for all agents or a standard deviation that used a model slightly different from ours that considered a pathloss factor.

For ISEE.U we continued to apply a communication range to each agent where its radius is again 55% of the length of the diagonal of the space. For the comparison with the benchmark the motion of the network in this method was also changed. Previously, in each iteration one agent randomly woke up and move to the best position. Now in each iteration of the algorithm every agent wakes up and moves sequentially, so in every iteration of the simulation all the agents of the network have the opportunity to move. After the movement of each agent an estimate for the target position for that agent is computed through consensus.

For the benchmark and following the Simulations Section presented in the corresponding paper, $120,000$ samples were used in the estimation layer and $6,000$ samples in the control layer.

For a defined number of iterations we ran both algorithms at the same time, and each algorithm produced an estimate for the target position and the control for all agents of the network for each run. This process was repeated 5 times so that the target estimate from both methods could be used to compute the MAE of estimation

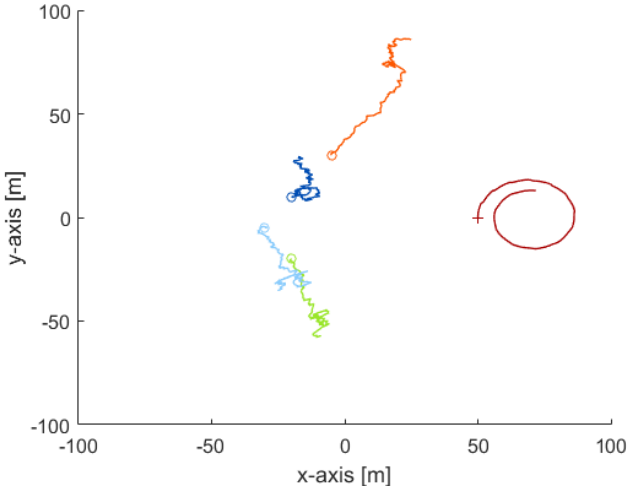
The simulation consisted in the usual network of four agents and one mobile target. The agents were deployed in random positions and the simulation ran for 150 iterations since this was enough to analyze the trajectories that result from both methods and the evolution of the estimation error. In each run the target state at some iteration t, $\mathbf{x}_{\text{target}}(t) = [p_1(t)\ p_2(t)\ \dot{p}_1(t)\ \dot{p}_2(t)]^T$ changes according to

$$\mathbf{x}_{\text{target}}(t) = \mathbf{x}_{\text{target}}(t-1) + \mathbf{\Gamma}(t) + \Lambda\mathbf{q}(t), \tag{24}$$

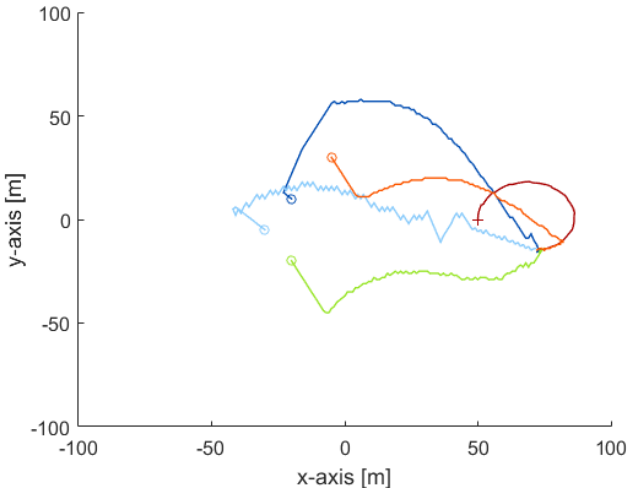
where $\mathbf{\Gamma}(t)$ is

$$\mathbf{\Gamma}(t) = \begin{bmatrix} R_t(t) * cos(\pi - \frac{v}{R(t)} * t) + c_1 \\ R_t(t) * sin(\pi - \frac{v}{R} * t) + c_2 \\ 0 \\ 0 \end{bmatrix}, \tag{25}$$

$\mathbf{q}(t) \in \mathbb{R}^2$ where $q_i(t) \sim \mathcal{N}(0, \tilde{\sigma}_q^2)$ is statistically characterized by a Gaussian distribution with zero mean and variance $\tilde{\sigma}_q^2 = 10^{-5}$ and where v is the linear velocity of the target that was kept constant and defined as 70% of the velocity of the group, $R_t(t)$ is the radius of the spiral that was decreased every 10 iterations according to $R_t(t) = 0.97R_t(t-1)$, where we defined $R_t(0) = 20$, and $\mathbf{c} = [c_1\ \ c_2]^T$ are constants that define the center of the spiral, since the coordinates of the center of the spiral are given by $(c_1 + p_1(0), c_2 + p_2(0))$, and chosen as $\mathbf{c} = [20\ \ 0]^T$. So the target will start at its normal starting position and will move clockwise in a spiral trajectory centered in $(70, 0)$.



(a) Benchmark.



(b) ISEE.U.

Figure 7: Obtained trajectories for the simulation for both methods. The trajectories are represented by lines having the same color of the circle of the agent that they correspond. The red line represents the trajectory of the target. The agents in **(a)** spread and loose track of the target as it can be seen by the absence of samples (represented by red dots), while in **(b)** the agents pursue and trap the target.

In the Figures 7 it is possible to observe the differences between the trajectories obtained for both methods, which were different because both methods used different cost functions to compute the respective control.

For the benchmark, we can see in Figure 7a that the agents start approaching the target and when they reach a distance from which it is not possible to decrease the measurement noise, the agents spread out to find a formation more suitable

to locate and track the target. The belief also should be represented in the form of samples but they are not in the graph due to the fact that when the target is moving during the first semi-circle the benchmark assumes that the target has a linear trajectory so when its movement abruptly changes the agents still assume that the target is moving in a linear trajectory and continue to produce estimates until reaching the boundary of the space. This can also explain the fact that the agents start to move in the direction of the target but since they loose it and start computing wrong estimates, they start spreading to try to minimize that error.

With ISEE.U, as seen in Figure 7b the agents move in the direction of the target and eventually end up trapping it. This is because the cost function used to compute the control is the volume of the error ellipsoid, which is a function of the covariance matrix of the estimator from equation (9), depends on the distance between the agents and the target via the covariance of the measurement data represented in equation (10). As the error is dependent on the distance, agents try to minimize their distance to the target.

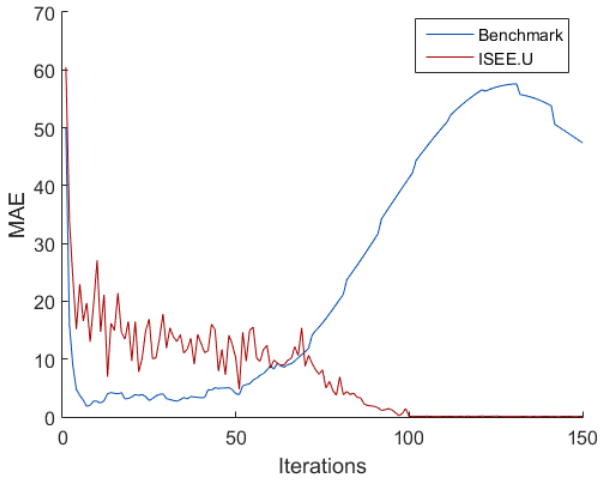


Figure 8: MAE obtained for both methods in the simulation. ISEE.U outperforms the benchmark after approximately 60 iterations. After 50 iterations the MAE for the benchmark increases.

In Figure 8 we can see that, after approximately 10 iterations, the benchmark reaches a low MAE and stays around the same value until approximately the 50-th iteration, where the MAE starts increasing. This coincides with the moment when the movement of the target abruptly changes and so the estimates start to be further away from the real target position and so the estimation error increases. ISEE.U needs more iterations to reach a low MAE but after after approximately 60 iterations it reaches the same value as the benchmark but continues decreasing until reaching a MAE very close to 0, outperforming the benchmark in these iterations.

We can conclude that the method presented in this work showed a very satisfactory performance when compared to a state-of-the-art method, supporting the claim that it is a good solution for the problem presented in Section I.

VI. Conclusion

In this work we proposed two methods to address the problems of estimating multiple target positions using a network of mobile agents in a distributed way, and of computing the control for the movement of the network in order to increase the estimation accuracy. As a byproduct, we developed a consensus + innovations scheme that achieves more precision that the known consensus and consensus + innovations algorithms, for the same number of consensus rounds, in our simulations.

These methods provide approximate solutions for a nonconvex least squares optimization problem by transforming it into a linear problem: one method uses a biased linear estimator to solve it while the other uses the unbiased linear estimator to solve another approximation to the problem.

Our ISEE.U estimator, based on the unbiased linear estimator, is a new consensus + innovations recursion that we proved to be centered. Our ISEE.U estimator, despite not being efficient, could achieve, in our simulations, more precision than the efficient traditional consensus and consensus + innovations for the same number of consensus rounds.

The control takes advantage of two key facts that have gone unnoticed until now: (i) in our distributed scheme every agent has access to an estimate of the covariance matrix for the overall estimator, (ii) the overall covariance can be seen as a sum of each agent's contributions. These two facts imply that an agent can take its estimate of the covariance for the whole network and test which future movement will minimize the uncertainty in the estimation.

When comparing the developed ISEE.U estimator with a state-of-the-art solution under different scenarios, numerical results indicate that, for a static or linear motion target, the benchmark increases localization precision at a faster pace in the beginning of the trajectory, but that ISEE.U catches up and even increases localization precision when compared with the benchmark. For more varied trajectories the results are even more interesting: ISEE.U clearly outperforms the benchmark in localization precision. The running times are also very competitive: ISEE.U takes a few seconds while the benckmark — a solution based on a particle filter — can run for hours.

In conclusion, we were able to find a flexible method that is easy to implement and adapt to very different scenarios and that can still compete in terms of performance with a much more complex state-of-the-art method.

Throughout this work some topics were left unaddressed and they provide great starting points for possible future work.

The first topic is to use the biased linear estimator to solve this problem. ISEE.U and the biased linear estimator were compared and it was possible to conclude that both could provide estimates with similar errors but the bias estimator still outperformed ISEE.U since it considered a better approximation to the original problem. Although this is the best method to solve this problem, it is more complex and a deeper analysis should be made as future work.

The second topic was already mentioned in Section III and is finding an efficient estimator based on ISEE.U which could perform better than the one presented in this work.

The final topic is integrating the refinement methods in our solution. As it was presented in Section V-C the refinement methods improved significantly the estimate for the target position in the first iterations of the run. Unfortunately, the two methods assume that the target is a cooperative agent that can receive data from the network and adjust the initial estimate given by the ISEE.U estimator to obtain a better estimate. In our setting it is not possible since the target is a non cooperative entity from which we have no information besides the noisy range. To solve this problem we need to find a way to try to substitute the target computations when estimating the target position using, for example, a consensus scheme.

REFERENCES

[1] R. Olfati-Saber and P. Jalalkamali, "Coupled distributed estimation and control for mobile sensor networks," *IEEE Transactions on Automatic Control*, vol. 57, no. 10, pp. 2609–2614, 2012.

[2] H. J. Rad, T. Van Waterschoot, and G. Leus, "Cooperative localization using efficient kalman filtering for mobile wireless sensor networks," in *Signal Processing Conference, 2011 19th European*. IEEE, 2011, pp. 1984–1988.

[3] Z. Liu, W. Chen, J. Wang, and H. Wang, "Action selection for active and cooperative global localization based on localizability estimation," in *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1012–1018.

[4] F. Morbidi and G. L. Mariottini, "Active target tracking and cooperative localization for teams of aerial vehicles," *IEEE transactions on control systems technology*, vol. 21, no. 5, pp. 1694–1707, 2013.

[5] S. Martínez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, no. 4, pp. 661–668, 2006.

[6] H. Wei, W. Lu, P. Zhu, G. Huang, J. Leonard, and S. Ferrari, "Optimized visibility motion planning for target tracking and localization," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 76–82.

[7] J. Banfi, J. Guzzi, A. Giusti, L. Gambardella, and G. A. Di Caro, "Fair multi-target tracking in cooperative multi-robot systems," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5411–5418.

[8] J. Tisdale, Z. Kim, and J. K. Hedrick, "Autonomous uav path planning and estimation," *IEEE Robotics & Automation Magazine*, vol. 16, no. 2, 2009.

[9] S. M. Esmailifar and F. Saghafi, "Moving target localization by cooperation of multiple flying vehicles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 1, pp. 739–746, 2015.

[10] G. Soatti, M. Nicoli, S. Savazzi, and U. Spagnolini, "Consensus-based algorithms for distributed network-state estimation and localization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 2, pp. 430–444, 2017.

[11] F. Meyer, H. Wymeersch, M. Fröhle, and F. Hlawatsch, "Distributed estimation with information-seeking control in agent networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 11, pp. 2439–2456, 2015.

[12] S. M. Kay, *Fundamentals of statistical signal processing*. Prentice Hall PTR, 1993.

[13] A. Beck, P. Stoica, and J. Li, "Exact and approximate solutions of source localization problems," *IEEE Transactions on signal processing*, vol. 56, no. 5, pp. 1770–1778, 2008.

[14] P. Stoica and J. Li, "Lecture notes-source localization from range-difference measurements," *IEEE Signal Processing Magazine*, vol. 23, no. 6, pp. 63–66, 2006.

[15] Z. Weng and P. M. Djuric, "Efficient estimation of linear parameters from correlated node measurements over networks," *IEEE Signal Processing Letters*, vol. 21, no. 11, pp. 1408–1412, 2014.

[16] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[17] G. C. Calafiore, L. Carlone, and M. Wei, "A distributed technique for localization of agent formations from relative range measurements," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 42, no. 5, pp. 1065–1076, 2012.

[18] C. Soares, J. Xavier, and J. Gomes, "Distributed, simple and stable network localization," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*. IEEE, 2014, pp. 764–768.

[19] B. T. Polyak, "Introduction to optimization. translations series in mathematics and engineering," *Optimization Software*, 1987.

[20] R. Fletcher, "On the barzilai-borwein method," *Optimization and control with applications*, pp. 235–256, 2005.

[21] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA journal of numerical analysis*, vol. 8, no. 1, pp. 141–148, 1988.

[22] A. Beck and Y. C. Eldar, "Sparsity constrained nonlinear optimization: Optimality conditions and algorithms," *SIAM Journal on Optimization*, vol. 23, no. 3, pp. 1480–1509, 2013.