Decentralized Localization in Homogeneous Swarms Considering Real-World Non-Idealities

Hanlin Wang and Michael Rubenstein

***Abstract*—This letter presents a decentralized algorithm that allows a swarm of identically programmed agents to cooperatively estimate their global poses using local range and bearing measurements. The design of our algorithm explicitly considers the phase asynchrony of each agent's local clock, moreover, the execution of our algorithm does not require each agent to actively keep the same neighbors over time. A theoretical analysis about the effect of each agent's sensing noise and communication loss is given, in addition, we validate the presented algorithm via experiments running on a swarm of up to 256 simulated robots and a swarm of 100 physical robots. The results from the experiments show that the presented algorithm allows each agent to estimate its global pose quickly and reliably. Video of 100 robots executing the presented algorithm as well as supplementary material can be found in [1] and [2].**

***Index Terms*—Swarms, distributed robot systems, sensor networks.**

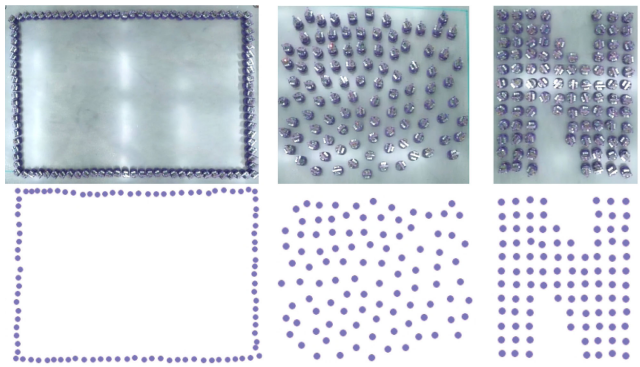

Fig. 1. (Top) 100 Coachbot V2.0 robots are placed in three patterns; (Bottom) The robots' position estimates obtained via the proposed algorithm.

I. Introduction

LOCALIZATION plays an important role in swarm systems. In many collective tasks, such as shape formation [3]–[5], shepherding [6], and more [7], it is valuable to have each agent know its pose in a global coordinate system.

Previous methods can be categorized into centralized methods [3], [8]–[10], and decentralized methods [11]–[21]. In centralized methods, agents obtain their poses from a central controller [8], [9], or a pre-setup external infrastructure [3], [10]. This type of method can work well in well-controlled environments, but cannot easily be deployed in unknown environments, and suffer from the *single point of failure* problem [22].

Decentralized methods, in contrast, are inherently more scalable and more robust to failures. Here, it is assumed that each agent is able to communicate with nearby agents. In addition, each agent can measure its distance [11]–[16], bearing [17], [18], or both [19]–[21] relative to its neighbors. The agents will estimate their global poses using the local communication and relative measurements. Decentralized methods can be categorized into two types: progressive methods [12]–[14], [18], [23] and concurrent methods [16]. In progressive methods, each agent has two possible states: localized and unlocalized. The unlocalized agents will localize themselves using the inter-agent measurements and the pose estimates from their already localized neighbors, then, these unlocalized agents will become localized agents and their pose estimates will later be used by their unlocalized neighbors. It was shown in [12], [13], [23] that with certain amount of pre-localized anchoring agents, it is possible to accurately localize a densely placed swarm using the local information only. However, each agent's position error will accumulate over its communication hops away from the anchoring agents. In addition, this type of method requires an additional phase to set certain agents as anchoring agents. The beacon-free solutions are proposed in [14], [18]. In [14], agents use inter-agent distance information to organize *robust quads* with their nearby agents. The adjacent *robust quads* can recover their relative positions using the information from the set of agents in common between two quads. However, it has not been shown that this method can estimate agent's orientation. The work presented in [18] allows agents to estimate their orientation using the inter-agent bearing information, in addition, this methods is also able to localize only a subset of the agents in the swarm. However, the coordinate system established by this method is scale-free, requiring an additional step to recover the coordinate system's scale [24].

Different from the progressive methods, in concurrent methods [16], [25], [26], agents do not explicitly hold a Boolean state of being localized or unlocalized. Instead, all the agents will constantly and cooperatively refine their pose estimates using their local measurements. The concurrent methods are generally more robust to sensing noise, in addition, these types of methods

Manuscript received February 24, 2021; accepted June 23, 2021. Date of publication July 7, 2021; date of current version July 22, 2021. This letter was recommended for publication by Associate Editor M. Ani Hsieh and Editor A. Prorok upon evaluation of the Reviewers' comments. This work was supported by the National Science Foundation under Grant CMMI-2024774. *(Corresponding author: Hanlin Wang.)*

Hanlin Wang is with the Department of Computer Science, Northwestern University, Evanston, IL 60601 USA (e-mail: hanlinwang2015@u.northwestern.edu).

Michael Rubenstein is with the Department of Computer Science and the Department of Mechanical Engineering, Northwestern University, Evanston, IL 60208 USA (e-mail: rubenstein@northwestern.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3095032, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3095032

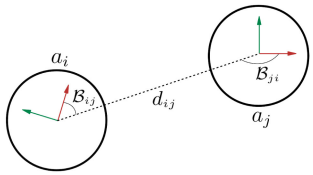

Fig. 2. Graphical illustration of the agent's sensing capability. The red and green arrow lines are each agent' local coordinate frame, where red arrow line is the x-axis.

are more robust to the unexpected external disturbances such as the removal or the addition of the agents. One drawback for this type of method is that, compared to the progressive methods, the concurrent methods' commutation cost is often more expensive, as each agent needs to frequently exchange its pose estimates with its neighbors.

In this letter, we present a fully decentralized concurrent localization algorithm for localizing a swarm of identically programmed agents. We assume that each agent can talk to nearby agents, and that when an agent receives a message, it can measure the transmitter's bearing and distance. The agents will estimate their global poses by cooperatively minimizing the disagreement between local measurements and their pose estimates. The novelty of our algorithm is that: The design of our algorithm explicitly considers the phase asynchrony of each agent's local clock, moreover, the execution of our algorithm does not require each agent to actively keep the same neighbors over time, which helps to avoid unnecessary communication overhead and makes our algorithm more flexible. A theoretical analysis about the effect of each agent's sensing noise and communication loss is given, in addition, the algorithm's performance is thoroughly investigated via experiments running on a swarm of up to 256 simulated agents as well as a swarm of 100 physical robots. The results from those experiments show that our algorithm is able to reliably localize the swarms with different sizes and configurations.

II. Preliminaries

In this section, we introduce the agent model used in the design as well as the analysis of the algorithm, and formally state the decentralized localization problem.

A. Agent Model

We assume that the agents are placed on a 2D plane. Each agent holds a local coordinate frame where the local coordinate frame's origin is fixed on the center of the agent and the x-axis' direction is aligned with the agent's heading. We assume each agent's clock has the same frequency but can be asynchronous in phase. In addition, it is assumed that each agent has a locally unique ID. We assume each agent is able to broadcast messages to all agents within its communication range R. Moreover, when an agent a_i receives a message from a neighbor a_j, the agent a_i is able to sense the transmitter a_j's relative bearing angle $\mathcal{B}_{ij}$ and the distance d_{ij}. See Fig. 2 for a graphical illustration of agent's sensing capabilities. One real-world example that fits our agent model is the Bluetooth 5.1 AOA device [20].

In order to make our agent model more realistic, we assume the inter-agent communication channel is lossy, that is: for an agent a_i, when a nearby neighbor a_j transmits a message, a_i will receive this message with a probability of λ, where $0 < \lambda < 1$. In addition, we assume the agent's bearing and range sensor is noisy, that is: let $\mathcal{B}_{ij}$ and d_{ij} be agent a_j's actual bearing angle and distance measured by agent a_i, respectively. When agent a_i attempts to measure those two quantities, the actual measurements returned from the sensor will be two random variables $\hat{\mathcal{B}_{ij}} = \langle \mathcal{B}_{ij} + \epsilon_{\mathcal{B}} \rangle_{2\pi}$ and $\hat{d_{ij}} = d_{ij} + \epsilon_d$, where $\langle \cdot \rangle_{2\pi} := \cdot + 2\pi \lfloor \frac{\pi - \cdot}{2\pi} \rfloor$ denotes the 2π modulus operator [27], which is an operator that wraps an angle from the real space to the interval $(-\pi, \pi]$, and $\epsilon_B \sim \mathcal{N}(0, \sigma_{\mathcal{B}}^2)$, $\epsilon_d \sim \mathcal{N}(0, \sigma_d^2)$ are two independent zero-mean Gaussian random variables.

B. Problem Statement

Our task is to design an algorithm that estimates each agent's pose in a global frame using local information only. In our algorithm, this task is achieved by forcing agents to constantly and cooperatively refine their pose estimates to reduce the inconsistency between their pose estimates and the local measurements. The global pose estimation will occur when each agent's pose estimate becomes consistent with its local measurements and its neighbors' pose estimates. More formally, let $\mathcal{A} = [a_1, a_2, \ldots, a_n]$ be a set of n agents, we use the undirected graph $\mathcal{G} = \{\mathcal{A}, \mathcal{E}\}$ to describe the network's communication topology. For a pair of agents a_i and a_j, $(i, j) \in \mathcal{E}$ iff they are located in each other's communication/sensing range R, and for each agent a_i, the set of all the agents located in its communication range is denoted as $N_i = \{a_j | a_j \in \mathcal{A}, (i, j) \in \mathcal{E}\}$. To simplify the description and analysis of the algorithm, it is assumed that $\mathcal{G}$ is always connected. In addition, we assume that each agent is stationary. Note that the assumption of each agent being stationary is only for the sake of analysis, the actual execution of our algorithm does not require this assumption. An example of the algorithm working in the situation where the swarm's communication topology is dynamically changing is shown in Section V-C. Each agent a_i uses the vector $[\theta_{ix}, \theta_{iy}]$ and vector $[x_i, y_i]$ to describe its orientation estimate and position estimate, respectively, where $\theta_i = \arctan 2(\theta_{iy}, \theta_{ix})$ is the agent's orientation estimate, and $[x_i, y_i]$ is the agent's estimate of its xy position. The agents will estimate their pose by cooperatively solving the following two problems:

Problem 1: (Orientation Estimation): Let $\mathcal{W}_{ij}$ be the true orientation difference between two agent a_i and a_j, we define the swarm's orientation estimate error f_o as:

$$\frac{1}{4} \sum_{a_i \in \mathcal{A}} \sum_{a_j \in N_i} \frac{1}{|N_i|} \frac{1}{|N_j|} \left\| \begin{bmatrix} \cos(\mathcal{W}_{ij}), & -\sin(\mathcal{W}_{ij}) \\ \sin(\mathcal{W}_{ij}), & \cos(\mathcal{W}_{ij}) \end{bmatrix} \begin{bmatrix} \theta_{ix} \\ \theta_{iy} \end{bmatrix} - \begin{bmatrix} \theta_{jx} \\ \theta_{jy} \end{bmatrix} \right\|_2^2$$

The task is to find each agent a_i's orientation estimate $[\theta_i^{x*}, \theta_i^{y*}]$ that minimizes the objective f_o above.

Problem 2: (Position Estimation): For each agent a_i, given a_i's orientation estimate $\theta_i = \arctan 2(\theta_{iy}, \theta_{ix})$, we define the swarm's position estimate error f_p as:

$$\frac{1}{4} \sum_{a_i \in \mathcal{A}} \sum_{a_j \in N_i} \frac{1}{|N_i|} \frac{1}{|N_j|} \left\| \begin{bmatrix} x_i \\ y_i \end{bmatrix} + d_{ij} \begin{bmatrix} \cos(\mathcal{B}_{ij} + \theta_i) \\ \sin(\mathcal{B}_{ij} + \theta_i) \end{bmatrix} - \begin{bmatrix} x_j \\ y_j \end{bmatrix} \right\|_2^2$$

The task is: given each agent's orientation estimate obtained from problem 1, find each agent a_i's position estimate $[x_i^*, y_i^*]$ that minimizes the objective f_p above.

The objective f_o and f_p can be intuitively interpreted as the normalized sum of the inconsistency between each pair of agents' pose estimates and their distance, orientation difference, as well as bearings relative to each other.

C. From Bearing Angle to Orientation Difference

One can see that in order to solve problem 1, each agent a_i needs to be able to measure nearby agent a_j's orientation difference $\mathcal{W}_{ij}$. On the other hand, as stated in Section II-A, agent a_i is not able to measure the neighbor a_j's orientation difference directly. It was shown in [18] that, for two agents a_i and a_j, given their bearing angles in the other's local frame $\mathcal{B}_{ji}$ and $\mathcal{B}_{ij}$, their orientation difference $\mathcal{W}_{ij}$ can be explicitly written as: $\mathcal{W}_{ij} = \langle \mathcal{B}_{ij} - \mathcal{B}_{ji} + \pi \rangle_{2\pi}$.

III. Approach

As we briefly discussed in Section II-B, in our algorithm, the agents will actively refine their pose estimates to minimize the objective f_o and f_p. When doing so, each agent will treat the objective f_o and the objective f_p separately: each agent a_i updates its orientation estimate $[\theta_{ix}, \theta_{iy}]$ only according to objective f_o (Algorithm 1, Line 20-21), and then uses the obtained orientation estimate $[\theta_{ix}, \theta_{iy}]$ to calculate objective f_p so as to update its position estimate $[x_i, y_i]$ (Algorithm 1, Line 22-24).

In the algorithm, each agent will consistently broadcast its current pose estimate to the neighbors at a fixed frequency of f (Algorithm 1, Line 28), meanwhile, at the same frequency, it will periodically update its pose estimate using the messages received (Algorithm 1, Line 10-26). For each agent, the message received from another agent can be characterised by a 3-tuple $msg = (data, \hat{\mathcal{B}}, \hat{d})$, where *data* is the payload of the message, $\hat{\mathcal{B}}$ is the measurement of transmitter's bearing angle, and $\hat{d}$ is the measurement of transmitter's distance. See Algorithm 1 for the detailed pseudo code of the proposed algorithm, and below is a detailed description of the main variables used in the algorithm:

- *id:* the agent's id, which is locally unique;
- α, β: the variables to control the step size of the update of the agent's pose estimate;
- *msg_buff:* the buffer to store the messages received in the latest $\frac{1}{f}$ amount of time;
- *last_check:* the variable to record when was the last time the agent transmitted a message;
- *clock():* the syscall that returns the time elapsed since the program started;
- $\theta_{ix}, \theta_{iy}, x_i, y_i$: the agent a_i's pose estimate;
- *echo_id, echo_B, echo_msg*: the variables that assist the agent to execute the "random echo" protocol so as to obtain its own bearing angle measured in the other's local frame;
- *msg_out:* the message to be transmitted;
- *msg_in:* a 3-tuple that contains the payload of received message, the measurement of message's transmitter's bearing angle, and the measurement of the message's transmitter's distance.

For each agent in swarm, every time when it receives a message, it will store it in the buffer *msg_buff* (Algorithm 1 Line 31-32). Each agent will periodically process the messages in buffer *msg_buff* (Algorithm 1, Line 9-26) and then empty the buffer (Algorithm 1, Line 29) at a fixed frequency of f. When an agent a_i attempts to update its orientation estimate $[\theta_{ix}, \theta_{iy}]$, the first task is to measure the orientation differences from its neighbor a_j. As stated in Section II-C, for an agent a_i, the calculation of its orientation difference from its neighbor a_j requires two measurements: $\hat{\mathcal{B}}_{ij}$, which is a_j's bearing angle measured by a_i, and $\hat{\mathcal{B}}_{ji}$, which is a_i's bearing angle measured by a_j. On the other hand, the second measurement $\hat{\mathcal{B}}_{ji}$ cannot be obtained by a_i via local sensing directly. In order to allow each agent to obtain its own bearing angle measured by its neighbors, the agents will cooperatively execute a "random echo" protocol: for each agent in the swarm, every time when forging *msg_out*, it will first uniformly and randomly select a message *echo_msg* in the buffer *msg_buff*, then embeds the id and the measurement of bearing angle of this *echo_msg*'s transmitter in the *msg_out* (Algorithm 1, Line 11-13, Line 26). By cooperatively doing so, each agent a_i will be able to obtain its own bearing angle measured by any neighbor a_j with a non-zero probability. See Fig. 3 for a minimal working example for this "random echo" protocol. In this example, there are three agents involved. Recall that we assume the agents' clocks are asynchronous in phase, therefore, from a global observer's perspective, despite that each agent is programmed to broadcast at the same frequency, they still might transmit messages at different times, as shown in Fig. 3. Due to the space constraints, in the example, we will only track the agent 0's behavior, which suffices to demonstrate the "random echo" protocol as all the agents are identically programmed. At t_0 and t_1, agent 0 receives messages from agent 1 as well as agent 2 and stores them in the buffer *msg_buff* (Algorithm 1, Line 33). At t_2, agent 0 generates a message and transmits it out. When generating this message, agent 0 uniformly and randomly selects a message in the *msg_buff* (which is the message from agent 1 at t_2), and embeds the id as well as the measurement of the bearing angle of this selected message's transmitter in the message to be transmitted (Algorithm 1, Line 11-13). This

Algorithm 1: Proposed algorithm (runs on each agent a_i).

1 **Input**: *id*, f, α, β
2 *msg_buff* $\leftarrow \emptyset$
3 *echo_id* $\leftarrow$ *id*
4 *echo_B* $\leftarrow 0$
5 *last_check* $\leftarrow$ *clock()*
6 Initialize the pose estimation $\theta_{ix}, \theta_{iy}, x_i, y_i$
7 *msg_out* $\leftarrow$ {*id*, $\theta_{ix}, \theta_{iy}, x_i, y_i$, *echo_id*, *echo_B*}
8 **while** *True* **do**
9 **if** *clock()* - *last_check* $> \frac{1}{f}$ **then**
10 *last_check* $\leftarrow$ *clock()*
11 **if** *msg_buff* is ***not*** *empty* **then**
12 *echo_msg* $\leftarrow$ uniformly and randomly choose a element in *msg_buff*
13 *echo_id* $\leftarrow$ *echo_msg.data.id*
14 *echo_B* $\leftarrow$ *echo_msg.*$\hat{\mathcal{B}}$
15 *msg_selected* $\leftarrow$ uniformly and randomly choose a element in *msg_buff*
16 **if** *msg_selected.data.echo_id* == *id* **then**
17 $\theta_{jx}, \theta_{jy}, x_j, y_j \leftarrow$ the pose estimates contained in *msg_selected*
18 $\hat{\mathcal{B}}_{ij} \leftarrow$ *msg_selected.*$\hat{\mathcal{B}}$
19 $\hat{d}_{ij} \leftarrow$ *msg_selected.*$\hat{d}$
20 $\hat{\mathcal{B}}_{ji} \leftarrow$ *msg_selected.data.echo_B*
21 $\hat{\mathcal{W}}_{ij} \leftarrow \langle \hat{\mathcal{B}}_{ij} - \hat{\mathcal{B}}_{ji} + \pi \rangle_{2\pi}$
22 $\delta_\theta \leftarrow \begin{bmatrix} \cos\left(\hat{\mathcal{W}}_{ij}\right), \sin\left(\hat{\mathcal{W}}_{ij}\right) \\ -\sin\left(\hat{\mathcal{W}}_{ij}\right), \cos\left(\hat{\mathcal{W}}_{ij}\right) \end{bmatrix} \begin{bmatrix} \theta_{jx} \\ \theta_{jy} \end{bmatrix} - \begin{bmatrix} \theta_{ix} \\ \theta_{iy} \end{bmatrix}$
23 $\theta_i \leftarrow \arctan 2(\theta_{iy}, \theta_{ix})$
24 $\theta_j \leftarrow \arctan 2(\theta_{jy}, \theta_{jx})$
25 $\delta_{xy} \leftarrow \begin{bmatrix} x_j \\ y_j \end{bmatrix} - \frac{\hat{d}_{ij}}{2} \begin{bmatrix} \cos\left(\hat{\mathcal{B}}_{ij} + \theta_i\right) - \cos\left(\hat{\mathcal{B}}_{ji} + \theta_j\right) \\ \sin\left(\hat{\mathcal{B}}_{ij} + \theta_i\right) - \sin\left(\hat{\mathcal{B}}_{ji} + \theta_j\right) \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix}$
26 $\begin{bmatrix} \theta_{ix} \\ \theta_{iy} \end{bmatrix} \leftarrow \begin{bmatrix} \theta_{ix} \\ \theta_{iy} \end{bmatrix} + \alpha \delta_\theta$
27 $\begin{bmatrix} x_i \\ y_i \end{bmatrix} \leftarrow \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \beta \delta_{xy}$
28 *msg_out* $\leftarrow$ {*id*, $\theta_{ix}, \theta_{iy}, x_i, y_i$, *echo_id*, *echo_B*}
29 transmit *msg_out*
30 *msg_buff* $\leftarrow \emptyset$
31 **else**
32 **if** *receive a message msg_in* **then**
33 *msg_buff* $\leftarrow$ *msg_buff* $\cup$ {*msg_in*}

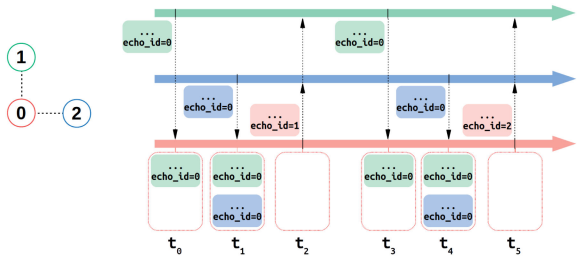

Fig. 3. (Left) Physical postion of agents: each disk represents an agent, the number on the disk indicates the agent's id, the dotted line connecting two agents indicates that those two agents are located within each other's communication range. (Right) The horizontal colored arrow lines are agents' local clocks running from the left to the right. Each vertical dotted arrow line is a broadcast event, which points from the transmitter to the receiver(s). Each filled box is a transmitted message, the box's color shows its transmitter. The array of unfilled boxes attached to agent 0's clock on the bottom shows the values of agent 0's variable *msg_buff* at different times $t_0, \ldots, t_5$, each filled box inside the *msg_buff* is a received message currently stored in agent 0's buffer *msg_buff*.

transmitted message will allow the agent 1 to obtain its own bearing angle measured by agent 0. Right after transmitting the message, agent 0 will empty its buffer *msg_buff*. The agent 0's behaviors at t_3, t_4 and t_5 are almost the same as t_0, t_1 and t_2, except that at t_5, it forges the message using the message from agent 2 instead of the message from agent 1.

As we discussed before, in the algorithm, each agent a_i uses the messages in the buffer *msg_buff* to update its pose estimate at a fixed frequency of f (Algorithm 1, Line 9-29). To do so, every $\frac{1}{f}$ amount of time, each agent a_i will first uniformly and randomly select a message in the buffer *msg_buff*, which is denoted as *msg_selected* in the algorithm, then, it will check if the value of the field *echo_id* in *msg_selected* matches its own id, i.e., if this *msg_selected* contains a_i's bearing angle measured by the other. If so, a_i will use the information in *msg_selected* to update its pose estimate (Algorithm 1, Line 16-26) and then empty the buffer *msg_buff* (Algorithm 1, Line 29). Otherwise, a_i will do nothing but empty the buffer *msg_buff* directly (Algorithm 1, Line 29). Each time when agent a_i updates its pose estimate (Algorithm 1, Line 25-26), the step size is controlled by the variables α and β.

IV. Theoretical Results

In this section, we study how each agent's sensing noise and communication loss will affect the swarm's behavior, and we show that when the agent's communication loss and sensing noise is low, the swarm's behavior can be approximated as the unbiased stochastic gradient descent (SGD) on the objective f_o and f_p. In addition, we give analysis of the algorithm's complexity.

A. Analysis of the Swarm's Behavior

First, we consider the swarm's behavior from a global observer's perspective. Say there is a global observer holding a clock that has the same frequency as each agent's local clock. This global observer will record every agent's pose estimate at a fixed frequency f, which is the same as the frequency at which each agent will attempt to update its pose estimate. By doing so, this global observer will be able to capture all the changes of each agent's pose estimate. Please see [2] Section VII for the formal proof of this conclusion.

Algorithm 2: Swarm's behavior (From a global observer's perspective).

for $k \leftarrow 1, 2, \ldots, \infty$ **do**
  for *each agent* $a_i \in \mathcal{A}$ **do**
    Assign each neighbor $a_j \in N_i$ a probability: $\frac{1-(1-\lambda)^{|N_j|}}{|N_j|}\frac{1-(1-\lambda)^{|N_i|}}{|N_i|}$, where $(1-\lambda)$ is the packet loss rate
    randomly select a neighbor a_j according to its probability assigned above
    $\hat{\mathcal{B}_{ij}} \leftarrow \langle \mathcal{B}_{ij} + \epsilon_{\mathcal{B}} \rangle_{2\pi}$, where $\epsilon_B \sim \mathcal{N}(0, \sigma^2_{\mathcal{B}})$
    $\hat{\mathcal{B}_{ji}} \leftarrow \langle \mathcal{B}_{ji} + \epsilon_{\mathcal{B}} \rangle_{2\pi}$, where $\epsilon_B \sim \mathcal{N}(0, \sigma^2_{\mathcal{B}})$
    $\hat{d}_{ij} \leftarrow d_{ij} + \epsilon_d$, where $\epsilon_d \sim \mathcal{N}(0, \sigma^2_d)$
    $\hat{\mathcal{W}}_{ij} \leftarrow \langle \hat{\mathcal{B}}_{ij} - \hat{\mathcal{B}}_{ji} + \pi \rangle_{2\pi}$
    $\delta_\theta \leftarrow \begin{bmatrix} \cos\left(\hat{\mathcal{W}}_{ij}\right), \sin\left(\hat{\mathcal{W}}_{ij}\right) \\ -\sin\left(\hat{\mathcal{W}}_{ij}\right), \cos\left(\hat{\mathcal{W}}_{ij}\right) \end{bmatrix} \begin{bmatrix} \theta^{k-1}_{jx} \\ \theta^{k-1}_{jy} \end{bmatrix} - \begin{bmatrix} \theta^{k-1}_{ix} \\ \theta^{k-1}_{iy} \end{bmatrix}$
    $\theta_i \leftarrow \arctan 2(\theta^{k-1}_{iy}, \theta^{k-1}_{ix})$
    $\theta_j \leftarrow \arctan 2(\theta^{k-1}_{jy}, \theta^{k-1}_{jx})$
    $\delta_{xy} \leftarrow \begin{bmatrix} x^{k-1}_j \\ y^{k-1}_j \end{bmatrix} - \frac{\hat{d}_{ij}}{2} \begin{bmatrix} \cos\left(\hat{\mathcal{B}}_{ij} + \theta_i\right) - \cos\left(\hat{\mathcal{B}}_{ji} + \theta_j\right) \\ \sin\left(\hat{\mathcal{B}}_{ij} + \theta_i\right) - \sin\left(\hat{\mathcal{B}}_{ji} + \theta_j\right) \end{bmatrix} - \begin{bmatrix} x^{k-1}_i \\ y^{k-1}_i \end{bmatrix}$
    $\begin{bmatrix} \theta^k_{ix} \\ \theta^k_{iy} \end{bmatrix} \leftarrow \begin{bmatrix} \theta^{k-1}_{ix} \\ \theta^{k-1}_{iy} \end{bmatrix} + \alpha\delta_\theta$
    $\begin{bmatrix} x^k_i \\ y^k_i \end{bmatrix} \leftarrow \begin{bmatrix} x^{k-1}_i \\ y^{k-1}_i \end{bmatrix} + \beta\delta_{xy}$

Proposition 1: Let $\theta^k_{ix}, \theta^k_{iy}, x^k_i, y^k_i$ be agent a_i's pose estimate observed by the global observer at time $k\frac{1}{f}$, the swarm's behavior can be described by the Algorithm 2 without loss of any information.

Proof: Please see [2], Section VII. ■

Note that the objective f_o and f_p are defined using the true inter-agent relative bearing angles and distances, whereas in reality, the actual measurements that each agent uses to update its pose estimate will be corrupted by the sensing noise. Next, we show how the sensing noise and the communication loss will affect the swarm's behavior.

Lemma 1: Let $\Theta^k_i = [\theta^k_{ix}, \theta^k_{iy}]^\intercal$ be agent a_i's orientation estimate at k^{th} iteration, in addition, let $\frac{\partial f_o}{\partial \Theta_i}^k$ be the value of partial derivative of objective f_o with respect to Θ_i calculated using agents' orientation estimates at k^{th} iteration, we have:

$$\mathbb{E}(\Theta^{k+1}_i - \Theta^k_i) = -\mu\alpha \left\{ \frac{\partial f_o}{\partial \Theta_i}^k + \gamma^k_\Theta \right\}$$

In which:

$$\mu = 1 - (1-\lambda)^{|N_i|} \tag{1}$$

$$\gamma^k_\Theta = \sum_{a_j \in N_i} \frac{1 - \exp -\sigma^2_{\mathcal{B}}\{1-(1-\lambda)^{|N_j|}\}}{|N_j|} \mathcal{R}(-\mathcal{W}_{ij})\Theta^k_j - \frac{(1-\lambda)^{|N_j|}}{|N_j|}\Theta^k_i \tag{2}$$

where $\mathcal{R}(\cdot) = \begin{bmatrix} \cos(\cdot), & -\sin(\cdot) \\ \sin(\cdot), & \cos(\cdot) \end{bmatrix}$ is the rotation matrix constructed from the angle $\cdot$

Proof: Please see [2], Section VIII. ■

Lemma 2: Assume the agents' orientation estimates have converged to a stable state. Let $\mathcal{X}^k_i = [x^k_i, y^k_i]^\intercal$ be agent a_i's position estimate at k^{th} iteration, moreover, let $\frac{\partial f_p}{\partial \mathcal{X}_i}^k$ be the value of partial derivative of objective f_p with respect to $\mathcal{X}_i$ calculated using agents' position estimates at k^{th} iteration, we have:

$$\mathbb{E}(\mathcal{X}^{k+1}_i - \mathcal{X}^k_i) = -\mu\beta \left\{ \frac{\partial f_p}{\partial \mathcal{X}_i}^k + \gamma^k_\mathcal{X} \right\}$$

In which:

$$\mu = 1 - (1-\lambda)^{|N_i|} \tag{3}$$

$$\gamma_{\mathcal{X}}^k = \sum_{a_j \in N_i} \frac{1 - \exp\{-\frac{\sigma_B^2}{2}\}\{1-(1-\lambda)^{|N_j|}\}}{2|N_j|}\mathcal{P}(i,j)d_{ij} + \frac{(1-\lambda)^{|N_j|}}{|N_j|}\{\mathcal{X}_j^k - \mathcal{X}_i^k\} \tag{4}$$

where $\mathcal{P}(i,j) = -\begin{bmatrix}\cos(\mathcal{B}_{ij}+\theta_i) - \cos(\mathcal{B}_{ji}+\theta_j) \\ \sin(\mathcal{B}_{ij}+\theta_i) - \sin(\mathcal{B}_{ji}+\theta_j)\end{bmatrix}$ is the matrix constructed from a pair of agents a_i and a_j's relative bearing angles as well as their orientation estimates.

Proof: Please see [2], Section IX. ∎

Lemma 1 and Lemma 2 suggest that at each iteration, in expectation, each agent will update its pose estimate along a direction that is slightly deviated from the negative gradient of objective f_p and f_o, with a step size that is slightly smaller than the user-specified one. As we can see in Lemma 1 and Lemma 2, the depreciation factor of the step size μ is introduced by the agent's communication loss, in other words, in expectation, the communication loss will "shorten" the agent's step size. In addition, the communication loss' effect will be reduced by the agent's degree. As to γ_Θ and $\gamma_\mathcal{X}$ (which are the deviation of the expected update direction from the objective's gradient negative), according to the 2 and 4, they are a result of both the communication loss and the sensing noise.

On the other hand, one can observe that, the bias of the update direction, and the depreciation of the step size will super-linearly decay over agent's communication loss and sensing noise. This suggests that, when communication loss and sensing noise are reasonably small, the update direction bias γ_Θ, $\gamma_\mathcal{X}$, and the depreciation of the step size μ will become negligible.

Assumption 1: Each agent a_i's degree $|N_i|$ is not too low and the packet loss rate $1-\lambda$ is not too high s.t. $(1-\lambda)^{|N_i|} \approx 0$.

Assumption 2: The variance of agent's bearing sensing noise $\sigma_\mathcal{B}^2$ is not too big s.t. $\exp\{-\frac{\sigma_B^2}{2}\} \approx 1$.

Remark: Assumption 1 and 2 are actually pretty mild. One can consider a case where the packet loss rate $1-\lambda$ is 10% and the lowest degree of any agent in the swarm is 2. In this case, $(1-\lambda)^{|N_i|} = 0.01 \approx 0$. For assumption 2, consider a case where each agent's bearing sensing noise has a std of 0.3 rad, which is around the same as the std of the angle measurements of Bluetooth 5.1 devices [20]. In this case, $\exp -\frac{\sigma_B^2}{2} = \exp\{-0.045\} \approx 1$.

Theorem 1: If Assumption 1 and 2 hold, then the swarm's behavior can be approximated as the unbiased stochastic gradient descent on objective f_o and f_p. Namely:

$$\mathbb{E}(\Theta_i^{k+1} - \Theta_i^k) \approx -\alpha \frac{\partial f_o}{\partial \Theta_i}^k, \quad \mathbb{E}(\mathcal{X}_i^{k+1} - \mathcal{X}_i^k) \approx -\beta \frac{\partial f_p}{\partial \mathcal{X}_i}^k$$

Proof: Theorem 1 can be easily obtained by substituting $(1-\lambda)^{|N_i|}$ with 1 and substituting $\exp\{-\frac{\sigma_B^2}{2}\}$ with equation 1 in 1, equation 2, equation 3, and equation 4. ∎

So far, we have shown that when communication loss and sensing noise is reasonably small, the swarm's behavior is equivalent to the unbiased SGD on the objective f_o and f_p. In other words, in expectation, the algorithm allows the agents to constantly reduce the disagreement between their pose estimates and their local measurements, showing the algorithm's correctness.

B. Complexity

First, we study the algorithm's memory complexity. It is straight forward to examine that the memory to execute the algorithm is dominated by the size of buffer *msg_buff*. Therefore, for an agent a_i, the algorithm's memory complexity is $\mathcal{O}(|N_i|)$, where $|N_i|$ is the number of a_i's neighbors.

Next, we investigate the algorithm's computation complexity. We assume that the complexity of querying a random number generator is $\mathcal{O}(1)$, then, the cost to execute Algorithm 1 Line 8-32 is $\mathcal{O}(1)$. In addition, during a unit of time, each agent a_i can receive at most $f|N_i|$ messages, that is, in a unit of time, a_i will execute Algorithm Line 8-32 at most $f|N_i|$ times. Thus, for an agent a_i, the computation complexity of the algorithm is $\mathcal{O}(f|N_i|)$.

Lastly, given the fact that the length of each message exchanged amongst the agents is $\mathcal{O}(1)$, one can easily conclude that the algorithm's communication complexity, i.e., the amount of data to be transmitted by each agent in a unit of time, is $\mathcal{O}(f)$.

V. Empirical Evaluation

In this section, we study the performance of proposed algorithm empirically in a 100-robot swarm and in simulation.

To qualitatively evaluate the algorithm's performance, we introduce two metrics *NOEE* (normalized orientation estimate error) and *NPEE* (normalized position estimate error), which are the metrics to evaluate the error of agents' orientation estimates and position estimates, respectively. The *NOEE* and *NPEE* are defined as follows: we use $\theta_i^k = \text{atan2}(\theta_{iy}^k, \theta_{ix}^k)$ and $[x_i^k, y_i^k]$ to denote agent a_i's orientation estimate and position estimate observed by the global observer at k^{th} iteration, in addition, we use θ_i^g and $[x_i^g, y_i^g]$ to denote agent a_i's true pose in a global coordinate system. At each iteration k, we first calculate the rotation and translation that can match the agents' estimated positions and their true positions the best, namely, find $\theta_*^k \in (-\pi, \pi], t_*^k \in \mathbf{R}^2$ that minimize the following objective:

$$\min \sum_{a_i \in \mathcal{A}} \left\| \begin{bmatrix}\cos(\theta_*^k), & -\sin(\theta_*^k) \\ \sin(\theta_*^k), & \cos(\theta_*^k)\end{bmatrix} \begin{bmatrix}x_i^k \\ y_i^k\end{bmatrix} + t_*^k - \begin{bmatrix}x_i^g \\ y_i^g\end{bmatrix} \right\|_2^2$$

Then, the *NOEE* and *NPEE* at k^{th} iteration are defined as:

$$NOEE^k = \frac{1}{n}\sum_{a_i \in \mathcal{A}} |\langle \theta_i^k + \theta_*^k - \theta_i^g \rangle_{2\pi}|$$

$$NPEE^k = \frac{1}{n}\sum_{a_i \in \mathcal{A}} \left\| \begin{bmatrix}\cos(\theta_*^k), & -\sin(\theta_*^k) \\ \sin(\theta_*^k), & \cos(\theta_*^k)\end{bmatrix} \begin{bmatrix}x_i^k \\ y_i^k\end{bmatrix} + t_*^k - \begin{bmatrix}x_i^g \\ y_i^g\end{bmatrix} \right\|_2$$

where n is the swarm size. The *NOEE* is the normalized geodesic distance between agents' transformed orientation estimates and their true orientations, and *NPEE* is the normalized Euclidean distance between agents' transformed position estimates and their true positions.

A. Simulation

In simulation, the robot's communication range is set to 0.3 m, the robot's communication frequency f is set to 30 Hz. In each test, we place the simulated robots in three patterns: the circle, the shape "N," and the random mesh. See Fig. 4 for a graphical illustration of these three patterns. In the circle pattern, n robots are evenly distributed on on a circle. The circle's radius is made such that the distance between two adjacent robots is 0.25 m,

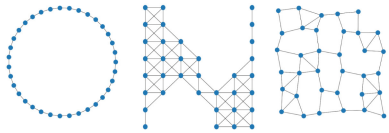

Fig. 4. From left to right: the circle pattern, the shape "N" pattern, and the random mesh pattern for a swarm of 35 robots. Each dot represents a robot, a line connecting two robots indicates that those two robots are located within each other's communication range.

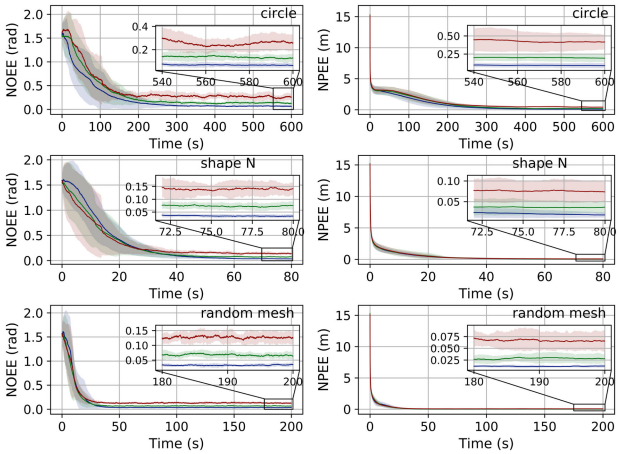

Fig. 5. Each colored solid line is the mean from 30 trials, and the colored shade areas show the confidence intervals at a confidence level of two σ. The color indicates the noise profile used in experiment: blue – $\sigma_{\mathcal{B}} = 0.05$ rad, $\sigma_d = 0.01$ m; green – $\sigma_{\mathcal{B}} = 0.1$ rad, $\sigma_d = 0.02$ m; red – $\sigma_{\mathcal{B}} = 0.2$ rad, $\sigma_d = 0.04$ m.

in addition, each robot's orientation is set to be such that: each agent faces straight towards to the center of the circle. In the shape "N" pattern, the robots' positions form a "N" shape on a grid, moreover, the distance between any pair of adjacent robots is 0.2 m. In the random mesh configuration, n agents are randomly placed in a $0.25\sqrt{n}$ m $\times$ $0.25\sqrt{n}$ m space, moreover, when generating each robot's position, we enforce the swarm's communication graph to be connected. In both shape "N" pattern and random mesh pattern, we set each robot's orientation randomly. We use these three patterns to investigate the effect of the swarm's connectivity on the algorithm's performance. On average, in circle pattern, each agent has 2 neighbors; in random mesh pattern, each agent has 4 neighbors; in shape "N" pattern, each agent has 6 neighbors. In all tests, each robot a_i's pose estimate is randomly initialized in a way that $x_i^0 \sim \mathcal{U}(-20, 20), y_i^0 \sim \mathcal{U}(-20, 20), \theta_i^0 \sim \mathcal{U}(-\pi, \pi)$, where $\mathcal{U}(a, b)$ stands for uniform distribution between the interval (a, b).

In the first test, we study the effect of sensing noise on the algorithm's performance. In this test, a swarm of 100 simulated robots estimate their poses with a step size of $\alpha = \beta = 0.2$. The communication loss rate $1 - \lambda$ is set to 0.1. We test the algorithm's performance with three noise profiles: $\sigma_{\mathcal{B}} = 0.05$ rad, $\sigma_d = 0.01$ m; $\sigma_{\mathcal{B}} = 0.1$ rad, $\sigma_d = 0.02$ m; and $\sigma_{\mathcal{B}} = 0.2$ rad, $\sigma_d = 0.04$ m, where $\sigma_{\mathcal{B}}, \sigma_d$ are the standard deviations of robot's bearing sensing noise and range sensing noise, respectively. For each noise profile, 30 trials were run. The results are shown in Fig 5. As we can see in the figure, for all three patterns, the convergence rate of swarm's pose estimate is almost the same for different noise profiles. On the other hand, as expected, the sensing noise will affect the accuracy of the swarm's pose estimate: bigger sensing noise will result in

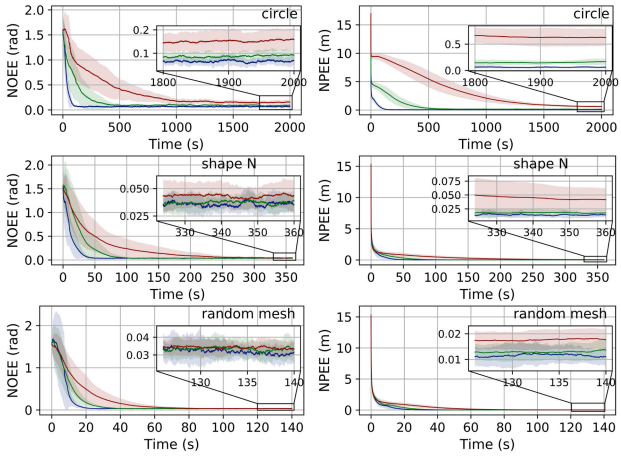

Fig. 6. Each colored solid line is the mean from 30 trials, and the colored shade areas show the confidence intervals for a confidence level of two σ. The color indicates the swarm size used in the experiment: blue – 64; green – 128; red – 256.

swarm's pose estimate converging to a state with higher error. Furthermore, one can see that compared the other two patterns, sensing error will affect the circle pattern more significantly.

A second test studies the algorithm's performance on different swarm size. In this test, swarms of 64, 128, 256 robots estimate their poses with a step size of $\alpha = \beta = 0.2$. The communication loss rate $1 - \lambda$ is set to 0.1. In addition, the sensing noise is set to $\sigma_{\mathcal{B}} = 0.05$ rad, $\sigma_d = 0.01$ m. For each swarm size, 30 trials were run. The results are shown in Fig. 6. Unsurprisingly, the results from experiments suggests that for all three patterns, the swarm size will affect both the convergence rate and the accuracy of the swarm's pose estimate: the larger the swarm size is, the slower the swarm's pose estimate will converge, and the higher the swarm's localization error will be.

In the third test, we study the effect of the step size α and β on the algorithm's performance. In this test, a swarm of 100 simulated robots estimate their poses using three different step sizes: $\alpha = \beta = 0.1$, $\alpha = \beta = 0.2$, and $\alpha = \beta = 0.3$. The communication loss rate $1 - \lambda$ is set to 0.1. The noise profile used in this test is $\sigma_{\mathcal{B}} = 0.05$ rad, $\sigma_d = 0.01$ m. For each step size, 30 trials were run. The results are shown in Fig 7. As we can see in the figure, a larger step size will enable the swarm's pose estimate to converge faster, at a cost of swarm's pose estimate's accuracy. In other words, there is a trade off between the algorithm's convergence rate and localization error. As we can see in the plots, for all three patterns, the step size $\alpha = \beta = 0.2$ seems to balance the convergence rate and localization error the best.

In the last test, we study the effect of agent's communication loss. In this test, a swarm of 100 simulated robots estimate their poses with a step size of $\alpha = \beta = 0.2$. The noise profile used in this test is $\sigma_{\mathcal{B}} = 0.05$ rad, $\sigma_d = 0.01$ m. We test the algorithm's performance with three different communication loss rates: $1 - \lambda = 0.1, 1 - \lambda = 0.3$, and $1 - \lambda = 0.5$. For each communication loss rate, 30 trials were run. The results are shown in Fig 8. As shown in the figure, unsurprisingly, the communication loss will slow down the swarm's convergence rate. In addition, compared to the other two patterns, the communication loss will affect the circle pattern more significantly. The reason is that: in the circle pattern, the agent's average degree is much lower than the other two patterns. This observation confirms our finding in

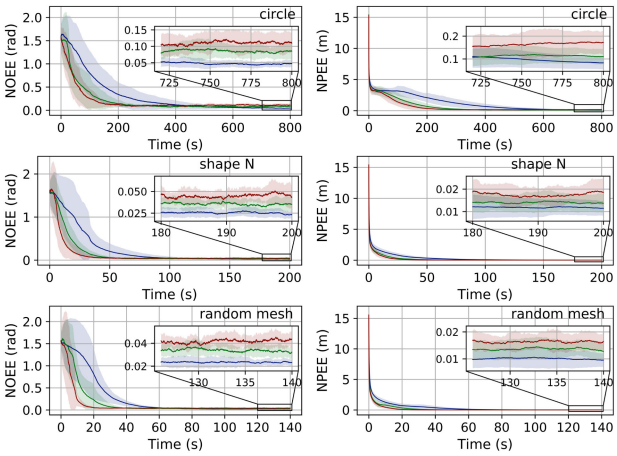

Fig. 7. Each colored solid line is the mean from 30 trials, and the colored shade areas show the confidence intervals for a confidence level of two σ. The color indicates the step size used in experiment: blue – $\alpha = \beta = 0.1$; green – $\alpha = \beta = 0.2$; red – $\alpha = \beta = 0.3$.

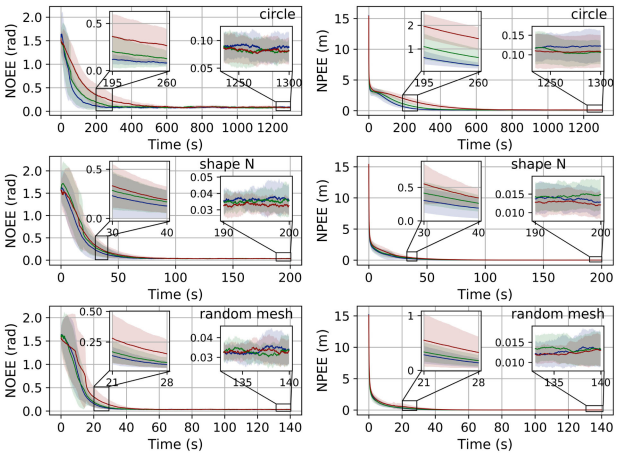

Fig. 8. Each colored solid line is the mean from 30 trials, and the colored shade areas show the confidence intervals for a confidence level of two σ. The color indicates the communication loss rate used in experiment: blue – $1 - \lambda = 0.1$; green – $1 - \lambda = 0.3$; red – $1 - \lambda = 0.5$.

Lemma 1 and Lemma 2 that, the effect of communication loss can be reduced by the agent's degree.

In all four tests, we can see that: compared to the other two patterns, it takes much longer for the circle pattern to converge. One possible explanation is that: given n agents, the circle pattern's communication diameter, i.e., the maximal pairwise inter-agent communication hop, is $\mathcal{O}(n)$, whereas the random mesh pattern and shape "N" pattern has a communication diameter of $\mathcal{O}(\sqrt{n})$. The communication diameter essentially characterizes the cost to spread an agent's information across the entire swarm. The larger the communication diameter is, the longer it will take to spread a agent's information across the swarm. As a result, the larger communication diameter makes circle pattern converge much slower than the others. In addition, the difference of each pattern's communication diameter can also be used to explain the observation that the circle pattern has a higher localization error, as the sensing error will accumulate over the communication hop.

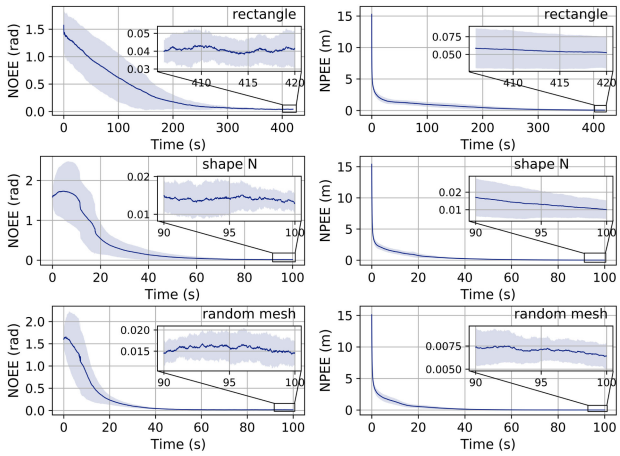

Fig. 9. The solid line is the mean from multiple trials, and the colored shade areas show the confidence intervals for a confidence level of two σ.

B. Experiments

In this section, we examined the algorithm's performance on a swarm of 100 Coachbot V2.0 robots [3]. Coachbot V2.0 is a differential driven mobile robot that is able to sense its position and orientation in a global coordinate system. Note that Coachbot V2.0 does not have a real bearing and range sensor. In order to implement our algorithm, in experiments, we embed the robot's position in the transmitted data packet, and the receiver can calculate the transmitter's bearing and distance by comparing its own pose with the position contained in the received message. The limited communication range is simulated in the same way as [3].

In the experiment, the robot's communication frequency f is 30 Hz, and the step size is set to $\alpha = \beta = 0.2$. The robots are placed in three patterns: the rectangle pattern, the random mesh pattern, and the shape "N" pattern. See Fig. 1 for a graphical illustration of these three patterns. In the rectangle pattern, the robots are densely placed on the perimeter of a rectangle, where the distance between two adjacent robots is 0.15 m, and each robot faces straight towards the center of the rectangle. The robot's communication range is set to 0.2 m in rectangle pattern so as to make each robot's degree to the consistent with the circle pattern used in the simulation. The remaining two patterns are the same as the ones used in the simulation, and in those two patterns, the robot's communication range is set to 0.3 m. For the random mesh pattern and the shape "N" pattern, 15 trials were run; for the rectangle pattern, 6 trials were run due to its long convergence time. The results are shown in Fig 9.

In all the trials, the algorithm reliably converge to all the robots accurately localizing themselves. For each pattern, its convergence rate is approximately the same as the result we obtained in the simulation. The random mesh pattern and the shape "N" pattern can converge in less than a minute, while it takes much longer for the rectangle pattern to converge. In addition, one can see that, for random mesh pattern and the shape "N" pattern, the average converged *NPEE*, i.e., normalized position error, is smaller than 2 cm, and the average converged *NPEE* for the rectangle pattern is around 5 cm. Given the fact that the robot is in a disk shape with a diameter of 13 cm, it can be concluded that, for all three patterns, the robots' average converged position error is smaller than the robot's footprint, and in the case of shape "N" pattern and the random mesh pattern, much smaller.

C. Example Use Case: Decentralized Robotic Shape Formation

One desirable feature of our algorithm is that the execution of our algorithm does not require each agent to actively keep the same neighbors over time, or synchronize its local clock's phase with the others. This feature makes it possible for the algorithm to work in the situations where the swarm's communication topology is dynamically changing. In this demonstration, 100 robots use our algorithm to execute the shape formation algorithm presented in [3]. Different from the original version of the algorithm presented in [3], where the agents need to acquire their poses from a global position system, in this experiment, the robots estimate their poses according to the local measurements and the odometry. Initially, each robot stays stationary and executes our algorithm to estimate its pose. Once the robot's pose estimate gets consistent with the local measurements and its neighbors' pose estimates, it starts to move to form the shape. When the robot moves, it uses on-board odometry to capture the change in its pose over time and updates its pose estimate accordingly, at the same time, it also constantly refines its pose estimate according to the local measurements using our algorithm. The video of this experiment is shown in [1]. The result shows that our algorithm successfully enabled the swarm to replace the use of the global position system with the local measurements.

VI. Conclusion

This letter presents a fully decentralized algorithm for localizing a swarm of identically programmed agents. The execution of the presented algorithm does not require each agent to actively keep the same neighbors over time, or synchronize its local clock's phase with the others. The theoretical analysis about the effect of each agent's sensing noise and communication loss is given, in addition, the correctness and performance of the algorithm was examined via the tests running on a swarm of up to 256 simulated robots, and 100 real robots. Extensive simulation trials and real robot experiments show that the algorithm can localize the swarms with different sizes and configurations quickly and reliably. Furthermore, beyond the situations where the agents are stationary, it was shown by a 100-robot experiment that when cooperating with the robot's on-board odometry, the presented algorithm can also be used to localize the swarms where the agents are dynamically moving.

References

[1] H. Wang and M. Rubenstein, "Decentralized localization in homogeneous swarms considering real-world non-idealities (Multimedia Materials)," 2021. [Online]. Available: http://users.eecs.northwestern.edu/~mrubenst/RAL-IROS-2021.zip

[2] H. Wang and M. Rubenstein, "Decentralized localization in homogeneous swarms considering real-world non-idealities (Supplementary Material)," 2021. [Online]. Available: http://users.eecs.northwestern.edu/~mrubenst/supplementary_material_ral2021.pdf

[3] H. Wang and M. Rubenstein, "Shape formation in homogeneous swarms using local task swapping," *IEEE Trans. Robot*, vol. 36, no. 3, pp. 597–612, Jun. 2020.

[4] M. Gauci, R. Nagpal, and M. Rubenstein, "Programmable self-disassembly for shape formation in large-scale robot collectives," in *Proc. Int. Symp. DARS*. Springer, 2018, pp. 573–586, doi: 10.1007/978-3-319-73008-0_40.

[5] M. Rubenstein and W.-M. Shen, "A scalable and distributed approach for self-assembly and self-healing of a differentiated shape," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 1397–1402.

[6] N. K. Long, K. Sammut, D. Sgarioto, M. Garratt, and H. A. Abbass, "A comprehensive review of shepherding as a bio-inspired swarm-robotics guidance approach," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 4, pp. 523–537, Aug. 2020.

[7] H. Wang and M. Rubenstein, "Walk, stop, count, and swap: Decentralized multi-agent path finding with theoretical guarantees," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1119–1126, Apr. 2020.

[8] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Trans. Sens. Netw.*, vol. 2, no. 2, pp. 188–220, 2006.

[9] S. Li, H. Sun, and H. Esmaiel, "Underwater tdoa acoustical location based on majorization-minimization optimization," *Sensors*, vol. 20, no. 16, p. 4457, 2020, doi: 10.3390/s20164457.

[10] M. Imperoli, C. Potena, D. Nardi, G. Grisetti, and A. Pretto, "An effective multi-cue positioning system for agricultural robotics," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3685–3692, Oct. 2018.

[11] M. L. Elwin, R. A. Freeman, and K. M. Lynch, "Distributed voronoi neighbor identification from inter-robot distances," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1320–1327, Jul. 2017.

[12] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[13] J. Klingner, N. Ahmed, and N. Correll, "Fault-tolerant covariance intersection for localizing robot swarms," *Robot. Auton. Syst.*, vol. 122, 2019, Art. no. 103306, doi: 10.1016/j.robot.2019.103306.

[14] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *Proc. 2nd Int. Conf. Embedded Networked Sensor Syst.*, 2004, pp. 50–61.

[15] S. M. Trenkwalder, I. Esnaola, Y. K. Lopes, A. Kolling, and R. Groß, "Swarmcom: An infra-red-based mobile ad-hoc network for severely constrained robots," *Auton. Robots*, vol. 44, no. 1, pp. 93–114, 2020.

[16] F. K. Chan and H.-C. So, "Accurate distributed range-based positioning algorithm for wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 4100–4105, Oct. 2009.

[17] M. Basiri, F. Schill, P. Lima, and D. Floreano, "On-board relative bearing estimation for teams of drones using sound," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 820–827, Jul. 2016.

[18] A. Cornejo, A. J. Lynch, E. Fudge, S. Bilstein, M. Khabbazian, and J. McLurkin, "Scale-free coordinates for multi-robot systems with bearing-only sensors," *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1459–1474, 2013.

[19] J. F. Roberts, T. S. Stirling, J.-C. Zufferey, and D. Floreano, "2.5 d infrared range and bearing system for collective robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 3659–3664.

[20] M. Cominelli, P. Patras, and F. Gringoli, "Dead on arrival: An empirical study of the bluetooth 5.1 positioning system," in *Proc. 13th Int. Workshop Wireless Netw. Testbeds*, Experimental Evaluation & Characterization, 2019, pp. 13–20.

[21] J. Garcia, S. Soto, A. Sultana, and A. T. Becker, "Localization using a particle filter and magnetic induction transmissions: Theory and experiments in air," in *Proc. IEEE Texas Symp. WMCS, 2020*, pp. 1–6.

[22] K. Dooley, *Designing Large Scale Lans: Help for Network Designers*. "O'Reilly Media, Inc." 2001.

[23] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a global coordinate system from local information on an Ad Hoc sensor network," in *Inf. Process. Sensor Netw.*. Springer, 2003, pp. 333–348.

[24] R. Spica and P. R. Giordano, "Active decentralized scale estimation for bearing-based localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 5084–5091.

[25] Y. Zou, L. Wang, and Z. Meng, "Distributed localization and circumnavigation algorithms for a multiagent system with persistent and intermittent bearing measurements," *IEEE Trans. Automat. Control.*, 2020, doi: 10.1109/TCST.2020.3032395.

[26] Z. Lin, M. Fu, and Y. Diao, "Distributed self localization for relative position sensing networks in 2 d space," *IEEE Trans. Signal Process.*, vol. 63, no. 14, pp. 3751–3761, Jul. 2015.

[27] L. Carlone and A. Censi, "From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization," *IEEE Trans. Robot*, vol. 30, no. 2, pp. 475–492, Apr. 2014.