Distributed Range-Based Relative Localization of Robot Swarms

Alejandro Cornejo and Radhika Nagpal

Harvard University
School of Engineering and Applied Sciences
Cambridge MA 02138

Abstract. This paper studies the problem of having mobile robots in a multi-robot system maintain an estimate of the relative position and relative orientation of near-by robots in the environment. This problem is studied in the context of large swarms of simple robots which are capable of measuring only the distance to near-by robots.
We compare two distributed localization algorithms with different tradeoffs between their computational complexity and their coordination requirements. The first algorithm does not require the robots to coordinate their motion. It relies on a non-linear least squares based strategy to allow robots to compute the relative pose of near-by robots. The second algorithm borrows tools from distributed computing theory to coordinate which robots must remain stationary and which robots are allowed to move. This coordination allows the robots to use standard trilateration techniques to compute the relative pose of near-by robots. Both algorithms are analyzed theoretically and validated through simulations.

1 Introduction

Most tasks which can be performed effectively by a group of robots require the robots to have some information about the relative positions and orientations of other nearby robots. For example in flocking [1] robots use the relative orientation of its neighbors to control their own heading and the relative position of its neighbors to ensure collision avoidance and group cohesion, in formation control [2] robots control their own position as a function of the relative position of their neighbors to reach a desired configuration, and in mapping [3] robots use the relative position and relative orientation of their neighbors to interpret and fuse the information collected by other robots. However, most of the existing work on localization requires landmarks with known positions on the environment, addresses localization of a single robot, requires complex computations, or relies on expensive sensors. Many environments of interest prevent the use of landmarks, and in swarm platforms, computation is limited and large or costly sensors are not available.

We study the problem of having each robot in a multi-robot system compute the relative pose (position and orientation) of close-by robots relying only on

distance estimates to close-by robots. This paper studies and compares algorithms which are appropriate for large populations of cheap and simple robots. The algorithms described in this paper are fully distributed, and the computations performed at each robot rely only on information available in its local neighborhood. This problem is ongoing, since for any mobile robot, the set of close-by robots and their relative pose changes throughout the execution.

We consider a general problem formulation which does not require explicit control over the motions performed by the robots. Specifically, the first algorithm we consider places no restrictions on the motion whatsoever. The second algorithm coordinates which robots are stationary and which robots are mobile, rotating robots between these roles in a fair and distributed fashion. This allows composing solutions to this problem with motion-control algorithms and implement different higher-level behaviors. Furthermore, we study this problem in a robot swarm setting, which imposes stringent sensor and computational restrictions on the solutions.

In a typical swarm platform, the communication, computation and sensing capabilities of individual robots are fairly limited. The communication limitations of the individual robots in a swarm platform rule out any strategy that requires collecting large amounts of data at hub locations, and yet, the simplicity of the individual robots demand some form of cooperation. Moreover, the computational constraints of individual robots exclude the possibility of storing and updating complex models of the world or other robots.

Therefore, to fully exploit the potential of a robot swarm platform, it is paramount to use decentralized strategies that allow individual robots to coordinate locally to complete global tasks. This is akin to the behavior observed in swarms of insects, which collectively perform a number of complex tasks which are unsurmountable by a single individual, all while relying on fairly primitive forms of local communication.

1.1 Related work

For the most part, existing work on multi-robot localization requires either stationary landmarks in the environment or the ability for the robots to measure something other than just the distance to their neighbors. More importantly, most of the existing localization algorithms are tailored for small groups of capable robots, and place an emphasis on detailed error models to prevent drift over time. We briefly describe some of the more relevant works in the paragraphs below. In contrast, this paper addresses the problem of providing relative localization service for large groups of very simple robots which can only sense the distance of close-by robots. In this setting drift of the estimates is less of a concern, since the information is meant to be used for simple position control, and not to perform path integration over longer time intervals. Concretely our goal is to allow robots to approximate the relative pose of their neighbors using only a couple of communication rounds and performing as little computation as possible.

The problem of localization using distance-only sensors has received a lot of attention, most of it focusing on landmark- or anchor-based localization. Using only connectivity information to stationary landmarks with known positions [4], it is possible to approximate the position of mobile nodes. When distance measurements to the landmarks with known positions are available (for example, via ultrasound) the Cricket Location-Support System is able to localize mobile nodes within predefined regions, and extending a similar setup it has been show how to obtain finer grained position information [5]. The more general case of fixed stationary landmarks with unknown initial positions has also been considered in the literature [6, 7].

The robust quadrilaterals algorithm [8], which is based on rigidity theory, is one of the few landmark-free localization methods that relies only on distance sensing between robots, and is the closest in spirit to the present work. However, the robust quadrilaterals algorithm was designed primarily for static sensor networks and cannot recover the relative orientation of the robots. More recently, global optimal solutions to this problem have also been proposed [9] which formulate localization as a weighted least squares estimation problem, and present algorithms in the same spirit to the first algorithm described in this paper.

There is also a large body of work on the problem of cooperative localization. One of the earliest works on cooperative localization [10] required bearing and (optionally) range-sensors and advocated for an approach where robots are divided into a group that is allowed to move and use odometry, and another group which plays the role of stationary landmarks. The work in [11] described a similar approach using range-sensors but requiring global all-to-all communication and sensing towards the landmarks. Both of these approaches neglect the distributed coordination problem of selecting which robots play the role of stationary beacons and which robots are allowed to move.

In the context of simultaneous localization and mapping, a Monte-Carlo Localization (MCL) approach shown to boost the accuracy of localization through cooperation of two or more robots [12]. Extended Kalman-Filters (EKF) have also been used with a similar effect [13]. Both of these works considered robots that have sensors capable of measuring the angle and distance to other robots, as well as sensors to sense the environment.

The MCL and EKF approaches have been subsequently extended and improved in recent works. For example[14] extended the EKF approach described in [13] to consider weaker forms of sensors, including distance-only sensing, and [15] described how to reduce the amount of state and communication required. The computational complexity of the EKF was further reduced in [16], and a communication-bandwidth aware solution was described in [17]. Similarly novel techniques to reduce the computational costs of the MCL approach have been proposed, for example [18] described a clustering technique to minimize the amount of state and communication required.

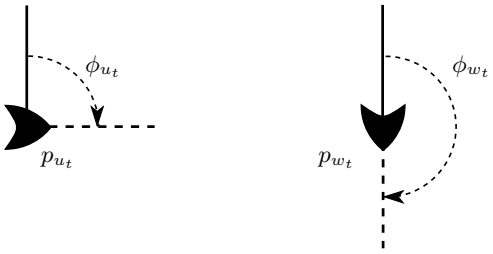


Fig. 1: In a global coordinate system u is pointing right and w is pointing down. In robot u's local coordinate system robot p_{w_t} is in front of robot p_{u_t}, and in robot w's local coordinate system robot p_{u_t} is to the right of robot p_{w_t}.

2 System Model

Let V be a collection of robots deployed in a planar environment. The *pose* (aka kinematic state) of robot $u \in V$ at time $t \in \mathbb{R}^+$ is described by a tuple $pose_{u_t} = \langle p_{u_t}, \phi_{u_t} \rangle$ where $p_{u_t} \in \mathbb{R}^2$ represents the *position* of robot u at time t, and $\phi_{v_t} \in [0, 2\pi)$ represents the *orientation* of robot u at time t. Robots do *not* know their position or orientation.

Each robot has its own local coordinate system which changes as a function of its pose. Specifically, at time t the local coordinate system of robot u has the origin at its own position p_{u_t} and has the x-axis aligned with its own orientation ϕ_{u_t}. All sensing at a robot is recorded in its local coordinate system (cf. Fig 1).

For $\theta \in [0, 2\pi)$ let R_θ and $\psi(\theta)$ denote rotation matrix of θ and a unit vector of angle θ. The position of robot w at time t' in the local coordinate system of robot u at time t is defined as $p_{w_{t'}}|_{u_t} = R_{-\phi_{u_t}}(p_{w_{t'}} - p_{u_t}) = \left\|p_{w_{t'}} - p_{u_t}\right\| \psi(\theta_{w_{t'}}|_{u_t})$, and the orientation of robot w at time t' in the local coordinate system of robot u at time t is defined as $\phi_{w_{t'}}|_{u_t} = \phi_{w_{t'}} - \phi_{u_t}$. Hence the pose of robot w at time t' in the local coordinate system of robot u at time t is described by the tuple $pose_{w_{t'}}|_{u_t} = \left\langle p_{w_{t'}}|_{u_t}, \phi_{w_{t'}}|_{u_t} \right\rangle$.

The communication graph at time t is a directed graph $G_t = (V, E_t)$, where $E_t \subseteq V \times V$ as a set of of directed edges such that $(u, v) \in E_t$ if and only if a message sent by robot u at time t is received by robot v. The neighbors of robot u at time t are the set of robots from which u can receive a message at time t, denoted by $N_{u_t} = \{v \mid (v, u) \in E_t\}$.

For simplicity and ease of exposition, it is assumed that computation, communication and sensing proceeds in synchronous lock-step rounds $\{1, 2, \ldots\}$. In practice synchronizers [19] can be used to simulate perfect synchrony in any partially synchronous system. If robot u receives a message from robot w at round i then robot u can identify the message originated from w, and estimate the distance $\|p_{v_i} - p_{w_i}\| = d_i(u, w)^*$.

*Many swarm of platforms, including the Kilobots[20], use the same hardware (i.e., infrared transceivers) as a cost-effective way to implement both communication and sensing.

Robots are capable of using odometry to estimate their pose change between rounds in their own local coordinate system. Specifically at round j a robot $u \in V$ can estimate its translation change $p_{u_i}|_{u_j}$ with respect to round $i < j$ and its orientation change $\phi_{u_i}|_{u_j}$ with respect to round $i < j$. It is assumed that odometry estimates are reliable over intervals of two or three rounds (i.e. $i >= j - 3$), but suffer from drift over longer time intervals.

2.1 Problem Formulation

Formally, the problem statement requires that at every round i, each robot u computes the relative pose $pose_{w_i}|_{u_i}$ of every neighboring robot $w \in N_{u_i}$. Robots can only perceive each other through distance sensing. For a robot u to compute the pose of a neighboring robot w at a particular round, it must rely on the distance measurements and communication graph in the previous rounds, as well as the odometry estimates of u and w in previous rounds.

The algorithms considered do *not* require controlling the motion performed by each robot; the first algorithm imposes no constraints, and the second algorithm requires only to coordinate when robots can move, but not the motion they execute. This allows these algorithms to be run concurrently with any motion control algorithm. Moreover, the algorithms are tailored for large swarms of simple robots, and as such the size of the messages or the computation requirements do not depend on global parameters such as the size or diameter of the network.

3 Localization without Coordination

This section describes a distributed localization algorithm that requires no motion coordination between robots and uses minimal communication. Each robot localizes its neighbors by finding the solution to a system of non-linear equations. For simplicity, this section assumes that distance sensing and odometry estimation is perfect (e.g. noiseless); a similar treatment is possible if considering zero-mean Gaussian noise. Section 5 describes how the results presented here can be easily extended to handle noisy measurements.

Consider any pair of robots a and b for a contiguous interval of rounds $I \subset \mathbb{N}$. To simplify notation let $p_{a_j \to b_k} = p_{b_k} - p_{a_j}$ denote the vector, in the global coordinate system, that starts at p_{a_j} and ends at p_{b_k}.

Observing Fig. 2 it is easy to see that starting at p_{a_i} (and in general starting at any p_{a_j} for some $j < k$) there are at least two ways to arrive to p_{b_k}. For instance, by first traversing a dotted line and then a solid line or vice versa. Indeed, this holds since by definition for all $j \leq k$ we have:

$$p_{a_j \to a_k} + p_{a_k \to b_k} = p_{a_j \to b_k} = p_{a_j \to b_j} + p_{b_j \to b_k}. \tag{1}$$

For $j = k$ equation (1) is vacuously true and for $j < k$ it represents a constraint on the relative pose of robots a and b in terms of quantities that individual

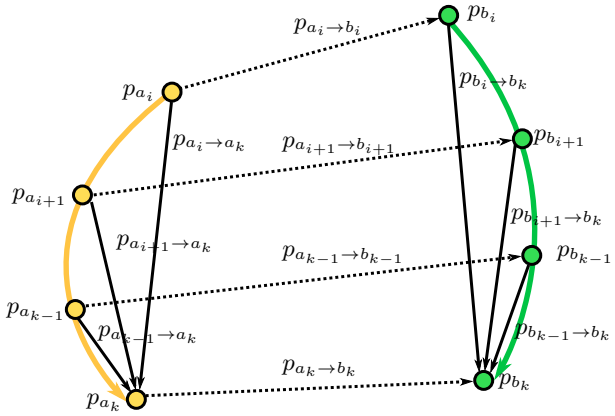


Fig. 2: Robot a and b in rounds $I = \{i, \ldots, k\}$.

robots can either sense or compute. Next, we massage the previous equation to represent a constraint on the relative pose in terms of known quantities.

$$
\begin{aligned}
p_{a_j \to a_k} - p_{b_j \to b_k} + p_{a_k \to b_k} &= p_{a_j \to b_j} \\
-R_{\phi_{a_k}} p_{a_j}|_{a_k} + R_{\phi_{b_k}} p_{b_j}|_{b_k} + R_{\phi_{a_k}} p_{b_k}|_{a_k} &= R_{\phi_{a_j}} p_{b_j}|_{a_j} \\
p_{a_j}|_{a_k} + R_{\phi_{b_k}|_{a_k}} p_{b_j}|_{b_k} + p_{b_k}|_{a_k} &= R_{\phi_{a_j} - \phi_{a_k}} p_{b_j}|_{a_j} \\
\left\| p_{a_j}|_{a_k} + R_{\phi_{b_k}|_{a_k}} p_{b_j}|_{b_k} + p_{b_k}|_{a_k} \right\| &= \left\| p_{b_j}|_{a_j} \right\| \\
\left\| -p_{a_j}|_{a_k} + R_{\phi_{b_k}|_{a_k}} p_{b_j}|_{b_k} + d_k(a,b)\psi(\theta_{b_k}|_{a_k}) \right\| &= d_j(a,b)
\end{aligned} \tag{2}
$$

Dissecting equation (2); $d_j(a,b)$ and $d_k(a,b)$ are known and correspond to the estimated distance between robot a and b at round j and k respectively; $p_{a_j}|_{a_k}$ and $p_{b_j}|_{b_k}$, are also known, and correspond to the odometry estimates from round j to round k taken by robot a and b respectively; finally $\phi_{b_k}|_{a_k}$ and $\theta_{b_k}|_{a_k}$ are both unknown and correspond to the relative position and orientation of robot b at round k in the local coordinate system of robot a at round k.

Considering equation (2) over a series of rounds yields a non-linear system that, if well-behaved, allows a robot to estimate the relative pose of another. To avoid an undetermined system we require at least two equations, since there are two unknowns. In practice we observed that even when the measurements are noisy, the additional information provided by the overconstrained system does not provide improvements to merit the additional computational cost, even when the measurements are noisy.

The following distributed algorithm leverages the constraints captured by a system of $\delta \geq 2$ equations to allow every robot to compute the relative pose of its neighbors.

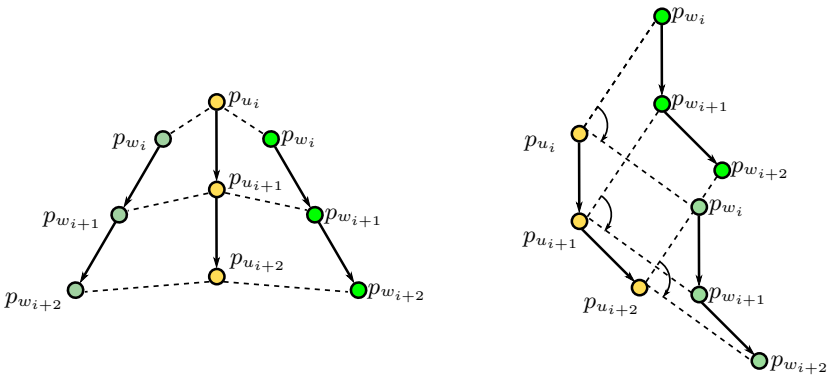
At each round of Algorithm 1 every robot sends a constant amount of information (its odometry measurements for that round) and therefore its message

Algorithm 1 Localization without Coordination

1: **for each** robot $u \in V$ and every round $k \in \{1, \ldots\}$ **do**
2: **broadcast** $\left\langle p_{u_{k-1}}|_{u_k}, \phi_{u_{k-1}}|_{u_k} \right\rangle$
3: **receive** $\left\langle p_{w_{k-1}}|_{w_k}, \phi_{w_{k-1}}|_{w_k} \right\rangle$ for $w \in N_{u_k}$
4: $I = \{k - \delta, k\}$
5: **for each** $w \in \bigcap_{j \in I} N_{u_j}$ **do**
6: integrate odometry $p_{u_j}|_{u_k}, \phi_{u_j}|_{u_k}$ for $j \in I$
7: find $\hat{\theta}_{w_k}|_{u_k}, \hat{\phi}_{w_k}|_{u_k}$ such that (2) holds $\forall j \in I$
8: $pose_{w_k}|_{u_k} \leftarrow \left\langle d_k(u, w)\psi(\hat{\theta}_{w_k}|_{u_k}), \hat{\phi}_{w_k}|_{u_k} \right\rangle$

complexity is $O(1)$. The computational complexity of Algorithm 1 is dominated by solving the system of non-linear equations (line 7), which can be done by numerical methods [21] in $O(\varepsilon^{-2})$ where ε is the desired accuracy.

Regardless of the choice of δ there are motion patterns for which any algorithm that does not enforce a very strict motion coordination (which includes Algorithm 1, which enforces no motion coordination) cannot recover the relative pose of neighboring robots. These motions are referred to as *degenerate*, and are described next (see Fig. 3). First, if during δ rounds two robots follow a linear trajectory, then the relative pose between these robots can only be recovered up to a *flip ambiguity*. Second, if during δ rounds one robot follows a displaced version of the trajectory followed by another robot, then it is possible to infer the relative orientation of the robots, but a *rotation ambiguity* prevents the recovery of the relative position. A degenerate motion can be a flip ambiguity, a rotation ambiguity, or a combination of both (cf. Fig 3).



(a) Flip Ambiguity (b) Rotation Ambiguity

Fig. 3: Due to generate motions yellow (light gray) robot cannot fully resolve the relative position of green (dark gray) robot.

Fortunately degenerate motions are rare. More precisely degenerate motions are a set of measure zero (for example, this implies that if the motions are random, then with probability 1 they are not degenerate). This can be shown

to be a consequence of the generic rigidity of a triangular prism in Euclidean 2-space, see [22] for a thorough treatment of rigidity. The next theorem formalizes the properties of the algorithm (all proofs were omitted due to lack of space).

Theorem 1 *If at round i, robots u and w have been neighbors for a contiguous interval of δ or more rounds, and perform non-degenerate motions, then at round i Algorithm 1 computes $pose_{w_i}|_{u_i}$ at u and $pose_{u_i}|_{w_i}$ at w.*

4 Localization with Coordination

This section describes a distributed localization algorithm that uses a simple stop/move motion coordinate scheme, and requires communication proportional to the number of neighbors. Using the aforementioned motion coordination scheme allows robots to compute the relative pose of neighboring robots through trilateration.

By collecting multiple distance estimates a moving robot can use trilateration to compute the relative position of a stationary robot; as before standard techniques can be used to extend this to the case of zero-mean noise, briefly detailed in Section 5. Two such distance estimates already suffice to allow the moving robot to compute the relative position of a stationary robot up to a flip ambiguity (i.e., a reflection along the line that passes through the coordinates at which the measurements were taken).

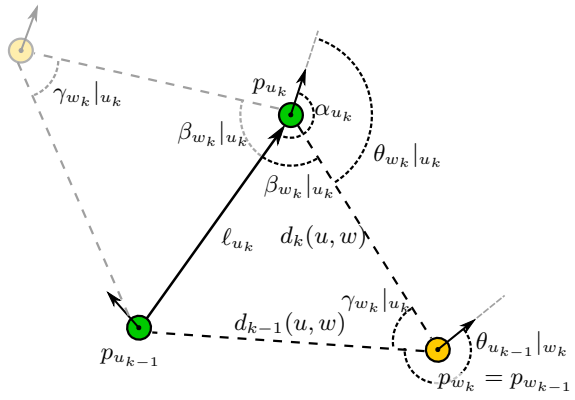


Fig. 4: Moving robot (green/dark gray) uses trilateration to compute the relative position of stationary robot (yellow/light gray) up to a flip ambiguity.

Consider two neighboring robots u and w where from round $k-1$ to round k robot u moves while robot w remains stationary (see Fig. 4). Robot u can compute the relative position $p_{w_k}|_{u_k}$ of robot w at round k up to a flip ambiguity, relying only on the distance measurements to robot w at round $k-1$ and round k, and its own odometry for round k. Specifically the cosine law yields the following.

$$\ell_{u_k} = \left\| p_{u_{k-1}}|_{u_k} \right\| \qquad \alpha_{u_k} = \measuredangle(p_{u_{k-1}}|_{u_k})$$

$$\beta_{w_k}|_{u_k} = \cos^{-1}\left(\frac{\ell_{u_k}^2 + d_k^2(u,w) - d_{k-1}^2(u,w)}{2\ell_{u_k} d_k(u,w)}\right) \tag{3}$$

$$\gamma_{w_k}|_{u_k} = \cos^{-1}\left(\frac{d_k^2(u,w) + d_{k-1}^2(u,w) - \ell_{u_k}^2}{2d_k(u,w)d_{k-1}(u,w)}\right) \tag{4}$$

$$\theta_{w_k}|_{u_k} = \alpha_{u_k} \mp \beta_{w_k}|_{u_k} \tag{5}$$

$$\theta_{u_k}|_{w_k} = \theta_{u_{k-1}}|_{w_k} \pm \gamma_{w_k}|_{u_k} \tag{6}$$

In order for robot u to fully determine the relative pose of robot w at round k (ignoring the flip ambiguity) it remains only to compute $\phi_{w_k}|_{u_k}$. Observe that given knowledge of $\theta_{u_{k-1}}|_{w_k}$, robot u can leverage Eq. 6 to compute $\theta_{u_k}|_{w_k}$ using the correction term $\gamma_{w_k}|_{u_k}$ computed through the cosine law. The following identity can be leveraged to easily recover $\phi_{w_k}|_{u_k}$ using $\theta_{u_k}|_{w_k}$ and $\theta_{w_k}|_{u_k}$.

$$\phi_{u_k}|_{w_k} = \theta_{w_k}|_{u_k} - \theta_{u_k}|_{w_k} + \pi \pmod{2\pi} \tag{7}$$

Summing up, if robot u moves from round $k-1$ to round k while robot w remains stationary, then using $d_{k-1}(u,w)$, $d_k(u,w)$ and $p_{u_{k-1}}|_{u_k}$ robot u can compute the relative position of robot w at time k. If knowledge of $\theta_{u_{k-1}}|_{w_k}$ is available robot u can also compute the relative orientation of robot w at time k. Both the position and orientation are correct up to a flip ambiguity.

A robot can resolve the flip ambiguity in position and orientation by repeating the above procedure and checking for consistency of the predicted position and orientation. We refer to motions which preserve symmetry and therefore prevent the flip ambiguity from being resolved (for instance, collinear motions) as degenerate (cf. Fig 5).

(a) Flip Ambiguity (b) Unambiguous

Fig. 5: Moving robot (yellow/light gray) localizing a stationary robot (green/dark gray) using distance measurements (dashed lines) and odometry (solid arrows).

Observe that the distance measurements between a stationary robot and a moving robot are invariant to rotations of the moving robot around the sta-

tionary robot (cf. Fig 6). This prevents a stationary robot from recovering the relative position of a moving neighbor using any number of distance estimates.

However, in order for robot u to recover the orientation of robot w, robot w —which remains stationary from round $k-1$ to round k— must compute $\theta_{u_{k-1}}|_{w_{k-1}} = \theta_{u_{k-1}}|_{w_k}$ and communicate it to robot u by round k.

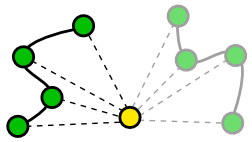


Fig. 6: Stationary robot (yellow) cannot compute the relative position of the moving robot (green), since all distance measurements (dashed lines) are invariant to rotations around the stationary robot.

Therefore, in order to leverage the previous trilateration procedure requires coordinating the motion of the robots in a manner that gives every robot a chance to move and ensures that when a robot is moving its neighbors remain stationary. Formally, a *motion-schedule* is an algorithm that at each round classifies every robots as being either mobile or stationary. A motion-schedule is *well-formed* if at every round i the set of robots classified as mobile define an independent set of the communication graph G_i (i.e. no two mobile robots are neighbors). The *length* of a motion-schedule is the maximum number of rounds that any robot must wait before it is classified as mobile. A motion-schedule is *valid* if it is well-formed and has finite length.

The validity of a motion-schedule ensures that mobile robots can use trilateration to compute the relative positions of all its neighbors, and having a motion-schedule of finite length guarantees every robot gets a chance to move. The next subsection provides a description of a distributed algorithm that produces a valid motion-schedule. Algorithm 2 describes a distributed localization algorithm that leverages a valid motion-schedule and trilateration.

At each round of Algorithm 2 every robot sends a message containing its own odometry estimates and $\Theta_{u_{k-1}}$, which is the set of previous position estimates (one for each of its neighbor), and therefore its message complexity is $O(\Delta)$. Mobile robots use trilateration to compute the relative position and relative orientation of its neighbors, and when possible stationary robots update the relative position and orientation of mobile robots using the received odometry estimates. In either case, the amount of computation spent by Algorithm 2 to localize each robot is constant.

Theorem 2 *(Assuming a valid motion-schedule.) If at round i, robots u and w have been neighbors for a contiguous set of rounds during which robot u performed a non-degenerate motion, then at round i Algorithm 2 computes $pose_{w_i}|_{u_i}$ at u.*

Algorithm 2 Localization with Coordination

1: $\Theta_{u_0} \leftarrow \emptyset \; \forall u \in V$
2: **for each** robot $u \in V$ and every round $k \in \{1, \ldots\}$ **do**
3: **broadcast** $\left\langle p_{u_{k-1}}|_{u_k}, \phi_{u_{k-1}}|_{u_k}, \Theta_{u_{k-1}} \right\rangle$
4: **receive** $\left\langle p_{w_{k-1}}|_{w_k}, \phi_{w_{k-1}}|_{w_k}, \Theta_{u_{k-1}} \right\rangle$ for $w \in N_{u_k}$
5: **if** state = mobile **then**
6: $\Theta_{u_k} \leftarrow \left\{\hat{\theta}_{w_k}|_{u_k} \text{ through Eq. (4-5)}\right\}$
7: $\hat{\phi}_{w_k}|_{u_k} \leftarrow$ use Eq. (6-7) $\forall w \in N_{u_k}$
8: use previous state resolve *flip* in Θ_{u_k}
9: **else**
10: update Θ_{u_k} through $\phi_{w_k-1}|_{w_k}, p_{w_{k-1}}|_{w_k}$
11: $\hat{pose}_{w_k}|_{u_k} \leftarrow \left\langle d_k(u,w)\psi(\hat{\theta}_{w_k}|_{u_k}), \hat{\phi}_{w_k}|_{u_k} \right\rangle \; \forall w \in N_{u_k}$
12: state $\leftarrow$ MOTION-SCHEDULER
13: **if** state = mobile **then**
14: move according to MOTION-CONTROLLER
15: **else**
16: remain stationary

4.1 Motion Scheduling

As a straw-man distributed algorithm that requires no communication and outputs a valid motion-schedule, consider an algorithm that assigns a single mobile robot to each round, in a round robin fashion (i.e. at round i let robot $k = i \mod n$ be mobile and let the remaining $n-1$ robots be stationary). Although the motion-schedule produced by such an algorithm is valid, it is not suited for a swarm setting, since it exhibits no parallelism and the time required for a robot to move is linear on the number of robots.

Finding a motion-schedule that maximizes the number of mobile robots at any particular round is tantamount to finding a maximum independent set (aka MaxIS) of the communication graph, which is NP-hard. Similarly, finding a motion-schedule with minimal length implies finding a vertex-coloring with fewest colors of the communication graph, which is also NP-hard.

Algorithm 3 describes a motion-schedule with the more modest property of having the set of moving robots at each round define a maximal independent set (aka MIS) of the communication graph. Once a robot is classified as being mobile, it does not participate on subsequent MIS computations, until each of its neighbors has also been classified as mobile. Given these properties, it is not hard to show that for any robot u and a round k, the number of rounds until robot u is classified as mobile is bounded by the number of neighbors of robot u at round k.

Theorem 3 *Algorithm 3 defines a valid motion-schedule with length* $\Delta + 1$.

The description of Algorithm 3 utilizes a distributed MIS algorithm as a subroutine (line 4 in the pseudo-code). However, it should be noted that the

Algorithm 3 Motion-Scheduler

```
1: if ∀w ∈ N_u state_w = inactive then
2:     state_u ← compete
3: if state_u = compete then
4:     if u is selected in distributed MIS then
5:         state_u ← inactive
6:         output mobile
7: output stationary
```

problem of finding an MIS with a distributed algorithm is a fundamental symmetry breaking problem and is far from trivial. Fortunately, the MIS problem has been studied extensively by the distributed computing community, and extremely efficient solutions have been proposed under a variety of communication models [23–25]. The classic solution [23] requires $O(\log n)$ communication rounds and every node uses a total of $O(\log n)$ [26] bits of communication. For a wireless network settings, it is known [24] how to find an MIS exchanging at most $O(\log^* n)^\dagger$ bits. Due to lack of space, for the purposes of this paper it should suffice to know that it is possible to implement a distributed MIS protocol in the lower communication layers without significant overhead.

5 Algorithm Evaluation

This section evaluates the performance of the proposed localization algorithms considering that both the distance measurements and the odometry estimates are subject to noise. We use a simulator environment tailored to closely resemble the physical characteristics of the Kilobot swarm platform.

Specifically we assume the distance measurements of each robot are subject to independent zero-mean Gaussian noise with variance σ_d and the odometry estimates is subject to two independent sources of noise; the orientation component is subject to zero-mean Guassian noise with variance σ_ϕ, and the translation component is subject to a two-dimensional symmetric zero-mean Guassian noise with variance σ_{xy}. We do not use the standard noise assumptions on the odometry model, since our odometry model is modeling the noise present in the external overhead computer vision system used to provide odometry on the Kilobot swarm platform (the stick-slip locomotion used by the Kilobot swarm produces movements that depend on the imperfections of the surface underneath each robot, so they cannot have odometry built-in).

Algorithm 1 relied on finding a zero in a non-linear system of equations constructed using the distance estimates and odometry estimates pertinent to that robot. When these estimates are subject to noise, the corresponding non-linear

†The iterated logarithm function counts the number of times the logarithm is applied to the argument before the result is less or equal to 1. It is an extremely slowly growing function, for instance the iterated logarithm of the number of atoms in the universe is less than 5.

system is no longer guaranteed to have a zero. To cope with noisy measurements it suffices to instead look for the point that minimizes the mean-squared error. This incurs in no additional computational overhead, since it can be accomplished using the same numerical methods used in the noiseless case.

The length of each simulation trial is 20 rounds of 6 second (2 minutes). A total of 50 trials were carried out for each different combination of noise parameters. In each trial, 20 robots are deployed randomly in a region of $10m \times 10m$, and at each round each robot is allowed to perform a motion with a random orientation change between $[-\pi/4, \pi/4]$ and a translation change which is normally distributed with a mean of 3m and a variance of 0.5m. The length of each trial is 20 rounds of 6 second (2 minutes). The plots below (cf. Figs 7 and 8) show the mean squared error (MSE) in the computed position (blue) and orientation (red) over 50 random trials for various different noise parameters. Since to initialize the position and orientation estimates Algorithm 2 requires at least three rounds, the first three rounds of every trial were discarded.

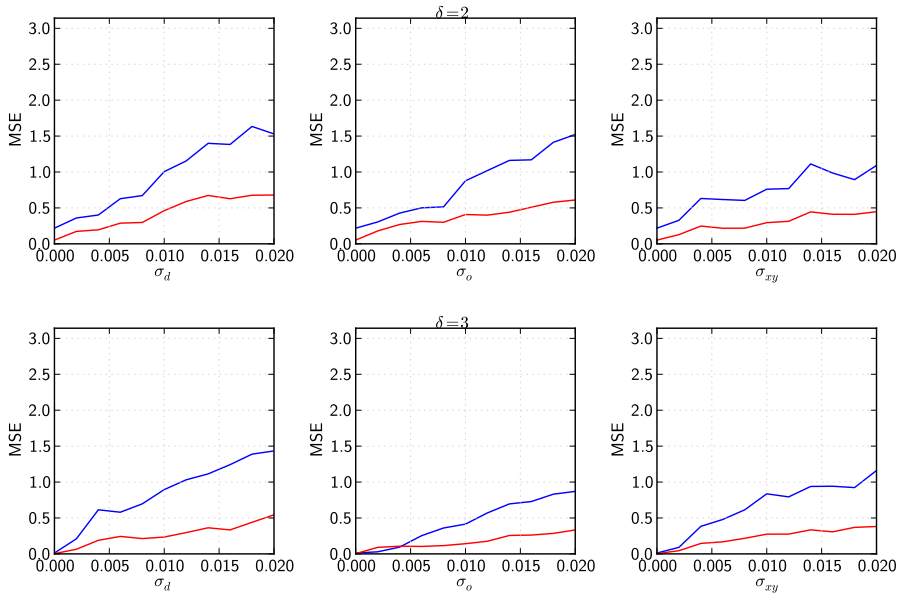


Fig. 7: Each plot shows MSE of the position (blue) and orientation (red) as a function of one component of the variance Σ. The vertical axis goes from 0 to π. From left to right, each column shows the MSE as a function of σ_d, σ_o and $\sigma_{x,y}$. The top row shows the results with $\delta = 2$ and the bottom row for $\delta = 3$.

Not surprisingly the results produced by Algorithm 1 are sensitive to errors in all axis, although it is slightly more robust to errors in the translation odometry than in the distance sensing. Furthermore, the relative orientation estimate was consistently more tolerant to noise than the position estimate. As it would be

expected, for all the different noise settings, increasing the parameter δ from 2 to 3 consistently reduced the MSE in both position and orientation produced by Algorithm 1. However, increasing δ also increases the computational costs of the algorithm and only gives diminishing returns.

Algorithm 2 is evaluated with the same parameters as Algorithm 1 with one exception; to keep the number of motions per trial for Algorithm 1 and Algorithm 2 roughly the same, the length of the trial was doubled to 40 rounds, since at each round, for every pair of nodes, only one of them will be mobile and the other will remain stationary.

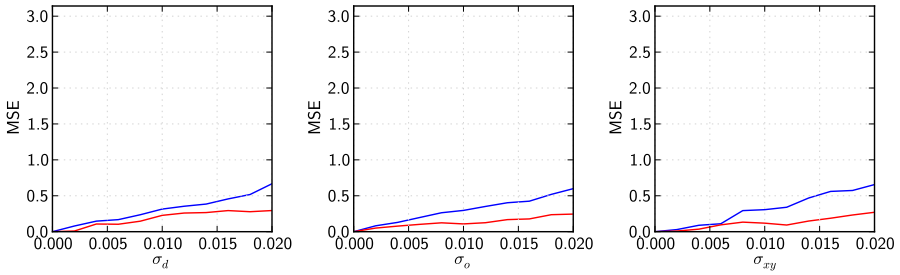


Fig. 8: Plots show MSE of the position (blue) and orientation (red) as a function of one component of the variance Σ. The vertical axis goes from 0 to π. From left to right, each column shows the MSE as a function of σ_d, σ_o and $\sigma_{x,y}$.

The pose estimates produced by Algorithm 2 are for the most part equally affected by noise in either of the dimension. As it was the case with Algorithm 1, the relative orientation estimate was consistently more tolerant to noise than the position estimate. Overall compared to Algorithm 1, the results show that Algorithm 2 is in all respects less sensitive to noise. This, together with its computational simplicity, make it more suitable for implementation on the Kilobot platform.

5.1 Motion Control and Localization

Here we explore the feasibility of composing existing motion control algorithms with the proposed localization algorithms. For its simplicity we consider the canonical problem of flocking [1]. Informally, flocking describes an emergent behavior of a collection of agents with no central coordination that move cohesively despite having no common a priori sense of direction.

Flocking behavior has received a lot of attention in the scientific community. Vicsek et al. [27] studied flocking from a physics perspective through simulations and focused on the emergence of alignment in self-driven particle systems. Flocking has also been studied from a control theoretic perspective, for example in [28, 29], where the emphasis is on the robustness of the eventual alignment process despite the local and unpredictable nature of the communication.

We study a flocking behavior where each robot aligns its heading with its neighbors and avoids colliding with close by neighbors. Namely, at each round every robot steers its own orientation to the average orientation of its neighbors, adjusting its speed to avoid getting to close to any of its neighbors. It has been shown [28, 29] that under very mild assumptions this converges to a state where all robots share the same orientation.

Fig. 9 shows the results of the described average-based flocking algorithm when combined with Algorithm 1 to provide relative orientation estimates. Initially the first rounds the robots move erratically while the position and orientation estimates are initialized, and soon after the orientations of all the robots converge. Increasing the error in the distance sensing and odometry measurements is translated in greater inaccuracy in the resulting relative orientation estimates, which affects the resulting flocking state.

Fig. 9: Final configuration of 6 robots after four 40 round runs of a flocking algorithm composed with Algorithm 1. From left to right the variance of all noise parameters was increased with same starting configuration.

Before the swarm reaches the steady state the distance measurements can be used to localize, and localization becomes impossible only when adjustments are no longer needed and the swarm is in steady state.

6 Conclusions and Future Work

We considered two distributed algorithms to solve the relative localization problem tailored for swarms of simple robots. The algorithms have different communication and computational requirements, as well as different robustness to sensing errors. Specifically, having greater communication and coordination allows us to reduce the required computational complexity and increase the robustness to sensing errors. In future work, we hope to further study whether this trade-off is inherent to the problem or not.

We are currently implementing the described algorithms on the Kilobot swarm platform. The Kilobot platform has no floating point unit and limited program memory (30k), as well as very limited bandwidth (24 bytes per second). Thus, even simple algorithms require careful tunning and optimization of all parameters in order to be implemented on the Kilobots. We are also investigating algorithms with fewer communication requirements.

References

[1] A. Turgut et al. “Self-organized flocking in mobile robot swarms”. In: *Swarm Intelligence* 2.2-4 (2008), pp. 97–120.

[2] T. Balch and R.C. Arkin. “Behavior-based formation control for multirobot teams”. In: *Transactions on Robotics and Automation (TRA)* 14.6 (1998), pp. 926–939.

[3] S. Thrun, W. Burgard, and D. Fox. “A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping”. In: *International Conference on Robotics and Automation (ICRA)*. Vol. 1. 2000, pp. 321–328.

[4] J. Luo and Q. Zhang. “Relative distance based localization for mobile sensor networks”. In: *Global Telecommunications Conference (GlobeCom)*. 2007, pp. 1076–1080.

[5] A. Savvides, C. Han, and M. Strivastava. “Dynamic fine-grained localization in ad-hoc networks of sensors”. In: *International Conference on Mobile Computing and Networking (MobiCom)*. 2001, pp. 166–179.

[6] J. Djugash et al. “Range-only slam for robots operating cooperatively with sensor networks”. In: *International Conference on Robotics and Automation (ICRA)*. 2006, pp. 2078–2084.

[7] E. Olson, J. Leonard, and S. Teller. “Robust range-only beacon localization”. In: *Journal of Oceanic Engineering* 31.4 (2006), pp. 949–958.

[8] D. Moore et al. “Robust distributed network localization with noisy range measurements”. In: *International Conference on Embedded Networked Sensor Systems (SenSys)*. 2004, pp. 50–61.

[9] Nikolas Trawny and Stergios I Roumeliotis. “On the global optimum of planar, range-based robot-to-robot relative pose estimation”. In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2010, pp. 3200–3206.

[10] R. Kurazume, S. Nagata, and S. Hirose. “Cooperative positioning with multiple robots”. In: *International Conference on Robotics and Automation (ICRA)*. 1994, 1250–1257 vol.2.

[11] L. Navarro-Serment and P. Paredis C. Khosla. “A beacon system for the localization of distributed robotic teams”. In: *International Conference on Field and Service Robotics (FSR)*. Vol. 6. 1999.

[12] D. Fox et al. “A Probabilistic Approach to Collaborative Multi-Robot Localization”. In: *Autonomous Robots* 8.3 (2000), pp. 325–344.

[13] S.I. Roumeliotis and George A. Bekey. “Collective localization: a distributed Kalman filter approach to localization of groups of mobile robots”. In: *International Conference on Robotics and Automation (ICRA)*. Vol. 3. 2000, pp. 2958–2965.

[14] A. Martinelli, F. Pont, and R. Siegwart. “Multi-Robot Localization Using Relative Observations”. In: *International Conference on Robotics and Automation (ICRA)*. 2005, pp. 2797–2802.

[15] K.Y.K. Leung, T.D. Barfoot, and H.H.T. Liu. “Decentralized Localization of Sparsely-Communicating Robot Networks: A Centralized-Equivalent Approach”. In: *IEEE Transactions on Robotics* 26.1 (2010).

[16] L.C. Carrillo-Arce et al. “Decentralized multi-robot cooperative localization using covariance intersection”. In: *Intelligent Robots and Systems (IROS)*. 2013, pp. 1412–1417.

[17] E.D. Nerurkar and S.I. Roumeliotis. “A communication-bandwidth-aware hybrid estimation framework for multi-robot cooperative localization”. In: *Intelligent Robots and Systems (IROS)*. 2013, pp. 1418–1425.

[18] A. Prorok, A. Bahr, and A. Martinoli. “Low-cost collaborative localization for large-scale multi-robot systems”. In: *International Conference on Robotics and Automation (ICRA)*. 2012, pp. 4236–4241.

[19] B. Awerbuch. “Complexity of network synchronization”. In: *Journal of the ACM (JACM)* 32.4 (1985), pp. 804–823.

[20] M. Rubenstein, C. Ahler, and R. Nagpal. “Kilobot: A low cost scalable robot system for collective behaviors”. In: *International Conference on Robotics and Automation (ICRA)*. 2012, pp. 3293–3298.

[21] K. Ueda and N. Yamashita. “On a Global Complexity Bound Levenberg-Marquardt Method”. In: *Journal of Optimization Theory and Applications* 147.3 (2010), pp. 443–453.

[22] B. Servatius and H. Servatius. *Generic and abstract rigidity*. 1999.

[23] M. Luby. “A Simple Parallel Algorithm for the Maximal Independent Set Problem”. In: *SIAM Journal on Computing* 15 (1986), pp. 1036–1053.

[24] J. Schneider and R. Wattenhofer. “A Log-Star Maximal Independent Set Algorithm for Growth-Bounded Graphs”. In: *International Symposium on Principles of Distributed Computing (PODC)* (2008).

[25] Y. Afek et al. “Beeping a Maximal Independent Set”. In: *Internatial Symposium on Distributed Computing (DISC)*. 2011, pp. 32–50.

[26] Y. Métivier et al. “An Optimal Bit Complexity Randomized Distributed MIS Algorithm”. In: *Colloquim on Structural Information and Communication Complexity (SIROCCO)* (2009).

[27] T. Vicsek et al. “Novel type of phase transition in a system of self-driven pinproceedingss”. In: *Physical Review Letters*. Vol. 75. 6. 1995, p. 1226.

[28] R. Olfati-Saber. “Flocking for multi-agent dynamic systems: Algorithms and theory”. In: *IEEE Transactions on Automatic Control*. Vol. 51. 3. 2006, pp. 401–420.

[29] A. Jadbabaie, J. Lin, and A.S. Morse. “Coordination of groups of mobile autonomous agents using nearest neighbor rules”. In: *IEEE Transactions on Automatic Control*. Vol. 48. 6. 2003, pp. 988–1001.