Distributed swarm localization using inter-agent ranging

Mark Halka
University of Waterloo
mahalka@uwaterloo.ca

Abstract

A robot swarm has many potential advantages over a single complicated robot, including robustness, adaptability, efficiency, and cost. However, to be able to execute complex tasks the swarm often needs a consistent global coordinate system. This problem is made difficult by the limited sensing, processing and communication capabilities of each individual robot. Especially in the face of miniaturization, techniques using large, expensive or complicated sensors such as GPS or cameras become unavailable. In this paper, we implement a robust, decentralized localization algorithm in a robot swarm using only intra-distance measurements. We show the robustness of the algorithm and demonstrate how it scales to large networks. Finally, we compare the accuracy and speed of our algorithm to others.

1. Introduction

A swarm of robots has many benefits over a single complicated robot. These include flexibility, scalability, robustness, adaptability and cost. However, these benefits come with the challenge of employing distributed control, and the confinement to simple sensors and systems. In our case, we utilize a single sensor capable of communicating with neighbours, as well as estimating the distance to neighbours. There are many sensors that satisfy these properties, such as magnetic [3], or infrared [1, 2], however, we do not restrict ourselves to a single implementation, rather, we aim to develop a general algorithm that can be used on a variety of platforms.

The problem of localization is canonical to swarm robotics. To be able to execute a large range of complicated tasks, a swarm must have a consistent global coordinate system, shared between all of its members. However, many modern implementations rely on external information, often in the form of ‘beacon’ or ‘anchor’ nodes which already know their global position through some means, often GPS or external markers. The problem then simplifies to completing a partially completed coordinate system. In contrast, our algorithm is focused on systems that are incredibly small and simple, such as the I-SWARM [4], which is approximately 8 mm^3 in size. At these scales, sensors such as GPS or cameras become intractable, and new approaches must be used. Our vision is to create a series of algorithms that are capable of accurately controlling and manipulating a swarm of potentially tens of thousands of individual members, each on the millimetre, or sub-millimeter scale. This paper represents our first algorithm, and will serve as the basis for our following work. We focus on two critical properties which our algorithm must have: robustness, and scalability. To be robust, the system must have no single point of failure. That is to say, at no point in time should the swarm be dependent on any individual robot functioning properly. For scalability, the error rate should grow no more than linearly with the size of the swarm, while the time

complexity should be constant per individual robot.

In summary, our algorithm has the following characteristics:

1. It is robust, with no single point of failure anywhere during the algorithm. It is also robust to noise in both measurements and movements
2. It is fully distributed, and requires no complicated or external sensors such as GPS or cameras, nor beacon or anchor nodes
3. It supports dynamic node insertion and mobility
4. It converges to a global coordinate system correctly identifying each agent with a high probability.

2. Related Work

Swarm localization has been a heavily researched topic in the past decade, although many of the proposed solutions make use of external information. For example, Goel et al. [6], proposed a system to localize a group of UAV's, however the accuracy of the system heavily relies on the use of GNSS (Global Navigation Satellite System). Similarly, Raghavendra et al. in [7, 8] localized a network of wireless sensors, but the algorithms make use of 'beacon nodes' which already know their position in a global reference frame, in this case with GPS. Similarly, Wu et al. [11] make use of external information in the form of externally placed markers, and internal memory of those markers in global coordinates. Some potential problems with these approaches are that they require specific environments to function, usually by requiring accurate GPS signals which would exclude applications in cities, caves, indoors etc. or by requiring markers to be previously placed in the environment, which again excludes a large range of environments. Solutions requiring beacon nodes also suffer from lack of robustness, as failures in the relatively small percentage of specialized beacon nodes could cause the entire swarm to fail. Finally, many of these techniques are not usable for swarms of robots on the millimetre or sub-millimetre scale as often the error from external means of localization (such as GPS) are orders of magnitude larger than the distances between individual agents. There have also been algorithms developed that make no use of external information such as Lee et al. [9], or Spletzer et al. [10] where they are able to localize a group of robots using visual odometry. However, these algorithms often have large, power-consuming, computationally expensive sensors, in this case cameras, that again are not tractable for large swarms of small robots.

Finally, [12-14] have proposed algorithms that meet our stringent requirements, of using simple sensors, and being entirely local. In [12], Moore et al. gave an excellent localization method, which builds on the ideas of robust quads. Although their method is very accurate for the nodes it is able to find, it requires high node density and is less adaptable to large-scale asynchronous systems. In [13] Li et al. propose a method to localize a group of quadcopters using only IMU and intra-range measurements. Although it is able to converge well for a small group of robots, due to its use of an EKF and IMU measurements, it is unlikely to scale well for

much larger swarms. Finally, [14] introduces a method somewhat similar to ours. It begins by building small local coordinate systems, before combining them. However, it requires a predefined local coordinate system to converge to (in this way it is closer to our baseline algorithm, described below), it also does not converge well for larger swarms.

Our algorithm differs from these approaches in that it:

- Scales well, to swarms in the hundreds
- Can compute the location of all nodes with acceptable error.
- Is completely distributed, and more robust than the above examples

3. Approach

For simplicity, we describe two-dimensional localization. However, our algorithm extends straightforwardly to three dimensions. In the following sections, we use the term's node, agent and robot interchangeably.

We begin with a swarm of agents, each with one sensor with a range of r. In addition, each node has a unique ID. We represent the ith node as n_i and its neighbourhood as D_i which contains all nodes within distance r of n_i.

Our algorithm consists of the following four steps: first, certain nodes will probabilistically become seed nodes, creating a new coordinate system. Next, nodes connected to seed nodes will use trilateration to determine their position within the seed node coordinate system. Afterwards, all nodes will agree on and converge to a single coordinate system. Finally, the position of all nodes is updated using trilateration and gradient descent. We demonstrate two implementations following the above steps. The first is a trivial baseline, which we will later use to compare to the second, more optimized algorithm.

3.1 Baseline

Step I: Initially, all nodes have no coordinate system or position. Each node has a probability p, of becoming a seed node. Before becoming a seed node, the node communicates with two of its neighbours. If they are not seed nodes, then all three nodes become seed nodes, and are assigned positions according to Algorithm 1. The new coordinate system shared by all seed nodes will have a Coordinate system ID, equal to the initial seed nodes ID.

Step II: On each time step, if a robot has no coordinate system, it searches for a valid neighbourhood of agents, described as at least 3 agents in its communication range with the same coordinate system. The agent then uses these neighbours to trilaterate its position in the new coordinate system. If multiple such valid neighbourhoods exist, the agent chooses the one with the smallest coordinate system ID. An example of trilateration is provided in *Fig 1*.

Step III: Nodes continue to search their neighbours for their coordinate system ID's, and if possible, they recalculate their positions in the smallest coordinate system.

Step IV: Nodes then iteratively update their positions based on a gradient descent algorithm minimizing the RMS error of the actual distances between a node and its neighbours, and the estimated distance, based

on their positions. The implementation is given in Algorithm 3.

Algorithm 1:

1. Let D_i be the neighbourhood of n_i
2. Choose three elements of D_i such that they have no coordinate system. If less than three such neighbours exists then terminate.
3. Let $d_{0,1}$ represent the measured distance between D_i^0 and D_i^1. Assign coordinates to each chosen element of D_i as follows:

$D_i^0 = (0,0)$
$D_i^1 = (d_i, 0)$
D_i^2 = *any point of intersection of the two circles centred at* D_i^0 *and* D_i^1 *with radius r*

Note: a consequence of the arbitrary choice for D_i^2 is that the coordinate system may be left or right-handed.

Algorithm 2:

1. Let D_i represent the neighbourhood of n_i
2. Sort each element of D_i by its coordinate system ID, let this sorted set be represented by C with C_{id} representing a set of nodes with the corresponding coordinate system ID.
3. Select the element in C with the smallest ID, that has at least 3 neighbours, if no sets satisfy this condition then terminate.
4. Let the selected element be C_{min}. Now use any three nodes within C_{min} to trilaterate the position of n_i in the new coordinate system.
5. If the trilateration is unsuccessful, terminate.

Algorithm 3:

1. Let D_i represent the neighbourhood of n_i
2. For each element of D_i calculate the total error, given as the RMS of the estimated distance between n_i and its neighbour using the sensor, and the estimated distance using their positions.
3. Then update n_i's position according to the standard gradient descent model, with some learning rate alpha.

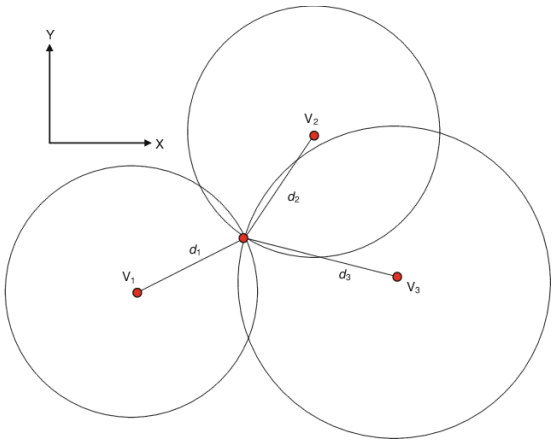

Fig 1. The above image represents trilaterating a new position from three given positions and their distances. In our case all three distances d_1,d_2, and d_3 are equal to r. It is important to note that trilateration has the following shortcomings: first, it can only be done when all points are in the same coordinate system. Second, it is an approximation method, whose error compounds with the amount of trilaterations preformed. Third, there exist several cases where a solution cannot be obtained.

Our baseline algorithm provides a simple and robust approach to the localization problem, and it can be shown to always converge to a single global coordinate system. However, one draw back to this system, is that it is usually sub-optimal. Consider an example where the majority of the swarm has converged to a single coordinate system, but a new seed is

created with a smaller coordinate system ID. This has the effect of forcing all agents to recalculate their positions, potentially multiple times. This both compounds the cumulative error, and the time to convergence of the algorithm.

3.2 Divide and Conquer

For optimization, we introduce a second algorithm which differs only in *Step III.* We notice that after *Step II.* Each node belongs to a coordinate system, each of which is approximately a voroni region, as depicted in *Fig 2*. The swarm must now achieve consensus on which coordinate system to converge to. A more efficient implementation would be to have the swarm converge to the largest area after *Step II.* This would result in the least number of coordinate transformations, and thus be the most computationally efficient. This brings us to our algorithm, which we dubbed "divide-and-conquer" or D&C for short. In this algorithm, larger areas have a high probability of 'invading' smaller areas, which is proportional to the difference in areas.

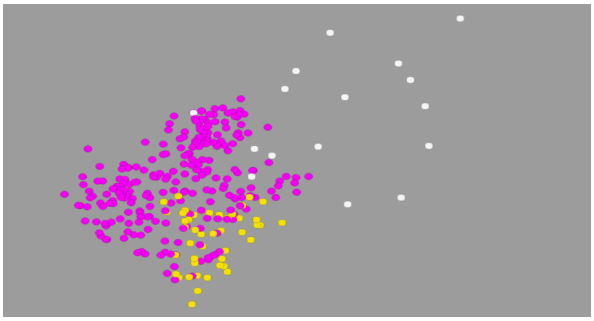

Fig 2. This image represents the state of a swarm after *Step II.* With each separate coordinate system a different colour

A few characteristics of D&C:

1. It is robust to introduction and removal of individual nodes
2. It is robust to the division and conjugation of large groups of nodes, which individually have converged to different coordinate systems
3. It has lower error than our baseline due to fewer total trilaterations necessary until convergence

Below we introduce a revised *Step III,* with two subsections:

Section I: Once an agent belongs to a coordinate system, it begins looking for other regions to invade. If one of its neighbours belongs to a different coordinate system, it has a probability defined by *Algorithm 4* of invading it.

Section II: If an agent decides to invade an area, it propagates an *invading* signal to the chosen area, initiating the invasion. This is show in detail in *Algorithm 5.*

Algorithm 4:

1. If n_i is currently invading another area, or being invaded, terminate. Otherwise, let D_i represent the neighbourhood of n_i
2. Sort each element of D_i by its coordinate system ID, let this sorted set be represented by C with C_{id} representing a set of nodes with the corresponding coordinate system ID.
3. Sort C with respect to the number of nodes in each element of C. Let the smallest such neighbourhood be C_{min}
4. Sort all elements in C_{min} by the magnitude of their distance (equivalent to their distance from the origin)
5. Use rank selection to choose an element from C_{min} giving a higher probability to the elements who are farther from their origin.

6. Let the selected element be represented by e, if e is already being invaded, or if it is invading n_i's coordinate system, terminate.
7. Otherwise, the probability of invasion is given by the threshold function: where t denotes the ratio between the magnitude of n_i and e's positions, minus 1.
8. If true, send and invade signal to e

Algorithm 5:

1. Let C_i represent all nodes that are in the neighbourhood of n_i which have the same coordinate system ID.
2. If n_i is being invaded, then for each element in C_i if they are not being invaded, there is a probability q of n_i propagating the invasion signal to its neighbours.
3. Otherwise, if n_i is not being invaded, and receives an invasion signal, it will calculate the probability of being invaded according to *Algorithm 4*.
4. If an agent has an invasion signal, it ignores all other signals. After t time steps of adopting a signal, the agent will abandon it and reset itself to be neutral.
5. If an agent currently has an invasion signal, it will attempt to search for a valid neighbourhood of 3 or more agents which belong to the invading region. If these agents are found, the agent will use trilateration to determine its new position

4. Analysis

We empirically analyze the properties of our algorithm. Our error metric is the average absolute difference between estimated and actual distance, for each pair of nodes.

$$\sigma = \frac{1}{N} \sum_{i}^{all\ nodes} \sum_{j,\, j \neq i}^{all\ nodes} \left| e_{ij} - a_{ij} \right|$$

Where N represents the total number of pairs of nodes. e_{ij} represents the estimated distance between n_i and n_j and a_{ij} represents the actual distance between a_i and a_j.

4.1 Simulation Results

In our simulations, nodes were distributed uniform randomly in a 300 cm x 300 cm test area. The range of each node was 50 cm. The hyperparameters choose where: q, the probability of propagating and invasion signal was 0.5. t, the number of time steps until forgetting a signal was 10. And p the probability of becoming a seed node was 0.01. For each algorithm, the maximum number of iterations was 50. For both D&C methods, *Step I* and *II* were run for the first 10 iterations, after which only *Step III* was used. Note, that *Step IV* was not used for any algorithm, as it is an optional method for improving upon an already established estimate. Each simulation was averaged over 10 runs.

We evaluated three different functions. The first is the optimized Divide and Conquer algorithm from 3.2. The second is the baseline from 3.1. And the third, Divide and Conquer with minimum area, includes the following

change to the algorithm from 3.2: in *Step II*, instead of having and uninitialized node choose the coordinate system with the smallest ID, it instead chooses the coordinate where it is closest to the origin. We hypothesized that this would lead to more regularly shaped voroni regions after *Step II*, without the overlaps as seen in *Fig 2*, which would help convergence.

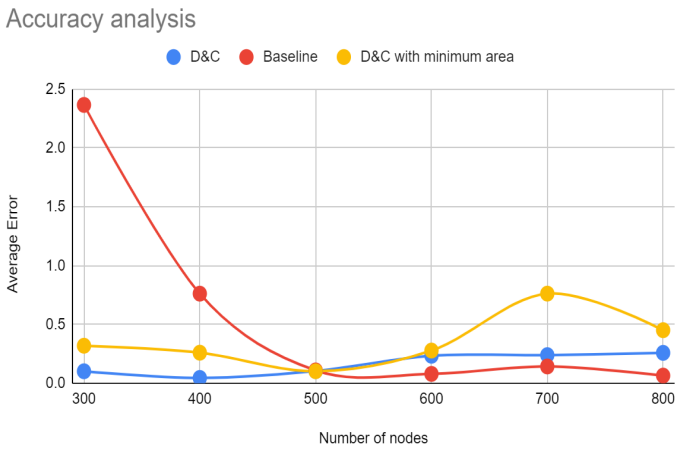

Fig 4. Analysis of three algorithms using the average error per node with variable number of nodes.

Our results are the following: first, nodes were localized with a high degree of accuracy, representing at most a 5% error from their measurement range. Note however, that these results were obtained with the assumption of perfect communication, and no movement. The true accuracy will most likely be worse, and implementation and testing on physical systems remains the goal of future research. Interestingly, the Baseline algorithm improves significantly with the number of nodes. This is due to the increased node-connectivity, as the number of nodes increases. We hypothesize that the performance would remain suboptimal for less dense networks, and the original arguments against the Baseline algorithm remain.

Next, we analyze the total convergence rate of each algorithm. Specifically, we find the number of nodes who failed to converge to the global coordinate system. We present the results in a log-scale format.

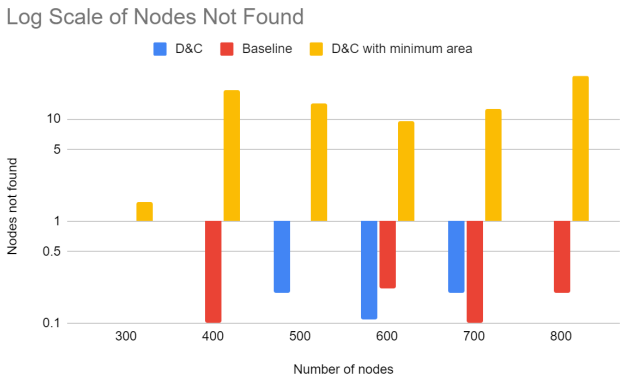

Fig 5. Analysis of the average number of nodes not found for each algorithm.

The worst performance by far was achieved by our modified D&C function. We hypothesize that this is due to modification in *Step II*. This creates many more local coordinate systems than the initial configuration, which by definition attempts to only create one. The network after *Step II* consists of many small local coordinate systems. For our time limit of 50 total iterations, there is not enough time for each coordinate system to converge, and thus some nodes are not localized. This can also explain the decrease in accuracy seen in *Fig 4*. With many small local coordinate systems, there will likely be many cycles of invasions before a single dominant coordinate system is chosen, these cycles compound the errors introduced by each trilateration.

Finally, comparing our algorithm to that of [12], we obtain similar results in terms of precision and convergence.

5. Conclusion

We have demonstrated an algorithm that is able to robustly localize robots in a swarm using only distance measurements, and no beacons or anchors. We compared our proposed algorithm, D&C, to a baseline and another similar algorithm, and we've demonstrated that D&C has shown faster overall convergence, with fewer errors, needing fewer connections per node. However, our algorithm is suspect to ambiguities and incorrect configurations, as opposed to [12], although these occur rarely. For future work, we hope to implement our algorithm on hardware, and test it in real-world situations. Furthermore, we aim to improve the convergence speed, and scalability of our algorithm, to allow for extremely large swarm sizes as outlined in our introduction.

[1] Arvin, Farshad & Samsudin, Khairulmizam & Ramli, Abdul. (2009). A Short-Range Infrared Communication for Swarm Mobile Robots. Signal Processing Systems, International Conference on. 454-458. 10.1109/ICSPS.2009.88.

[2] Rao T, Rama & Balachander, D.. (2013). Short-range near-floor RF propagation experiments in indoor corridors for Wireless Sensor Communications. 3549-3553.

[3] Won, Yun-Jae & Kang, Shin-Jae & Kim, Sunhee & Choi, David & Lim, Seung-Ok. (2009). A communication system using magnetic fields. 265 - 269. 10.1109/WIRELESSVITAE.2009.5172460.

[4] Seyfried, Jörg & Szymanski, Marc & Bender, Natalie & Estana, Ramon & Thiel, Michael & Wörn, Heinz. (2004). The I-SWARM Project: Intelligent Small World Autonomous Robots for Micro-manipulation. Lecture Notes in Computer Science. 3342. 70-83. 10.1007/978-3-540-30552-1_7.

[5] Lee, Young-Hee & Zhu, Chen & Giorgi, Gabriele & Gunther, Christoph. (2020). Cooperative swarm localization and mapping with inter-agent ranging. 353-359. 10.1109/PLANS46316.2020.9110227. (camera)

[6] Goel, Salil. (2017). A Distributed Cooperative UAV Swarm Localization System: Development and Analysis. 10.33012/2017.15217.

[7] Vinodini Ramesh, Maneesha & Divya, P. & Kulkarni, Raghavendra & Prabha, Rekha. (2012). A swarm intelligence based distributed localization technique for wireless sensor network. 10.1145/2345396.2345457.

[8] Kulkarni, Raghavendra & Venayagamoorthy, Ganesh & Cheng, Maggie. (2009). Bio-Inspired Node Localization in Wireless Sensor Networks. Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics. 205 - 210. 10.1109/ICSMC.2009.5346107.

[9] Lee, Young-Hee & Zhu, Chen & Giorgi, Gabriele & Gunther, Christoph. (2020). Cooperative swarm localization and mapping with inter-agent ranging. 353-359. 10.1109/PLANS46316.2020.9110227.

[10] Spletzer, John & Das, Aveek & Fierro, R. & Taylor, C.J. & Kumar, V. & Ostrowski, Jim. (2001). Cooperative localization and control for multi-robot manipulation. IEEE International Conference on Intelligent Robots and Systems. 2. 631 - 636 vol.2. 10.1109/IROS.2001.976240.

[11] Han Wu, Shizhen Qu, Dongdong Xu, Chunlin Chen, "Precise Localization and Formation Control of Swarm Robots via Wireless Sensor Networks", *Mathematical Problems in Engineering*, vol. 2014, Article ID 942306, 12 pages, 2014. https://doi.org/10.1155/2014/942306

[12] Moore, David & Leonard, John & Rus, Daniela & Teller, Seth. (2004). Robust Distributed Network Localization with Noisy Range Measurements. SenSys'04 - Proceedings of the Second International Conference on Embedded Networked Sensor Systems. 10.1145/1031495.1031502.

[13] Li, S., Coppola, M., Wagter, C., & Croon, G.D. (2020). An autonomous swarm of micro flying robots with range-based relative localization. *ArXiv, abs/2003.05853*.

[14] Capkun, Srdjan & Hamdi, Maher & Hubaux, Jean-Pierre. (2002). GPS-free Positioning in Mobile Ad-Hoc Networks. Cluster Computing. 5. 10.1023/A:1013933626682.