Self Contained Relative Localization with a Low-Cost Multi-Robot System

I. D. Miller[1] and J. Wallace[2]

***Abstract*— A key limitation of current multi-robot systems is a lack of relative localization, particularly in environments without GPS or motion capture systems. This article presents a centralized method for relatively localizing a 2D swarm using sensors and beacons on the robots themselves. The UKF-based algorithm as well as the requisite novel and cost-effective sensing hardware are discussed. Comparisons with a motion capture system show that the method is capable of localization with errors on the order of the size of the robots.**

***Index Terms*— Localization, Multi-Robot Systems, Range Sensing.**

I. Introduction

SINCE its inception, robotics has been inspired by biology, and recent multi-robot systems can emulate the swarming behavior seen in the animal kingdom. In an early paper, Kube successfully built a simple ground-based swarm system using simple rules inspired by insects [1]. Since then, swarms have taken to the sky [2] and even the water [3]. Robot swarms are not only a fascinating achievement, but also have useful applications ranging from search and rescue [4] to satellite constellations [5]. Swarms have key advantages over traditional single robot systems with respect to redundancy, resiliency, and parallelization [6].

Knowledge of system state is a key requirement for most robot platforms, and in the case of a multi-robot system or swarm, the system state includes the relative locations and orientation of the individual members. Centralized and real-time knowledge of this state is desirable for a human or autonomous controller. A simple and often-used method for state estimation employs an external motion capture system. This approach is often used for swarms such as the Robotarium [7], but such external systems often preclude practical deployment.

A conceptually straightforward approach to multi-robot localization is to individually globally localize each member, allowing any member or controller to easily compute relative locations with minimal communication. Global localization can be performed with GPS or visual odometry as in [8], but GPS is expensive, limited in accuracy, and cannot be used indoors, while odometry methods gradually drift and accrue error over time.

Given the difficulties of global localization, attention has focused on robot systems that estimate relative robot position using onboard sensors and combine this information to obtain the global system state. In [9] and [10] the Kalman filter was employed, but the method was not tested on real hardware and no solution for obtaining the initial system state was given. More recently, in [11] a method using particle filters is described and tested on actual Khepera III robots, but the approach has significant computational demand. Several sensing methods have been proposed to obtain robot bearing and range, including optical tags [12], sonar [13], and infrared (IR) [11], [14]. These strategies rely either on relatively expensive hardware or, in the case of [14], provide only bearing information.

Given this landscape, experimental swarm research is out of the reach of many interested researchers, either due to the high cost or non-deployable nature of the specialized hardware. Therefore, the purpose of this paper is to present a robot platform employing a local sensing method that is deployable in real scenarios, yet has a very low cost. Specifically, we present a system capable of obtaining real-time position information of swarm members with an accuracy better than the size of a robot, without requiring known initial conditions. An external motion capture system is only used to study position estimation error and is not required in actual deployment. Further, low cost was a critical objective, which was kept within $200 per node.

We note similarity to the work in [10], but unlike the distributed sensing algorithm there that requires inter-robot communication, we limit our present scope to centralized computation, where nodes report sensor data directly to a controller. We also introduce a novel low-cost sensor system for gathering the requisite relative pose information. Details of the hardware and software components developed in this project are available at no cost to interested researchers [1].

II. Robotic Platform

The robot design focused on maximizing versatility while minimizing cost. As depicted in Fig. 1, the system consists of an `RFM69` radio, dual stepper motors and drivers, IR beacons and sensors, and a Teensy 3.2 ARM micro-controller. Communication with the controlling computer is achieved with another `RFM69` connected through a micro-controller via USB to a computer. Fig. 2 depicts a single robot, employing two stepper motors driven with interrupt-controlled stepper drivers. Two bearing balls are used as low-friction draggers on the front and back for stability.

An IR LED on the top of the robot pulses at a frequency between 1-50 kHz (unique to each robot) to allow node identification. The LED is covered with a white foam ball to evenly distribute the signal. Two phototransistors are used as detectors, placed on either side of a vertically mounted PCB attached to a servo motor. As the servo rotates back

[1]Ian D. Miller was with the Dept. of Electrical and Computer Engineering, Lafayette College, Easton, PA, USA, and is now with the University of Pennsylvania GRASP Lab, Philadelphia, PA, USA. `iandm@seas.upenn.edu`

[2]Jon Wallace is with the Dept. of Electrical and Computer Engineering, Lafayette College, Easton, PA, USA. `wallacjw@lafayette.edu`

[1]https://github.com/iandouglas96/swarm-thesis/ under the MIT license.

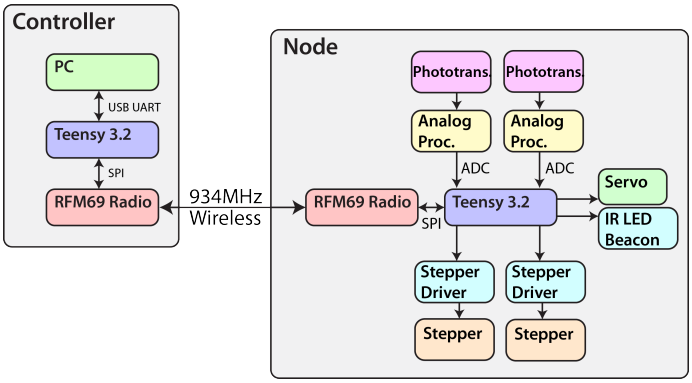

Fig. 1. High-level hardware block diagram of swarm robot and central controller.

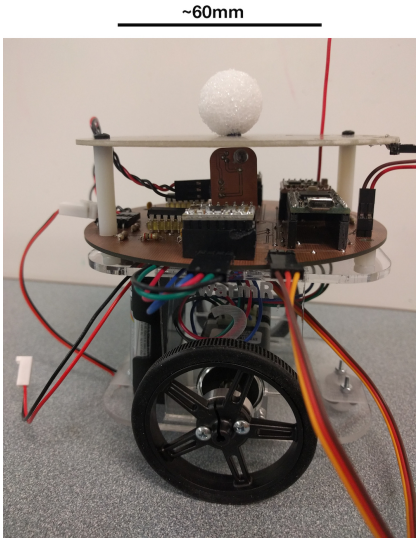

Fig. 2. Photograph of a single robot.

and forth, each phototransistor sweeps out a 180° swath on each side of the robot. A custom analog circuit filters the raw phototransistor output, which is fed directly into one of the micro-controller's onboard ADCs.

A. Sensor Signal Processing

An FFT is performed on the incoming data at each sweep angle,[2] and because the IR LEDs are pulsed, the raw data consists of superimposed square waves of varying amplitude as depicted in Fig. 3(a). Specific frequency bins correspond to each robot, and the values of only these bins are stored at each angle.

Once an entire 360° sweep is obtained (from simultaneous 180° sweeps), the magnitude of each bin at each angle can be plotted, as shown in Fig. 3(c), where neighboring robots appear as signal peaks. The frequency bin, angular location, and height of the peaks determine the nearby robot's identity, bearing, and range, respectively.

[2]The Teensy audio library is used to perform the FFT, which uses 1024 samples at about 30kHz (standard audio sample rate). This library has the benefit of using the DSP instructions available in the ARM core, so it is quite fast.

The relationship between peak magnitude and distance can be modeled with a power law, given by

$$D = \frac{\text{constant} * \text{luminosity}}{\text{surface area}} = AM^{-2} \tag{1}$$

where M is the peak magnitude and A is a constant. In practice, there are significant variations between individual sensors and components used in the analog electronics, requiring each robot to be calibrated as shown in Fig. 4. A power-law fit is used to allow the exponent to vary in the event that anisotropies of the transmitter cause deviation from the ideal inverse square law. Note that the low-range readings were ignored for the purposes of fitting, since they deviate strongly from the power-law curve due to saturation and occlusion by the transmitter mount.

B. Determination of Noise Matrices

As described in Sec. III-B, the Kalman filter requires the error from the motion model and sensors to be characterized. Generally, we have chosen error standard deviations that are slightly higher than measured or estimated values, which may be pessimistic, but they have yielded good experimental results. To quantify error in the sensor model (σ_{dist}), two robots were placed 61 cm apart, and the estimated distance was recorded while one robot rotated in place. Fig. 5 shows a representative histogram from this procedure.

Although standard deviation of range sensing in this example is 18.6 cm, most of the readings are quite accurate, as indicated by a standard deviation of 7.2 cm when outliers (distances $>$ 90 cm) are removed. The outliers are caused primarily by blind spots from the supporting standoffs that hold up the IR LED board. Choosing to ignore outliers in computation of σ_{dist} results in a filter that generally tracks well but diverges when a highly erroneous value is received. An alternate approach uses the actual skewed value of σ_{dist}, giving a slower converging (but more stable) filter. The latter approach was adopted herein with $\sigma_{\text{dist}} = 15$ cm. A possible more sophisticated approach would be to pre-filter sensor data and throw out readings which are vastly different from expected.

The variance in sensor angle is ideally only limited by the resolution of the angular scan, or 2°, but is practically higher due to robot movement and delay in the sensing method. The update rate is approximately 1 Hz, giving a typical delay of 0.5 s. Considering robot rotation speed to be $20^\circ/\text{s}$ on average, the average error is 10°. For the following experiments, a standard deviation of $\sigma_{\text{angle}} = 0.15$ rad or 8° was used.

The system propagation matrices required for the Kalman filter are more challenging to determine. Typically, the model is very accurate because we are using steppers, but occasionally the robots slip on the floor for about 1 s. The average error after a single predict step (run at a 30 Hz update rate) is found assuming a typical forward speed of 3 cm/s, giving an error of 0.1 cm as the wheels slip and suggesting the choice of $\sigma_x = \sigma_y = 0.1$ cm. Taking the typical rotation value of $20^\circ/\text{s}$, the maximum change in angle after 1/30 s is 0.02 rad, and $\sigma_\theta = 0.03$ rad was used.

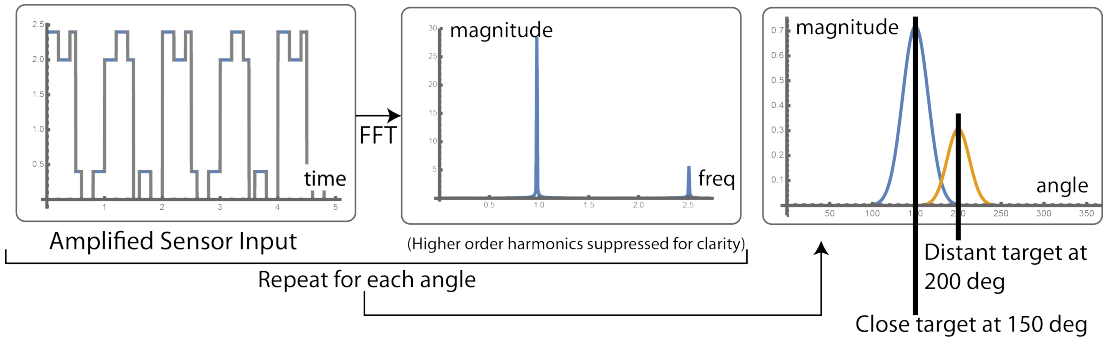

Fig. 3. Data flow for converting raw sensor data to target range and bearing.

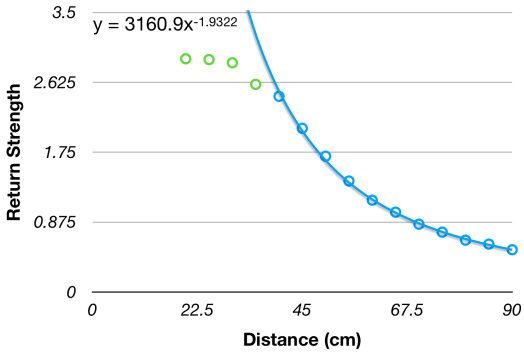

Fig. 4. Plot showing the sensor signal magnitude for a target at varying distances, as well as the power-law best fit.

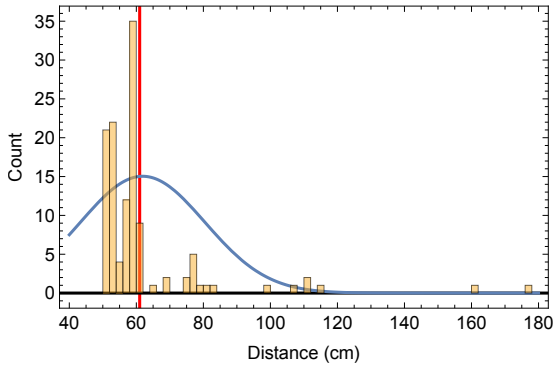

Fig. 5. Histogram of distance measured between two robots, with actual distance shown in red. A Gaussian distribution with the same mean and variance is shown in blue.

III. Localization Algorithms

The sensing method and hardware described in Sec. II yield range and bearing estimates of robots that are 1 m away or closer. Individual robots report this information to a central controller, who must combine the estimates to obtain a global picture of the swarm. This problem will first be solved using a direct estimation procedure that does not require the initial state of the swarm to be known. Due to the computational complexity and limitations of the direct method, we later develop a more efficient tracking method based on Kalman filtering that more optimally deals with sensor error.

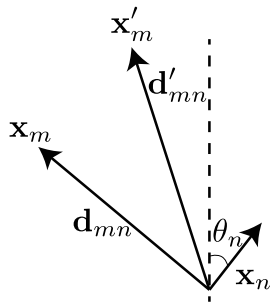

Fig. 6. Geometry of Detected and Actual Location of Adjacent Robots

A. Direct Estimation Method

We follow an approach similar to [15], seeking to minimize the least-square error between the global robot map and reported local positions. However, here we do not assume stationary landmarks nor do we exploit knowledge of robot velocities. Consider the geometry shown in Figure 6, where $\mathbf{x}_p$ and θ_p give the global estimated position and rotation of robot p, respectively, where $p \in \{m, n\}$ for the mth and nth robots. The displacement vector $\mathbf{d}_{mn}$ can be computed between the robots in the global model. Likewise, robot n obtains a local estimate of the displacement vector to robot m as $\mathbf{d}'_{mn}$ using local range and bearing (θ'_{mn}) estimates, resulting in a perceived location of $\mathbf{x}'_m$.

Given N total robots and that robot n sees the set of robots denoted $\mathcal{M}_n$, the goal of the estimation procedure is to find the $\mathbf{d}_{mn}$ that minimize

$$\phi = \sum_{n=1}^{N} \sum_{m \in \mathcal{M}_n} |\mathbf{d}_{mn} - \mathbf{d}'_{mn}(d_{mn}, \theta'_{mn})|^2, \qquad (2)$$

which is non-trivial to solve for more than two robots. In this work, minimization of ϕ was accomplished using the sequential least-squares programming minimization algorithm (SLSQP) from the Python `SciPy` library. Special care was taken to program the objective function (2) as a matrix operation using `NumPy` to allow fast computation in Python.

Note that the minimization algorithm only uses sensor information on the *relative* positions of the robots, as robots know nothing about the *global* environment. Therefore, the global rotation and translation of the complete swarm are free

parameters. For evaluation purposes, the semi-random values of these parameters as returned by the minimization algorithm are modified to optimally fit the estimated positions to the known positions. This is accomplished by aligning centroids and applying the Kabsch algorithm [16] to find the appropriate rotation.

Fig. 7(a) shows the solution given by this minimization method for 20 robots, assuming that each robot can see all others and that there is no local estimation error. Error in the global map is effectively zero, and this computation required roughly 0.5 s. Fig. 7(b) shows the effect of including moderate sensor limitations and estimation error. Here, we introduced random Gaussian noise in distance measurements with a standard deviation given by the radius of the blue circle, Gaussian angular noise with a standard deviation of 6°, and a sensing range of half of the field size, given by the green circle. Results deviate slightly from the exact solution, but average error is on the order of the robot size. Computational time did not increase appreciably compared to the ideal case.

Fig. 7(c) considers a case when the algorithm fails, where the maximum sensing range is decreased further, resulting in an incorrect solution. This occurs due to insufficient data about the robot adjacencies, since there are other incorrect solutions that may minimize error. There are a total of $3N-3$ degrees of freedom (DOF), since each robot has three DOF and the arbitrary rotation and position of the swarm also has three DOF. Each sensor reading provides two pieces of information (bearing and range). We clearly require

$$2\sum_n |\mathcal{M}_n| \geq 3N-3, \tag{3}$$

where $|\cdot|$ denotes set cardinality. This requirement can be better understood by assuming each node sees M neighbors, and in this case we require $2NM \geq 3N-3$, or $M \geq 3(N-1)/(2N) \approx 3/2$ for large N.

B. Kalman Filtering

The minimization strategy of the previous section works well for initialization, but is highly sensitive to sensor noise and can only update at the sensor refresh rate. To solve these problems, we turn to Kalman Filters, using a method similar to the landmark localization problem described in [17] and [18] where the landmarks are in fact mobile robots themselves, as described in [10].

A distributed algorithm is desirable, since it would theoretically scale better for large swarms, but this would additionally require inter-robot communication. Distributing the algorithm is partially supported by using separate Kalman filters to track the state of each robot. However, the algorithm still uses the states of other robots in the update step, which is required to track the complete swarm. For our predict step, we implement a differential drive model as described in [19]. We assume independent variables, so the system noise covariance is simply a diagonal matrix of parameter variances.

We first develop the measurement function $\mathbf{h}$, which transforms from state space to measurement space. Each other robot has its own filter generating its own state estimate, so we can treat these states as known landmarks, and then apply the same landmark measurement model used in [20].

If we have a robot n that sees robots $m_1, m_2, ... \in \mathcal{M}_n$, then

$$\mathbf{h}(n) = \begin{bmatrix} |\mathbf{d}'_{m_1 n}|^2 \\ \tan^{-1}(x(\mathbf{d}'_{m_1 n})/y(\mathbf{d}'_{m_1 n}) - \theta_n \\ |\mathbf{d}'_{m_2 n}|^2 \\ \tan^{-1}(x(\mathbf{d}'_{m_2 n})/y(\mathbf{d}'_{m_2 n}) - \theta_n \\ \vdots \end{bmatrix}. \tag{4}$$

where $x(\mathbf{d})$ and $y(\mathbf{d})$ simply extract the x and y components of the vector $\mathbf{d}$. In practice we use the two-argument inverse tangent function (`atan2`) to guarantee that the resulting sign is correct. As with the motion model, we also model the noise, in this case from the sensor. Assuming independent variables, we have $\mathbf{R} = \mathrm{diag}([\sigma^2_{\text{dist}}, \sigma^2_{\text{angle}}, \sigma^2_{\text{dist}}, \sigma^2_{\text{angle}}, ...])$, where $\mathrm{diag}(\cdot)$ forms a diagonal matrix from its vector argument.

In [10], an extended Kalman filter (EKF) is used to handle the system nonlinearities. We found, however, that an EKF tended to diverge when faced with highly nonlinear robot trajectories. To handle these cases, we instead implement an unscented Kalman filter (UKF). Where the EKF simply linearizes the model around an operating point, the UKF takes a set of "sigma points" which it passes through the function. These sigma points are chosen using the method in [21], with the choice of number of sigma points $n = 3$, since we have 3 state variables, $\kappa = 3 - n = 0$, $\beta = 2$, and $\alpha = .00001$. These are the choices made by Labbe in his similar landmark localization problem in [17]. We then run these sigma points through the unscented transform to effectively fit a Gaussian distribution to the result.

To implement this filter, Labbe's `FilterPy` library was used, only requiring small adjustments for this application. For the residual calculation in the update step, a special function must be provided to properly calculate the difference of angles due to modulo angular arithmetic (e.g. $359^\circ - 1^\circ = -2^\circ$, not 358°). Furthermore, the number and identity of robots that any given robot can see at any time may, and most likely will, vary from sensor scan to scan. When calculating the measurement residual, we therefore automatically set appropriate matrix entries in the residual to 0 for robots for which we have no sensor data. To handle the varying size of $\mathbf{h}$ (the measurement matrix), modifications to the `FilterPy` library were made to automatically copy $\mathbf{R}$ to scale it to the appropriate size on each update step.

The overall localization process is diagrammed in Fig. 8. Note that the direct estimation method Sec. III-A is used to determine the initial conditions, and control is then passed to the UKF.

IV. Testing

For comparison and testing purposes, each robot was equipped with ArUco tags [22], which were then processed through OpenCV to provide ground-truth position information. The relative locations determined by the UKF were then translated and rotated to best fit to the known locations

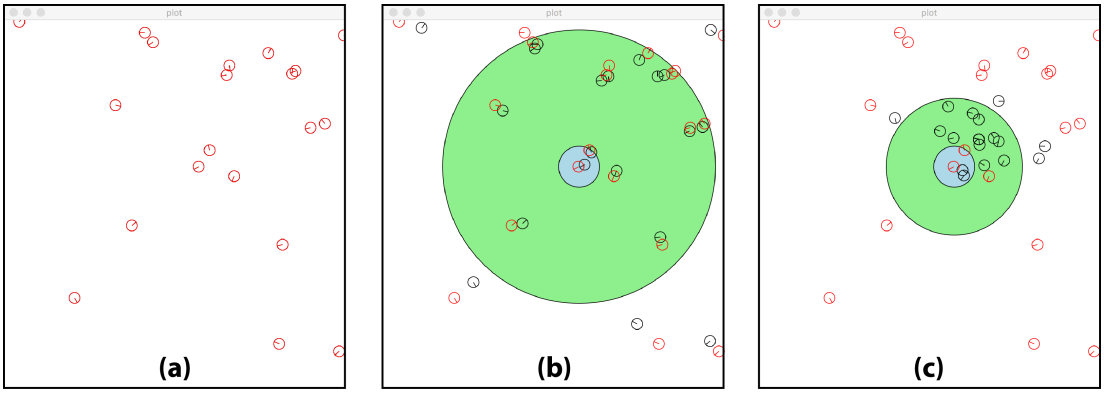

Fig. 7. Direct estimation algorithm simulation with (a) no error, (b) simulated noise, and (c) insufficient data. The red circles are the actual positions, the black the calculated positions.

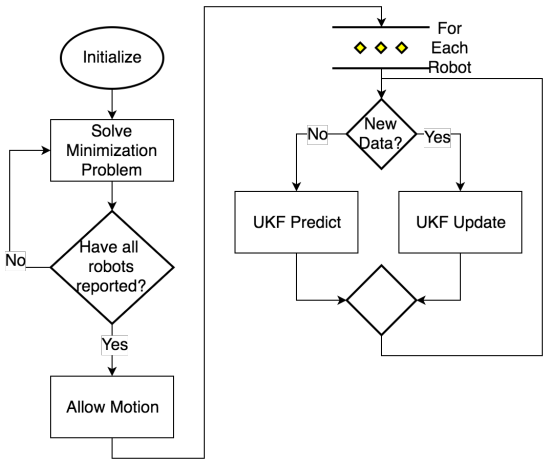

Fig. 8. Flowchart showing the startup process for the UKF. The direct estimation algorithm is used to determine the initial state before control is transferred to the UKF.

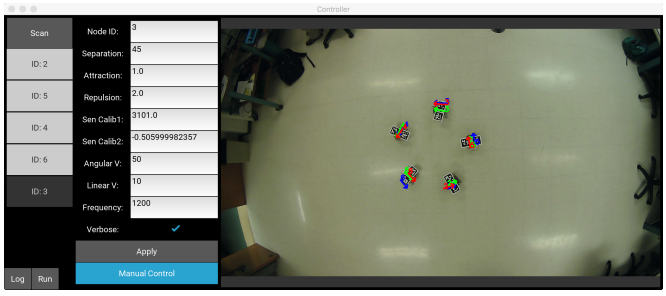

Fig. 9. Image of the motion capture system running on 5 robots. The different arrows correspond to the different localization systems: green for motion capture, red for minimization, blue for UKF.

using the Kabsch algorithm [16]. Simultaneously, the direct-estimation algorithm was run for comparison purposes. The motion-capture system running is shown in Fig. 9.

Plots of the average error versus time for various maneuvers are shown in Fig. 10, where the time scales are on the order of tens of seconds. The primary source of error is periodic points of large range-error. This can arise from a number of different sources, including blind spots created by support posts on the robots as well as cables becoming faulty on the sensor boards due to constant rotation. However, these effects do not cause the filter to diverge, and the overall average error remains about 10 cm, or on the order of the size of a robot.

The large amount of systematic error seen for the line formation is caused by accumulated systematic calibration error. If each robot perceives its neighbors to be slightly further away than they actually are, then this error quickly accumulates in a line-like configuration. This case highlights that relatively small systematic errors can quickly multiply into large problems in larger swarms. A more sophisticated calibration process or general sensing system would be needed to mitigate this issue.

V. Conclusion

This paper has presented a low-cost multi-robot platform and accompanying algorithm based on the UKF that can estimate relative positions of robots in a small swarm. The method was demonstrated for 5 robots, where the main limiting factors were sensor accuracy and robustness. Accuracy of the position information was shown to be similar to that obtained with external capture systems, indicating that low-cost and deployable robot swarms are feasible.

In order for the system to be practical, it is important for the operator to view not only the current relative configuration of the swarm, but also the configuration of the swarm in its current space. This would make the system truly capable of mapping areas, which is a very promising application of swarms. However, our robot nodes currently have no sensors capable of gathering information about their environment. Care would be required to achieve this at low cost and without interfering with the existing sensing method.

Further work is also needed to develop a truly distributed version of our proposed UKF algorithm. The current method relies on a centralized controller, but since each robot has a separate UKF, it should be feasible to distribute computation onto the robots themselves. Another direction of future work could involve the design of a system with multiple robots employing reuse of the IR modulation frequencies, since this feature will be necessary to support of swarms of arbitrary size.

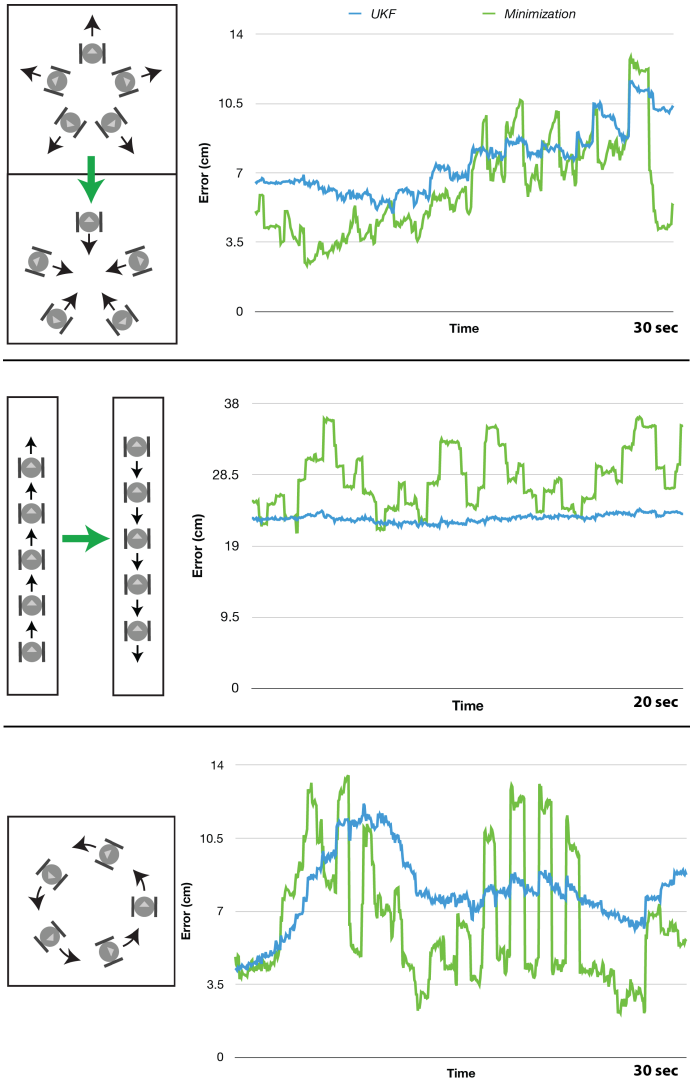

Fig. 10. Plot of average error between calculated and actual relative positions of robots over time for various 5-robot formations.

REFERENCES

[1] C. R. Kube and H. Zhang, "Collective robotics: From social insects to robots," *Adaptive Behavior*, vol. 2, no. 2, pp. 189–218, 1993.

[2] Y. Mulgaonkar, A. Makineni, L. Guerrero-Bonilla, and V. Kumar, "Robust aerial robot swarms without collision avoidance," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 596–603, Jan 2018.

[3] T. J. G. Waduge and M. Joordens, "Fish robotic research platform for swarms," in *2017 25th International Conference on Systems Engineering (ICSEng)*, Aug 2017, pp. 212–217.

[4] A. S. Kumar, G. Manikutty, R. R. Bhavani, and M. S. Couceiro, "Search and rescue operations using robotic darwinian particle swarm optimization," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sept 2017, pp. 1839–1843.

[5] E. Gill, "Editorial," *Acta Astronautica*, vol. 123, p. 363, 2016, special Section: Selected Papers from the International Workshop on Satellite Constellations and Formation Flying 2015.

[6] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[7] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1699–1706.

[8] A. Weinstein, A. Cho, G. Loianno, and V. Kumar, "Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1801–1807, July 2018.

[9] S. I. Roumeliotis and G. A. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, Oct 2002.

[10] A. Martinelli, F. Pont, and R. Siegwart, "Multi-robot localization using relative observations," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 2797–2802.

[11] A. Prorok, A. Bahr, and A. Martinoli, "Low-cost collaborative localization for large-scale multi-robot systems," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 4236–4241.

[12] M. Saska, "Mav-swarms: Unmanned aerial vehicles stabilized along a given path using onboard relative localization," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 894–903.

[13] W. Boussetta, J. Knani, and H. Tlijani, "Multi-robot localization in a landmark-free environment using binaural sonar," in *2017 14th International Multi-Conference on Systems, Signals Devices (SSD)*, March 2017, pp. 169–174.

[14] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 3293–3298.

[15] A. Ahmad, G. D. Tipaldi, P. Lima, and W. Burgard, "Cooperative robot localization and target tracking based on least squares minimization," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 5696–5701.

[16] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, no. 5, pp. 922–923, Sep 1976.

[17] R. Labbe, "Kalman and bayesian filters in python," https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python, 2014.

[18] G. Péter, B. Kiss, and G. Kovács, "Kalman filter based cooperative landmark localization in indoor environment for mobile robots," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 1888–1893.

[19] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*, 2nd ed. Cambridge University Press, 2010.

[20] R. Negenborn, "Robot localization and Kalman filters," Master's thesis, Utrecht University, 2003.

[21] R. Van Der Merwe, "Sigma-point Kalman filters for probabilistic inference in dynamic state-space models," Ph.D. dissertation, 2004.

[22] S. Garrido-Jurado, R. Munoz-Salinas, F. Madrid-Cuevas, and M. Marin-Jimenez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014.