Proceedings of the National Conference
On Undergraduate Research (NCUR) 2005
Virginia Military Institute
Washington and Lee University
Lexington, Virginia
April 21 – 23, 2005

Relative Localization in Colony Robots

Christopher Atwood[*], Felix Duvallet[*], Aaron Johnson[*], Richard Juchniewicz[*], Ryan Kellogg[*], Katherine Killfoile[*], Alison Naaktgeboren[†], Suresh Nidhiry[‡], Iain Proctor[§], Justine Rembisz[*], Steven Shamlian[*], Prasanna Velagapudi[*‡]

[*]Carnegie Institute of Technology [†]Mellon College of Science
[‡]School of Computer Science [§]College of Humanities and Social Sciences
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213. USA

Faculty Advisor: Dr. Brett Browning

Abstract

This project demonstrates the coordinated discovery and capture of a mobile target using a colony of low-cost robots. Many existing robot colonies use high-end components to guarantee results. We show that a colony of inexpensive robots, which alone are insufficient to the task, can together overcome their shortcomings. Information about the environment is gathered via simple one-dimensional sensors, such as IR beacons, and shared wirelessly. The robots localize relative to one another by sharing their 1D range data. All sensors are noisy and inaccurate, but when filtered, shared interpretation allows useful state information to emerge. No one robot is in charge or knows the entire state of the world, but as a communal intelligence the colony does. To search the world space for the target, the robots create regular formations and move as a coordinated mass. Once the heated target is located via pyroelectric sensors, the robots communicate the target's position and form a strategy to capture it using behavioral paradigms. We will demonstrate robots moving in a coordinated formation to surround and capture a target.

1. Introduction

Our goal is to create a colony of low cost, mobile, autonomous robots that work together to complete a variety of tasks. There have been many colonies of robots in the past, but none have low cost solutions to complex problems such as relative localization. One of the most successful colonies is the MIT Swarm. This colony can relatively localize and create intelligent arrangements, but at a very high per-robot cost [1]. A lower cost colony robot will allow for larger colonies at lower costs. This allows more people to have access to colonies for further research.

Colonies of robots have many applications. They can solve larger problems than individual robots. The Idaho National Engineering and Environmental Laboratory has developed a colony of robots that seeks out liquid spills. They surround the spill by having each robot locate part of the edge of the spill [2]. Our behavior goal is to track and surround a mobile heated target. This will be accomplished by the use of inexpensive sensors and simple, but clever, algorithms.

This paper discusses aspects of relative localization in low cost colony robots. The following section discusses the background research of distributed sensor networks and the hardware and algorithms which is utilized to support such networks. Section 3 presents the hardware used for relative localization on our own system of colony robots, whereas Section 4 describes the algorithm these robots use to relatively localize. Section 5 and 6 report the results of actual experimentation on our sensor sub-systems and our relative localization algorithm. Section 7 discusses the advantages and disadvantages of our solutions to the technical challenges of colony robots. Finally, the last section offers some concluding remarks and the future direction our project will take.

2. Background

Recently there has been considerable interest in the localization problem for colony robots, as it is central to the development of distributed autonomous systems. Two primary localization paradigms exist: centralized and distributed. In centralized localization, sensor nodes exchange data with a common location, where computation is performed to calculate positions for each node in the network. Distributed localization does not require a dominate node, and sensor nodes determine their position by communicating with neighboring nodes.

Numerous methodologies have emerged to address multi-robot localization. Monte Carlo Localization (MCL) combines filtering with probabilistic models. The algorithm iteratively executes a prediction phase and an update phase, responding to increased positional uncertainty with new measurements to continually update its predicted location. One group implements the Monte Carlo algorithm for distributed, range-free localization using wireless communication [3]. Their findings indicate mobility can actually improve the accuracy of localization.

Markov localization is also a probabilistic algorithm that uses dead reckoning and environment measurements. Detections are ignored, and hence each robot maintains a belief over its position that is independent from other robot positions. In the most general case, the initial position of all robots is unknown and the system is initialized to a uniform distribution. Markov localization has been implemented in a network of mobile robots equipped with cameras and laser range-finders [4]. They found that their approach reduced uncertainty when compared to conventional single-robot localization.

Kalman filtering recursively computes each robot's state based on its previous state. A "centralized Kalman filter estimator" has combined sensor information across multiple robots to produce estimates of position at uniform accuracy [4]. Their implementation collected environment information using encoders, gyroscopes, and cameras.

These algorithms utilize complex, high-accuracy sensors. Sensors commonly employed include laser range finders, the global positioning system (GPS), color cameras, and omni-directional cameras [5-7]. Laser range finders are an expensive (~$5000) yet common way to accurately calculate range [5,8]. These sensors are 15cm x 15cm x 18cm" and weigh 4.5 kg [9]. Although GPS (ranging from ~$100 to ~$2000) can calculate position information up to an accuracy of one meter, it is not available in all areas, for example in buildings [10]. This limits use of GPS, because many colonies are developed indoors. In addition, an accuracy of one meter is not precise enough to colony robots. These sensors are not feasible to use on small, low-cost robots; although their calculations are accurate, they are prohibitively expensive.

3. Hardware

To allow the creation of a large robot colony on a restricted budget, inexpensive robots must be used. While cost considerations were paramount in the design of this robot, it must also be able to provide accurate and meaningful sensor data for use by the localization algorithm. Our solution to this is a small $300 robot weighing just over 0.5 kilograms, pictured in Figure 3.1.

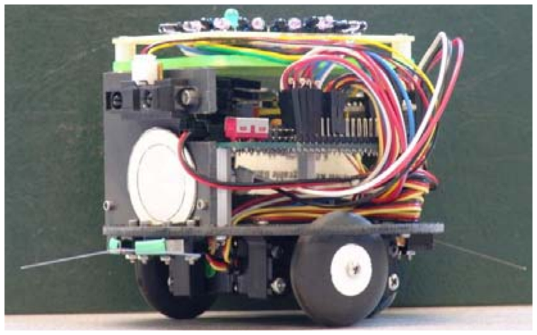
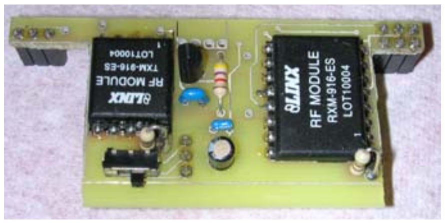
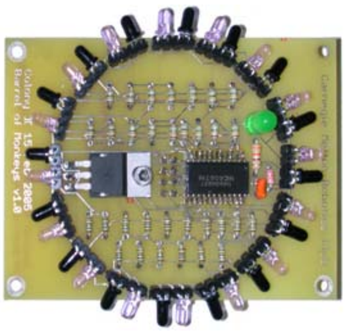


Figure 3.1 One of the colony robots

The brain of each individual robot is the Cerebellum, a PIC16F877-based micro-controller produced by Botrics, LLC. A Cerebellum costs $95 and can read analog sensor values, control the robot's servos, perform computation, as well as utilize a wireless board to send and receive data. The Cerebellum is located centrally on the robot to allow easy electrical connection to all of the robot components.

To allow the robots to share world information, a RF wireless board (Figure 3.2a) was custom-designed after no existing small and inexpensive wireless solutions were found. This wireless board attaches to the Cerebellum as a daughter-card, capturing the serial port of the PIC processor, in addition to the necessary control lines. Total parts cost of the board is under $40 per robot. One limitation of the Linx ES RF transmitters used on this board is that only one transmitter in the entire colony can be powered at a time or collisions render the data useless. In order to avoid these transmission collisions, a token ring network layer was developed for the PIC processor.


(a)


(b)

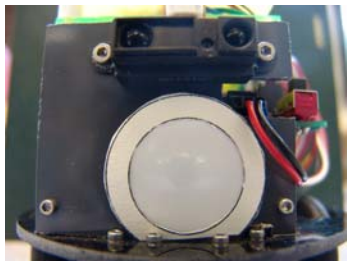

(c)

Figure 3.2 (a)The wireless board circuitry; (b)BOM circuit board; (c)Eltec IR-EYE and Sharp GP2D12 IR sensors

With this network protocol, robots are able to communicate with one another and share the position data required for relative localization. Since our relative localization approach was based on triangulation, it was also necessary to get either range or angle data between robots. Originally, we attempted to gauge distance using the RF signal strength provided by the wireless receivers, but we found this data to be wildly inaccurate. Thus, another source of relative location data was necessary.

Further investigation revealed no practical low-cost solutions to the range finding problem, prompting the development of a new board that could obtain angle information instead of distance. This yielded the Bearing and Orientation Sensor (BOM), a ring of 16 infrared (IR) emitter/detector pairs (Figure 3.2b), which could be constructed relatively inexpensively (~$17). One robot lights all of its emitters, creating a ring of uniform IR light, while the rest of the robots in the colony use their IR detectors to determine the angle of the light source relative to the detecting robot. The use of only 16 detectors provides discrete angle information (accurate to ±11.25 degrees). The board is elevated above the rest of the robot and requires line-of-sight from the emitting to the detecting robot. This is trivially achieved because all the robots operate in the same plane and are physically identical.

Eventually, we hope to demonstrate and apply our relative localization algorithm in colony applications such as herding and obstacle avoidance, and have added hardware that will be used towards these applications in the future. Now that the robots have relative localization, they need to be mobile and un-tethered to complete tasks. Mobility is accomplished by two wheels driven by two standard Hitec HS-311 servo motors costing $12 each. The robots are self-contained with a $10 battery pack that lasts approximately 2 to 3 hours. In addition to relative localization sensors, each robot also needs the ability to detect obstacles in its immediate environment. Four collision sensors ($6 per robot) were installed, two in the front and two in the rear. Each collision sensor consists of a leaf spring switch actuated by a 3" long piece of piano wire crimped to the leaf of the switch.

The BOM board yields angle data, but distance is also necessary for real-world applications. For this purpose, we use a Sharp GP2D120 infrared range finder ($12). The combination of these sensors generates enough information for the robots to execute a cooperative task, such as the coordinated discovery and capture of a mobile target. One such application is to model the capture of living beings that emit heat. To detect heat, we therefore employ an Eltec IR-EYE ($60), which is a pyroelectric sensor (Figure 3.2c). When a target is in range of this sensor, the IR rangefinder can be used in tandem to establish the distance to the target.

All components of the robot designed in the course of this project are open source and can be found on the Carnegie Mellon University Robotics Club website, http://www.roboticsclub.org/.

4. Algorithm for Relative Localization

4.1 local geometric solution

The main component of the algorithm uses a simple geometric method to solve the location of a point in a plane from the position of two other points and the angles between those points. This can be computed using the Law of Sines, which states that for any triangle ABC (as depicted in Figure 4.1), with angles ∠A, ∠B, ∠C, and sides AB, BC, CA, there exists the equality specified in equation (4.1).

$$\frac{\sin(\angle A)}{|BC|} = \frac{\sin(\angle B)}{|CA|} = \frac{\sin(\angle C)}{|AB|} \tag{4.1}$$

Assume that two points, A and B, and angles ∠A, ∠B, and ∠C are known. Then it is possible to solve the position of the third point, C, in one of two ways.

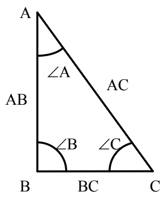


Figure 4.1 Example triangle with labeled sides, vertices, and angles

First, we can solve for the length of the side BC, using equation (4.2).

$$|BC| = \frac{|AB|\sin(\angle A)}{\sin(\angle C)} \tag{4.2}$$

Once this is computed, we simply travel a distance |BC| from point B, using the angle ∠B and the global angle of point A relative to point B, which is essentially atan2($A_y - B_y$, $A_x - B_x$) to determine the direction (atan2 is the arctangent function, sign corrected to output an angle in the range $[-\pi, \pi)$.

Alternately, solve for the length of the side CA, using equation (4.3).

$$|CA| = \frac{|AB|\sin(\angle B)}{\sin(\angle C)} \tag{4.3}$$

Using a similar method, we can determine the angle at which to travel length |CA| from point A. If our known variables are exact, then this ought to yield the same result in both cases. However, if the known values are not exact, as is the case, a better approximation may be obtained from averaging results.

The computational cost of this procedure is primarily bounded by two specific high-cost operations, since robots rely on a stored sine and cosine lookup table. The first is performing the distance measurement between points A and B. The measurement requires a square root computation and an arctangent calculation necessary to find the direction in which point C lies, after the appropriate side is calculated. It is possible to eliminate the latter cost by using two pieces of data outside of the point abstraction for robots. Each robot is stored with an orientation and raw sensor data relating the directions from the robot to other detected robots. By combining the two, the direction of point C can be ascertained with negligible expense.

4.2 local iterative solution

The sensor data robots receive is extremely low-resolution. However using the above method, we require only two robots to find a possible solution for the position of an unknown robot. Thus with *n* robots, we can construct $({}_nC_2)/2$ distinct solutions for the location of an unknown robot. Each solution is averaged with the previous approximation, which prevents sudden glitches in position, and then stored as the best known approximation of the given robot. It is then possible to reconstruct the position of the third robot to a much higher degree of accuracy than the individual results provide.

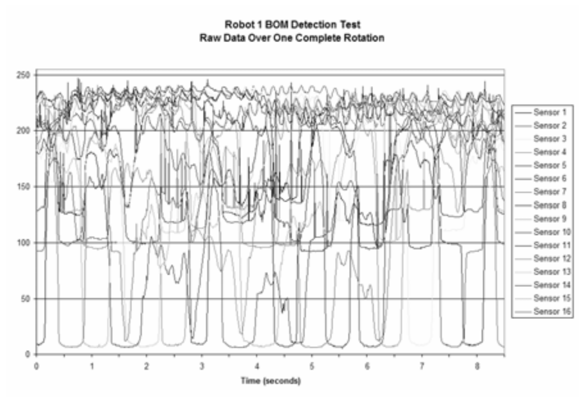
4.3 global iterative solution

The robots are arbitrarily ordered in a circular sequence. Each robot maintains the approximated location and orientation of all of the other robots, along with an array of the angles at which the robot detects all other robots.

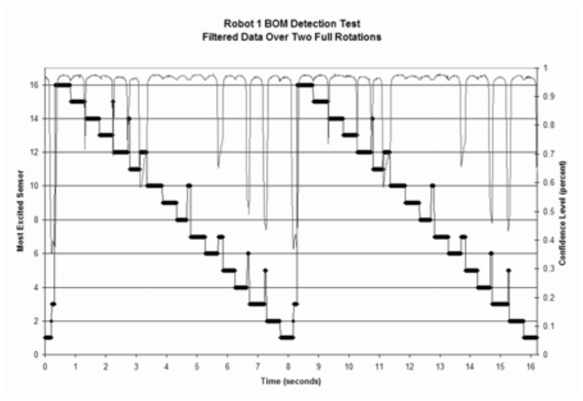
When the robots are initialized, these values are all zeroed. Each robot then takes turns resolving its own position and orientation. When it completes this operation, it broadcasts its updated position and orientation to the other robots, as well the most recent copy of its own sensor data. This is followed by the next robot solving its position and orientation, and so forth. As each robot refines its position calculation, the other robots use the updated location to improve their own positions. Within 1 to 10 iterations, the system reaches a stable arrangement. Robot movement subsequent to this point can be regarded as systematic error, and as such, is constantly compensated for as the algorithm iterates.

5. Hardware Analysis
5.1 the Bearing and Orientation Module

Since the Bearing and Orientation Module ("BOM") is the first sensing device of its kind for relative localization, it is important to characterize sensor error to determine the sensor's usefulness. To do so, two robots were positioned with a center-to-center distance of 30cm. One robot, tethered to a constant voltage source, constantly emitted IR light through its sixteen LEDs. The other robot was allowed to rotate in place for two rotations. Every 10ms, analog readings were taken from each of the sixteen phototransistors and recorded. Figure 5.1a is the raw data of such a test, where each trace is a series of phototransistor (sensor) readings from BOM number 1.



(a)



(b)

Figure 5.1 Raw and filtered BOM data

Sensor values recorded are lower as the perceived light intensity increases. A cursory inspection of the data shows a general trend of coverage from each sensor, nearly completely filling the appropriate 22.5 degree span of each detector. To figure out in which direction the robot is "seeing" the emitting robot, the microcontroller simply finds the sensor reporting the smallest value and therefore receiving the greatest amount of light. To determine the confidence of a particular reading, the microcontroller considers the minimum value of all the sensors; the larger this value, the smaller the amount of confidence the robot has in the reading. Figure 5.1b is a graph of the raw data, filtered to show the assumed direction and confidence level. The confidence level is expressed as a percentage, which is computed with equation (5.1).

$$\frac{255 - MIN(sensor\ readings)}{255} \qquad (5.1)$$

There are very few instances of the "wrong" sensor reading that it is more directly pointed toward the beacon source, though there is one notable example of sensor 10 mistakenly believing that it is pointed at the beacon between sensors 8 and 7 near 4.75s and 12.75s. This error occurred without a corresponding low confidence. We believe that this error can be remedied by adding shielding over each of the phototransistors to keep light from inadvertently entering the detector. All other errors in detection have corresponding drops in confidence. Besides adding shielding, we believe that we can reduce these errors by using IR phototransistors with a slightly wider angular response. This will remove the confidence level discrepancy present at sensor-to-sensor boundaries.

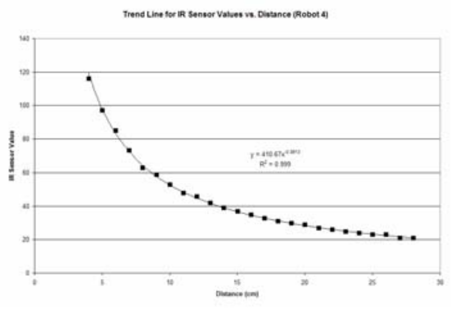
5.2 Sharp GP2D12 infrared range sensor

Because each robot has a Sharp IR range finder sensor, it is important to collect data that describes the typical behavior of these sensors as well as the variance between them. The test for gathering data involved placing a detectable object in front of a robot with the IR range finding sensor at different distances and recording the sensor output. Recordings occurred at distances that were 1cm apart with the first distance at 0cm away from the sensor and the last distance at 28cm from the sensor. This test was repeated for each robot.

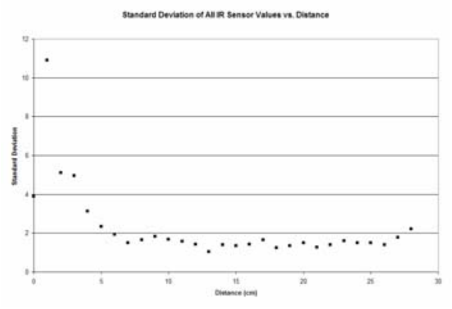
The data showed what was already commonly known about these IR range finders. The sensor output against the distance of the recording follows a power model of the form of equation (5.2) where *y* is the output value, *x* is the distance of the object from the center of the robot, and *a* and *b* are constants.

$$y = ax^b \tag{5.2}$$

As expected, this model fails when the sensor detects objects that are closer than ~3cm away. Readings in this range drastically deviate from the power model due to the build of the sensor. A power model was fit for the data of each sensor (for distances greater than ~3cm) and the constants *a* and *b* fall in the rough ranges of 370 to 500 and negative 0.98 to negative 0.83, respectively. It is important to note that some sensors fit their corresponding model better than others. For example, the sensor on Robot 4 has an R squared value of 0.999 (as seen in Figure 5.3a) while the sensor on Robot 3 only has an R squared value of 0.9928.



(a)



(b)

Figure 5.3 IR sensor values with best fit line and sensor standard deviation

It is more important, however, to describe the sensors collectively, namely how their performance varies from one to another. The standard deviation was calculated using the collected sensor outputs at each recorded distance, so that the standard deviation of all the sensors can be described in terms of distance. It was found that the standard deviation was very high when detecting objects at close range, roughly 0cm to 5cm. These high values have a peak at the distance of 1cm, indicating that not only do the sensors not follow the power model at close range, but also have very large variability between each other. The standard deviation is low and varies between 1.05 and 1.94 while detecting objects in the range of 5cm to 28cm. From this analysis, it is seen that the sensors are more predictable, collectively speaking, when detecting far objects and less predictable when detecting close range objects. The standard deviation is shown in Figure 5.3b.

6. Software Analysis

The localization algorithm was run repeatedly on random sets of robot colonies to characterize the error of the algorithm. The randomly placed robots' sensor data was created by finding inter-robot angles to a 22.5 degree accuracy, with a random Gaussian error. The generated sensor information was fed through the algorithm for ten iterations allowing each robot a chance to solve the layout. The algorithm generates a world with arbitrary center, rotation and scale. The layout makes it hard to compare the generated colony to the real colony. To account for this, the difference in centering, rotation, and scaling for each robot between the calculated and actual values was recorded and averaged. The average was used to transform (center, rotate, and scale) the entire colony to roughly the same position and orientation as the actual values. Since the algorithm is only designed to calculate relative localization, which is maintained during these calculations, this transformation does not affect the accuracy of the data. There the error was noted as the distance between each robot's calculated position and actual position. Error for a given colony was taken as the average of each robot’s error.

We found that the algorithm performs well overall, however occasionally a rouge robot will disrupt the entire colony’s position. To mitigate this error, we dropped the robots with the highest errors from the final transformation calculations. This made individual error higher, even though the rest of the colony’s error dropped.

The units of error here were arbitrary. The magnitude of the error is correlated to the robots placement, randomly, in a 1000 x 1000 unit box. Graphical representations of colonies with average error of 18, 37, and 120 are given for reference in Figure 6.1. A visual analysis reveals that even the robots with the highest error are not far from accurate. Generally speaking, adjacency and relative position were maintained even if part of the colony is displaced somewhat. In the case of higher error, there are some portions that are close to accurate, and others that have been distorted but not completely wrong in terms of inter robot positions.

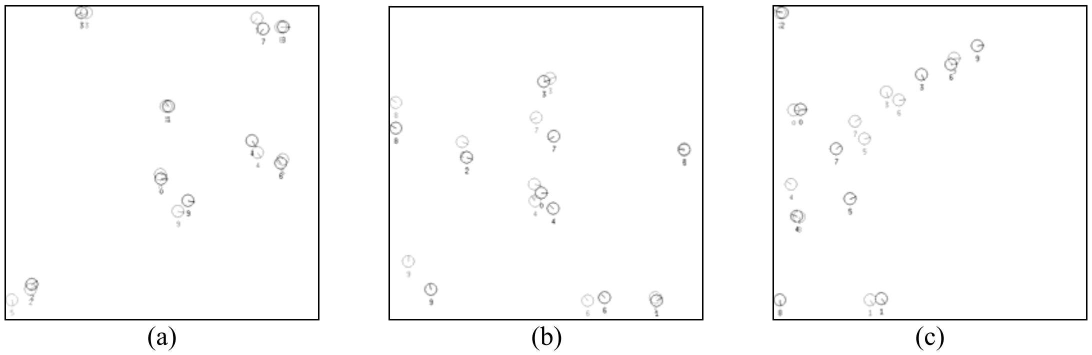

(a) (b) (c)

Figure 6.1 Colony simulations with average error of (a) 18 (b) 37 (c) 120, where black circles are real positions and grey are calculated (note that robot 5 in (a) is almost perfectly accurate).

After test consisting of 1000 runs of ten robot colonies, a mean error of 63 units was determined. However the mean is a poor measure of the general accuracy of the method since it is heavily thrown by a few high outliers. The median of 36 is a better measure, and the mode of 17 is the highest point on the error distribution graph (Figure 6.2). The dotted line on the graph represents the distribution of robot error with the refined algorithm that stops the most inaccurate robots from influencing calculations, while the solid line uses the original algorithm.

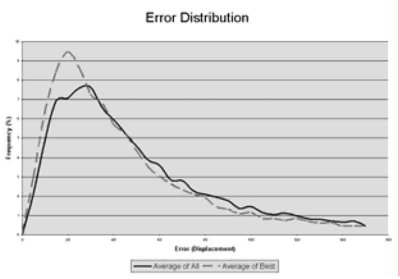


Figure 6.2 Robot error distribution over 1000 runs of ten robots each.

7. Discussion

Why would one build a colony of robots with limited functionality when robust and functional colonies have been built by other researchers? The short answer is money. By limiting the size and mechanical capability of the robot, and using less accurate sensors, the total cost of the robot is very low. This increases colony research in two ways. First, well-funded colony research groups can now build colonies with greater numbers of robots, which allows different algorithms and techniques to be explored. Secondly, colony research is now possible for groups who could not previously afford it.

Of course, when inexpensive and inaccurate components are used, new problems are presented. For example, the angular input to the relative localization algorithm has error of plus or minus 11.25 degrees. This increases the complexity of the position solver as well as making its output less precise. Had expensive and accurate sensors been used the software could have been simplified, but the quantity of data would greatly increase, pushing up computational requirements. Obviously, there is a trade off between accuracy and performance over cost. Our aim is to show that useful work can be performed at the low cost-end.

The colony is not up and running due to architectural limitations of our processor. We have, however, proved that all the main subsystems work. The token ring network is functional, the BOM works as an angular sensor, the pyroelectric and infrared distance sensors are accurate and repeatable. Future work will mainly be focused on adding software functionality to the robots. The relative localization algorithm has been successfully ported to the PIC processor. While the entirety of the algorithm takes fewer than 2000 assembly instructions, the required arrays of position and sensor data do not leave room for robot control, networking, or behavioral state. It is expected that the colony robots will move to an ATMEL-based architecture or that a second PIC processor will be added to increase processing capability and data storage. Work on the localization algorithm will also include better error identification and rejection, as well as static obstacle detection and reporting on the world map. Hardware improvements include adding a few more sensors to the BOM to fill in detection gaps. Additionally, features necessary for large colonies like automatic robot recharging and automatic distributed programming will be investigated.

8. Conclusion

We have developed hardware and software that has the potential to form a low-cost colony robot. We have demonstrated the effectiveness of our custom hardware, sensor package and relative localization algorithm. Future research will be conducted on integrating these sub-systems into a fully functional colony.

9. Acknowledgements

The authors wish to thank the Carnegie Mellon Robotics Club for the use of their equipment, money, space, and expertise, and Botrics, LLC for offering us discounts on their products. The authors also wish to thank Brian Kirby for his advice on wireless communication, and our advisor, Dr. Brett Browning, for having an awesome accent.

This research was supported in part by two Small Undergraduate Research Grants from the Undergraduate Research Office at Carnegie Mellon University under the direction of Dr. Janet Stocks.

10. References

[1] McLurken, James. "An Obsession with Robots." [Online Document]. Available: http://www.pbs.org/wgbh/nova/sciencenow/3204/03-talk.html

[2] Dalton, Louisa Wray. "Creating a Robot Colony." [Online Document]. Available: http://www.inel.gov/featurestories/12-01robots.shtml

[3] L. Hu, and D. Evans. "Localization for Mobile Sensor Networks." [Online Document]. Available: http://www.cs.virginia.edu/~evans/pubs/mobicom2004.pdf

[4] S. Roumeliotis, and G. Bekey. "Distributed Multirobot Localization," in IEEE Transactions on Robotics and Automation, Vol 18, No. 5, [Online Document], 2002. Available: http://cc.ee.ntu.edu.tw/~ncslab/StudyGroup/MultiVehicle/RoBe0210Localization.pdf

[5] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. "A Probabilistic Approach to Collaborative Multi-Robot Localization," in Special Issue of Autonomous Robots on Heterogenous Multi-Robot Systems, [Online Document], 2000. Available: http://www.vision.ethz.ch/kruppa/publications/fox00probabilistic.pdf

[6] J. Spletzer, A. K. Das, R. Fierro, C. J. Taylor, V. Kumar, and J. P. Ostrowski. "Cooperative Localization and Control for Muti-Robot Manipulation." [Online Document]. Available: http://www.cse.lehigh.edu/~spletzer/publications/iros01.pdf

[7] G. Pereira, V. Kumar, and M. Campos. "Localization and Tracking in Robot Networks." [Online Document]. Available: http://citeseer.ist.psu.edu/581417.html

[8] S. Thrun, W. Burgard, and D. Fox. "A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping," in IEEE International Conference on Robotics and Automation, [Online Document], 2000. Available: http://www-2.cs.cmu.edu/~thrun/papers/thrun.map3d.pdf

[9] C. Bader. "Sic Transits Transit." [Online Document]. Available: http://www.forester.net/gec_0007_sic.html

[10] R. Madhavan, K. Fregene, and L. Parker. "Distributed Cooperative Outdoor Multirobot Localization and Mapping." [Online Document], 2004. Available: http://www.cs.utk.edu/~parker/publications/AutoRob2004.pdf