Effect of Sensor and Actuator Quality on Robot Swarm Algorithm Performance

Nicholas Hoff, Robert Wood, and Radhika Nagpal
Harvard University, Electrical Engineering and Computer Science

***Abstract*— The performance of a swarm of robots depends on the hardware quality of the robots in the swarm. A swarm of robots with high–quality sensors and actuators is expected to out–perform a swarm of robots with low–quality sensors and actuators. This paper directly investigates the relationship between hardware quality and swarm performance. We take three common components of swarm algorithms (trail following, swarm expansion, and shape formation) and measure how they are affected by two common types of hardware inaccuracy (communication bearing reception error, and movement error) both in simulation and with E-Puck robots. We find that large amounts of both types of hardware error are required before performance appreciably decreases.**

I. Introduction

When constructing large swarms of robots, there is significant pressure for each robot to be simple and inexpensive. Cost, power, and manufacturing concerns all drive swarm robots toward fewer sensors, less computational capability, and less accurate locomotion. Our research ultimately drives toward swarms of robots that weigh ~5 g and cost ~$10 each. In this realm, every gram, penny, and milliwatt counts. Locomotion tends to be unreliable (either from walking legs, wheels, or vibration, for example), and there are few sensors viable for these scales and costs[4], [1], [15]. Improving any aspect of the hardware capabilities requires justification for the increase in weight, price, or power consumption.

However, robots with such minimal capabilities may not execute swarm algorithms well. For example, a robot may not be able to reliably follow a path if its movement is 50% random. The robots are under significant hardware pressure, but must still execute the algorithms. This raises the question, how good does the hardware have to be, or more generally, what is the relationship between hardware quality and algorithm performance? This is the central question addressed in this paper. To address it, we will take several swarm tasks and run them in simulation and on physical robots (E-Pucks) and artificially degrade the hardware capabilities of the robots while measuring the effect on in algorithm performance.

The swarm algorithms in the literature tend to be composed of a few common primitive behaviors, such as trail following, expansion, and shape formation [5], [11], [3]. A foraging algorithm, for example, could combine expansion (to find the target) with trail following (to return it to the nest). A flocking algorithm could use shape formation to form and maintain a flock, and homing (a special case of trail following) to move the flock. A coverage algorithm could use expansion on its own, combined with appropriate communication. Instead of measuring the effect of hardware quality on a few specific swarm algorithms, we will measure the effect on these three primitives in order to broaden the applicability of the results.

Contact nhoff@eecs.harvard.edu.

We have chosen two specific types of hardware capability to study. The first is the accuracy with which bearing data on incoming local communications can be measured. In order to coordinate with their neighbors, robots need to communicate and often need to measure the range and bearing to their neighbors. Measuring bearing usually requires multiple receivers arrayed around the robot. Using fewer sensors would yield savings in space, complexity, and power, but would yield a less accurate bearing measurement. The second type of hardware degradation is movement accuracy. With small robots, the locomotion tends to be imprecise. Making robots which can accurately move and turn requires accurate actuators and precision timing, or expensive feedback sensors which eat up processing time, power, and weight. Noisy open–loop locomotion would allow rough construction and would eliminate the feedback sensors.

We have chosen a very common robot model: simple tank steering and limited local communication. This model matches well with the small scale robots we will ultimately use, and is also common throughout the literature [17], [10], [4], [1] (see examples in Figure 1). We vary the amount of noise in robot movement, and the accuracy of the bearing measurement on communication receptions.

We find that the hardware must degrade considerably before a decrease in performance is observed. The tasks we test generally exhibit only mild decreases in performance even with $\pm 20\%$ movement error and 4 quadrants of bearing reception. Degrading the hardware further from this point causes large drops in performance. This suggests a design point at which hardware is as degraded as possible without appreciably impacting performance.

A. Related Work

Robustness and error tolerance are often–claimed aspects of multi–agent swarm algorithms [6], but quantitative hardware–based confirmations are less common. Often, a new algorithm is shown to perform well in the face of the particular hardware imperfections already present in the system ([17], [9]) or unplanned changes in the environment ([8], [18]), but it is not common for hardware quality to be independently varied while performance is measured.

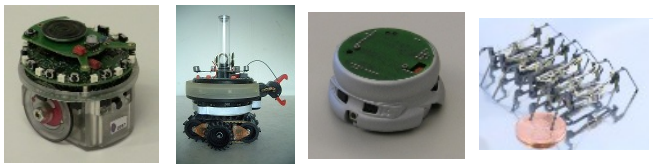

Fig. 1. An E-Puck robot with IR ring communication structure [12], [2], a SWARM-BOT [10] (photos kindly provided by Marco Dorigo, project coordinator), a Khepera II robot [13], and a crawling robot [1].

In one study which takes some hardware quality dependent measurements, a coverage algorithm is shown to be robust to some degradation in position measurement accuracy and communication range [16]. In this case, position error is modeled as a probability of calculating a completely random position. It has also been shown for search tasks that randomness of the target position and robot movement error can increase the attractiveness of random algorithms relative to coordinated ones [14]. We are not studying the benefit of coordinated algorithms, but rather the effect of hardware inaccuracies on the performance of coordinated algorithms. Our approach also differs in that we study several primitive behaviors rather than a few complete algorithms, covering a broader space of algorithms.

There is little information published using physical robots to measure the effect of the quality of a robot's sensors and actuators on the performance of common tasks running on those robots. In this work, we systematically vary sensor and actuator quality as the independent variable, and do this both in simulation and hardware.

II. Models, Tasks, and Hardware Variables

A. Robot and Environment Models

We use a simple robot model consisting of tank–like steering, a single forward–facing bump sensor, and the ability to send and directionally receive simple messages (a single byte) between robots within a small communication range. When a robot receives a communication, it can measure the range and bearing to the transmitting robot (see Figure 3).

The environment is modeled as a bounded region, possibly with obstacles (which block both movement and communication). Robots can not occupy the same physical space, and can not move obstacles. Snapshots of the model, both in simulation and the physical testbed, are shown in Figure 2.

B. Task Descriptions

Trail Following Task: The goal of the trail following task is for a robot to detect a trail marked by other robots, and repeatedly traverse the trail from one endpoint to the other. A set of robots is positioned to create a trail, remaining stationary. The first robot sends a signal indicating it is the start of the trail (perhaps the 'nest' in a foraging task, or the base station in a search–and–rescue task), and the last robot indicates that it is the end (the 'food' or 'victim'). The other robots that make up the trail broadcast integers representing their hop–count from each end of the trail (see Figure 2a). A 'walker' robot can navigate to either end by always moving toward the beacon with the smallest hop–count. The hop–count trails are hard–coded into the robots in these tests, so that the results focus solely on the trail–following ability of the walker. Distributed algorithms for developing the gradient are well known in general, and one is described for this hardware specifically in [7].

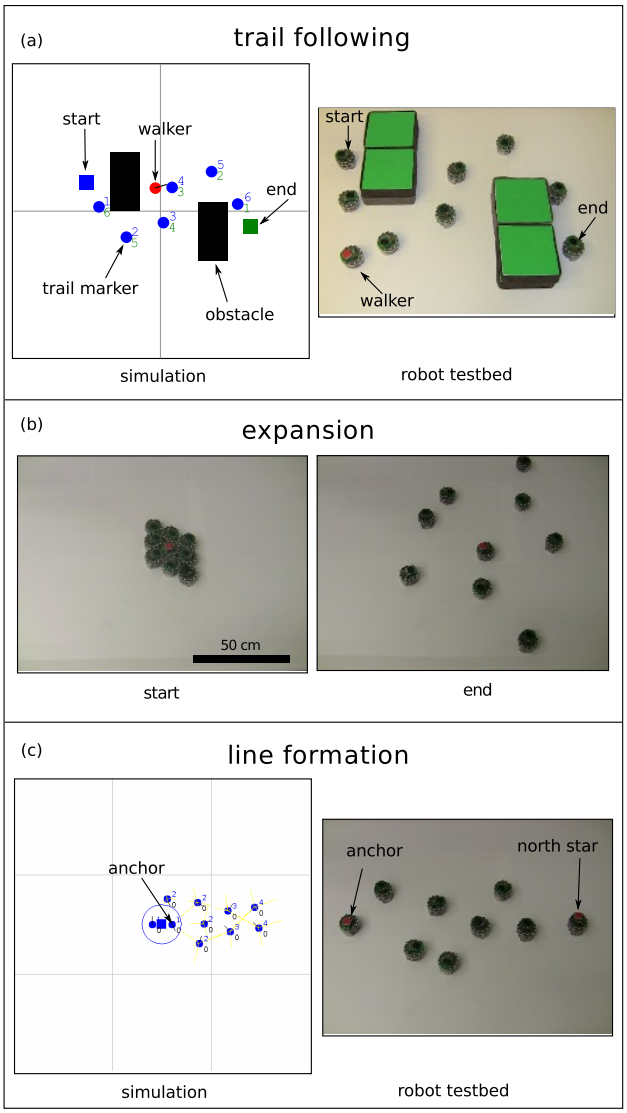

Fig. 2. Subfigure (a) shows the trail following task mid–execution. Subfigure (b) shows start and end configurations in the expansion task. Subfigure (c) shows the final state of the line formation task in simulation and on the robot testbed.

When the walker robot senses an obstacle or another robot in front of it (the bump sensors can not distinguish obstacles from robots), it simply turns left until it can no longer sense the obstruction, moves forward a short distance, then resumes its previous navigation. This results in the walker walking around obstacles to the left.

Our success metric for the trail following task will be the number of complete traversals the walker is able to make in 10 minutes, capturing both correctness and speed. Once the walker sees the robot marking the end, it turns around and navigates to the start, and repeats. Moving from one side of the trail to the other counts as a single traversal.

Expansion Task: In the expansion task, the swarm must expand to cover as much of the world as possible while maintaining a connected network of robot–to–robot communication. Example starting and final configurations are shown in Figure 2b. The expansion algorithm is simple: if a robot can hear more than three neighbors, it moves randomly, otherwise it remains stationary. Eventually, the robots will have expanded from the starting point such that each robot has three neighbors. If a search target were within this covered area, the swarm would be able to find it, so we are interested in how much area is covered by the swarm, and how fast the swarm reaches this final coverage.

Line Formation Task: In this task, the robots start in a clump and must form a line. This is done using a virtual–forces algorithm. Each robot measures the range and bearing to each of its neighbors, calculates a virtual force as if there were a spring between itself and the neighbor, sums the forces from all robots in its communication range, and moves in that direction. Robots which are close together will be forced apart, and robots too far apart will be pulled together. One robot does not move regardless of the virtual forces acting on it, thus acting as an anchor. Each robot also feels an additional small force which is always directed in the same global direction (arbitrarily called "north"). This could be achieved by placing a compass or sun sensor on the robots. E-Pucks do not have compasses, so in our tests it was achieved by placing a stationary robot at the edge of the field, transmitting a special message at full power, acting as the "north star". All robots placed an additional force on themselves pointing toward the north star robot.

After some amount of time, a line of robots is formed (example shown in Figure 2c), reaching from the anchor robot and stretching to the north. We are interested in how often this virtual–forces method succeeds in creating a line, and when it does, how long the line takes to form.

C. Hardware Variables

We test two types of hardware quality: bearing quantization and motor accuracy.

Bearing Quantization: The robots receive signals from other robots and can measure the range and bearing to each transmitter. This bearing measurement would likely be achieved by placing a ring of receivers around the robot. One could interpolate between the received intensities at each receiver to calculate a continuous bearing to the receiver. Our epucks calculate a continuous bearing measurement in this manner (with 12 sensors arranged around the robot). This requires high quality receivers and significant computation to do the interpolation. If there were only eight sensors, bearing would be obtained by simply knowing which sensor received the signal, and would only be known to an accuracy of 45°. In other words, there would only be eight quantized possibilities for the bearing measurement. This would be a significantly simpler design, requiring fewer sensors, less wiring, less power usage and weight, and fewer possibilities for failure.

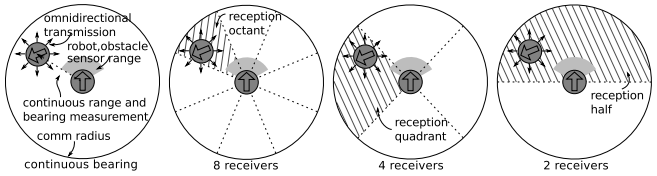

Fig. 3. With continuous bearing measurement, the receiver can calculate the bearing to the transmitter (to within the error of the sensors). With quantization of 4, for example, the receiver only knows which quadrant the transmitter is in.

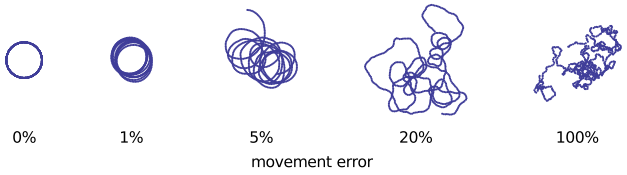

Fig. 4. In simulation, a single robot is commanded to move in a 2m–diameter circle ten times, with varying amounts of movement error. With $\pm 20\%$ error, there is still approximate looping behavior, and with $\pm 100\%$ error, it looks like completely random movement.

We test four possibilities for this bearing quantization: none (continuous bearing measurement), eight sensors, four sensors, and two sensors. With only two sensors, the robot only knows if the transmitter is in front of or behind it. These possibilities for bearing quantization are diagrammed in Figure 3. To achieve this on the actual robots, we artificially degrade the sensors in software.

Motor Accuracy: We assume that control of motion is implemented by velocity control of two motors. Uniformly distributed noise is artificially added to the commanded velocity to measure the effect of motor quality. We test four amounts of noise: $\pm 0\%$, $\pm 5\%$, $\pm 20\%$, and $\pm 100\%$. With $\pm 100\%$ noise, each wheel could rotate at a speed anywhere between 0 and twice the commanded rate. Because the noise on each wheel is independent, when a robot intends to go straight, it could actually veer off course.

An illustration is provided in Figure 4. As a simple example, a robot in simulation is commanded to walk ten times around a circle whose diameter is 25 times the robot diameter. This is repeated with various amounts of motor error. With $\pm 1\%$, the circles drift slowly, and with $\pm 100\%$, it resembles random walk.

Building robots with small movement error is difficult because it requires accurate actuators, feedback sensors (which require mass, power, and computation), or tight tolerances on construction (precise wheel diameter, leg length, power regulation). An algorithm which can perform well on robots with poor locomotion will be more useful because the robots will be easier to build, less costly, and simpler.

Error of $\pm 0\%$ is possible to achieve in the simulator, but the physical robots will, of course, have some minimum noise. Informal measurements indicate that this error is less than 1% for the motors on the E-Pucks. Strictly, movement error is added to the error already present in the system, which for the simulator is 0% to within the accuracy of a Java® `double`, and for the robots is less than 1%.

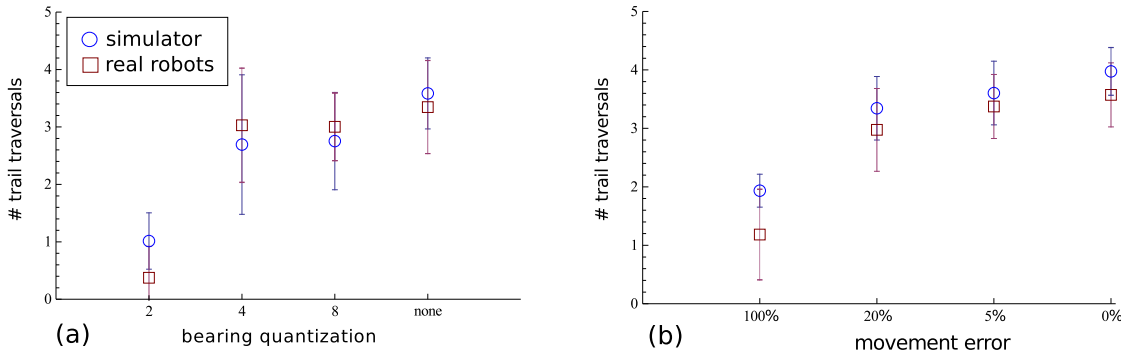

Fig. 5. Trail following performance. Simulation data points represent the average of 100 runs, physical robot data points represent the average of 5 runs, and error bars show one standard deviation. On the horizontal axes, hardware performance increases to the right. Bearing quantization must decrease to 2 until an appreciable decrease in performance is observed. Similarly, movement error must reach $\pm 100\%$ before performance decreases.

III. Tests and Results

We measure the effect that bearing quantization and motor accuracy have on the metrics described above: number of complete path traversals, amount of area coverage, speed of coverage, line formation success rate, and speed of line formation. Each test is done both in simulation and on physical robots. The data is shown in Figures 5, 6, and 7.

Robot size, movement speed, communication radius, and obstacle sensing range are all matched between simulation and hardware. Dropped communications do occur in the real robots but are not modeled in the simulator, sometimes causing a small difference between the robot performance and the simulator performance.

A. Trail Following Experiment

1) Number of Complete Path Traversals: With no bearing quantization, the robots make between 3 and 4 traversals in 10 minutes (Figure 5a). With only eight or even four sensing regions, the performance does not substantially fall. There is a substantial drop with only two sensors.

Movement error shows a similar trend. $\pm 5\%$ or even $\pm 20\%$ show no substantial drop in performance from $\pm 0\%$. Only with $\pm 100\%$ error does the performance fall. For both bearing quantization and movement error, the simulation and physical robots have comparable performance.

B. Expansion Experiments

1) Total Area Coverage: Neither bearing quantization nor movement error have an effect on area coverage, as seen in Figures 6a and 6b. This makes sense because the bearing to a transmitter has no effect on robot movement during this procedure; the only quantity that matters is the number of other robots in the sensing range. The robots simply move randomly when they can hear more than three other robots and stand still otherwise, regardless of where those other robots are. It also makes sense that the movement error has no effect because this metric measures the area covered when the expansion is finished, regardless of how long it took. So, with $\pm 100\%$ error, one might expect that coverage takes longer, but it still covers the same area in the end.

In the bearing quantization data, the amount of area covered by the physical robots is noticeably worse than in simulation. This is likely because of intermittent communications failures. In reality, the edge of the communication radius is not sharply defined. Sometimes communications are just lost, and sometimes they travel farther than normal. This can cause robots to wander too far away and get lost, and therefore not be counted.

2) Time Until Coverage: As expected, with $\pm 100\%$ error the swarm does take longer to reach its final coverage than with $\pm 0\%$ (Figure 6d). $\pm 5\%$ or $\pm 20\%$ show mostly non-degraded performance. Bearing quantization has no effect (Figure 6c), which makes sense for the same reason as above — the location of the receptions is not used in the expansion algorithm. The same effects of communications failures can also be seen here, with the results from the robots being consistently worse than the results from the simulation.

C. Line Formation Experiments

1) Line Formation Success Rate: In simulation, the robot swarm is almost always able to form the line, as seen in Figures 7a and 7b. With physical robots, however, failures are more common. Because of dropped communications, some of the virtual forces can be temporarily lost causing robots to move in the wrong directions, or even get completely lost. With only two reception directions (Figure 7a), the situation is even worse. In this case, the robots often fail, partly because of communications failures, but partly because they don't have enough information on which to calculate their movements. The virtual force always points directly ahead or backward. If the force points backward, the robot will turn until it is directly ahead (not necessarily a full 180° turn) and move. With so little information, the communication failures overwhelm them.

2) Line Formation Time: The time required for the swarm to form a line (when it does so successfully) is roughly the same with no bearing quantization as with a quantization level of eight or four. When only two sensors are present (bearing quantization of two) it takes twice as long or more, as seen in Figure 7c. Surprisingly, the amount of movement error seems to have no significant effect. Even with $\pm 100\%$ error, it takes approximately as long as with no error. During the process of forming the line, the robots 'jostle' around as they are moved by the virtual forces. Most of their movement

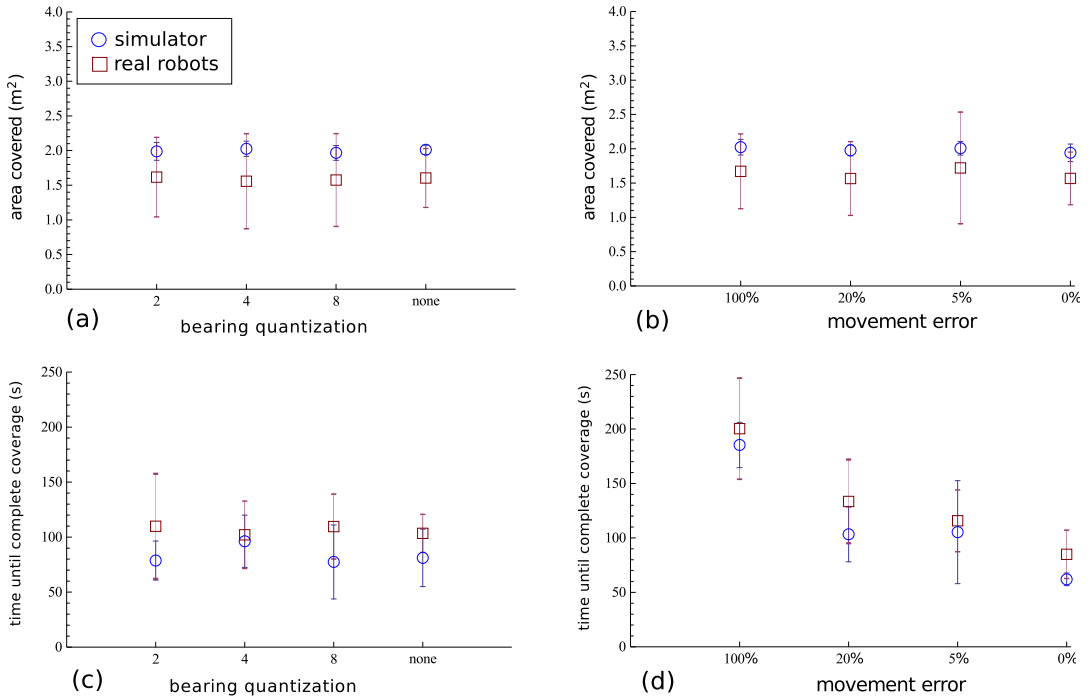

Fig. 6. Expansion performance. Simulation data points represent the average of 100 runs, physical robot data points represent the average of 5 runs, and error bars show one standard deviation. On the horizontal axes, hardware performance increases to the right. Bearing quantization has no effect on the total area covered nor the time it takes to reach coverage. Movement error does not affect total area coverage either, and must reach $\pm 100\%$ before it degrades the time until complete coverage.

appears to be this jostle, and over time they drift to the correct final positions. Apparently, adding even $\pm 100\%$ error just adds to the jostle, but the robots are still able to reach the final positions in roughly the same time.

IV. Conclusion

This paper explored the relationship between hardware quality and swarm algorithm performance. Specifically, we varied movement accuracy and bearing measurement quantization, and measured the effect on trail following, expansion, and line formation performance. Minimal hardware is desirable when constructing a swarm of robots, because high quality hardware is expensive and could reduce reliability. One goal of this work is to answer the hypothetical question from an engineer building a robot swarm: "how good do the robots need to be?". The data provide a partial answer. For the trail following task, it appears that a communication system which can only distinguish four reception directions and movement with $\pm 20\%$ error is good enough, meaning that it will achieve performance comparable to a robot with continuous bearing resolution. $\pm 20\%$ seems to also be sufficient for the expansion task. For the line formation task, quantization of four offers similar performance to continuous resolution, and even $\pm 100\%$ movement error is fine.

Overall, robotic hardware offering $\pm 20\%$ movement error and bearing quantization of four yield mostly undegraded performance, suggesting those are good design points for swarm robots. Slight gains may be achieved moving from $\pm 20\%$ to $\pm 0\%$, or from quantization of four to continuous resolution, but these are only worth paying for if the cost of those higher–quality sensors and actuators is commensurate with the small gain.

These relationships are important to understand as we move toward swarms of smaller and more constrained robots. Eventually, we aim to implement swarm algorithms on very small robots [1] on which sensor and actuator precision is very expensive.

In the future, we will expand this work to test other primitives [3], [11], and to test whether combinations of primitives show the same hardware degradation relationships. If so, then the data could be used to assess the implications for whole algorithms. Additionally, although bearing quantization and movement error are two important hardware variables, others could be more relevant to some particular robots, so a broader range of hardware variables would be useful.

This work was funded by the National Science Foundation under grant #IIS-0811571. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Appendix A. Task Pseudocode

The trail following, expansion, and shape formation tasks all have straightforward implementations. For clarity, pseudocode for each is shown in Figure 8.

References

[1] A. Baisch, R. Wood. Design and Fabrication of the Harvard Ambulatory Micro-Robot. *Intern. Symp. on Robotics Research*, 2009.

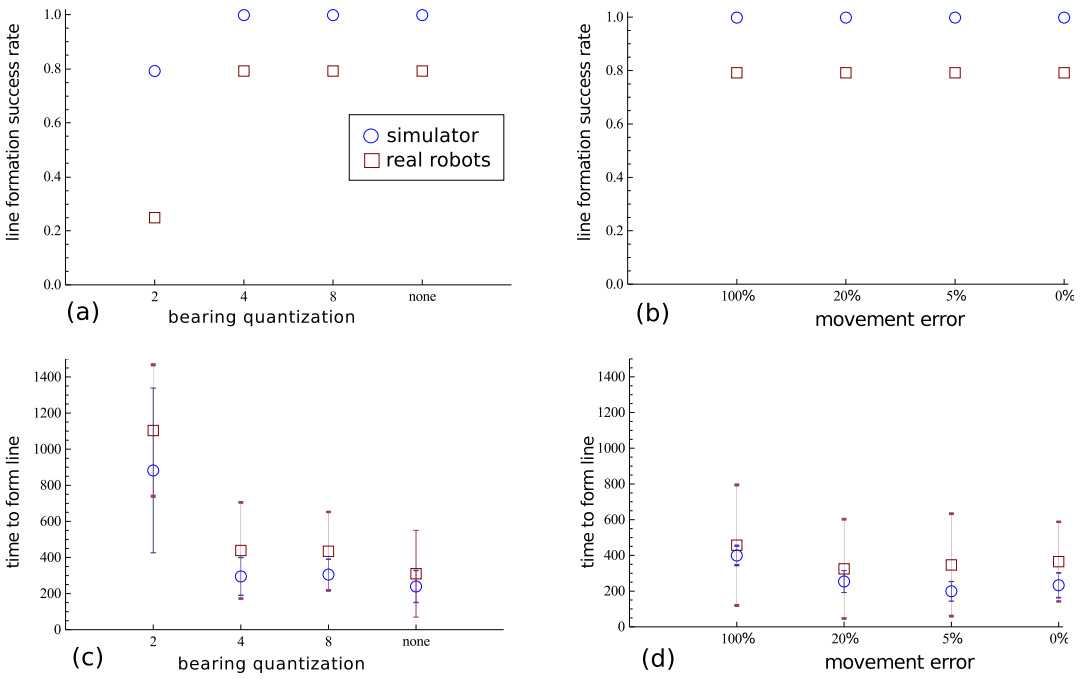

Fig. 7. Line formation performance. Simulation data points represent the average of 10 runs, physical robot data points represent the average of 5 runs, and error bars show one standard deviation. On the horizontal axes, hardware performance increases to the right. The swarm is capable of forming the line most of the time, except under a bearing quantization of 2, when it also takes much longer to form the line.

trail following

```
if outbound
  detect all outbound gradient values in range
  if detect endpoint
    begin inbound, return
  if pointing in direction of smallest value
    if facing obstacle
      turn left until no obstacle, then move forward
    else
      move forward
  else
    turn toward direction of smallest value
else
  (similar procedure to move inbound)
```

expansion

```
n = number of receptions in comm range
if n > 3
  move randomly, do not transmit
else
  stand still and transmit
```

line formation

```
detect all robots in comm range
for each reception
  calcualte virtual spring force
vector sum virtual spring forces
add virtual north force
if pointing in the direction of overall force
  if facing obstacle
      turn left until no obstacle, then move forward
  else
      move forward
else
  turn toward direction of overall force
```

Fig. 8. Pseudocode for each task.

[2] A. Gutierrez et. al. Open E-Puck Range & Bearing Miniaturized Board for Local Communication in Swarm Robotics. *International Conference on Robotics and Automation*, 2009.

[3] J. Bachrach, J. Beal, and J. McLurkin. Composable continuous space programs for robotic swarms. *Neural Computing and Applications, Special Issue on Swarms*, 19(6):825–847, 2010.

[4] G. Caprari, T. Estier, and R. Siegwart. Fascination of down scaling – Alice the sugar cube robot. *Journal of Micro-Mechatronics, VSP, Utrecht*, 1(3):177–189, 2002.

[5] Chris Jones and Maja Mataric. Behavior-Based Coordination in Multi-Robot Systems. *Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications*, 2005.

[6] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence*. Oxford University Press, 1999.

[7] Nicholas Hoff, Amelia Sagoff, Robert J. Wood, and Radhika Nagpal. Two foraging algorithms for robot swarms using only local communication. In *IEEE International Conference on Robotics and Biomimetics, ROBIO*, 2010.

[8] B. Hrolenok, S. Luke, K. Sullivan, and C. Vo. Collaborative foraging using beacons. In *Proc. of 9th Int. Conf. on Autonomous Agents and Multi-agent Systems (AAMAS 2010)*, 2010.

[9] J. Svennebring and S. Koenig. Building Terrain-Covering Ant Robots. *Autonomous Robots*, 16(3):313–332, 2004.

[10] Dorigo M., Tuci E., Trianni V., Groß R., Nouyan S., Ampatzis C., Labella T.H., O'Grady R., Bonani M., and Mondada F. Design and implementation of colonies of self-assembling robots. *Computational Intelligence: Principles and Practice, Gary Y. Yen and David B. Fogel (eds.), IEEE Computational Intelligence Society*, pages 103–135, 2006.

[11] James McLurkin. *Stupid Robot Tricks: A Behavior-Based Distributed Algorithm Library for Programming Swarms of Robots*. S.M. thesis, Massachusetts Institute of Technology, 2004.

[12] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, Floreano D., and A. Martinoli. The e-puck, a robot designed for education in engineering. *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, 1(1):59–65, 2009.

[13] F. Mondada, E. Franzi, and A. Guignard. The development of khepera. In *In Proc. of the 1st Intern. Khepera Workshop*, pages 7–13, 1999.

[14] J. Pugh and A. Martinoli. The cost of reality: Effects of real–world factors on multi-robot search. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

[15] Mike Rubenstein and Radhika Nagpal. Kilobot: A robotic module for demonstrating behaviors in a large scale (2^{10} units) collective. *Modular Robotics Workshop, IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010.

[16] S. Rutishauser, N. Correll, and A. Martinoli. Collaborative coverage using a swarm of networked miniature robots. *Robotics and Automated Systems*, 2009.

[17] M. Schwager, J. McLurkin, J. J. E. Slotine, and D. Rus. From theory to practice: Distributed coverage control experiments with groups of robots. In *Proceedings of International Symposium on Experimental Robotics*, Athens, Greece, jul 2008.

[18] Chih-Han Yu. *Biologically–Inspired Control for Self–Adaptive Multi-agent Systems*. PhD thesis, Harvard University, 2010.