Self-Organized Flocking with Agent Failure: Off-Line Optimization and Demonstration with Real Robots

Adam T. Hayes, Parsa Dormiani-Tabatabaei

Abstract— **This paper presents an investigation of flocking, the formation and maintenance of coherent group movement, by teams of autonomous mobile robots using principles of Swarm Intelligence. First, we present a simple flocking task, and we describe a leaderless distributed flocking algorithm (LD) that is more conducive to implementation on embodied agents than the established algorithms used in computer animation. Next, we use an embodied simulator and reinforcement learning techniques to optimize LD performance under different conditions, showing that this method can be used not only to improve performance but also to gain insight into which algorithm components contribute most to system behavior. Finally, we demonstrate that a group of real robots executing LD with emulated sensors can successfully flock (even in the presence of individual agent failure) and that systematic characterization (and therefore optimization) of real robot flocking parameters is achievable.**

Keywords— **Collective Autonomous Robotics, Flocking, Off-line Optimization, Swarm Intelligence, Self-Organization**

I. Introduction

FLOCKING, the formation and maintenance of coherent group movement, has long been studied in natural systems, and more recently efforts have been made to reproduce this type of behavior in artificial systems. The first such work appeared in the context of computer animation [1], and since then this behavior has been extensively studied in simulation (e.g. [2]), and less so on real robots [3], [4]. Theoretical treatments of the stability of flocking behavior have also been presented [5], [6]. The study of flocking is distinct from that of formation control (e.g. [7], [8]), because the goal of flocking is simply to achieve and maintain coherent group movement rather than to govern specific inter-agent position relationships. Flocking is better suited for implementation on large groups of agents (hundreds to thousands) where the overhead of extensive inter-agent communication and unique agent identification renders formation control inefficient. Also, like formation control, flocking is not an end in itself, but rather can be used as a component of a larger multi-agent system, perhaps simplifying the transport of large numbers of agents or organizing the nodes of a distributed sensing system.

This paper presents an investigation of flocking by groups of autonomous mobile robots using principles of Swarm Intelligence (SI), a computational and behavioral metaphor for solving distributed problems that takes its inspiration from biological examples provided by social insects. In most biological cases studied so far, robust and capable group behavior has been found to be mediated by nothing more than a small set of simple interactions among individuals and between individuals and the environment [9]. The application of SI principles to autonomous collective robotics aims to develop robust task solving by minimizing the complexity of the individual units and emphasizing parallelism, exploitation of direct or indirect interactions, and distributedness. There are three main advantages of this approach: first, scalability from a few to thousands of units, second, flexibility, as units can be dynamically added or removed without explicit reorganization, and third, increased system robustness, not only through unit redundancy but also through the design of minimalist units. Several examples of collective robotics tasks solved with SI principles can be found in the literature: aggregation [10], segregation [11], foraging [12], and odor localization [13].

Solving a task using the SI approach reduces to determining a set of local rules which, when carried out in parallel by a group of agents, has the desired global effect. These rules can involve the control of behavior (software mediated) and/or direct physical interactions (hardware mediated). Each rule can have a set of associated parameters, and once the rules have been chosen, maximizing team performance involves solving a global optimization problem. Because SI systems depend heavily on sensitive agent-to-agent and agent-to-environment interactions, performance is often stochastic, and evaluative, rather than gradient based, search methods are appropriate. This type of control optimization has been extensively studied for the case of a single agent [14], [15], [16], as well as for multiple agents [17], [18], [19].

II. The Flocking Task

A. Task Definition

The flocking task examined in this paper is similar in form to the cooperative movement task studied in [20]. The agents begin each trial at random positions and orientations within an area A located in the corner of a square arena of length L. During the trial the agents move diagonally across the arena through an obstacle field to an area B in the opposite corner. There is some uniform probability of failure per time period θ that an agent will 'fail' during traversal, meaning that it stops moving but other agents can still see it. To reduce the number of trials that can

Collective Robotics Group
136-93 California Institute of Technology, Pasadena CA 91125
athayes, parsa2k@caltech.edu - http://www.coro.caltech.edu

never complete the task, the number of failures is capped at no more than half of the total number of agents, and the trial is declared finished when half of the agents have entered area B.

For the purposes of this work we define the system performance to be a combination of the time required to complete the task T, the average distance traveled by each of the successful agents D, and the average inter-agent distance of operational agents I. These factors can be combined to form a cost metric C:

$$C = \alpha T + \beta D + \gamma I \tag{1}$$

α is taken to be the cost per unit time of not completing the task, β is the cost per unit distance of running each agent, and γ the cost incurred per unit distance of inter-agent separation (e.g. if the agents provide mutual protection when grouped together, looser groups would be associated with lower protection and higher costs due to agent loss). C represents the total cost incurred before the task is completed. By choosing specific values for α, β, and γ the appropriate relationship between time required, energy used, and inter-agent spacing can be generated for evaluating any particular application.

B. The LD Flocking Algorithm

[1] identifies three behavior types that lead to simulated flocking: separation, alignment, and cohesion. However, much of the robotic work on flocking ([3], [4]) relies solely on balanced combinations of separation and cohesion (i.e., flock centering) to produce flocking behavior. It is likely that the inclusion of an alignment term into robotic flocking algorithms will improve performance (particularly by speeding up flock formation times), but there is a cost to making heading information explicitly available within a system. LD is essentially an extension of the flock centering algorithm presented in [2], incorporating an explicit collision avoidance mechanism (as they suggest) as well as an implicit velocity matching behavior (i.e., an alignment term – via the comparison of sequential flock centering data). Thus LD should exhibit better flocking performance than previous robotic algorithms (though comparative data is unavailable) while not significantly complicating implementation on real robots. Because LD does not explicitly use the alignment of other group members, individual agents need not be able to sense their neighbors' orientation, and range- and-bearing information suffices.

Specifically, LD is defined as follows. There are two basic behaviors, collision avoidance and velocity matching flock centering. Collision avoidance is activated whenever an agent's collision sensors detect the presence of an obstacle (which may be either an environmental obstacle or another team member), and it mediates a turn away from the obstacle. Flock centering is active whenever collision avoidance is not, and it involves the generation of a target vector and a target difference vector as well as a mapping from those vectors to wheel speed commands.

After every sensory input cycle, each agent can utilize information from up to N closest neighbors residing in a region surrounding the agent defined by a maximum range M, as shown in Figure 1. Range ($\|n_i\|$) and bearing ($-\pi \leq \hat{n}_i \leq \pi$) information from this set of m neighbors ($i = 1...m, 0 \leq m \leq N$), along with the desired cushion distance C between each agent and its neighbors, can be used to generate an instantaneous center of mass vector CoM for each agent.

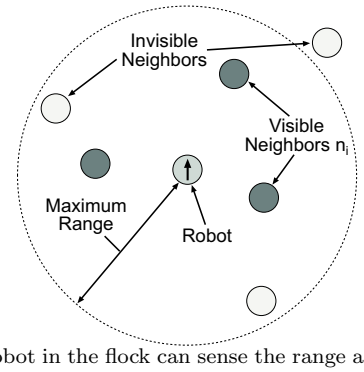


Fig. 1. Each robot in the flock can sense the range and bearing of up to N neighbors within a sensory area defined by a maximum range M. In this example $N = 3$.

$$CoM = \sum_{i=1}^{m} \left(\frac{\|n_i\| - C}{N}, \hat{n}_i \right) + (P, \hat{b}) \tag{2}$$

CoM is normalized by the maximum number of neighbors to reduce the vector sizes seen at large values of N. P is a tunable system parameter that represents the strength of the attraction to the goal area, and $\hat{b}$ is the (agent centered) heading of the goal area (e.g. supplied by a GPS type signal). Because the flocking task being studied not only favors coherent movement with flock neighbors but also directed movement toward the goal, this vector is added to CoM to encourage movement in the proper direction.

To generate ΔCoM, the value of CoM generated in the previous sensory cycle (CoM_{prev}) is transformed into the current agent coordinates (CoM') and combined with the current CoM:

$$CoM' = \left[\left(CoM_{prev} - \frac{\Delta h}{2} \right) - e \right] - \frac{\Delta h}{2} \tag{3}$$

$$\Delta CoM = CoM - CoM' \tag{4}$$

Δh is the agent's change in heading between sensory cycles, and e is the agent's change in position. The relationships between the algorithm components are summarized in Figure 2.

The agent has access to its desired position with respect to its neighbors CoM as well as how that location is moving with respect to the agent ΔCoM. These values are used to generate the motor commands:

$$U = \frac{M + K_2 \|CoM\| \cos(\hat{CoM})}{M} \tag{5}$$

$$LSpeed = (V - K_0(\hat{CoM'} + K_1 \hat{CoM})) * U \tag{6}$$

$$RSpeed = (V + K_0(\hat{CoM'} + K_1 \hat{CoM})) * U \tag{7}$$

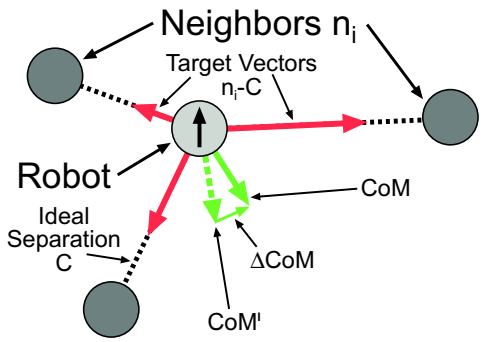


Fig. 2. A summary of the generation of CoM^+.

The motor speeds are biased at a desired travel speed V. They are changed differentially to rotate toward the heading specified by a weighted sum of the direction of the desired location and the direction of movement of the desired location. K_0 is a weighting parameter that determines how fast an agent can approach this target heading. K_1 weights the influence of the desired location direction versus the desired location movement direction. A small K_1 will induce agents to align with their neighbors (thus minimizing $\hat{CoM'}$) rather than to move toward their desired locations, although once alignment is achieved the agents will gradually steer toward CoM. The agents also uniformly speed up or slow down to approach CoM using $\|CoM\|$ and another gain parameter K_2. Note that it is not necessary to calculate the optimal movement necessary to reach the goal position in order to have a functional system. As long as the commanded wheel speeds bring each agent closer to its desired position during each sensory cycle (and CoM cannot move faster than the agent itself), in steady state all agents will approach their goal positions. Formal stability conditions and proofs are not examined in this paper, although stable flocking systems were observed over a broad range of algorithm parameters. There are 8 tunable algorithm parameters, as shown in Table I.

TABLE I
LEADERLESS DISTRIBUTED FLOCKING ALGORITHM PARAMETERS

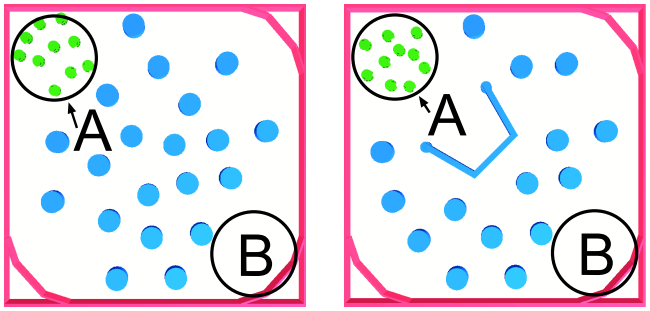
V	Desired forward speed
N	Maximum number of neighbors
C	Desired distance between agents
M	Maximum sensor range
$K0-2$	Motor speed gain parameters
P	Target attraction

III. Materials and Methods

A. Embodied Simulation

To maintain a close correspondence with the structure and function of the real robots, we used Webots [21], a 3D sensor-based, kinematic simulator, originally developed for Khepera robots [22], to systematically investigate the performance of LD in simulation. This embodied simulator has previously been shown to generate data that closely matches real Khepera [23], [10], and Moorebot [13] (see Section III C) experiments, so we were confident that real robot behavior was accurately captured.

The physical arena was captured in Webots to allow comparison with data generated by the real robots, and two different obstacle fields were studied. The simpler of the two (`Obs1`) contained only cylindrical obstacles that were twice as large as each agent, while the more complex (`Obs2`) also contained a three-sided barrier that obstructed the direct path between the start (A) and goal (B) areas. These environments are shown in Figure 3a and b.



(**a**) (**b**)

Fig. 3. (**a**) `Obs1` and (**b**) `Obs2`, seen from above. The start and goal areas are indicated. The large disks are the obstacles, and the smaller disks (shown here within the start area) are the agents.

B. Off-Line Machine Learning Optimization

When a multi-agent system returns highly stochastic performance values and possesses complex interactions among its different sub-tasks, the use of homogeneous controllers with a global reward signal provides a way of addressing the credit assignment problem for off-line control optimization. By making the learning agent operate in the space of algorithm parameters and providing only measures of group performance (rather than feedback from individual actions), there effectively becomes one agent and one reward signal and the credit assignment problem no longer applies [24]. This may be an extreme simplification of the problem of learning in distributed multi-agent systems, but it allows the optimization of team performance when evaluation is expensive, as is often the case with real-world environments that include a strong stochastic component.

The optimization procedure for this flocking task involves the off-line tuning of 8 parameters. Since a full 8-dimensional optimization is not computationally feasible, we instead perform 8 sequential 1-dimensional optimizations (termed an optimization cycle), with each parameter optimized while the others remain fixed. While this restriction may make finding the optimal parameter set difficult in some search domains, it does not do so in the particular case being studied (see Section IV), and it allows performance improvements to be achieved in a reasonable amount of time. In this initial study the selection of design points (i.e., specific parameter values over which to optimize) is done a priori, although there are techniques for selecting

them adaptively [16], [25] which may be utilized in further studies. Each parameter space is bounded and linearly discretized to include a range of important values, as determined by preliminary experiments. At the beginning of each optimization run the variable values are randomly initialized.

For a given parameter, once the design points x_i ($i = 1...R_p$, where R_p is the total number of points for parameter p), are selected, the optimization is performed as follows:

1) Initialize the set of active points B to include all x_i.

2) At each iteration j, simulate a trial at each x_i in B, storing the result $-C = y_i^j$ in Y_i. C is defined in equation 1. The negative of the cost function is used so that the maximum value is the optimum.

3) If $j > \epsilon$, first, using Tukey's HSD procedure [26], determine the critical difference D (to significance γ which must be equalled or exceeded by the difference of two means for that difference to be declared significant. Next, let

$$P_{max} = \max_i E(Y_i) = E(Y_m) \tag{8}$$

For each $x_i \in B$, if

$$(1 - \eta)P_{max} - E(Y_i) > D \tag{9}$$

where $i \neq m$, remove x_i from A. $E(x)$ represents the expected value of x.

4) If more than one x_i remains in B, go to *Step 2*.

At the end of the process, the remaining point x_{Max} represents the best guess at the optimum value for the parameter currently being optimized given the other fixed parameter values. After each cycle through all parameters, the resulting parameter set sampled (via 30 trials) and then used as the input set for the next cycle. This algorithm is defined by the initial design choice method and three parameters: η, γ, and ϵ. η defines the margin around x_{Max} in which it is defined to be not cost effective to further optimize (e.g., if $\eta = .1$ and all remaining design points are determined to be less than 10% greater than the maximum, the optimization stops). γ defines the desired level of certainty of achievement of the margin defined by η. ϵ sets the minimum number of trials necessary so that Tukey's HSD procedure is accurate. Tukey's HSD procedure was chosen for this multiple comparison task because it has been shown to be relatively robust to violations in assumptions about the data (namely normality of data and equal variances among samples).

The cycle stopping condition for an optimization run ideally would be reached when the input and output of a cycle are the same parameter set. However, due to the stochastic nature of the optimization process and the size of the parameter space this event is unlikely. Therefore, in this work we set the number of cycles per run to 10.

C. Real Robots

We use a group of 10 Moorebots , as shown in Figure 4. The flocking arena is 6.7 by 6.7 m, and the robots are 24 cm in diameter. The layout of the arena is the same as shown in Figure 3, in this case a single obstacle was placed in the center of the arena. In addition to the standard configuration, as described in [27], each robot is equipped with four Sharp GP2-D02 infra-red range sensors for collision avoidance.

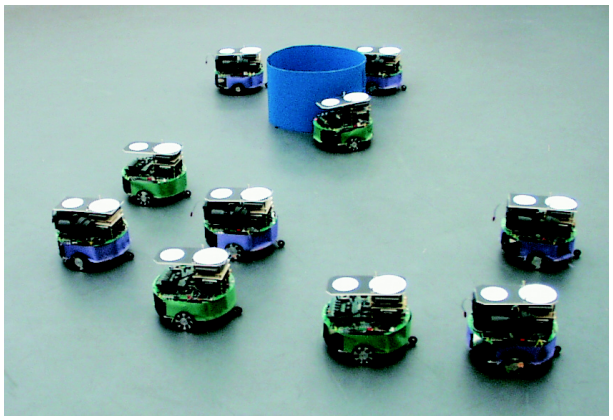


Fig. 4. 10 Moorebots flocking.

An overhead camera tracking system, combined with a radio LAN among the robots and an external workstation, is used to log position data during the trials, reposition the robots between trials, and emulate the range and bearing sensor signals.

IV. Results and Discussion

A. Optimization with the Embodied Simulator

We optimized under four different conditions consisting of 10 runs each: `Obs1` and `Obs2`, each with `F` and without failures `NF`. Optimization parameter values were as follows: $\eta = .05$, $\gamma = .05$, and $\epsilon = 10$.

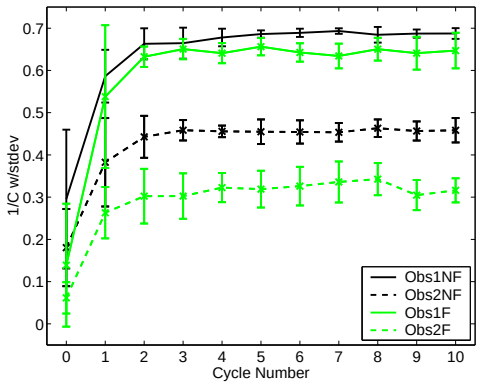


Fig. 5. Per-cycle flocking performance for each experimental condition.

Figure 5 shows the flocking performance at each cycle for each of the four conditions. Cycle 0 data represents the performance of the initial parameter sets. `Obs1NF` converges to the highest performance value, with `Obs1F` slightly worse, followed by `Obs2NF` and then `Obs2F`. This shows that `Obs2` is a more difficult environment than `Obs1`, and the presence of agent failures can hurt performance. Under all four

conditions the means and standard deviations stabilize after 4 optimization cycles, showing that optimization does improve performance and is complete after a small number of cycles. The fact that all conditions have a small standard deviation across runs once optimized (after cycle 4) suggests that even though the optimization algorithm searches only one dimension at a time, it is performing an effective search of the fitness landscape and is not susceptible to being trapped in local minima. The standard deviations for the `F` and `Obs2` conditions are larger because in these environments occasional runs fail to complete within the timeout period, and thus the performance metrics (for individual parameter sets) have higher variances.

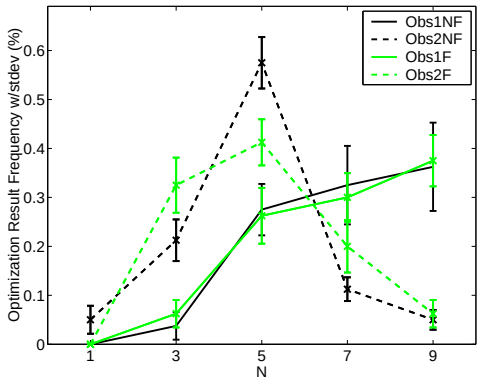


Fig. 6. The optimal result frequency curves for N, the maximum number of neighbors observed while flocking.

The optimization procedure can do more than improve system performance, because by looking at the optimized parameter values one can gain insight into the operation of the algorithm itself. We analyzed the optimized parameters first by combining cycle results 4-10 from each run, and then averaging the selected parameter distributions across the 10 runs for each condition (this results in an optimization result frequency curve). Figure 6 shows that there are optimal values of N for each environment and that they are different, with `Obs2` preferring smaller neighborhoods. This result makes intuitive sense because when an agent is listening to fewer neighbors, it is less likely to be impeded by a neighbor that is caught behind a barrier (which is more common in `Obs2`), while larger neighborhoods allow for tighter flocks and thus a higher performance level. Using pointwise one-way ANOVA comparisons ($p < .01$) and a threshold of > 1 significant difference, we determined that the optimization result frequency curves for the `Obs1` conditions differ from those of `Obs2`, while they remain the same within each simulated environment. In fact, for the `Obs1` conditions the result frequency curves did not differ for any parameter, indicating that the best solution in that environment remained the best even in the presence of agent failure. This makes sense because a failed agent simply becomes another circular obstacle, although it might have been expected that the presence of failed agents would reduce the size of the agent neighborhood (so failed agents do not impede the progress of those still active).

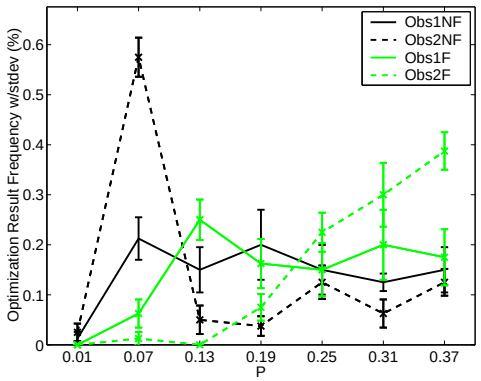


Fig. 7. The optimal result frequency curves for P, the attractive power of the goal area.

In the `Obs2` environment, the optimal parameter values are influenced by the presence of agent failure, as shown in Figure 7. In the absence of agent failure, a low value of P is optimal, so that the agents are able to move around the barrier as a group. When agents can fail, however, the task becomes so difficult (because failed agents can trap others within the barrier) that the best solution is to move as individuals toward the goal whenever the opportunity presents itself. Note that for the `Obs1` conditions, there is a broad region of the parameter space over which the performance landscape is effectively flat. these types of findings suggest that in some cases the parameter space being searched and the amount of discretization may be reduced, resulting in faster optimization runs without a loss of performance.

B. Real Robots

Because local range and bearing hardware has not yet been completed, the Moorebots must rely on emulated sensory information from the overhead camera system to perform LD. The processing burden thus placed on the camera system limits the maximum speed of the robots, as the camera system must be able to track the robots from frame to frame by position only. This restriction, along with the fact that control is not truly distributed, renders extensive experimental effort unwarranted. However, to demonstrate that we have the capability to quantitatively characterize real robot flocking performance (and thus in principle can reproduce the simulated optimization experiments presented above in the real world), we chose a set of reasonable parameter values and looked at the influence of varying N on the flocking performance of a group of 10 real robots.

For each value of N, 10 trials were run ($T = 210$ [sec]) under both the `F` and `NF` conditions, and the resulting performance values are shown in Figure 8. LD at $N = 0$ represents a baseline traversal behavior (because there is no interaction between agents), and for this case the normalization by N was omitted. The data demonstrates that LD does enable this group of robots to flock, as flocking performance is greater at $N = 4$ than $N = 0$ (significant

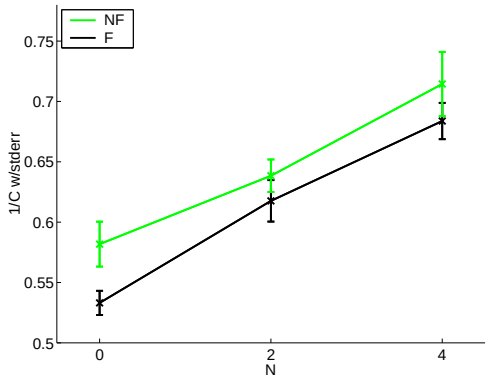


Fig. 8. Flocking performance of a group of 10 real robots vs. N, the maximum number of visible neighbors.

via ANOVA to $p < .01$), while agent failure does not significantly influence performance (via ANOVA to $p < .01$ – although it is likely that a difference would be significant at larger sample sizes). Because the specifics of these results are likely to be highly dependent on the particular parameter values chosen (most of which are arbitrary rather than optimized), detailed comparison with the simulation results is not meaningful.

V. Conclusion

In this paper we presented a simple flocking task, and we described a leaderless distributed flocking algorithm (LD) that is more conducive to implementation on embodied agents than the established algorithms used in computer animation. The key point of this algorithm is that it uses the time derivative of the perceived center of the flock to align the robots without explicit knowledge of robot heading. We also used an embodied simulator and reinforcement learning techniques to optimize LD performance in different environments, showing that this method can be used not only to improve performance but also to gain insight into which algorithm components contribute most to system behavior. An issue for further study is the automation of the selection, or perhaps improvement of, the parameter ranges and discretization levels that are searched. Also, using optimization data it may eventually be possible to construct models that directly relate environmental characteristics to parameter values. Finally, we demonstrated that a group of real robots executing LD with emulated sensors can successfully flock and that systematic characterization of real robot flocking parameters is achievable. Members of the lab are currently working on the hardware necessary to implement LD fully locally, and this will enable full verification of the optimization experiments to be performed on the real robots.

Acknowledgments

We would like to thank Alcherio Martinoli, Richard Murray, Owen Holland, and Rodney Goodman for their advice and comments on this work. This work is supported in part by the Center for Neuromorphic Systems Engineering as part of the NSF ERC Program under grant EEC-9402726. Support has also been received from DARPA under grant DAAK60-97-K-9503, the ONR under grant N00014-98-1-0821, and from the ARO under MURI grant DAAG55-98-1-0266. Adam Hayes is supported by a NSF Graduate Research Fellowship.

References

[1] C. W. Reynolds, "Flocks, herds, and schools: A distibuted behavioral model," *Computer Graphics*, vol. 21, no. 1, pp. 79–98, 1987.

[2] D. C. Brogan and J. K. Hodgins, "Group behaviors for systems with significant dynamics," *Autonomous Robots*, vol. 4, pp. 137–153, 1997.

[3] I. D. Kelly and D. A. Keating, "Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots," in *Proc. of the The Third Int. Conf. on Mechatronics and Machine Vision in Practice*, Guimaraes, Portugal, 1996, pp. 1/1–1/4.

[4] Mataric M., *Interaction and Intelligent Behavior*, Ph.D Thesis, MIT, Boston, May 1994.

[5] J. Toner and Y. Tu, "Flocks, herds, and schools: a quantitative theory of flocking," *Physical Review E*, vol. 58, no. 4, pp. 4828–4858, 1998.

[6] H. Yamaguchi and G. Beni, "Distributed autonomous formation control of mobile robot groups by swarm-based pattern generation," in *Proc. of the Second Int. Symp. on Distributed Autonomous Robotic Systems DARS-96*. 1996, pp. 141–155, Springer Verlag.

[7] T. Balch and R. C. Arkin, "Behavior-based formation control for multi-robot teams," *IEEE Robotics and Automation*, vol. 14, no. 6, pp. 926–939, December 1998.

[8] J. Fredslund and M. J. Mataric, "Robot formations using only local sensing and control," in *Proceedings, International Symposium on Computational Intelligence in Robotics and Automation (IEEE CIRA 2001)*, Banff, Alberta, Canada, July 2001.

[9] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York, US, 1999.

[10] A. Martinoli, A. J. Ijspeert, and F. Mondada, "Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots," *Robotic and Autonomous Systems*, vol. 29, pp. 51–63, 1999.

[11] O.E. Holland and C. Melhuish, "Stigmergy, self-organization, and sorting in collective robotics," *Artificial Life*, vol. 5, pp. 173–202, 1999.

[12] M. J. B. Krieger and J.-B. Billeter, "The call of duty: Self-organised task allocation in a population of up to twelve mobile robots," *Robotics and Autonomous Systems*, vol. 30, pp. 65–84, 2000.

[13] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Swarm robotic odor localization," in *Proc. of the IEEE Conf. on Intelligent Robots and Systems*, Wailea, HI, October 2001, pp. 1073–1078, IEEE Press.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning*, MIT Press, Cambridge, MA, 1998.

[15] D. Floreano and Mondada F., "Evolution of homing navigation in a real mobile robot," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 26, pp. 396–407, June 1996.

[16] S. Yakowitz, P. L'Ecuyer, and F. Vazquez-Abad, "Global sthocastic optimization with low-dispersion point sets," *Operations Research*, vol. 48, no. 6, pp. 939–950, November-December 2000.

[17] L. E. Parker, "Lifelong adaptation in heterogeneous multi-robot teams: Response to continual variation in individual robot performance," *Autonomous Robots*, vol. 8, no. 3, pp. 239–267, 2000.

[18] M. J. Mataric, "Reinforcement learning in the multi-robot domain," *Autonomous Robots*, vol. 4, no. 1, pp. 73–83, March 1997.

[19] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, 2000.

[20] T. Balch, *Behavioral Diversity in Learning Robot Teams*, Ph.D Thesis, College of Computing, Georgia Institute of Technology, December 1998.

[21] O. Michel, "Webots: Symbiosis between virtual and real mobile robots," in *Proceedings of the First International Conference on*

Virtual Worlds, VW'98, Paris, France, July 1998, pp. 254–263, Springer Verlag.

[22] F. Mondada, E. Franzi, and P. Ienne, "Mobile robot miniaturization: A tool for investigation in control algorithms," in *Proc. of the Third International Symposium on Experimental Robotics ISER-93*, T. Yoshikawa and F. Miyazaki, Eds., Kyoto, Japan, 1993, pp. 501–513, Springer Verlag.

[23] A. J. Ijspeert, A. Martinoli, A. Billard, and L. M. Gambardella, "Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment," *Autonomous Robots*, vol. 11, no. 2, pp. 149–171, 2001.

[24] C. Versino and L. M. Gambardella, "Learning real team solutions," in *DAI Meets Machine Learning, Lecture Notes in Artificial Intelligence*, G. Weiss, Ed., pp. 298–311. Springer Verlag, Berlin, 1997.

[25] J. R. Koehler, A. A. Puhalskii, and B. Simon, "Estimating functions evaluated by simulation: A bayesian/analytic approach," *The Annals of Applied Probability*, vol. 8, no. 4, pp. 1184–1215, 1998.

[26] P.H. Ramsey, "Multiple comparisions of independent means," in *Applied analysis of variance in behavioral science*, L. K. Edwards, Ed., vol. XI, pp. 25–62. Marcel Dekker, New York, 1993.

[27] A.F.T. Winfield and O.E. Holland, "The application of wireless local area network technology to the control of mobile robots," *Microprocessors and Microsystems*, vol. 23, pp. 597–607, 2000.