Robot Herds: Group Behaviors for Systems with Significant Dynamics [†]

Jessica K. Hodgins
David C. Brogan
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
[jkh|dbrogan]@cc.gatech.edu

Abstract

Birds, fish, and many other animals are able to move gracefully and efficiently as a herd, flock, or school. We would like to reproduce this behavior for herds of artificial creatures with significant dynamics. This paper develops an algorithm for grouping behaviors and evaluates the performance of the algorithm on two types of systems: a full dynamic simulation of a legged robot that must balance as well as move with the herd and a point mass with minimal dynamics. Robust control algorithms for group behaviors of dynamic systems will allow us to generate realistic motion for animation using high-level controls, to develop synthetic actors for use in virtual environments, mobile robotics, and perhaps to improve our understanding of the behavior of biological systems.

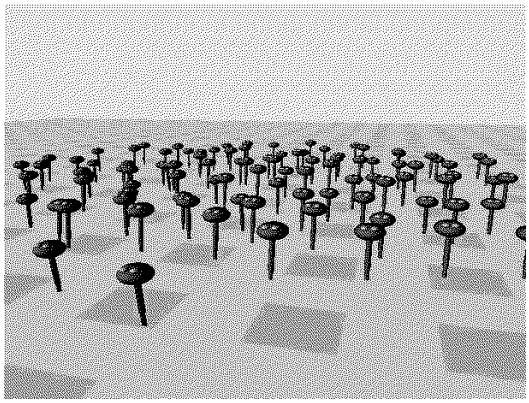

Figure 1: Photograph of a herd of 100 simulated one-legged robots. The herd had run stably for 5 minutes before this photograph was taken.

1 Introduction

To run as a herd, animals must remain in close proximity while changing direction and velocity and while avoiding collisions with other members of the herd and obstacles in the environment. In this paper, we explore the performance of a control algorithm for modulating the motion of each individual in a herd of dynamically simulated legged robots. A photograph of 100 simulated robots running as a herd is shown in figure 1.

The herding algorithm computes a desired velocity for each individual based on the location and velocity of its visible neighbors. This desired velocity is then used as an input to the locomotion control system for the robot. We compare the performance of this algorithm on a herd of point-mass objects and a herd of dynamically simulated running robots for a test suite of four problems: steady-state motion, acceleration and deceleration, turning, and avoiding obstacles. For this test suite, all individuals in the herd of robots remained upright and only a small number of collisions occurred. However, the performance of this herd was not as robust as that of the point-mass system.

In contrast to most previous implementations of algorithms for group behaviors, we are using this algorithm to control a robot herd where the individuals have significant dynamics. The problem of controlling the robot herd more closely resembles that faced by biological systems because of the underlying dynamics of the individuals in the herd. Each robot in the herd is a dynamic simulation of a physical robot and a control system. As such, the robots have limited acceleration, velocity, and turning radius. Furthermore, the control algorithms are inexact, resulting in both transient and steady-state errors in velocity control. Required changes in velocity are delayed by as much as half a running step because the control system can influence

[†] This paper will appear in the Proceedings of Artificial Life IV.

velocity during only the stance phase of the running cycle. To understand the effect of the underlying dynamics, we compared the performance of the herding algorithms on the robots with full dynamics and on particle systems with perfect velocity control.

Algorithms for high-level behaviors of dynamic simulations are needed for the construction of synthetic actors with robust and realistic motion that can respond interactively to changes in the environment. A dynamic simulation in concert with a control system will provide natural looking motion for low-level behaviors such as walking, running, and climbing. High-level behaviors such as obstacle avoidance, grouping, and rough terrain locomotion will allow the actor to function in and interact with a complex and unpredictable environment.

2 Background

Recent advances in robotics have produced autonomous agents capable of performing a variety of tasks in different domains. At the same time, researchers in the artificial life community have contributed to our understanding of the evolution of complex behaviors through the use of simulations that produce emergent behaviors. By building on work from these fields, we should be able to create multi-agent robotic systems that mimic the elegant grouping behaviors of biological organisms.

Herding, flocking, and schooling behaviors of animals have been studied extensively over the past century and this research serves as motivation for the creation of artificial creatures with similar skills. Groupings exemplify an attraction that modulates the desire of each member to join the group with the desire to maintain a particular separation distance from nearby creatures (Shaw 1970). As an example of this attraction, Cullen, Shaw, and Baldwin (1965) report that the density of fish is approximately equal in all planes of the school, as if each fish had a sphere around its head with which it wished to contact the sphere of another fish. Herding benefits the average group member by limiting the average number of encounters with predators (data summarized in Veherencamp 1987). Grouping behaviors allow animals to hunt more powerful animals than those they could overpower as individuals. Due to the success of behaviors such as these in biological systems, it seems reasonable to assume that it would be advantageous to reproduce them in robotic systems.

Early work in the simulation of grouping behaviors was performed by Reynolds (1987). Actors in his system are bird-like objects and are similar to the point masses used in particle systems except that each bird also has an orientation. The birds maintain proper position and orientation in the flock by balancing their desires to avoid collisions with neighbors, to match the velocity of nearby neighbors, and to move towards the center of the flock. Each bird uses only information about nearby neighbors. This localization of information simulates perception and reaction in biological systems and allows for proper balancing of the three flocking tendencies. Reynolds's work demonstrates that realistic-looking animations of group formations can be created by applying simple rules to determine the behaviors of the individuals in the flock.

Yeung and Bekey (1987) propose a decentralized approach to the navigation problem for multiple agents. Their system first constructs a global plan without taking into account moving obstacles. When a collision is imminent, the system locally re-plans using inter-robot communication to resolve the conflict. Because of the two levels of planning, this solution requires the communication overhead associated with grouping behaviors only when a pair of robots perceive an impending collision.

Sugihara and Suzuki (1990) demonstrated that multiple robots can form stable formations when each robot executes an identical algorithm for position determination within the group. In their simulation, each robot can perceive the relative positions of all other robots and has the ability to move one grid position during each unit of time. By adjusting the position of each robot relative to either the most distant or the closest neighbor, a regular geometric shape such as a circle can be formed by the robots in the world. Furthermore, the movement of one robot in a formation can cause a chain reaction that results in a translation of the group in world coordinates. By carefully constructing the algorithms that each robot uses in determining intra-group position, formations will emerge without *a priori* knowledge about the total number of robots. Designation of leaders allows the simple rules of the group to create leader-following algorithms and to demonstrate the division of a formation into smaller groups.

Wang (1991) investigated the navigation of multiple robots in formation and the resulting group dynamics. Each robot in the model is simulated as a point mass and perceives other robots in the region contained in a cone extending from the center of the robot and heading in the direction of travel. Formations are represented as a set of offsets from a predefined reference robot. In this way, a formation can be directly defined as a set of positions for each robot relative to the leader, closest neighbor, or set of closest neighbors.

Wang shows that the error in desired position relative to actual position diminishes to zero for each independent robot in the formation and therefore the desired formation is asymptotically stable. Simulations for up to four point-mass robots demonstrate that these navigation strategies can produce stable group formations.

Arkin explored the question of communication in a group of interacting mobile robots using schema-based reactive control (Arkin 1992, Arkin and Hobbs 1992). Example schemas are *move-to-goal*, *move-ahead*, and *avoid-static-obstacle*. Each behavior computes a velocity vector that is combined with the velocity vectors from the other behaviors and is used to control the robot. Arkin demonstrated that for some tasks robots can interact with no communication other than observations of the environment or with very limited explicit communication. The herding algorithms we implemented are also examples of an algorithm in which there is no explicit leader and all communication is through observations of the environment.

Mataric researched emergent behavior and group dynamics in the domain of wheeled vehicles. These robots, like Arkin's, do not explicitly communicate state or goals and the system has no leaders. This work demonstrated that combinations of such simple behaviors as attraction and repulsion can produce complex relationships such as dispersion and flocking in physical robots in the laboratory (Mataric 1992a,b). The robots utilize the knowledge that they are all identical when executing behaviors, but an extension to these results found that heterogeneous agents do not perform significantly better than homogeneous ones (1993). In these experiments, a hierarchy is created in which an ordering between the agents determines which agent will move first in completing tasks such as grouping and dispersing.

3 Algorithms for Herding

The herding algorithms described in this paper were evaluated on two systems: a one-legged robot simulation with full dynamics and a particle simulation with minimal dynamics. The next two sections describe the herding algorithms and the two simulations.

The herding algorithm consists of three parts: a perceptual model to determine the visible creatures for each individual in the herd, a placement algorithm to determine a desired position for each individual given the locations of the creatures that are visible to it, and a spring/damper control system to compute a desired velocity given the current position and the desired position. The herding algorithm is run for each individual in the herd to compute a desired velocity for that individual. The control system for each legged robot then uses the desired velocity provided by the herding algorithm to determine how the leg should be positioned during flight to achieve the desired change in forward velocity. The particles in the point-mass herd use this desired velocity as their actual velocity on the next time step.

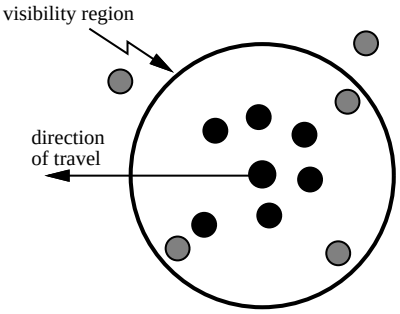

Figure 2: One creature is visible to another if it is within a certain radius and is one of the n closest visible creatures (n is six for this example). The black circles represent visible creatures and the grey represent creatures that can not be seen by the individual under consideration.

The herding algorithms for an individual robot are run each simulation time step while the robot is in flight. For the particle system, the herding algorithms compute a new desired velocity for each simulation step and a new set of visible particles every ten simulation steps.

3.1 Perception Model

Each individual in the herd can perceive the location and velocity of the n nearest creatures that are within a circle of radius r. In the trials reported in this paper n was 30 and r was 24 m and the herd included 105 individuals. For most configurations the circle was large enough to include all members of the herd. Figure 2 illustrates the perception model.

3.2 Desired Position and Velocity

The list of visible creatures provided by the perceptual model is used to compute a desired position for each individual in the herd. A desired position relative to each visible creature is computed and then these desired positions are combined with a weighted average. The desired position of an individual relative to each of the visible creatures is a constant distance D away from the visible creature on the line between the two creatures (figure 3). In these experiments D was 2.5m. This set of desired positions (one for each visible creature) is averaged with a weighting of $1/d^2$ to compute a

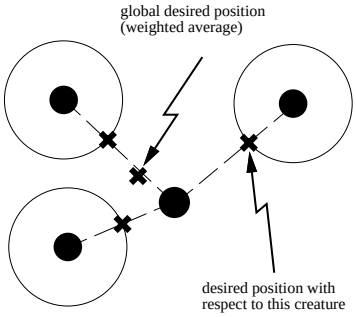

Figure 3: The locations of the visible creatures are used to compute a global desired position for the individual under consideration. The algorithm computes a desired position with respect to each visible robot by finding the point on the line between the individual and the visible creature that is a constant distance D away from the visible creature. These desired positions are averaged with a weighting equal to $1/d^2$ where d is the distance between the two creatures.

global desired position where d is the distance between the two creatures.

The global desired position for an individual is used to compute a desired velocity for that creature using a spring and damper system:

$$\dot{x}_{\mathrm{d}} = k_{\mathrm{p}} e - k_{\mathrm{v}} \dot{e} + \dot{x}_{\mathrm{nom}} \qquad (1)$$

where $\dot{x}_{\mathrm{d}}$ is the desired velocity in the plane, e is the error between the current position of the creature and the global desired position, $\dot{e}$ is the rate of change of the error, k_{p} and k_{v} are the proportional and derivative gains, and $\dot{x}_{\mathrm{nom}}$ is the nominal velocity. For the experiments reported here $k_{\mathrm{p}} = 0.5$ and $k_{\mathrm{v}} = 0.3$. The nominal velocity $\dot{x}_{\mathrm{nom}}$ was the average of the desired velocities of the visible creatures. To provide the user with control of the herd, one creature is selected by the user. The nominal velocity in equation 1 for that creature is set by the user rather than computed by averaging the desired velocity of the visible creatures.

4 Simulating the Herd

The herd simulation consists of the equations of motion for either the robot or the particle system, a copy of the state vector for each individual in the herd, control algorithms for running, a graphical image for viewing the motion of the herd, and an interface that allows the user to control the parameters of the simulation. For the robot herd, the equations of motion represent a rigid body model of a one-legged robot and control algorithms that allow the robot to run at a variety of speeds and flight durations. At each simulation time step, the control system computes forces or torques for each joint of the robot based on the actual and desired state vector for that individual. The equations of motion of the system are integrated forward in time, and the resulting motion of the individuals in the herd is displayed graphically and recorded for later use. The equations of motion for the individuals in the herd do not take into account the physical effects of collisions between two members of the herd, although collisions are detected and a count of collisions is recorded for use in analyzing the data. The details of the robot and particle models are described below.

Link	Mass (kg)	Moment of Inertia (x, y, z kgm^2)		
Body	23.1	0.9	0.9	0.602
Upper Leg	1.4	0.018463	0.017297	0.001441
Lower Leg	0.64	0.0197	0.0197	0.000176

Table 1: Parameters of the rigid body model of a one-legged robot. The moment of inertia is computed about the center of mass of each link.

Link	COM to Proximal (m)	COM to Distal (m)
Body		0.0
Upper Leg	0.095	-0.095
Lower Leg	0.221	

Table 2: The distance from the center of mass of each link to the distal and proximal joints in z for the canonical configuration of the robot (the distance in x and y is zero for this model).

4.1 One-legged Robot Simulation

The equations of motion for the robot were generated using a commercially available package (Rosenthal and Sherman 1986). The package generates subroutines for the equations of motion using a variant of Kane's method and a symbolic simplification phase. The parameters of the robot are given in table 1 and table 2. The reference angles of the model are shown in figure 4.

The locomotion algorithms for the one-legged robot control flight duration, body attitude, and forward and sideways velocity. Flight duration is controlled by extending the leg during stance. Body attitude (pitch,

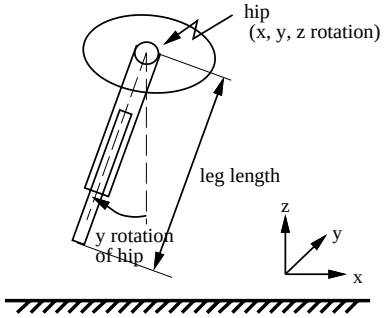

Figure 4: The reference angles for the controlled degrees of freedom of the robot. The controlled degrees of freedom are three degrees of freedom at the hip and the length of the leg.

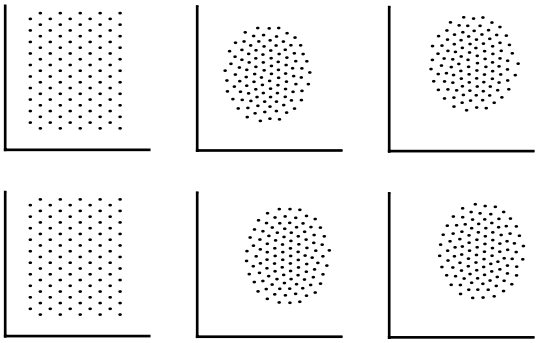

Figure 5: The configuration of the herd at the start state and every 100 s thereafter for a commanded steady-state velocity of 2.0 m/s in the x direction for the user-controlled creature. The top set of graphs shows the motion of the robot herd; the bottom set shows the motion of the particle system. Each graph represents the x, y position of each robot or particle in the herd.

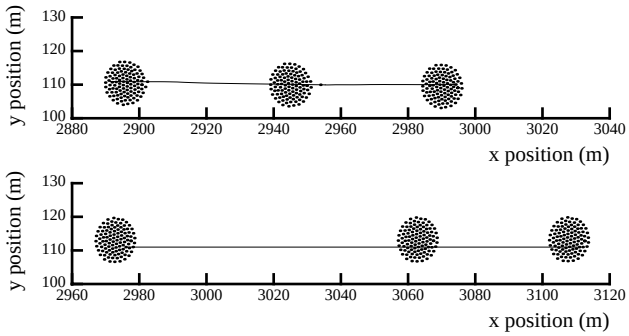

Figure 6: The configuration of the herd at the start state and every 30 s thereafter for an initial steady-state desired velocity of 2 m/s in the x direction, followed by a desired velocity of 3 m/s for 30 s and 1.5 m/s for 30 s for the user-controlled creature. The top graph represents the herd of robots; the bottom graph represents the herd of particles. The path shown between the snapshots of the herd is the trajectory that the user-controlled robot or point mass followed.

roll, and yaw) is controlled by exerting a torque between the body and the leg during stance. The velocity is controlled by the position of the foot with respect to the center of mass of the body at touchdown. For a constant velocity, the foot is positioned in the center of the distance that the body is expected to travel while the foot is on the ground. To increase the speed, the foot is positioned closer to the hip. To decrease the speed, the foot is positioned further from the hip. The details of the locomotion control algorithms are given in Raibert (1986).

4.2 Particle Simulation

The particle simulation has minimal dynamics. The desired velocity computed by the herding algorithm is used by the dynamic simulation as the actual velocity

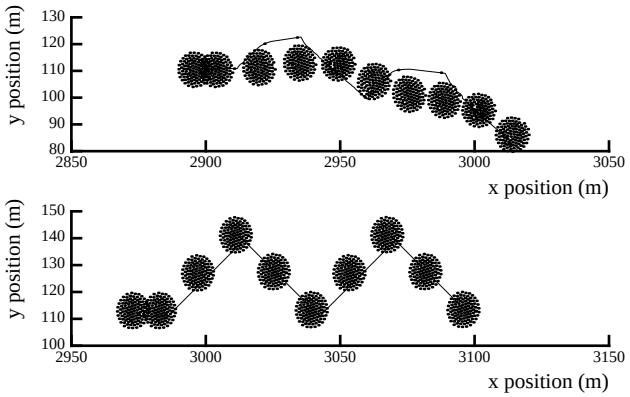

Figure 7: The configuration of the herd as the user-controlled creature follows a zig-zag pattern. Beginning with a steady-state run of 2.0 m/s in the x direction for 5 s, the y desired velocity was increased to 1.41 m/s and the x desired velocity was decreased to 1.41 m/s. After 20 s the y desired velocity was set to -1.41m/s for the next 20 s. This pattern was repeated a second time. The top graph represents the herd of robots, the bottom graph, the herd of particles. The path shown between the snapshots of either herd is the trajectory that the user-controlled robot or point mass followed.

on the next time step. There are no limits on acceleration or velocity. The particle system differs from the robots in that there is no delay in the implementation of a new desired velocity and the new velocity exactly matches the desired velocity.

5 Results

We tested the herding algorithms in four situations: steady-state movement, acceleration, turning, and avoiding obstacles. For steady-state movement, both the herd of robots and particles began in the same configuration, and the user-controlled creature was commanded to move at 2.0 m/s for 300 s. As is shown in the snapshots of the herd configurations in figure 5 both herds contracted to form a nearly circular shape.

The second test began with the ending point of the steady-state test for each system. The commands to the user-controlled creature were an acceleration to 3m/s in the x direction for 30 s and then a deceleration to 1.5 m/s for 30 s (figure 6). Both the particle system and the robots continued to move as a herd although the user-controlled robot moved ahead of the herd during the acceleration phase of the experiment and dropped back into the herd during the deceleration.

The third test involved turning. Beginning with a steady-state run of 2.0 m/s in the x direction, the y desired velocity was increased to 1.41 m/s and the x desired velocity was decreased to 1.41 m/s. After

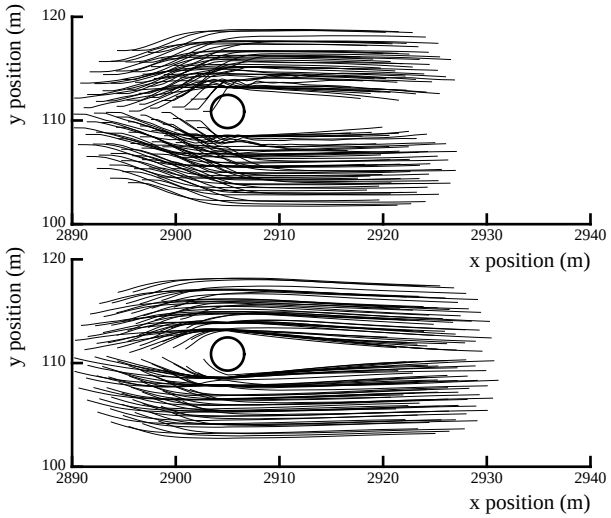

Figure 8: The trajectory of the members of the robot herd and the particle system herd as the creatures avoid an obstacle. The top graph represents the herd of robots, the bottom graph, the herd of particles. Both herds ran for 15 s at a nominal speed of 2 m/s in this experiment and the herds were moving from left to right.

20 s the y desired velocity was set to -1.41m/s for the next 20 s. This zigzag pattern was repeated a second time (figure 7). The particle system had no collisions but the robot herd had multiple collisions as the herd changed direction. The particle system herd tracked the user-controlled point-mass much more closely than the robot herd tracked the user-controlled robot although both herds retained an approximately circular shape. The user-controlled robot does not follow the desired zigzag pattern closely because its velocity is affected by its position relative to the other members of the herd.

The final test involved obstacle avoidance (figure 8). The creatures on a collision path with the obstacle moved to avoid the obstacle by aiming for a point out to the side of the obstacle at a distance of 1.5 times the radius of the obstacle. This sideways motion was incorporated into the calculation for the desired position with a weighted average. The herd of point masses were able to avoid the obstacle and quickly rejoined to form a single herd on the far side of the obstacle. The first robot in the robot herd was unable to avoid the obstacle and this herd was slower to regroup on the far side of the obstacle. In easier tests where the herds had more time to react the performance of the point mass herd and the robot herd were similar.

The herding algorithms used for these two sets of tests were identical and differences in performance can be attributed to differences between the two dynamic systems. The point-mass herd ran more tightly under changes in magnitude and direction of velocity because of the exact control of velocity. The behavior of the robot herd was not as robust as that of the point-mass system because the herd did not track the user-controlled robot as closely. The robot herd had more variability and motion within the herd and tests more often resulted in collisions between members of the herd. In more difficult tests than those reported here, an individual in the herd sometimes lost its balance and fell down. The particle systems had no notion of balance or of maximum speed or acceleration and could not fail in this way.

In other ways, the performance of the robot herd was superior to that of the particle system. The natural damping of the individual behavior of the robots appears to increase the stability of the herd in some situations and the robot herd was more stable than the point mass herd for tests where the number of visible creatures was reduced below 30.

A serious limitation of this herding algorithm is the knowledge required by a robot about the desired velocity of a neighbor because this information could not be measured with a sensor. An implementation that used actual rather than desired velocity would be preferable but was not stable for the robot herd. The dynamic interaction of the leg with the ground and inaccuracies in the locomotion control system prevent the robots from running at exactly the commanded velocity. A linear fit between the actual and desired velocities was not a sufficiently accurate model to correct this problem, and the herd ran increasingly faster or slower depending on the constant chosen for the linear fit. In the particle system, actual and desired velocities were identical and there is no difference between these two approaches.

There are other limitations to the herding algorithm we implemented. In some situations, the desired velocity moved two individuals closer to a collision. In our current implementation there is no reflexive reaction to an impending collision.

With this algorithm, a breakaway group of sufficient size will not rejoin the main herd unless a member of the main herd is visible to a member of the breakaway group. This problem could be solved by the addition of a separate behavior that causes individuals to look further afield for another herd to join.

We experimented with other perceptual models, adding occlusion and reducing the visibility of creatures behind as opposed to in front of the individual in question. Occlusion reduced the stability of the particle system without qualitatively changing the behavior of the robot herd. When the list of visible creatures changes because of the addition of a previously occluded individual, the desired position and velocity may change significantly thereby causing an immedi-

ate ripple effect in the particle system simulation. The natural inertia of the robot simulation appears to mitigate this effect.

Although reducing the visibility of creatures that are behind the one under consideration might appear to be a more natural perceptual model for an animal or a human, it was not a good heuristic for this simple herding algorithm. Unequal front and back visibility caused the creatures in front to contribute more heavily to the desired position than the creatures in back. The desired position would then be in front of the current position, and the velocity of the robot or particle would continually increase.

We have not yet explored the question of how the algorithms will perform on a heterogeneous population. Currently the robots have identical mass and inertia properties and identical control systems, but we plan to vary the parameters of the system and to introduce noise to study how the performance of the herding algorithms is affected. A further extension would be to develop "personalities" for the individuals as Bates did in his woggles simulation (Bates et al 1993). In the case of the dynamic robot simulation, a simple personality might consist of adjustments to the gains in the herding algorithm and the locomotion control system so that the robot appears to behave in an aggressive or timid fashion.

Although the simulation of the robots is a full dynamic simulation, many factors are missing in the simulation that would be present in a physical robot. The simulated motors do not have a maximum torque or limited bandwidth, the joint and perceptual sensors do not have noise or delay, and the environment used for testing the herding algorithms does not contain uneven or slippery terrain.

One application of this work is to provide high-level controls of simulated creatures for use in computer animations or virtual environments. To be useful in interactive virtual environments, the motion of simulated actors must be computed in real time (simulation time must be less than wall clock time). Our implementation of a single one-legged robot runs faster than real time on a Silicon Graphics Indigo^2 Computer with a R4400 processor. We anticipate that with improved simulation techniques and the continued increase in workstation speed, a small herd of robots or more human-like models will run in real time within a few years.

Acknowledgments

This project was supported in part by NSF Grant No. IRI-9309189 and funding from the Advanced Research Projects Agency.

References

Arkin, R. C., 1992. Cooperation Without Communication: Multiagent Schema Based Robot Navigation, *Journal of Robotic Systems* 351–364.

Arkin, R.C., and Hobbs, J. D., 1992. Dimensions of Communication and Social Organization in Multiagent Robotic Systems. *Proceedings of the Second International Conference on Simulation of Adaptive Behavior: From Animals to Animats 2* 486–493.

Bates, J., Altucher, J., Hauptman, A., Kantrowitz, M., Loyall, A. B., Murakami, K., Olbrich, P. Popovic, Z., Reilly, W. S., Sengers, P., Welch, W., Weyhrauch, P., Witkin, A., 1993. Edge of Intention. *Siggraph 1993, Visual Proceedings*, 113–114.

Cullen, J.M., Shaw, E., Baldwin, H.A. 1965. Methods for Measuring the Three-Dimensional Structure of Fish Schools. *Animal Behavior* 13:534–543.

Mataric, M., 1992a. Minimizing Complexity in Controlling a Mobile Robot Population. *Proceedings of the 1992 IEEE International Conference on Robotics and Automation* 830–835.

Mataric, M., 1992b. Designing Emergent Behaviors: From Local Interactions to Collective Intelligence. *Proceedings of the Second International Conference on Simulation of Adaptive Behavior: From Animals to Animats 2* 432–441.

Mataric, M., 1993. Kin Recognition, Similarity, and Group Behavior. *Proceedings of the Fifteenth Annual Cognitive Science Society Conference.*

Raibert, M. H. 1986. *Legged Robots That Balance.* Cambridge: MIT Press.

Reynolds, C. W. 1987. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics* 21(4): 25–34.

Rosenthal, D. E., Sherman, M. A., 1986. High Performance Multibody Simulations Via Symbolic Equation Manipulation and Kane's Method. *Journal of Astronautical Sciences* 34(3):223–239.

Shaw, E., 1970. Schooling in Fishes: Critique and Review. Development and Evolution of Behavior. Aronson, L., Tobach, E., Leherman, D., and Rosenblatt, J.(eds) 452–480.

Sugihara, K. and Suzuki, I., 1990. Distributed Motion Coordination of Multiple Mobile Robots. *Proceedings of the 1990 IEEE International Conference on Robotics and Automation* 138–143.

Wang, P. K. C., 1991. Navigation Strategies for Multiple Autonomous Robots Moving in Formation. *Journal of Robotic Systems* 8(2):177–195.

Yeung, D. Y. and Bekey, G. A., 1987. A Decentralized Approach to the Motion Planning Problem for Multiple Mobile Robots. *Proceedings of the 1987 IEEE International Conference on Robotics and Automation* 1779–1784.

Veherencamp, S., 1987. Handbook of Behavioral Neurobiology, Volume 3: Social Behavior and Communication, Marler, P. and Vandenbergh, J. G. (eds.) 354–382.