Group Behaviors for Systems with Significant Dynamics [†]

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

Abstract

Birds, fish, and many other animals travel as a flock, school, or herd. Animals in these groups must remain in close proximity while avoiding collisions with neighbors and with obstacles. We would like to reproduce this behavior for groups of artificial creatures with significant dynamics. In this paper we describe an algorithm for creatures that move as a group and evaluate the performance of the algorithm with three simulated systems: legged robots, human-like bicycle riders, and point-mass systems. Both the legged robots and the bicyclists are dynamic simulations that must control balance, facing direction, and forward speed as well as movement with the group. The point-mass systems have minimal dynamics and are included to facilitate our understanding of the effects of the dynamics on the performance of the algorithms.

Introduction

To run as a group, animals must remain in close proximity while changing direction and velocity and avoiding collisions with other group members and obstacles in the environment. We would like to create multi-agent systems that replicate the complexity and variability of natural groups by using simple communication, cooperation, and coordination strategies. In this paper, we explore the performance of an algorithm for group behaviors. The groups are made up of dynamically simulated legged robots, bicycle riders, and point-masses with minimal dynamics. An image of six simulated bicyclists riding as a group is shown in figure 1.

The algorithm for group behaviors computes a desired position for each individual based on the location and velocity of visible neighbors, visible obstacles, and a desired group velocity. The desired position is known only to each individual creature and the navigational intent of each creature is communicated to the others only by their observation of its actions. We compare the performance of this algorithm on the three systems for a test suite of three problems: steady-state motion, turning, and avoiding obstacles. The point-mass system demonstrates the most robust performance in all tests because the dynamics of the system do not interfere significantly with the control exerted by the group behaviors. When the inherent delay in the control of velocity in the one-legged robots is taken into account, their performance is also good. The bicyclists are the least robust of the three systems because the underlying control system for steering and balance is unable to execute some of the changes in velocity and direction requested by the higher-level algorithms for group behaviors.

Figure 1: Image of a group of six simulated bicycle riders.

In contrast to most previous implementations of algorithms for group behaviors, we use this algorithm to control a group in which the members have significant dynamics. The problem of controlling these individuals more closely resembles that faced by quickly moving mobile robots and by biological systems because each individual is dynamically simulated and has limited acceleration, velocity, and turning radius. Furthermore, the control algorithms are inexact, resulting in both transient and steady-state errors in the control of velocity. In the case of the legged robots, required changes in velocity are delayed by almost a full running cycle because the control system can influence velocity only during the stance phase of the running cycle.

Algorithms for high-level behaviors are needed for the construction of cooperating robots with robust

[†]This paper appears in IEEE/RSJ International Conference on Intelligent Robot and Systems, 1995.

and agile movements. When a robot moves quickly enough to have significant dynamics, the algorithms that control the high-level behaviors must include models of the underlying dynamics and the limitations of the low-level control. Algorithms for high-level behaviors of dynamic simulations are also needed for the construction of virtual 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 such low-level behaviors as walking, running, bicycling, and climbing. Such higher-level behaviors as obstacle avoidance, grouping, and rough terrain locomotion would allow the actor to interact with the user and with a complex and unpredictable environment.

Background

Herding, flocking, and schooling behaviors of animals have been studied extensively over the past century, and this research has stimulated the creation of robots and simulated 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 sufficient distance from nearby creatures[12]. As an example of this attraction, Cullen, Shaw, and Baldwin[4] 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 group members by limiting the average number of encounters with predators (data summarized in Veherencamp[17]). Group behaviors also allow animals to hunt more powerful animals than those they could overpower as individuals. The success of behaviors such as these in biological systems makes it reasonable to assume that it would be advantageous to reproduce them in robotic systems.

Early work in the simulation of group behaviors was performed by Reynolds[10]. Actors in his system are bird-like objects similar to point-masses except that each bird also has an orientation. The birds maintain position and orientation within the flock by balancing three desires: avoiding collisions with neighbors, matching the velocity of nearby neighbors, and moving towards the center of the flock. Reynolds's work demonstrated 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[16] proposed 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 replans using interrobot communication to resolve the conflict. This solution reduces the communication overhead associated with group behaviors. Sugihara and Suzuki[13] demonstrated that robots can form stable formations without *a priori* knowledge about the total number of robots when each robot executes an identical algorithm for determining position within the group.

Wang[15] investigated the asymptotic stability of multiple robots in formation. Each robot in the model is simulated as a point mass and perceives other robots in a cone-shaped region 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, the closest neighbor, or set of closest neighbors.

Takeuchi, Unuma, and Amakawa[14] implemented path planning in simulated multiagent systems where the attraction between agents is dependent on properties of the agents. The simulated agents are point-masses where forces are applied to the agents based on the vector sum of attractions to observable agents. This method was used to formulate a path for a butterfly among flowers, to describe the paths of schooling fish when approached by a predator, and to generate the paths of humans avoiding a car.

Arkin explored the question of communication in a group of interacting mobile robots using schema-based reactive control ([1] and [2]). Example schemas are *move-to-goal*, *move-ahead*, and *avoid-static-obstacle*. Each behavior computes a velocity vector that is combined with the vectors from other behaviors. Arkin demonstrated that for some tasks robots can interact with no communication other than observations of the environment or with very limited explicit communication.

Mataric investigated emergent behavior and group dynamics with 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([7] and [8]).

Group Behaviors

The algorithms for group behaviors described in this paper were evaluated on three simulated systems: a one-legged robot, a rigid body model of a human riding a bicycle, and a point-mass system with minimal dynamics. For the point-mass system and the legged robots each group included 105 individuals; the group of bicyclists included 6 individuals.

The algorithm for group behaviors consists of two parts: a perception model to determine the creatures visible to each individual in the group and a placement algorithm to determine a desired position for each individual given the locations and velocities of the creatures that are visible to it. The output of the algorithm for group behaviors is a desired position for each individual in the group. The low-level control algorithms for each creature use the desired position to compute a desired velocity and then attempt to achieve that velocity.

Each individual in a group can perceive the relative locations and velocities of the n nearest creatures within a circle of radius r (figure 2). In most of the trials reported in this paper, n was 30 and r was 24 m; for most configurations, the circle was large enough to include all members of the group.

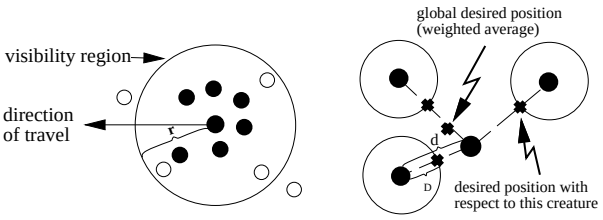

Figure 2: One creature is visible to another if it is within a certain radius, r, and is one of the n closest visible creatures. In the left figure, the black circles represent visible creatures and the white those that cannot be seen by the individual under consideration. The figure on the right illustrates how the locations of the visible creatures are used to compute a global desired position for the individual under consideration.

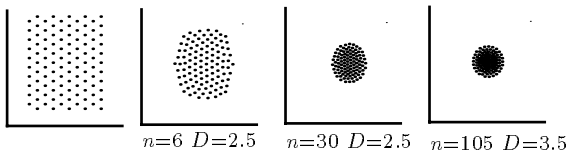

Figure 3: The first graph shows the initial configuration for three experiments. The other three show the configuration of the group of robots after 80 s of simulation with each graph representing a different choice for the number of perceptible robots (n) and the desired separation distance (D). When the robots were able to perceive a greater number of robots (n), the actual separation distance to the closest neighbors (d) was reduced even though the desired separation distance (D) to all visible robots increased.

The list of visible creatures provided by the perception model is used to compute a desired position for each individual in the group by computing a desired position relative to each visible creature and combining these desired positions with a weighted average. The desired position of an individual relative to each of the visible creatures is a constant desired separation distance D away from the visible creature on the line between the two (figure 2). In these trials 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 global desired position where d is the distance between the two creatures. Figure 3 illustrates the effect of the number of visible creatures and the separation distance.

In addition to avoiding collisions with other individuals in the group, the creatures avoid collisions with obstacles. If a creature is on a path that will cause it to collide with an obstacle, its desired position includes a weighted term for a desired position to the left or right of the obstacle.

Although the algorithms for group behaviors are identical for the three systems up to this point, the three systems use the information about desired position in different ways. The one-legged robots and the bicyclists adjust speed and facing direction in an attempt to eliminate the error in position. In contrast, the error in position for the point-masses is reduced by applying a force to the point-mass. These differences are described in detail in the following sections.

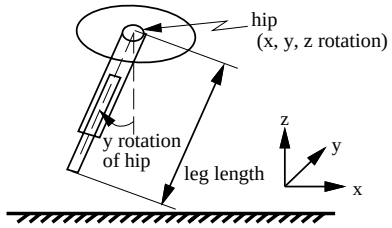

Figure 4: The reference angles for the degrees of freedom of the one-legged robot. The controlled degrees of freedom are the three degrees of freedom at the hip and the length of the leg.

Simulating Groups

A simulation of a group of creatures consists of the equations of motion and a state vector for each, control algorithms for running or bicycling, a graphical image for viewing the motion of the group, and an interface that allows the user to control the parameters of the simulation. For the group of robots, each simulation includes the equations of motion for 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. For the group of bicyclists, each simulation includes the equations of motion for a rigid body model of the bicycle and human rider and the control algorithms for steering and propelling the bike forward.

The equations of motion for the robot and the bicyclist were formulated using a commercially available package[11]. The equations of motion for the individuals in the group do not take into account the physical effects of collisions between two members of the group, although collisions are detected and a count of collisions is recorded for use in analyzing the data. The details of the robot, the bicycle rider, and the point-mass models are described below.

One-legged Robot

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 to make up for losses in the system. Body attitude (pitch, 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[9]. The reference angles of the model are shown in figure 4.

The desired position computed by the algorithms for group behaviors is used to compute a desired velocity. The desired position is instantaneous in that the individual would be in the right position if it were at that position at this moment in time. Running creatures, however, cannot change velocity during

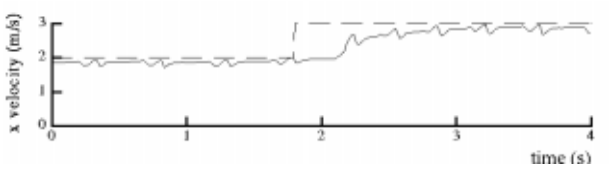

Figure 5: The response of the simulated one-legged robot to a step change in desired velocity. The solid line shows the actual velocity, and the dotted line shows the desired velocity. When the desired velocity changes at the end of the flight phase, the actual velocity does not change until the next stance phase.

flight and the control uses a model of this delay to improve the performance of the algorithms. A change in velocity is effected by repositioning the leg during flight in preparation for the next touchdown. During the subsequent stance phase, the velocity changes to the new desired velocity, which remains constant during the following flight phase. The response to a step change in desired velocity is illustrated in figure 5. To compensate for the delay in the control of the velocity, the error at the end of the next locomotion step is predicted and used to compute the desired velocity. The predicted position at the end of the next step is

$$x_p = x + \dot{x}(t_s + t_f) \tag{1}$$

where x is the current position, $\dot{x}$ is the velocity, t_s is duration of the stance phase and t_f is the duration of the flight phase. The desired position at the end of the next step is predicted in a similar fashion:

$$x_{dp} = x_d + \dot{x}_a(t_s + t_f) \tag{2}$$

where x_d is the desired x position computed by the algorithm for group behaviors and $\dot{x}_a$ is the average velocity of the members of the group that are visible to this individual. The average velocity of the visible group members is used to approximate the desired position of the individual on the next step because the future positions of the neighbors will influence the new desired position. The predicted error in position at the end of the next step is

$$e = x_{dp} - x_p. \tag{3}$$

We model a change in velocity as a linear ramp from the current velocity to the new velocity during stance and a constant velocity during the subsequent flight phase. The control system attempts to eliminate the error in position prior to the end of the next flight phase by computing an appropriate change in velocity. The error in desired position must cause a change in velocity that will make the position of the robot at the end of the next flight phase match the desired position:

$$\dot{x}(t_s + t_f) + e = \frac{\dot{x} + \dot{x}_d}{2} t_s + \dot{x}_d t_f. \tag{4}$$

Solving for $\dot{x}_d$:

$$\dot{x}_d = \dot{x} + \frac{e}{\frac{t_s}{2} + t_f}. \tag{5}$$

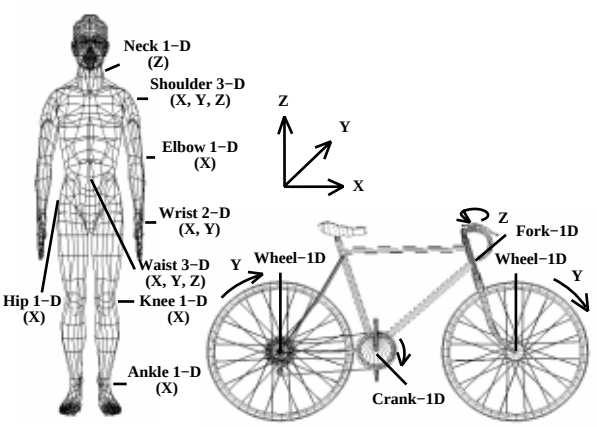

Figure 6: The controlled degrees of freedom of the human and bicycle models. The human model has fourteen joints, and the diagram shows the number of degrees of freedom at each joint. The four degrees of freedom of the bicycle model are shown in the figure. The human rider is attached to the bicycle by a pivot joint between the seat and the pelvis. The polygonal models were purchased from Viewpoint Datalabs.

We calculate the new desired velocity to be the sum of the current velocity and the average of the error in global desired velocity and the velocity due to the position error:

$$\dot{x}_{\mathrm{d}} = \dot{x} + \frac{1}{2}\left(\dot{x}_{\mathrm{gl}} - \dot{x} + \frac{e}{\frac{t_s}{2} + t_f}\right) \tag{6}$$

where $\dot{x}_{\mathrm{gl}}$ is the global desired velocity for the group. A similar model would be required for any creature with a ballistic flight phase during which speed and facing direction cannot not be altered.

Bicyclist Simulation

The human bicycle rider is modeled by a 15-segment rigid-body model connected by rotary joints with 22 controlled degrees of freedom. Some joints, like the knee, are modeled as a single-axis pin joint; others, like the wrist and shoulder, are modeled by two- and three-axis gimbal joints. The volume, mass, center of mass, moments of inertia, and distance between the joints are calculated from a polygonal representation of the human body (figure 6). The algorithm used to calculate the properties of the polygonal model integrates over the set of tetrahedra formed by the triangular faces of the model and the origin[6]. Density data obtained from the anatomical literature were used in calculating the dynamic properties of the body segments. The degrees of freedom of the bicycle model are shown in figure 6.

The simulated human rider controls the facing direction and speed of the bicycle by applying forces to the handlebars and the pedals. Spring and damper systems connect the hands to the handlebars, the feet to the pedals, and the crank to the rear wheel. The connecting springs are two-sided, and the bicyclist is able to pull up on the pedals as if the bicycle were equipped with toe-clips and a fixed gear.

The control system adjusts the velocity of the bicycle by moving the bicyclist's legs to produce a torque at the crank. The desired torque at the crank is

$$\tau_{\text{crank}} = k(v - v_d) \tag{7}$$

where k is a gain, v is the magnitude of the velocity, and v_d is the desired velocity. The force that can be applied by each leg is dependent on the angle of the crank because we assume that the legs are most effective when pushing downwards. When the crank is horizontal, the front leg can generate a positive torque and the rear leg can generate a negative torque. To compensate for the crank position, the desired forces for the legs are scaled by a weighting function between zero and one that depends on the crank position, θ_{crank}:

$$w = \frac{\sin(\theta_{\text{crank}}) + 1}{2}. \tag{8}$$

If $\tau_{\text{crank}} > 0$, the force on the pedal that the left leg should produce is

$$f_l = \frac{w\tau_{\text{crank}}}{l} \tag{9}$$

where l is the length of the crank arms. The desired pedal force for the right leg is

$$f_r = \frac{(1-w)\tau_{\text{crank}}}{l} \tag{10}$$

If τ_{crank} is less than zero, the equations for the left and right leg are switched. A kinematic model of the legs is used to compute hip and knee torques that will produce the desired pedal forces.

To steer the bicycle, the control system computes a desired angle for the fork based on the errors in roll and yaw:

$$\theta_{\text{fork}} = -k_\alpha(\alpha - \alpha_d) - k_{\dot\alpha}\dot\alpha + k_\beta(\beta - \beta_d) + k_{\dot\beta}\dot\beta \tag{11}$$

where α, α_d, and $\dot\alpha$ are the roll angle, desired roll, and roll velocity, respectively, and β, β_d, and $\dot\beta$ are the yaw angle, desired yaw, and yaw velocity. k_α, $k_{\dot\alpha}$, k_β, and $k_{\dot\beta}$ are gains. Inverse kinematics is used to compute the shoulder and elbow angles that position the hands on the handlebars with a fork angle of θ_{fork}; proportional-derivative servos move the shoulder and elbow joints towards those angles.

These control laws leave the motion of several joints of the bicyclist unspecified. The wrists and the waist are held at a constant angle with proportional-derivative controllers. The ankle joints are controlled to match data recorded from humans[3].

The desired position for the bicycle that is computed by the algorithm for group behaviors is used to compute a desired velocity for the bicycle:

$$\dot{x}_{\text{d}} = k_{\text{p}}e + k_{\text{v}}(\dot{x}_{\text{gl}} - \dot{x}) \tag{12}$$

where $\dot{x}_{\text{d}}$ is the desired velocity in the plane, e is the error between the current position of the creature and the desired position, k_{p} is the proportional gain on position, k_{v} is the proportional gain on velocity, and $\dot{x}_{\text{gl}}$ is the group's global desired velocity. The control system for the bicycle does not include a model of the delay in the control of velocity.

Point-mass Simulation

The point-masses have a mass equal to that of the one-legged robots. The desired position computed by the algorithm for group behaviors is used to compute a force that is applied to the point-mass:

$$f = k_{\text{p}}e + k_{\text{v}}(\dot{x}_{\text{gl}} - \dot{x})$$

where k_{p} and k_{v} are gains, $\dot{x}$ is the velocity of the point-mass and $\dot{x}_{\text{gl}}$ is the group's global desired velocity. There are no limits on the velocity. The point-mass system differs from the robots and bicyclists in that only the inertia of the mass prevents a given point-mass from reaching its new desired location.

Results

We tested the algorithms on three maneuvers: steady-state movement, turning, and avoiding obstacles. For steady-state movement, the initial configuration and velocity for the robots and point-masses were the same. The group of bicyclists had fewer members and the initial velocity was higher. Both the robots and the point-mass system contracted to form nearly circular shapes. Due to the minimal dynamics present in the point-mass system, the group of point-masses reached steady-state in under 7 seconds while the one-legged robots required nearly 21 seconds to reach an equally stable formation (figure 7). The group of bicyclists formed a pentagon with one bicyclist in the center. These shapes reflect the effects of the group behavior: each individual desires to be a specified distance from all visible neighbors. When this simple behavior is aggregated over all members of a group, a regular group formation results.

The second test involved turning. Beginning with a steady-state run the groups were commanded to turn first to the left and then to the right (figure 8). The bicyclists could not turn as sharply as the robots and point-masses. Both the bicycles and the point-mass systems completed the path without collisions, but the legged robots had collisions when the direction of the global desired velocity changed.

The final test involved obstacle avoidance (figure 8). The group of point-masses was able to avoid the obstacle and quickly rejoined to form a single group on the far side of the obstacle. The robots were also able to avoid the obstacle but were slower to rejoin on the far side of the obstacle. The bicycles were less able to adjust lateral velocity due to kinematic and dynamic constraints and had a larger global desired velocity. Although the bicycles came closer to the obstacle, they behaved in a manner similar to that of the robots and the point-masses.

The algorithms for group behaviors used for these trials were similar and most differences in performance can be attributed to differences between the underlying dynamics and control for each simulated system.

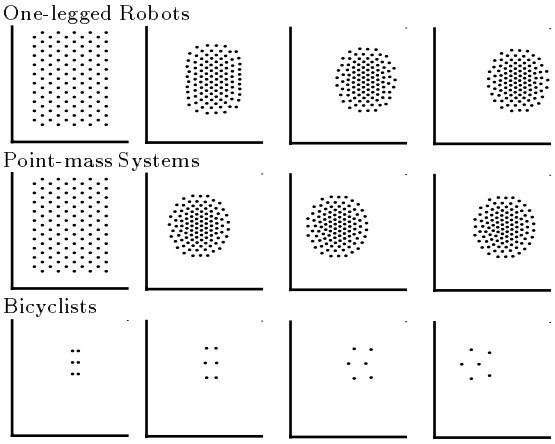

Figure 7: The top two rows of graphs show the group of robots and the point-mass systems at a start state and every 7 s thereafter as the two groups are commanded to travel at 2.0 m/s. The bottom row of graphs shows the bicyclists at a start state and every 10s thereafter while they are riding at 4.8m/s. Each graph represents the x and y position of each individual in the group.

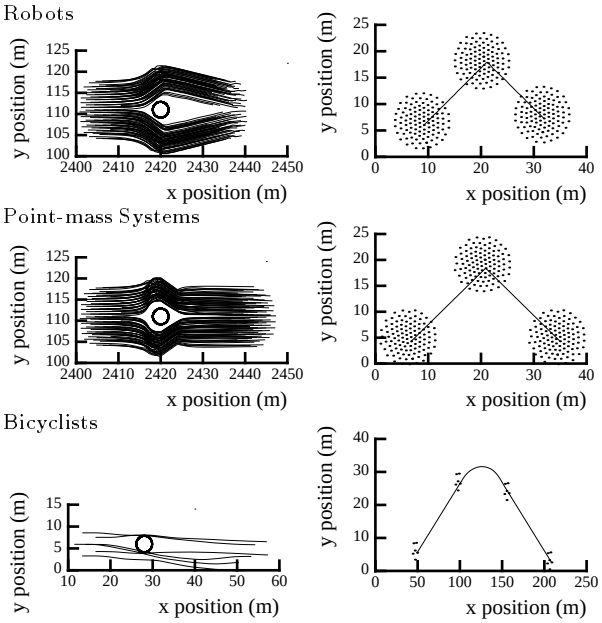

Figure 8: The left column illustrates the trajectories of the individuals in the groups of robots, bicyclists and point-masses as the creatures avoid an obstacle. The groups were moving left to right. In each case, the front edge of the group was positioned 8 meters from the obstacle before the obstacle was perceived. The radius of the obstacle was 2.0 m. In the right column the configurations of the groups are drawn to illustrate each simulation's performance as the desired velocity is changed to cause the group to turn. The robots and point-masses started in a steady-state at 2.0 m/s before they were asked to turn 45 deg to the left. After 20 s they were asked to turn 90 deg to the right. The bicyclists could not follow this path and required more time to complete smaller turns. In this experiment the bicyclists started in a steady-state at 5.5 m/s before they were asked to turn 22.5 deg to the left. After 20 seconds they were asked to turn 45 deg to the right. Snapshots of the group formations were taken every 10 s and the path of one individual in the group is traced through the whole turn.

The group of point-masses moved more tightly under changes in magnitude and direction of velocity because of the more exact control of velocity. The robot group had more variability and motion within the group, and tests more often resulted in collisions between members of the group. The collisions could have been prevented by increasing the desired separation distance between the creatures.

In more difficult tests than those reported here, an individual in the group of robots or the group of bicyclists sometimes lost its balance and fell. A maximum acceleration was enforced for the bicycles and the robots to prevent limitations in the low-level control from causing many of these failures. The point-mass systems had no notion of balance or maximum speed and could not fail in this way.

Communication as a means of coordination is implicit in the group's global knowledge of a global desired velocity, $\dot{x}_{\mathrm{gl}}$. In the turning test, for example, a synchronous change in direction is caused by changing this global desired velocity. However, reactions on a local scale require knowledge of the neighbors' intentions as opposed to those of the group. This knowledge is obtained by observing the positions and velocities of the n nearest neighbors. Larger values of n dampen the effects of local fluctuations because the quick movement of some neighbors is averaged with the stable behaviors of other neighbors. Alternatively, low values of n result in erratic and unstable behavior as local disturbances transfer undamped through the group.

There are many limitations to the algorithm for group behaviors we implemented. In some situations, the averaging of desired positions moved two individuals closer to collision, and there is no reflexive reaction to an impending collision. With this algorithm, a breakaway group of sufficient size will not rejoin the main group unless a member of the main group is visible to a member of the breakaway group. Presumably both problems could be solved by additional behaviors that cause individuals to react strongly to collisions and to look further afield for another group to join.

Our perceptual model assumes more complete and accurate information than that produced by sensors on physical robots. We experimented with other perceptual models by adding occlusion and reducing the visibility of creatures behind as opposed to in front of the individual in question. When the list of visible creatures changes because of the addition of a previously occluded individual, the desired position and velocity may change significantly causing a ripple effect throughout the group, thus increasing the probability that an individual will lose its balance in the robot and bicycle simulations.

Although the robots and the bicyclists are dynamic simulations, many factors are missing in the simulation that would be present in a physical system. 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 algorithms for group behaviors does not contain uneven or slippery terrain. The parameters for the one-legged robot are similar to those of robots that have been built[9], but the parameters for the bicyclist match those for a human and are superior to the materials available for robot construction.

We have not explored the question of how the algorithms will perform on a heterogeneous population. Currently all the robots and all the bicyclists have identical mass and inertia properties and identical control systems. We plan, however, to vary the parameters of the system and to introduce noise to study how the performance of the algorithms is affected. Heterogeneous groups that mix the types of simulated creatures could be studied by experimenting with low-speed, legged robots and higher-speed bicyclists on the same terrain. When tests such as obstacle avoidance and the turning are performed on homogeneous groups, the reaction of each individual is matched by those of other members of the group because each individual has the same limitations in low-level control. For example, one-legged robots cannot react while they are in the flight and must predict their motion and the motion of other creatures. Unlike the robots, the bicyclists are able to make small changes in direction at any time, but there is a delay while they adjust their lean and the angle of the front wheel to achieve large changes in direction. To work well, a group involving both these creatures must distribute this low-level knowledge to properly anticipate and react to the dynamic situations depending on the individuals involved.

Acknowledgments

This project was supported in part by NSF Grant Nos. IRI-9309189 and IRI-9457621, funding from the Advanced Research Projects Agency, and from Mitsubishi Electric Research Laboratory.

References

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

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

[3] Cavanagh, P., and Sanderson, D., 1986. The Biomechanics of Cycling: Studies of the Pedaling Mechanics of Elite Pursuit Riders. *Science of Cycling.* Burke, E. (ed), Chapter 5.

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

[5] Dempster, W.T., and Gaughran, G.R.L., 1965. Properties of Body Segments Based on Size and Weight. *American Journal of Anatomy*, 120: 33–54.

[6] Lien, S., Kajiya, J. T. 1984. A Symbolic Method for Calculating the Integral Properties of Arbitrary Nonconvex Polyhedra. *IEEE Computer Graphics and Applications*, 4(5):35–41.

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

[8] 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.

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

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

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

[12] 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.

[13] 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.

[14] Takeuchi, R., Unuma, M., and Amakawa, K., 1992. Path Planning and Its Application to Human Animation System. *Computer Animation 1992*, 163–175.

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

[16] 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.

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