

Group Behaviors for Systems with Significant Dynamics

DAVID C. BROGAN

dbrogan@cc.gatech.edu

College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280

JESSICA K. HODGINS

jkh@cc.gatech.edu

College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280

Received September 29, 1995. Revised September 30, 1996.

Editor: George A. Bekey

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 simulated creatures traveling fast enough that dynamics plays a significant role in determining their movement. In this paper, we describe an algorithm for controlling the movements of creatures that travel as a group and evaluate the performance of the algorithm with three simulated systems: legged robots, humanlike 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 position within the group. The simpler point-mass systems are included because they help us to understand the effects of the dynamics on the performance of the algorithm.

Keywords: multi-agent, herds, group navigation, dynamic simulation, legged locomotion

1. Introduction

To run as a group, animals must remain in close proximity while avoiding collisions with other members of the group and with 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 one such algorithm for group behaviors. The groups are made up of dynamically simulated legged robots, bicycle riders, and point masses. Examples of the systems we have simulated are shown in figure 1.

The algorithm for group behaviors computes a desired position for each individual based on the location and velocity of its visible neighbors, the location of visible obstacles, and a global desired group velocity. Each creature's desired position and velocity are known only to that creature although members of the group can estimate this information based on their

observations of the creature's actions. We compare the performance of this algorithm on the three dynamic simulations 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 control exerted by the grouping behaviors causes rapid and predictable changes in position and velocity. These predictable changes allow the point masses to navigate without collisions even with little advance notice of large obstacles. Predicting the movement of one-legged robots is more difficult because their velocity varies over a running cycle. Furthermore, the control of velocity is not as exact for the one-legged robots as it is for the point masses. For these reasons, grouping behaviors for the robots must compute less aggressive changes in desired position and velocity to avoid causing crashes. When the delay inherent in the control of velocity is taken into account, however, the one-legged robots are able to complete without collisions

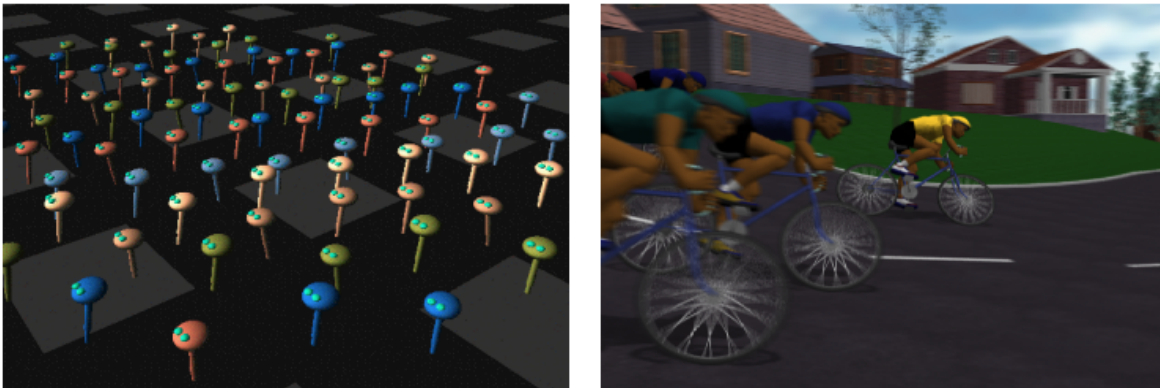


Fig. 1. Images of 105 simulated one-legged robots and 5 simulated bicycle riders.

the tests of obstacle avoidance and turning presented in this paper.

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 algorithm for group behaviors. When the higher-level algorithm requires better control than is available, individual bicyclists crash to the ground or into another member of the group. We reduced the difficulty of the turning test to allow the bicyclists to complete the tests without collisions.

In contrast to most previous implementations of algorithms for group behaviors, we are exploring algorithms for groups in which the members have significant dynamics. By significant dynamics, we mean that the motion of each individual is significantly affected by its dynamic properties and that a kinematic or point-mass simulation would not be an adequate representation. The dynamic system for an individual creates inherent limitations on acceleration, velocity, and turning radius. These limitations restrict what the low-level control algorithms can do and result in transient and steady-state errors. For example, changes in velocity of the one-legged robots may be delayed by almost a full running cycle because the low-level control system can influence velocity only during the stance phase of the cycle. The bicyclists also experience a delay because in order to turn to the right they must first steer to the left and lean right.

Algorithms for higher-level behaviors are needed for the construction of cooperating robots with robust and agile movements. When a robot moves quickly

enough that the dynamics of the system plays a significant role in the resulting motion, the algorithm that controls the higher-level behaviors must include explicit models of both the underlying dynamics and the limitations of the low-level control. It is our hope that by exploring the performance of this algorithm on dynamically simulated creatures, we can gain an understanding of principles that will carry over to the control of groups of physical robots performing useful tasks.

Algorithms for higher-level behaviors are also needed for the construction of virtual actors that can move and interact with a dynamic environment robustly and realistically. 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. Higher-level behaviors such as obstacle avoidance, grouping, and rough terrain locomotion would allow the actor to interact with the user and with a complex and unpredictable environment.

2. Background

Herding, flocking, and schooling behaviors of animals have been studied extensively over the past century, and this research has stimulated attempts to create robots and simulated creatures with similar skills. Biologists have found that groupings in animals are created through an attraction that modulates the desire of each member to join the group with the desire to maintain a sufficient 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 a school, as if each fish had a sphere around its head with which it wished to contact the spheres of other fish. Biologists have found that herding benefits group members by limiting the average number of encounters with predators (data summarized in Veherencamp (1987)). 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 argues the merit of exploring their use in robotic systems. An understanding of these behaviors is essential for realistic creatures in virtual environments.

Early progress in the simulation of group behaviors was made by Reynolds (1987). Actors in his system are birdlike objects similar to the point masses used in particle systems except that each bird also has an orientation. The birds maintain proper position and orientation within the flock by balancing their desire 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 one aspect of perception and reaction in biological systems and helps to balance 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) proposed a decentralized approach to the navigation problem for multiple agents. Their simulation 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. Because of the two levels of planning, this solution requires the communication overhead associated with group behaviors only when a pair of robots perceive an impending collision.

Sugihara and Suzuki (1990) demonstrated that multiple robots can form stable formations in simulation when each robot executes an identical algorithm for position determination within the group. 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 simulation. By carefully constructing the algorithm that each robot uses in determining intragroup 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-follower algorithms and to demonstrate the division of a formation into smaller groups.

Wang (1991) investigated the asymptotic stability of multiple simulated robots in formation. 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 proved 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.

Takeuchi, Unuma, and Amakawa (1992) implemented path planning in simulated multi-agent systems where the attraction between agents is dependent on properties of the agents. The simulated agents are Newtonian particles where forces are applied to the agents based on the vector sum of attractions to other 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 in the laboratory 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. The combined velocity vector is used to control the robot. Arkin demonstrated that multiple robots can effectively complete group tasks such as foraging and can retrieve large quantities of goal items with little or no explicit communication. The algorithm for group behaviors described in this paper is similar to Arkin's approach in that it is an example of an algorithm in which there is no explicit leader and all communication is through observations of the environment.

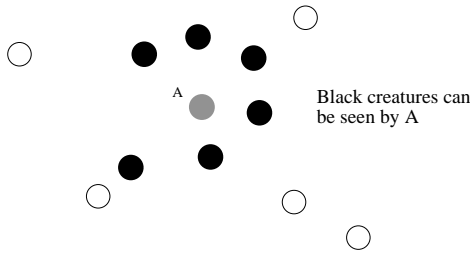


Fig. 2. One creature is visible to another if it is one of the n closest creatures (n is 6 for this example). The black circles represent the set of visible creatures, N , and the white those that cannot be seen by A.

Mataric explored emergent behavior and group dynamics in a group of 20 wheeled vehicles in the laboratory. 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 aggregation and dispersion can produce such complex relationships as flocking in physical robots in the laboratory ((Mataric, 1992a), (Mataric, 1992b)). 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 (Mataric, 1993). In these experiments, a hierarchy is created in which an ordering between the agents determines which agent will move first in completing such tasks as grouping and dispersing.

Parker (1993) investigated the advantages and disadvantages of using local and global knowledge when designing control laws for cooperative agent teams. Simulated wheeled robots were used to explore the tradeoffs in tasks where the robots are commanded to move in formation. She found that knowledge of global goals and global state contributes to improved performance but at the cost of increased interagent communication. Behavioral analysis is an alternative to communicating global knowledge when the actions of the agents are recognizable. The experiments indicate that global knowledge should be used to control longer-term actions whereas local knowledge is best used to control short-term actions that fit within the context of global goals. In this way, local information grounds the global knowledge in the current robot state.

3. Group Behaviors

The algorithm for group behaviors described in this paper was evaluated on three simulated systems: a

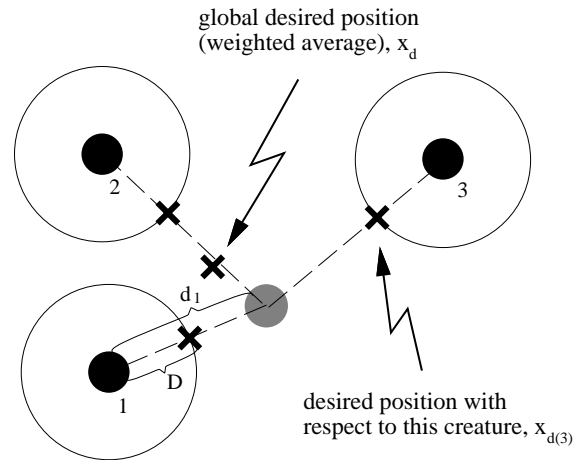


Fig. 3. The locations of the visible creatures are used to compute a global desired position for the individual under consideration. The algorithm computes a desired position with respect to each visible robot, $x_{d(i)}$, by finding the point on the line between the individual and the visible creature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

one-legged robot, a rigid-body model of a human riding a bicycle, and a point-mass system. In most trials the groups of point-mass systems and of legged robots each contained 105 individuals; the group of bicyclists contained 18 individuals.

The algorithm for group behaviors has two parts: a perception model to determine the creatures and obstacles visible to each individual in the group and a placement algorithm to determine the desired position for each individual given the locations and velocities of the visible creatures and obstacles. 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 set of visible creatures, N (figure 2). This information is used to compute a desired position, $(x_{d(i)}, y_{d(i)})$, relative to each of these creatures. This position is a distance D away from the visible creature on the line between the two creatures (figure 3):

$$y = \frac{y_i - y_A}{x_i - x_A} x \quad (1)$$

where (x_i, y_i) is the current position of creature i , and (x_A, y_A) is the current position of creature A . When $x_i = x_A$, the two creatures are on a line parallel to the y axis and the slope of the line is assumed to be a

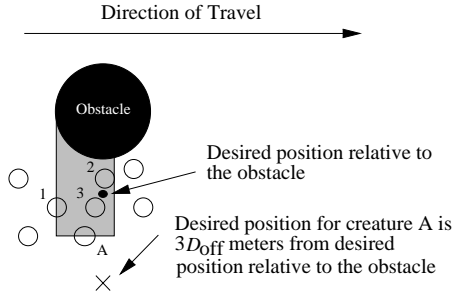


Fig. 4. The desired position for creature A takes into account both the obstacle and the neighbors that will move sideways as they approach the obstacle. First, the algorithm calculates a desired position that will prevent the creature from hitting the obstacle. Then the algorithm creates a rectangular region between creature A and the obstacle, calculates the number of visible neighbors within that region, and multiplies the desired separation distance, D_{off} , by the number of creatures. In the figure, creatures 1, 2, and 3 are members of n and are contained within the shaded rectangular region, so D_{off} is multiplied by three and that amount is added to the desired position with respect to the obstacle.

large, but not infinite, number. The desired separation distance, D , specifies that $(x_{d(i)}, y_{d(i)})$ is D meters away from creature i :

$$D = \sqrt{x_{d(i)}^2 + y_{d(i)}^2}. \quad (2)$$

Using equations 1 and 2, we can solve for $(x_{d(i)}, y_{d(i)})$:

$$x_{d(i)} = \frac{D}{\sqrt{\left(\frac{y_i - y_A}{x_i - x_A}\right)^2 + 1}}. \quad (3)$$

$y_{d(i)}$ is computed in a similar fashion. After computing the desired position relative to each creature in N , the set of desired positions is weighted by the actual distance between each pair of creatures, d_i , and averaged to compute a global desired position:

$$x_d = x_A + \frac{\sum_{i \in N} \text{sgn}(x_A - x_i) \frac{x_{d(i)}}{d_i^2}}{\sum_{i \in N} \frac{1}{d_i^2}}. \quad (4)$$

$\text{sgn}(p)$ returns -1 when p is negative and 1 otherwise. The desired y position, y_d , is computed similarly.

In addition to avoiding collisions with other individuals in the group, the creatures must avoid obstacles. The effect of an obstacle on a creature's behavior increases as it nears the obstacle, and an offset to the left or right of the obstacle is added to the average of the desired positions relative to neighbors:

$$y_{d_{obs}} = y_{obs} \pm \frac{D_{obs}}{d_{obs}^2} - y_A \quad (5)$$

where y_{obs} is the y position of the obstacle, D_{obs} is the desired separation distance from the obstacle, and d_{obs} is the actual distance between the creature and the obstacle. In addition, the algorithm for grouping behaviors allows space between the creature and the obstacle for the creature's neighbors by calculating the number of visible neighbors between the creature and the obstacle and multiplying this number by an additional separation distance, D_{off} , to produce an additional offset to the desired position for avoiding the obstacle (figure 4).

After the individual passes the center of the obstacle, the average of the desired positions relative to neighbors includes a term for a position towards the center of the obstacle. This term serves to bring the two groups on either side of the obstacle together. Within a short distance, the two groups perceive and react to creatures from the other group, merge into one, and return to steady state.

Although the group behaviors are identical for the three systems up to this point in the algorithm, the 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 desired 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.

4. Simulating Groups

A simulation of a group of creatures consists of the equations of motion and a state vector for each creature, 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 humanlike 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

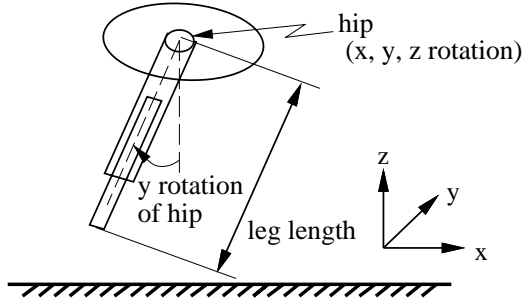


Fig. 5. The reference angles for the degrees of freedom of the one-legged robot. The controlled degrees of freedom are the three degrees of freedom at the hip and the length of the leg.

package (Rosenthal and Sherman, 1986). 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. There were no collisions in the tests described in this paper. The details of the robot, the bicycle rider, and the point-mass models are described below.

4.1. One-legged Robot

The locomotion algorithm for the one-legged robot controls flight duration, body attitude, and forward and sideways velocity. Flight duration is controlled by extending the telescoping leg during stance to make up for losses in the system:

$$l_d = l_{td} + l_{th} \quad (6)$$

where l_d is the desired length of the leg, l_{td} is the length of the leg at touchdown, and l_{th} is the desired thrust. Body attitude (pitch, roll, and yaw) is controlled by exerting a torque between the body and the leg during stance:

$$\tau_\theta = k_\theta(\theta_d - \theta) - k_\dot{\theta}\dot{\theta} \quad (7)$$

where θ is the pitch, roll, or yaw angle, τ_θ is the hip torque for the corresponding axis, k_θ is the proportional gain, $k_\dot{\theta}$ is the derivative gain, θ_d is the desired angle, and $\dot{\theta}$ is the roll, pitch or yaw velocity. The forward and sideways velocities are controlled

Table 1. 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).

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

Table 2. 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	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

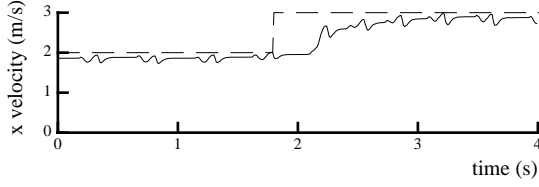
by the position of the foot with respect to the hip at touchdown:

$$x_{fh} = \frac{1}{2}t_s\dot{x} + k_{\dot{x}}(\dot{x} - \dot{x}_d) \quad (8)$$

where x_{fh} is the position of the foot with respect to the hip in the x direction, t_s is the expected time of stance, \dot{x} is the velocity, $k_{\dot{x}}$ is the gain for the control of velocity and \dot{x}_d is the desired velocity. The velocity in the y direction is controlled with a similar equation. Equation 8 implies that 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 farther from the hip. The details of the locomotion control algorithm is given in Raibert (1986). The reference angles of the model are shown in figure 5. The parameters of the robot are given in tables 1 and 2.

The algorithm for group behaviors results in a desired position for each individual in the group. The individual would be in the perfect position according to the goals of the algorithm if it could move to that position instantaneously. Because none of the three systems can change velocity instantaneously, the desired position is used to calculate a desired velocity, thereby introducing a delay into the system. Running creatures have the additional constraint that they cannot change velocity during flight, and the control uses a model of this delay to improve the performance of the algorithm.

One-legged Robot



Bicyclist

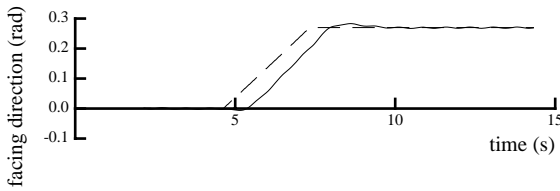


Fig. 6. The dynamics of the system can cause delays in response times. The top graph shows the response of the simulated one-legged robot to a step change in desired velocity. The bottom graph shows the response of the bicyclist to a linear ramp in desired facing direction. Solid lines show the actual velocity/facing direction; dashed lines show the desired. When the desired velocity of the one-legged robot changes at the end of the flight phase, the actual velocity does not change until the next stance phase. Similarly, the bicyclist requires approximately 0.5 s to respond significantly to the change in desired facing direction.

In the one-legged robots, 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 approaches the new desired velocity and then remains constant during the next flight phase. The response to a step change in desired velocity is illustrated in figure 6. To compensate for the delay in the control of the velocity, the predicted error at the end of the next locomotion step is used to compute the desired velocity. The predicted position at the end of the next step is

$$x_p = x + \dot{x}(t_s + t_f) \quad (9)$$

where x is the current position, \dot{x} is the velocity, t_s is the 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:

$$x_{dp} = x_d + \dot{x}_a(t_s + t_f) \quad (10)$$

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

creatures 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

$$e = x_{dp} - x_p. \quad (11)$$

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 uses the error in desired position to compute a velocity, \dot{x}_d , that will make the position of the robot at the end of the next flight phase match the desired position:

$$\dot{x}(t_s + t_f) + e = \frac{\dot{x} + \dot{x}_d}{2}t_s + \dot{x}_d t_f. \quad (12)$$

Solving for \dot{x}_d :

$$\dot{x}_d = \dot{x} + \frac{e}{\frac{t_s}{2} + t_f}. \quad (13)$$

Thus, the change in velocity required to match the desired position is $\frac{e}{\frac{t_s}{2} + t_f}$ and the change in velocity required to match the global desired velocity, \dot{x}_g , is $\dot{x}_g - \dot{x}$. We calculate the new desired velocity to be the sum of the current velocity and the average of the two changes in velocity calculated above:

$$\dot{x}_d = \dot{x} + \frac{1}{2} \left(\dot{x}_g - \dot{x} + \frac{e}{\frac{t_s}{2} + t_f} \right). \quad (14)$$

A similar model would be required for any creature with a ballistic flight phase during which speed and facing direction cannot be altered.

4.2. 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 7 and table 4). 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 (Lien

Table 3. Parameters of the rigid-body model of a human. The moments of inertia are computed about the center of mass of each link. The densities for the human model are given in Dempster and Gaughran (1965).

Link	Density (g/cm ³)	Mass (kg)	Moment of Inertia (x, y, z kgm ²)		
Head	1.17	5.89	0.030	0.033	0.023
Torso	1.01	29.27	0.73	0.63	0.32
Pelvis	1.03	16.61	0.23	0.18	0.16
Upper Leg	1.04	8.35	0.15	0.16	0.025
Lower Leg	1.08	4.16	0.055	0.056	0.007
Foot	1.07	1.34	0.002	0.008	0.007
Upper Arm	1.07	2.79	0.025	0.025	0.0050
Lower Arm	1.10	1.21	0.0050	0.0054	0.0012
Hand	1.07	0.551	0.002	0.002	0.001

and Kajiy, 1984). Density data obtained from the anatomical literature were used in calculating the dynamic properties of the body segments (Dempster and Gaughran, 1965) (table 3). The degrees of freedom of the bicycle model are shown in figure 7 and the parameters of the model are given in table 4.

The simulated human rider controls the facing direction and speed of the bicycle by applying forces to the handlebars with the hands and to the pedals with the feet. 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

$$\tau_c = k(v - v_d) \quad (15)$$

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 depends 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_c :

$$w = \frac{\sin(\theta_c) + 1}{2}. \quad (16)$$

If $\tau_c > 0$, the force on the pedal that the left leg should produce is

$$f_l = \frac{w\tau_c}{l} \quad (17)$$

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

$$f_r = \frac{(1-w)\tau_c}{l} \quad (18)$$

If τ_c 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:

$$\theta_f = -k_\alpha(\alpha - \alpha_d) - k_{\dot{\alpha}}\dot{\alpha} + k_\beta(\beta - \beta_d) + k_{\dot{\beta}}\dot{\beta} \quad (19)$$

where α , α_d , and $\dot{\alpha}$ are the roll angle, desired roll angle, and roll velocity, respectively, and β , β_d , and $\dot{\beta}$ are the yaw angle, desired yaw angle, 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 would position the hands on the handlebars with a fork angle of θ_f ; proportional-derivative servos move the shoulder and elbow joints toward 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 (Cavanagh and Sanderson, 1986).

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

$$\dot{x}_d = k_p e + k_v(\dot{x}_g - \dot{x}) \quad (20)$$

where \dot{x}_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}_g is the group's global desired velocity. In these experiments, k_p and k_v were 4 in the obstacle test and 1 and 4 respectively in the turning test. The higher-level control system for the bicycle does not include a model of the delay in the control of velocity and hence requires these empirically determined gains (figure 6).

Table 4. The distance from the center of mass of each link to the distal and proximal joints in x , y , and z . A positive distance along the y axis refers to a location on the left side of the body; a negative distance refers to the right side. The z axis is vertical and the x axis is positive in the direction that the model is facing.

Link	COM to Proximal (x, y, z m)			COM to Distal (x, y, z m)		
	x	y	z	x	y	z
Torso to neck				0.012	0.0	0.32
Torso to waist				0.012	0.0	-0.22
Torso to shoulder				-0.048	± 0.164	0.12
Head	-0.009	0.0	-0.064			
Pelvis	0.023	0.0	0.103			
Pelvis to hips				0.005	± 0.098	-0.11
Pelvis to bike seat				0.0	0.0	-0.15
Upper Leg	0.024	± 0.006	0.120	-0.05	± 0.019	-0.21
Lower Leg	0.005	± 0.019	0.165	-0.002	± 0.009	-0.25
Foot	-0.046	± 0.009	0.048			
Upper Arm	-0.0002	± 0.056	0.120	-0.005	± 0.036	-0.17
Lower Arm	-0.025	± 0.007	0.090	0.012	± 0.014	-0.11
Hand	-0.026	0.0	0.085			
Frame to bike seat	-0.15	0.0	0.028			
Frame to rear wheel				-0.42	0.0	-0.36
Frame to fork				0.46	0.0	0.09
Frame to crank				0.022	0.0	-0.36
Fork to front wheel				-0.026	0.0	0.085
Fork	-0.05	0.0	0.108			

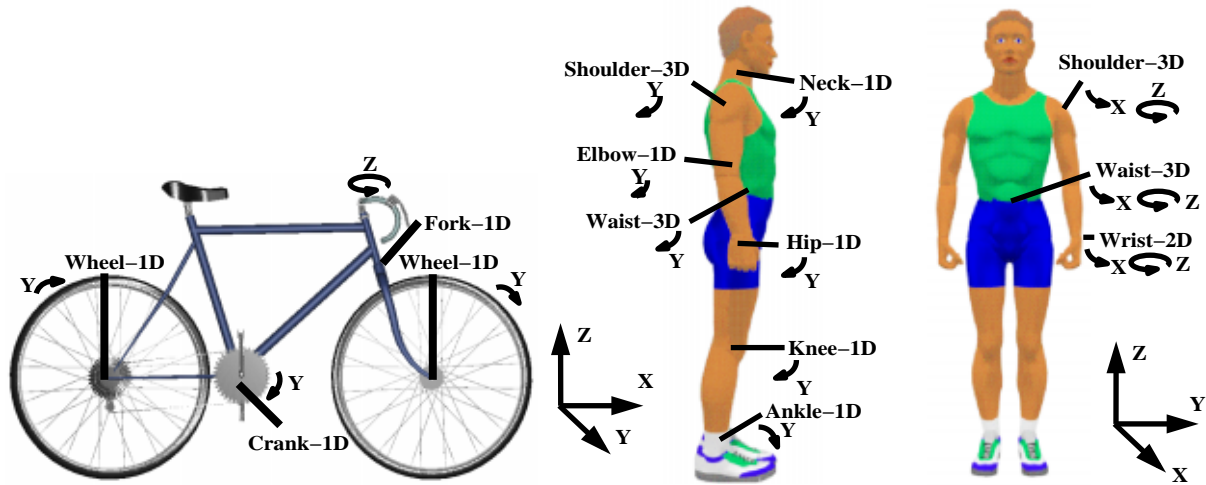


Fig. 7. The controlled degrees of freedom of the bicycle and human models. The human model has fourteen joints; the degrees of freedom at each joint are shown in the diagram as are the four degrees of freedom of the bicycle model. The human rider is attached to the bicycle by a pivot joint between the seat and the pelvis. The polygonal models were purchased from Viewpoint Datalabs.

4.3. Point-mass Simulation

$$f = k_p e + k_v (\dot{x}_g - \dot{x}) \tag{21}$$

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:

where k_p and k_v are gains, e is the error in position, \dot{x} is the velocity of the point mass and \dot{x}_g is the global desired velocity. In these experiments k_p was 6000 and k_v was 2000. There are no limits on the velocity. The point-mass system differs from the robots and

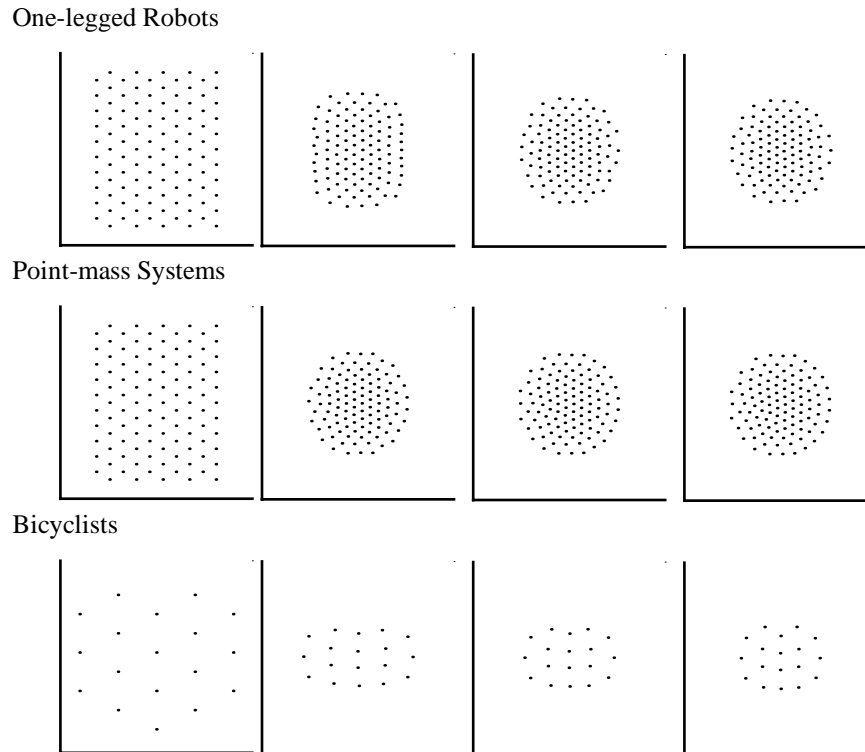


Fig. 8. The top two rows of graphs show the group of robots and the point-mass systems at a start state and every 7 s thereafter with a global desired velocity of 2.0 m/s. The bottom row of graphs shows the 18 bicyclists at a start state and every 20 s thereafter while they are riding at 7.5 m/s. In each of these graphs the number of visible creatures, n , was set equal to the total number of creatures. Each point on the graphs represents the x and y position of an individual in the group. As a measure of the convergence to steady state, we computed the ratio of the x dimension to the y dimension of the bounding box around each group. A circular formation would have a ratio of 1. One-legged robots: 0.78, 0.78, 0.93, 0.97 (for 0, 7, 14, 21 s). Point masses: 0.78, 0.98, 0.99, 0.99 (for 0, 7, 14, 21 s). Bicyclists: 1.33, 1.96, 1.56, 1.27 (for 0, 20, 40, 60 s).

bicyclists in that only the inertia of the mass prevents a given point mass from reaching its new desired location within a single time-step.

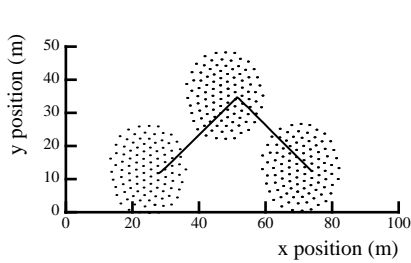
5. Results

We tested the algorithm on three maneuvers: steady-state movement, turning, and avoiding obstacles. For steady-state movement, the initial configuration of the one-legged robots, the point masses, and the bicyclists was a grid. The size of the groups of robots and point masses were the same (105) as were the initial velocities of the individuals in the groups (2.0 m/s). The group of bicyclists differed in that it had fewer members (18) and the initial velocity was higher (7.5 m/s). In our experiments, we considered systems to have formed a circle when the ratio of the x dimension to the y dimension of the bounding box surrounding the group was between 0.95 and 1.05. Once groups formed

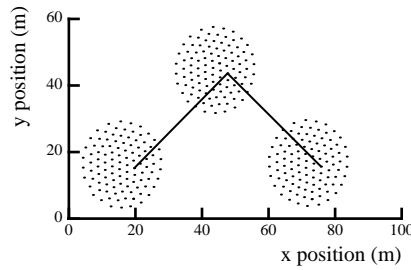
a circle, there was very little change in the dimensions of the bounding box and steady state was achieved. Due to the minimal dynamics present in the point-mass system, the group of point masses reached steady state in under 7 s while the one-legged robots required slightly more than 14 s to reach steady state (figure 8). After 60 s, the group of bicyclists formed an ellipsoidal shape that was slowly becoming more circular. In this experiment, the number of visible creatures, n , was set to equal the total number of creatures in the group. These circular 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 that approximates a circle.

To navigate and accomplish tasks in a complex environment, creatures must not only be able to form groups but they must also be able to turn and to avoid obstacles without collisions between the members of

One-legged Robots



Point-mass Systems



Bicyclists

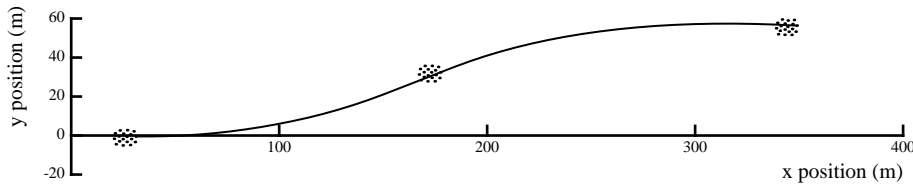


Fig. 9. The robots and point masses started in steady state at 2.0 m/s before the global velocity was changed to cause a 45 deg turn to the left. After 20 s the global desired velocity was turned 90 deg to the right. The bicyclists could not follow this path and required smaller turns. The bicyclists started in steady state at 7.5 m/s before turning 22.5 deg to the left. After 20 s the global desired velocity was changed to turn 45 deg to the right. Snapshots of the group formations were taken every 20 s and the path of one individual in the group is traced through the whole turn.

the group. The second test of the algorithm involved several turns. Beginning with a steady-state run, the global desired velocity was set first to cause a turn to the left by 45 degrees and then after 20 s to cause a turn to the right by 90 degrees (figure 9).

The bicyclists could not turn as sharply as the robots and point masses. As a result, the left turn for the bicyclists involved a 10 s period where the global desired velocity changed gradually followed by a 10 s period of constant desired velocity. The right turn was implemented in a similar fashion. The desired turning angle for the bicyclists was half that of the points and robots. In these tests, all three systems completed the path without collisions. The desired separation distance, D , was 5.0 m for the robots and point-mass systems and 3.5 m for the bicyclists. When D was smaller, the robots had collisions near the point in time when the global desired velocity was changed. The number of visible neighbors, n , was set to 30 for the robots and point-mass systems and to 9 for the bicyclists.

To test whether the higher-level algorithm would allow the groups to navigate a course with obstacles, we positioned an obstacle in front of each type of group when its members were moving in steady state. The values of n and D were the same as the turning test. The robots and the point masses were allowed to

perceive the obstacle 5 m before they reached it; the bicyclists detected the obstacle when it was 13 m from the leading edge of the group. Because the bicyclists were traveling at 7.5 m/s and the robots were running at 2 m/s, both groups had approximately the same length of time in which to react to the obstacle. 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 (figure 11). Both the robots and the bicyclists were able to avoid the obstacle but were slower to rejoin and form a single group (figure 10).

We used additional tests of obstacle avoidance to explore further the performance of the algorithm for grouping behaviors. The first set of tests contained three trials which varied the position of the obstacle so that it was not centered in front of the group of one-legged robots (figure 12). All three trials were completed without collisions. To permit the group to avoid the off-center obstacle without collisions, the amount of space allowed for each neighbor in the rectangle between the individual and the obstacle (D_{off}) had to be doubled (figure 4).

A second set of trials tested the effect of varying the size of the group. An obstacle was centered in front of groups of 20, 40, 60, and 80 one-legged robots (figure 13). In these trials the leading edge of the

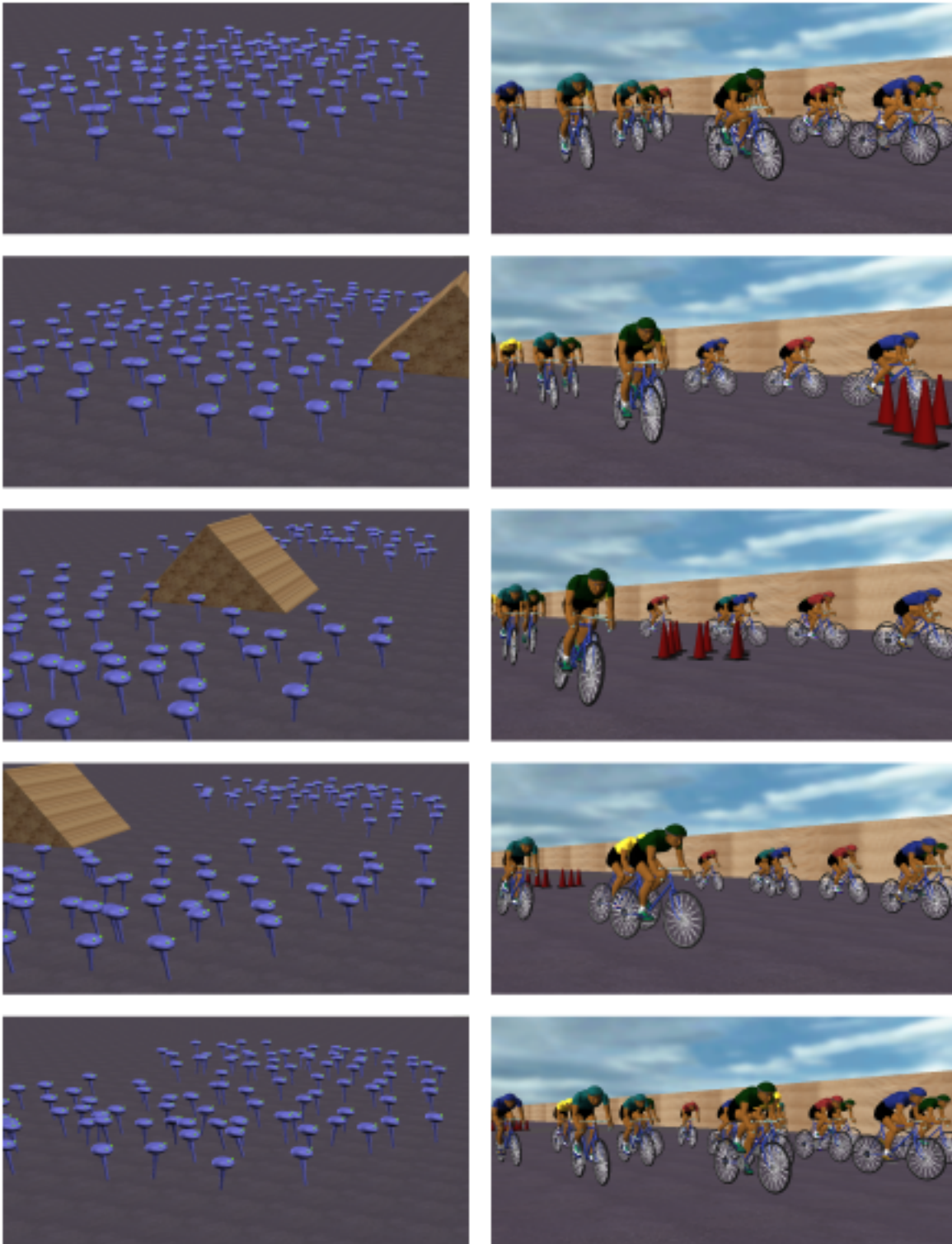
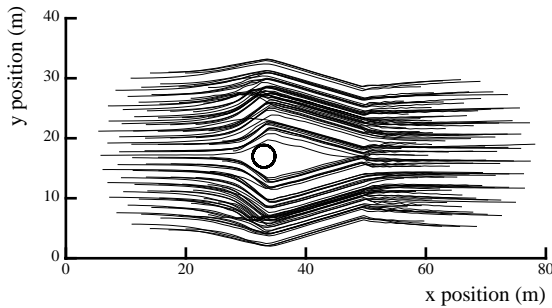
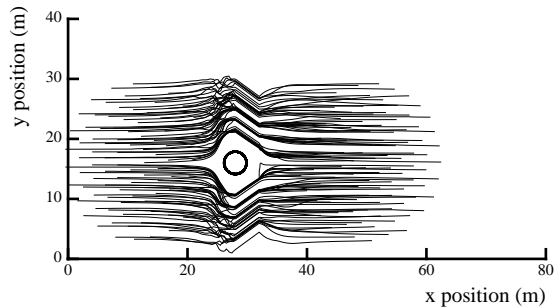


Fig. 10. Five pictures of the robots and the bicyclists as they avoid an obstacle.

One-legged Robots



Point-mass Systems



Bicyclists

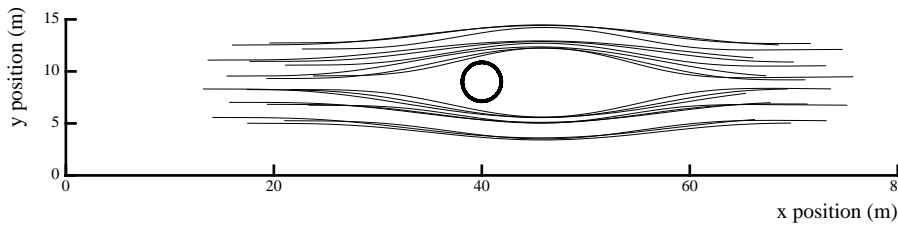


Fig. 11. The trajectories of the individual robots, bicyclists, and point masses as the groups avoid an obstacle. The groups were moving left to right. In the case of the one-legged robots and the point-mass systems, the front edge of the group was positioned 5 m from the obstacle with an initial velocity of 2 m/s. The front edge of the bicyclists was positioned 13 m from the obstacle with an initial velocity of 7.5 m/s. In each instance, the radius of the obstacle was 2.0 m.

robot group was 10 m from the center of the obstacle. The number of visible neighbors, n , and the desired separation distance, D , were scaled proportionally for the number of creatures in the group: $n = 6, 13, 20, 25$ and $D = 3, 4, 4.5, 5$ m for groups of 20, 40, 60, and 80, respectively. The four sizes of robot groups were able to navigate past the obstacle without collisions.

6. Discussion

The algorithms for group behaviors used for these trials were similar for the three systems and most differences in performance can be attributed to differences in the underlying dynamics and control. 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 the separation distance was made larger to prevent collisions between members. The control system for the bicyclists was not as robust, and the bicyclists were not able to perform as well on the turning test. In more difficult tests than those reported here, an individual in

the group of robots or bicyclists sometimes lost its balance and fell. A maximum acceleration was enforced for the bicyclists 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.

6.1. Global and Local Communication

Communication as a means of coordination is implicit in the global desired velocity, \dot{x}_g . In the turning test, for example, a synchronous change in direction is caused by changing this global desired velocity. Reactions on a local scale, however, require local knowledge in addition to global state information. This knowledge is obtained by observing the positions and velocities of the n nearest neighbors. In the obstacle test, for example, the position of the obstacle is known by all creatures, but local information about the positions of other creatures and a model of their behavior was required to eliminate collisions when the reaction time

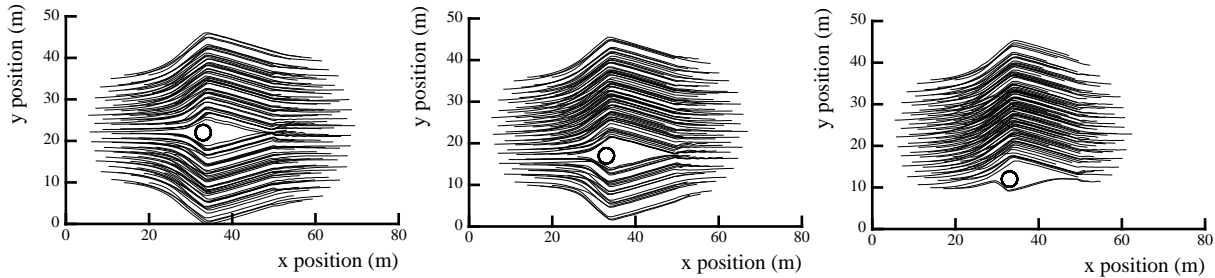


Fig. 12. A group of robots was simulated with three different obstacle placements (centered in front of the group, 5 m to the side, and 10 m to the side). The robots are traveling from left to right. The trajectories of the individuals are traced as the groups avoid the obstacle. Robots at the outer edges of the more populated sides had to travel farther to avoid both the obstacle and the other robots that were passing on the same side of the obstacle.

was small. In tests where the group reaction time was large, both the model of neighbors' behavior and the visibility of the obstacle throughout the group were removed and the obstacle was avoided without collisions.

The group of bicyclists was the only system that required adjusting the gains on position error and global velocity error, k_p and k_v . The bicyclist simulation used the same ratio of gains as the robots for the obstacle avoidance test, but k_p was one-fourth as large in the turning test. By more heavily weighting the bicyclists' reaction to global velocity, the bicyclists were able to respond more quickly to changes in global velocity while avoiding collisions with neighbors. With imperfect models and imperfect perception, it is difficult to determine how to weight global and local information.

6.2. Number of Visible Neighbors

The value of n has significant effects on group performance. The number of visible creatures affects both the general shape of the group as well as the overall responsiveness of the group. Because the values of n used in most of the tests reported here are large (one-third the size of the group), the averaging of the desired positions used by the grouping behaviors results in some neighbors that are closer than D and some neighbors that are farther apart. For creatures in the interior of the group, the number of neighbors in all directions is equal. Creatures on the edge of the group, however, have neighbors on only one side and forces that bring them closer to their neighbors will be directed towards the center of the group. The net effect of all the creatures along the edge pushing inward is

to compress the center of the group, therefore, larger values of n result in greater compression of the group and a smaller separation distance. This effect can be reduced by weighting the desired position for a particular neighbor by the inverse of the distance to that neighbor (as in equation 4) or by increasing the desired separation distance. Our experiments with the steady-state test indicate that large values of n produce stable, round configurations more quickly than small values of n . Because n is high, the edges of the group have stronger attractions to the center of the group and the overall configuration settles more quickly. Figure 14 illustrates the effect of the number of visible creatures and the separation distance.

In addition to affecting the shape of the group, n affects the responsiveness of the group. Low values of n isolate a creature's perception to a local region and increases the effect of a nearby neighbor. Therefore, lower values of n result in higher-level control that is more responsive to local state and will allow the system to react more quickly to small obstacles. But low values of n can result in erratic and unstable behavior as local disturbances transfer undamped through the group. Obstacle avoidance experiments with the one-legged robots indicate that a small value of n , 5, results in a safer passing distance for those robots close to the obstacle but greater lateral movement by all members of the group.

Higher values of n reduce local reactions but cause the motion of distant neighbors to affect the desired position of an individual. Perception of distant neighbors can provide an early warning for such upcoming changes in the local environment as an approaching but not yet visible obstacle. In our trials, response to a large obstacle was more fluid and resulted in fewer

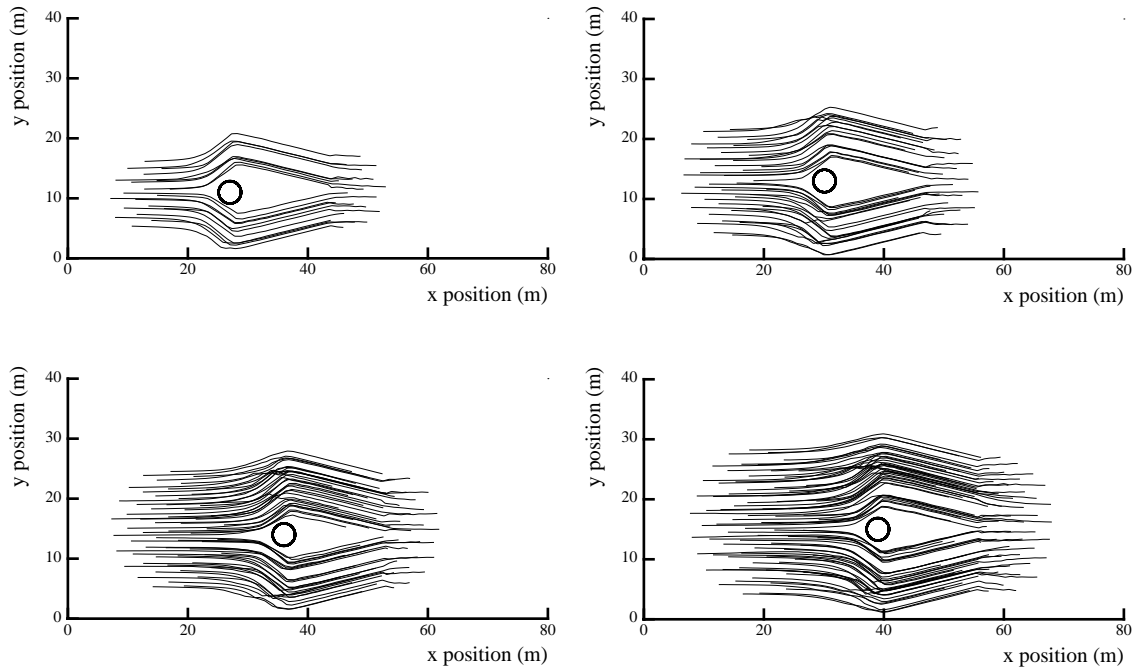


Fig. 13. Three groups of robots were simulated with a central obstacle location. The robots are traveling from left to right. The trajectories of the individuals are traced as the groups avoid the obstacle. The groups have 20, 40, 60, and 80 members, respectively. In each graph, the leading edge of the robots is placed 10 m from the center of the obstacle. The value of n has been scaled in proportion to the size of the group: $n = 6, 13, 20,$ and 25 respectively. The desired separation distance has been scaled to maintain a consistent average separation distance between robots: $D = 3, 4, 4.5, 5$ m.

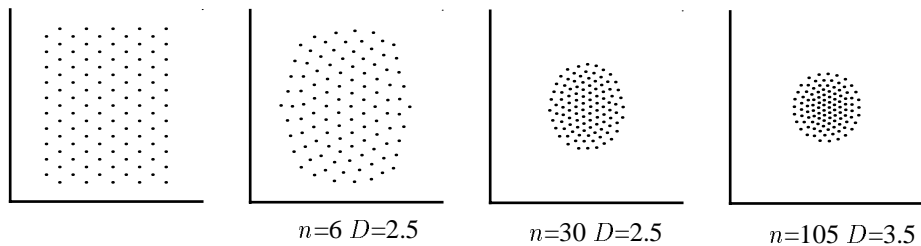


Fig. 14. The first graph shows the initial configuration of robots for three experiments. The other three graphs show the configurations of the group after 80 s of simulation with each graph representing a different choice for the number of visible robots (n) and the desired separation distance (D). 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 actual average separation distance between an individual and the set of neighbors in N was 2.36, 2.76, 4.78 m for the three trials. The average minimum separation distance between pairs of creatures was 1.98, 1.11, 0.95 m.

collisions when n was large. On the other hand, collisions may result if changes that occur only in the local environment are ignored because of the stability of the more distant environment.

Smaller values of n may prevent a breakaway group of sufficient size from joining another group because no individuals in the other group are visible to those in the breakaway group. For the value of n used in

the obstacle test, the desired position relative to the obstacle had to be moved inward after the creature was safely past the obstacle to ensure that the two groups always rejoined to form a single group. In more complex environments with many obstacles, the decision to remain as separate groups or to join will be more difficult. By increasing the complexity of the perception algorithm, creatures should have the ability

to distinguish nearby groups from more distant groups that can be ignored.

6.3. Limitations of the Grouping Behaviors

The algorithm for group behaviors that we implemented has several limitations. In some situations, the averaging of desired positions moved two individuals closer to collision, and there is no reflexive reaction to an impending collision beyond the averaging of desired positions. We found that reflexive reactions can create discontinuities in the calculation of the desired position that cause the dynamic simulations to become unstable. However, infrequent reflexive reactions may avert some collisions and provide more realistic-looking motion while maintaining the stability of the creature.

A second limitation is that 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 an individual as opposed to those in front. When the set 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 and increasing the probability that an individual will lose its balance.

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 algorithm 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 (Raibert, 1986), 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 algorithm will perform on a heterogeneous population. Currently, each individual has the same dynamic properties and control system. We plan, however, to vary the parameters for each individual to study the effect of a similar but nonhomogeneous population on the performance of the algorithm.

Heterogeneous groups that mix types of simulated creatures could be studied by experimenting with low-speed, legged robots and higher-speed bicyclists in the same environment. When tests such as obstacle avoidance and turning are performed on homogeneous groups, the reaction of each individual is matched by those of other members of the group because each member has the same fundamental limitations in low-level control. To work well, algorithms for nonhomogeneous groups must model these low-level limitations in order to predict more accurately the motion of members of the group.

7. Acknowledgments

An earlier version of this paper appeared in the 1995 *IEEE/RSJ International Conference on Intelligent Robot and Systems*. 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: Multi-agent Schema Based Robot Navigation. *Journal of Robotic Systems*, Vol. 9(3), 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*. E. Burke (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., and 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. Mataric, M., 1993. Kin Recognition, Similarity, and Group Behavior. *Proceedings of the Fifteenth Annual Cognitive Science Society Conference*, 705--710.

10. Parker, L.E., 1993. Designing Control Laws for Cooperative Agent Teams. *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, 582-587.
11. Raibert, M.H., 1986. *Legged Robots That Balance*. Cambridge: MIT Press.
12. Reynolds, C.W., 1987. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21(4): 25--34.
13. 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.
14. Shaw, E., 1970. Schooling in Fishes: Critique and Review. *Development and Evolution of Behavior*. L. Aronson, E. Tobach, D. Leherman, and J. Rosenblatt (eds), W. H. Freeman: San Francisco, CA, 452--480.
15. 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.
16. Takeuchi, R., Unuma, M., and Amakawa, K., 1992. Path Planning and Its Application to Human Animation System. *Computer Animation 1992*, 163--175.
17. Wang, P.K.C., 1991. Navigation Strategies for Multiple Autonomous Robots Moving in Formation. *Journal of Robotic Systems*, 8(2):177--195.
18. 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.
19. Veherencamp, S., 1987. *Handbook of Behavioral Neurobiology, Volume 3: Social Behavior and Communication*, P. Marler and J. G. Vandenbergh (eds.), Plenum Press: New York, NY, 354--382.