Control of Complex, Physically Simulated Robot Groups

David C. Brogan
University of Virginia
Charlottesville, Virginia

March 20, 2006

Abstract

Actuated systems such as robots take many forms and sizes but each requires solving the difficult task of utilizing available control inputs to accomplish desired system performance. Coordinated groups of robots provide the opportunity to accomplish more complex tasks, to adapt to changing environmental conditions, and to survive individual failures. Similarly, groups of simulated robots, represented as graphical characters, can test the design of experimental scenarios and provide autonomous interactive counterparts for video games. The complexity of writing control algorithms for these groups currently hinders their use. A combination of biologically inspired heuristics, search strategies, and optimization techniques serve to reduce the complexity of controlling these real and simulated characters and to provide computationally feasible solutions.

Keywords: Multiagent, dynamic simulation, group navigation, herds, swarms

1   Introduction

Animated characters are needed to play the role of teachers or guides, teammates or competitors, or just to provide a source of interesting motion in virtual environments. The characters in a compelling virtual environment

Figure 1: Images of 105 simulated one-legged robots and 6 simulated bicycle riders.

must have a wide variety of complex and interesting behaviors and must be responsive to the actions of the user. The difficulty of constructing such synthetic characters currently hinders the development of these environments, particularly when realism is required. In this paper, we describe one approach to populating graphical environments, using dynamic simulation to generate the motion of characters (Figure 1).

Motion for characters in virtual environments can be generated with keyframing, motion capture, or dynamic simulation. All three approaches require a tradeoff between the level of control given to the animator and the automatic nature of the process. Animators require detailed control when creating subtle movements that are unique or highly stylized. Generating expressive facial animations usually requires this low level of control. Automatic methods are benecial because they can interactively produce motion for characters based on the continuously changing state of the user and other characters in the virtual environment.

Keyframing requires that the animator specify critical, or key, positions for the animated objects. The computer then fills in the missing frames by smoothly interpolating between those positions. The specification of keyframes for some objects can be partially automated with techniques such as inverse kinematics, but keyframing still requires that the animator possess a detailed understanding of how moving objects should behave over time as well as the talent to express that information through the conguration of the character. A library of many keyframed animations can be generated off-line and subsequently accessed in an interactive environment to provide the motion for a character that interacts with the user.
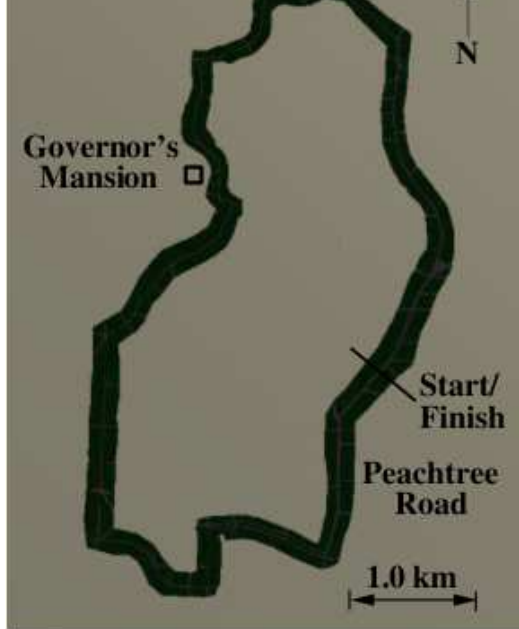
Figure 2: The 13-kilometer race course from the 1996 Olympics. This graphical course captures the elevation, side streets, and surrounding terrain of the streets from Atlanta, Georgia where the race was held.

To illustrate the use of dynamically simulated characters, we created a group of simulated human bicyclists and a group of alien bicyclists that ride on a bicycle race course (figure 2). Our earlier results indicate 1,2 that we can generate algorithms that support characters of different types and groups of varying size, however, manual tuning was required to obtain good performance. In this paper we describe automatic tuning methods and algorithms that generate improved group performance.

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 characters 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 characters.[1] As an example of this attraction, Cullen, Shaw, and Baldwin[2] 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). Group behaviors also allow animals to hunt more powerful animals than those they could overpower as individuals. The success of behaviors
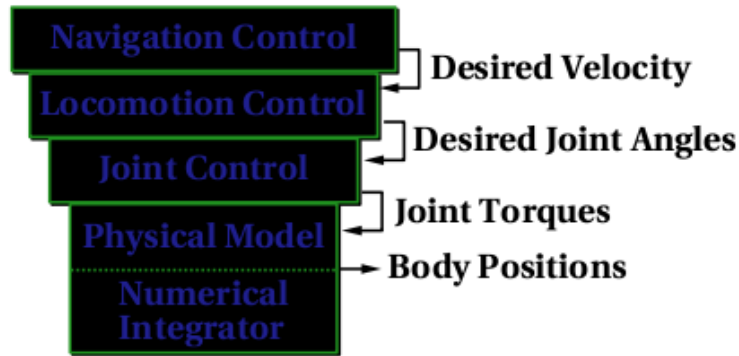
Figure 3: The simulation system used to animate physically simulated characters. The locomotion controller obtains a desired velocity from the navigation controller and computes the joint positions that will achieve it. The desired joint positions are passed to the joint controller, where joint torques are computed to eliminate errors in joint position. The joint torques are used by the numerical integrator to compute new positions for all the character's body parts.

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 characters in virtual environments.

3   Bicycle Simulation Locomotion Controller

The bicyclists in these groups consist of three components: physical simulation, locomotion controller, and navigation controller (figure 3). The physical simulation is dened by equations of motion that represent a hierarchy of rigid body parts and the rotary and telescoping joints that connect them. The equations of motion for the bicyclist were formulated using a commercially available package.[3] A character's locomotion control ler computes how to actuate its joints in order to move at a specified desired velocity. Due to kinematic and dynamic constraints, the locomotion controllers cannot instantaneously eliminate errors between a character's desired velocity and its actual velocity. These limitations to a character's maneuverability, or mobility constraints, are realistic and intuitive to the user, but they make it more difficult for navigation controllers to compute a desired velocity for each character that accomplishes group behaviors, obstacle avoidance, and path following.

The locomotion control system adjusts the speed of the bicycle by moving the bicyclist's legs to produce a torque at the crank. The desired torque at the crank is $\tau = k(v - v_b)$, where k is a gain, v is the magnitude of the velocity, and v_d is the desired velocity (specified by the navigation controller). 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 :

$$w = \frac{\sin(\theta_c) + 1}{2} \qquad (1)$$

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

$$f_l = \frac{\omega \tau_c}{l} \qquad (2)$$

where l is the length of the crank arms.

4 Bicycle Simulation Navigation Controller

The bicyclists in these groups consist of three components: physical simulation, locomotion controller, and navigation controller (Figure 3). The physical simulation is dened by equations of motion that represent a hierarchy of rigid body parts and the rotary and telescoping joints that connect them. The equations of motion for the bicyclist were formulated using a commercially available package.12 A characters locomotion control ler computes how to actuate its joints in order to move at a specied desired velocity. Due to kinematic and dynamic constraints, the locomotion controllers cannot instantaneously eliminate errors between a characters desired velocity and its actual velocity. These limitations to a characters maneuverability, or mobility constraints, are realistic and intuitive to the user, but they make it more dicult for navigation control lers to compute a desired velocity for each character that accomplishes group behaviors, obstacle avoidance, and path following.

4.1 Computing Desired Positions

In this stage of the navigation control algorithm, a set of desired positions is computed for each individual in the group. These desired positions correspond to the herding and path following goals of the navigation controller. Each character computes a desired position relative to its visible neighbors that preserves a close grouping while avoiding collisions. Each individual would be in the perfect position according to the grouping goals of the navigation controller if it could move to this position instantaneously. However, each character must also follow the path, and a desired position is computed that satisfies this goal. In this subsection we describe the algorithms that specify these two desired positions.

4.1.1 Desired position relative to neighbors

Each character's desired position algorithm uses the positions of the n visible neighbors (Figure 6) to compute a desired position, $(x_{d(i)}, y_{d(i)})$, relative to each of these characters. This position is a distance D away from visible character, i, on the line between the two characters (Figure 6):

$$y = \frac{y_i - y_A}{x_i - x_A} x, \tag{3}$$

where (x_i, y_i) is the current position of character i, and (x_A, y_A) is the current position of character A.

4.1.2 Desired position relative to path

In addition to avoiding collisions with other individuals in the group, the characters must also ride along a path. The path is constrained to be on a plane, but it is free to twist left and right. The center of the path is modeled as a Catmull-Rom spline and the path edges are 5.0 meters to either side of the center.

4.2 Computing Desired Velocities

The desired velocity algorithm must combine a characters desired positions, current state, neighbor velocities, and the user-specied nominal velocity to compute an appropriate desired velocity. This desired velocity must preserve the character's balance while eectively moving towards the desired positions.

Therefore, the navigation control algorithm must take into account each characters physical qualities and locomotion controller. For example, human and alien bicyclists possess dierent mass distributions, which in turn affect how these characters must compute velocities to eliminate position errors. The navigation controller requires that the desired velocity algorithms be tuned to account for unique attributes of each character.

4.2.1 Desired velocity for herding

The desired velocity algorithm first determines a desired riding direction and speed that will eliminate the error in position, (e_x, e_y), between the character's current position and the desired position relative to its neighbors, $(herd_x, herd_y)$:

$$e_x = herd_x - x \qquad e_y = herd_y - y \qquad (4)$$

The character's desired riding direction is:

$$herd_{yaw} = \arctan \frac{\dot{y} + k_p e_y + k_d \dot{e}y}{\dot{x} + k_p e_x + \dot{e}_x}, \qquad (5)$$

where $(\dot{x}, \dot{y})$ is the characters current velocity and the spring gain, k_p, and damping gain, k_d, are multiplied by the character's error in position and the velocity of the position error, respectively.

5 Results and Discussion

Results reported elsewhere[1, 2] indicate that these navigation controllers are robust to group size and character type. The controllers generate groups of human bicyclists as they avoid obstacles and ride along a path. The results verify that distributed navigation controllers can be used with the locomotion controllers of physically simulated characters to generate goal-oriented behavior. However, the unique dynamic abilities of each character impose limitations on the characters ability to react to the environment. The navigation controllers described in this paper partially account for these variations by performing simulated annealing tuning experiments for the herding gains and the path following look-ahead time. Additional manual adjustments must be made to prevent the navigation controllers from specifying unreasonable desired velocities to the locomotion controllers.

$$\sqrt[3]{\frac{x + y}{e = mc^2}}$$

The equation above is the root of the universe.

Appendix A. Tables

Table 1: 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)		
Torso to neck				0.012	0.0	0.32
Torso to waist				0.012	0.0	-0.32
Head	-0.009	0.0	-0.064			

Table 2: Human model's rigid-body parameters. The moments of inertia are computed about each link's COM

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
Upper Leg	1.04	8.35	0.15	0.16	0.025

References

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

[2] D. C. Brogan, R. A. Metoyer, and J. K. Hodgins, "Dynamically simulated characters in virtual environments", *IEEE Computer Graphics & Applications 18*, pp. 58-69, September/October 1998.

[3] D. E. Rosenthal and M. A. Sherman, "High performance multibody simulations via symbolic equation manipulation and kane's method, *Journal of Astronautical Sciences 34(3)*, pp. 223-239, 1986.

[4] John December and Neil Randall, The World Wide Web Unleashed, Sams Publishing, 1994.

[5] Helmut Kopka and Patrick W. Daly, A Guide to LATEX, Addison-Wesley Publishing Co., Inc., 1993.

[6] Nikos Drakos and Ross Moore, LaTeX2HTML Translator Version 99.2 beta8(1.43), Macquarie University, Sydney, 1999.

[7] Walker, Janice R. et al., "The Columbia Guide to Online Style", 1995. http://www.columbia.edu/cu/cup/cgos/idx_basic.html (August 11, 2000)