Simulation Level of Detail for Multiagent Control

David C. Brogan[†]
University of Virginia
dbrogan@cs.virginia.edu

Jessica K. Hodgins[†]
Carnegie Mellon University
jkh@cs.cmu.edu

ABSTRACT

Many classes of applications require multiagent navigation control algorithms to specify the movements and actions of heterogeneous groups containing thousands of characters. The scale and complexity of these interacting character groups require navigation control algorithms that are both generalizable and specifically tuned to particular character platforms. We propose a technique called simulation level of detail (LOD) that provides a simulation-based interface between navigation control algorithms and the specific mobile characters on which they operate. A simulation LOD efficiently models a character's ability to move given its dynamic state and provides this simplified version of the character to navigation controllers for use in run-time search algorithms that compute locomotion actions. We develop our simulation LOD algorithms on groups of physically simulated human and alien bicyclists and demonstrate reusable controllers that provide improvements in path following and herding tasks.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

General Terms

Algorithms

Keywords

Mobile agents, multiagent simulation, path planning

1. INTRODUCTION

Continued reductions in the cost of physical robots and simulation platforms permit the development of systems containing thousands of interacting characters. As character groups attain such large sizes, the number of potential character interactions scales exponentially and provides both a greater opportunity to coordinate and a costlier penalty for interference. When the characters are mobile, navigation control algorithms must utilize available actuators to move each character through the environment effectively. Due to the diversity of physical environments and character locomotion abilities it is currently difficult to develop navigation control algorithms that generalize across characters or physical settings. Yet we believe such navigation strategies as *move-to-goal* and *avoid-collision* are reusable provided they are adjusted to suit the physical capabilities of each character. For example, the drivers of buses and automobiles share common navigation goals and driving strategies when on the freeway, but the navigation control algorithms used by these two types of drivers are carefully adjusted to suit the locomotion capabilities of their underlying vehicles. The bus, for example, may move towards an exit lane earlier than a car due to its inability to quickly accelerate into a gap in traffic at the last moment. Similarly, a bicyclist may avoid an obstacle differently if he is leaned over in a turn or riding straight. To accomplish both improved character control and reusable navigation controllers, we propose a technique called simulation level of detail (LOD) that provides an interface between navigation control algorithms and the specific mobile characters on which they operate.

A simulation LOD is a simplified version of a physically simulated character that serves as a computationally efficient alternative. To improve character movements, the simulation LOD must supply the navigation controller with data regarding the feasibility of available actions. An action's feasibility is a complex relationship between the character's kinematics and dynamics and its state in the environment. With this feasibility information, navigation controllers will not attempt to accomplish high-level goals that

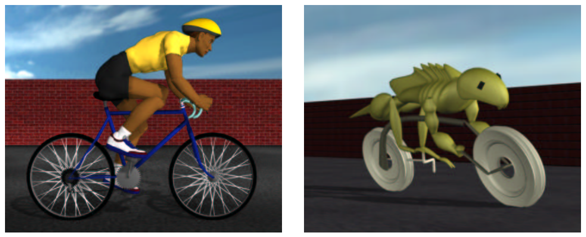
Figure 1: Physically simulated human and alien bicyclists.

[†]Research conducted while at GVU Center, Georgia Tech

require impossible movements and the realization of feasible goals will be more efficient. Furthermore, simulation LODs create a common interface between a navigation controller and many characters, thus allowing characters with diverse kinematic and dynamic properties to be used interchangeably. To demonstrate the generalizability of this technique, we created simulation LODs for human and alien bicyclists, each of which has distinct locomotion capabilities (figure 1).

This paper continues the development of our earlier systems, which generated groups of characters that could move as a group and avoid obstacles [2, 3]. The characters in these groups consist of three components: navigation controller, locomotion controller, and physical simulation. The navigation controller, which is the focus of this paper, must generate a desired velocity that causes each character to move in a way that accomplishes high-level goals. The desired velocities are used by locomotion controllers to compute joint torques, which are then integrated by the physical simulation to accomplish desired limb movements.

A character's locomotion controller computes how to actuate its joints in order to move at a specified desired velocity. The locomotion controller eliminates errors in the bicyclist's velocity by adjusting the facing direction and speed of the bicycle. The controller adjusts facing direction by applying forces to the handlebars with the hands and controls speed by applying forces to the pedals with the feet.

The physical simulation is defined by equations of motion that represent rigid body parts and the rotary and telescoping joints that connect them. The human bicycle rider is modeled by a 12-segment rigid-body model connected by rotary joints with 17 controlled degrees of freedom (DOFs). The alien bicycle rider is modeled by a 15-segment model with 27 DOFs. For both characters, some joints, like the knee, are modeled as a single-axis pin joint; others, like the waist and shoulder, are modeled by 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 graphical bodies. Density data obtained from the anatomical literature [6] were used in calculating the dynamic properties of the body segments.

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.

We hand crafted our early navigation controllers to anticipate a character's mobility constraints and to compute a desired velocity that achieves high-level goals given the available actions. The size and complexity of these early systems were constrained by the manual tuning required by the navigation controllers. Our previously published navigation control algorithms contain very simple representations of the bicyclists' mobility constraints and the characters are limited to smooth, gradual actions to do anything. Not only do these navigation algorithms produce overly cautious maneuvers but they also compute the same actions for all characters, independent of their size, shape, or dynamic state.

In this paper, we address these problems by constructing simulation LODs that better capture the subtle relationships between a character's mobility constraints and the goals of its navigation controller. Our simulation LOD models a character's locomotion capabilities and limitations by predicting the translation of its center of mass (COM) along the ground in response to such navigation control actions as turning. Because there are infinite turning actions and initial dynamic states for the bicyclists, we automatically build the simulation LOD through a series of off-line trials that sample the character's ability to move through the environment. This simulation LOD improves character control by encapsulating the information required by navigation controllers to model the character's mobility constraints and compute desired velocities. We will describe how to construct this simulation LOD, how to effectively use it to improve navigation control, and test the reusable navigation controller performance on single-character path following and herding tasks.

2. BACKGROUND

Multiagent robotic research demonstrates that models of mobility constraints can improve the performance of robots moving in formation. Wang modeled robots as point masses and proved the asymptotic stability of multiple simulated robots moving in formation [14]. Despite the simplicity of this simulated system, the provably stable group formation demonstrates the advantages of locomotion modeling. Tan and Lewis developed control algorithms for groups of nonholonomic simulated robots moving in formation [13]. A model of the robot's dynamic abilities estimates the positions each robot can reach at the next time step and provides a penalty value for all spots the robot cannot reach. The evaluation function uses this model of a robot's locomotion capabilities when evaluating the numerous translations and rotations that could be applied to the robots' formation.

Stone and Veloso developed a reinforcement learning approach to training simulated autonomous robosoccer players [12]. Because the state space is prohibitively large for run-time evaluation, the authors use pre-game, off-line experimentation to develop a model of each robot's ability to pass and intercept the ball. Similar off-line tests generate simplified simulation LODs for the bicyclists in this paper and provide navigation controllers with necessary data about their locomotion capabilities.

In computer graphics, Carlson and Hodgins applied simulation LOD techniques to a graphical environment populated with multiple, physically simulated one-legged robots [4]. Three simulation LOD models represent varying animation qualities: point-mass model with no animated degrees of freedom, a point-mass model with kinematically animated degrees of freedom, and a fully simulated version. Higher-quality LODs are reserved for characters at very dynamic moments, as when avoiding or experiencing collisions, and those near the viewer in the field of view. They demonstrated that a group of characters that dynamically switches between LODs can sufficiently replicate the performance of a fully simulated group.

Faloutsos et al. developed a system that accomplishes complex tasks by evaluating multiple controllers automatically and selecting an appropriate sequence [7]. The authors use an off-line machine learning technique to build a database that models the effectiveness of each controller for a character in a given state. Runtime queries of the database identify controllers that best suit the current state of the character and its desired final state. The database of controllers is similar to our simulation LODs because it supports the evaluation, comparison, and selection of controllers. However, the simulation LODs store more controllers and switch between them more frequently.

Additional graphics researchers are investigating ways to simplify physical simulations to reduce computation costs. Grzeszczuk et al. developed a technique that uses neural network approximations to emulate physically simulated characters' equations of motion [9]. After the neural network is trained to sufficiently model the original system, it can produce motions more efficiently than a full dynamic simulation. O'Sullivan and Dingliana investigate the opportunities to replace simulated particles with simplified counterparts when imperceptible to the viewer [11]. In particular, they find inaccurate dynamics are less noticeable in the peripheral vision and when the movements are complex. O'Brien et al. describe a method to automatically simplify particle simulations through clustering into spatially localized groups [10]. Chenney et al. describe ways to approximate the motion of periodic motions [5]. These papers present opportunities and methods to use simulation LOD for graphical environments, but they do not integrate the simplified systems with the controllers of the fully simulated versions.

3. CREATING A BICYCLIST SIMULATION LOD FOR CONTROL

A simulation LOD must reproduce the mobility constraints of each fully simulated character and provide a common interface between diverse character types and the navigation controller. Integrating the equations of motion of a fully simulated character for δt seconds generates the changes in position and velocity of all the DOFs that result from external forces and a navigation control action:

$$state_{t+\Delta t} = sim_{full}\left(state_t, action, \Delta t\right). \qquad (1)$$

Because the simulation LOD need only supply navigation controllers with information about a character's mobility constraints, it only generates the (x, y) trajectories of the COM:

$$(x_{t+\Delta t}, y_{t+\Delta t}) = sim_{LOD}\left(state_t, action, \Delta t\right). \qquad (2)$$

To satisfactorily represent a fully simulated character, the simulation LOD must be configurable to match a character's run-time state and must accurately predict how any navigation control action may affect the trajectory of the fully simulated character's COM. Because the simulation LOD will be used to generate many COM trajectories from the available navigation control actions, it must be computationally simple.

Unfortunately, there is no analytic way to simplify the character's simulation algorithms to generate only the COM trajectory instead of the entire body's trajectories. The system must compute the COM trajectory by applying the control action to the fully simulated character's control algorithms and equations of motion. The computational cost of the bicyclist characters prohibits executing such simulations at run time, but off line we conduct many simulation experiments that are stored in a database as part of the simulation LOD. This database, or *mobility map*, is constructed by applying each member of a discrete set of navigation control actions (turn left or turn right by x rad) to a fully simulated character initialized in multiple initial states.

The size of the mobility map is equal to the number of discretized navigation control actions multiplied by the number of initial states. To minimize the size of the mobility map, we wish to select initial states that characterize as wide a range of the fully simulated character's states as possible. In particular, we must select initial states that represent sources of a character's physical limitations, its mobility constraints.

Mobility constraints limit the ways a character can navigate through a simulated world. Fundamentally, however, the restrictions on a character's movements derive from kinematic relationships between its body parts, the bodies' kinetic states, and the character's ability to perform actions that add energy to the system. An understanding of a character's physical abilities and limitations permits identification of initial states that greatly affect LOD accuracy. In the case of the human and alien bicyclist models, we have eliminated many degrees of freedom and identified the subset of state parameters that significantly affect each character's COM trajectory and therefore require explicit modeling in the mobility map.

3.1 Reduce State Parameters

Many degrees of freedom bear little relationship to the character's ability to navigate and we can use the same COM trajectory model independent of their values. Cosmetic DOFs, such as the bicyclist's neck, have little effect on the riding abilities of the bicyclist (assuming we ignore issues related to the bicyclist's perception). Similarly, DOFs with nearly static values do not require models in the mobility map. The mobility map, therefore, does not include initial states that vary the values of the wrist and waist angles or the wheel velocities.

Many DOFs are vital to the movement of the character, and their values are not constant, but their particular values do not affect the character's ability to move through the environment. Due to the symmetry of the bicycle wheels, their angular rotation about the axle does not influence the characters' ability to ride. Similarly, the homogeneity of the simulated world causes the bicyclists' locations to have no effect on performance.

Degrees of freedom can also be eliminated when they move in unison with others. A bicyclist's elbow angle, for example, changes as the bicyclist turns the handlebars. However, the bicyclist's elbow is part of a closed kinematic chain that is formed by the frame of the bike, the pelvis, the torso, the upper arm, the lower arm, the hand, and the handlebars. This closed kinematic chain allows us to discard all upper body DOFs and include a single *steering angle* DOF. The bicyclist's legs are connected to the bicycle at the hip and at the feet. Inverse kinematics define the bicyclist's hip and knee angles as a function of the feet positions, which are dependent only on the crank angle. These kinematic chains may be under constrained, but we make assumptions that allow us to compute inverse kinematic solutions that mimic the fully simulated character. For example, the y-axis of the shoulder is not used and joint limits prevent the bicyclist from bending its knees backwards.

Through an iterative process of elimination, DOFs are eliminated from the set that will be used to create the initial states for the mobility map. In the case of the two bicyclist simulations, the only two DOFs that are included in the

DOF	Removal Reason	DOF	Removal Reason
x	Homog. World	$\dot{x}$	Homog. World
y	Homog. World	$\dot{y}$	Homog. World
z	$f(\theta_{roll})$	$\dot{z}$	$f(\dot{\theta}_{roll})$
θ_{roll}	Not removed	$\dot{\theta}_{roll}$	$f(\theta_{roll}, \dot{\theta}_{yaw})$
θ_{pitch}	Constant	$\dot{\theta}_{pitch}$	Constant
θ_{yaw}	Homog. World	$\dot{\theta}_{yaw}$	Not removed
θ_{fwheel}	Symmetrical	$\dot{\theta}_{fwheel}$	Constant
θ_{rwheel}	Symmetrical	$\dot{\theta}_{rwheel}$	Constant
θ_{crank}	Symmetrical	$\dot{\theta}_{crank}$	Constant
θ_{fork}	$f(\theta_{roll})$	$\dot{\theta}_{fork}$	$f(\theta_{roll}, \dot{\theta}_{yaw})$
θ_{pelvis}	Constant	$\dot{\theta}_{pelvis}$	Constant
θ_{waist}	Constant	$\dot{\theta}_{waist}$	Constant
θ_{wrist}	Constant	$\dot{\theta}_{wrist}$	Constant
θ_{elbow}	$f(\theta_{fork})$	$\dot{\theta}_{elbow}$	$f(\dot{\theta}_{fork})$
θ_{shldr}	$f(\theta_{fork})$	$\dot{\theta}_{shldr}$	$f(\dot{\theta}_{fork})$
θ_{hip}	$f(\theta_{crank})$	$\dot{\theta}_{hip}$	$f(\dot{\theta}_{crank})$
θ_{knee}	$f(\theta_{crank})$	$\dot{\theta}_{knee}$	$f(\dot{\theta}_{crank})$
θ_{ankle}	$f(\theta_{crank})$	$\dot{\theta}_{ankle}$	$f(\dot{\theta}_{crank})$

Table 1: The simulation level of detail reduces the DOFs of the bicyclists to just two: θ_{roll} and $\dot{\theta}_{yaw}$. Each DOF in the above table is removed for one of the following reasons: constant, homogeneous world, symmetrical, or it is a function of another DOF.

mobility map state parameter vector are the roll angle, θ_{roll} and the derivative of the yaw angle, $\dot{\theta}_{yaw}$.

3.2 Select Initial States for Testing

We have reduced the number of parameters that may affect the COM trajectory models stored in the mobility map from the original 57 state parameters (67 for the alien bicyclist) to two, θ_{roll} and $\dot{\theta}_{yaw}$. We now explore the relationship between these two variables and the behavior of the bicyclists' COMs.

We need not build a mobility map that is able to model the effect of all values of the θ_{roll} and $\dot{\theta}_{yaw}$ variables on the COM trajectory. Instead, we need only conduct experimental trials for values that span the space of θ_{roll} and $\dot{\theta}_{yaw}$ values observed at run time. We can analytically determine the valid range of θ_{roll} values (leaning too far to the side causes the bicyclist's pedal to hit the ground, for example), but the range of possible values for $\dot{\theta}_{yaw}$ is less clear. Instead, we observe $\dot{\theta}_{yaw}$ during a comprehensive set of experiments to determine the maximum and minimum values the bicyclists can accomplish.

We need to independently vary the θ_{roll} and $\dot{\theta}_{yaw}$ variables within their respective ranges to identify their effect on the COM trajectory. Unfortunately, it is difficult to perturb a fully simulated bicyclist state vector to match desired test θ_{roll} and $\dot{\theta}_{yaw}$ values. Not only are these two variables coupled by the dynamics of the system, but changing their values also requires simultaneously adjusting most other variables. Instead, we sample the simulation as it executes a series of dynamic maneuvers, attempting to ensure that the sampled system states span the space of the

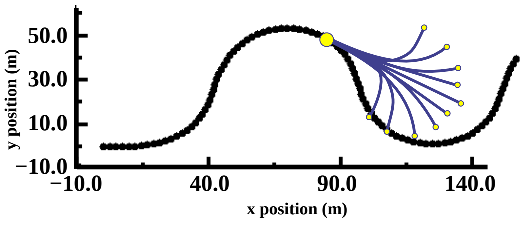

Figure 2: In this graph we depict how the turning experiments are initialized with the state of the bicyclist as sampled from a sinusoidal trajectory.

θ_{roll} and $\dot{\theta}_{yaw}$ that we anticipate in the run-time simulated environment. Our experience with the simulated bicyclists indicates their riding limitations occur when leaned far to the left and right, and also when turning quickly and slowly. To reproduce these dynamic conditions and to generate the initial states, the bicyclists are commanded to follow sinusoidal trajectories at a nominal speed of 12.0 m/s (human bicyclist) and 6.7 m/s (alien bicyclist). The states of the bicyclists are sampled at 100 evenly spaced intervals along one period of these trajectories.

3.3 Conduct Mobility Tests

The initial state vectors obtained by sampling the sinusoidal trials are used as the initial conditions in a series of mobility tests. Each mobility test specifies a control action, to change the desired riding direction in the range of $-\pi$ to π. Only a small subset of these potential actions can be accomplished by a bicyclist in a particular initial state because extreme changes in desired riding direction cause the locomotion controller of the bicyclist to fail and lose balance. A trial and error process is used to find maximum and minimum changes in desired riding direction that are successful for most initial conditions. We then generate ten turns by uniformly sampling within these endpoints to model the less aggressive control inputs. The human bicyclist simulation is tested with changes in desired riding direction in the amounts of 0.12, 0.35, 0.58, 0.82, and 1.0 rad to the left and to the right. The alien bicyclist is less robust to instantaneous changes in desired riding direction and turn values of 0.03, 0.89, 0.15, 0.2, and 0.26 rad were used.

We populate the mobility map with COM trajectories by conducting ten turning experiments for each of the 100 initial states sampled from the sinusoidal trajectory (figure 2). Each simulation executes for 5.0 s (or until the failure if the character should lose balance and fall), which provides sufficient time for the simulations to complete the transition from an initial state to the desired riding direction (figure 3). The state information recorded during the experiment, the (x, y) position of the COM, is stored in the mobility map as cubic splines fit to the time-sampled data.

4. USING A BICYCLIST SIMULATION LOD FOR CONTROL

Every 0.1 s, the navigation controller specifies a desired velocity for the run-time character. This desired velocity satisfies the path following goals of the character as well as the herding goals. The navigation controller requires three steps to use the simulation LOD to generate a desired velocity for path following and herding. First, the simulation

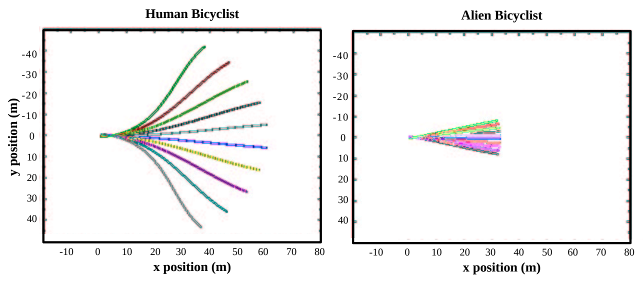

Figure 3: These two graphs plot the (x, y) trajectories that result from a set of 10 turning experiments. The human bicyclist was instructed to execute turns ranging from ± 1.0 rad and the alien bicyclist from ± 0.26 rad. All experiments lasted 5.0 s.

LOD must identify examples of discretized initial states in its mobility map that approximate the current state of the run-time simulation. It must then evaluate the corresponding COM trajectories that result from executing different control actions. Lastly, the simulation LOD interpolates between the two control actions with the most desirable COM trajectories to generate an appropriate change in desired riding direction for the navigation controller. In this manner, the navigation controller computes desired velocities that match each character's abilities with its high-level goals.

4.1 Initialize Simulation LOD State

The first step in using the simulation LOD for navigation control requires that the system search the set of precomputed experiments in the mobility map database and extract those with initial states that are similar to the run-time simulation state. The similarity between an example in the database and the current simulation state is defined by the weighted distance between the relevant components of the two state vectors:

$$\text{distance} = \sum_{s=\theta_{roll}, \dot{\theta}_{yaw}} w_s \cdot (x_{lod}[s] - x_{full}[s])^2, \qquad (3)$$

where x_{lod} is the initial state of a sample from the mobility map, and x_{full} is the current simulation state. The weights, w_s, are used to normalize the ranges of the state vector parameters:

$$w_s = \frac{w_s}{(s_{max} - s_{min})}, \qquad (4)$$

where s_{max} and s_{min} are the largest and smallest values of state parameter s in the entire set of navigation experiments. The navigation controller selects the initial state in the database with the smallest distance from the current simulation state.

The selected initial state indexes into the mobility map and provides a set of ten COM trajectories that result from applying a range of control actions to the bicyclist (figure 3). All these trajectories are rooted at the origin and oriented in the direction of the positive x-axis. To evaluate the desirability of each tested control action, these COM trajectories must be mapped to the bicyclist's local coordinate system. This is a simple transformation because of the homogeneous ground surface.

4.2 Using Simulation LOD Actions

For a bicyclist in a particular state, the simulation LOD provides ten COM trajectories produced by executing different actions. These COM trajectories are used by the navigation controller to evaluate available control actions. The bicyclist navigation controllers we consider here must accomplish two unique tasks, to follow trajectories and to shift their location sideways with only a momentary disruption of the preferred riding direction. The former allows a bicyclist to ride a race course and the latter permits bicyclists to adjust their position relative to neighbors that may be too close or far away. In both cases, the ten COM trajectories must be evaluated to select one navigation control action, a new desired velocity.

The navigation controller uses two metrics to evaluate how well the bicyclists are following a path. The distance between the bicyclist and the path reflects an error in position while the difference between a bicyclist's riding direction and the tangent to the path indicates an error in the bicyclist's θ_{yaw}. Each of the COM trajectories generated by the simulation LOD is compared to the path and evaluated by integrating the weighted position and yaw errors over the duration of the 5.0 s trial (figure 4). In these experiments, we discretize the integration and generate a weighted sum of position and yaw errors, e_{pos} and e_{yaw}:

$$eval = \sum_{t=0}^{duration} (w_{pos}(t) \cdot e_{pos}(t) + w_{yaw}(t) \cdot e_{yaw}(t)). \qquad (5)$$

The weights, $w_{pos}(t)$ and $w_{yaw}(t)$ account for the relative importance of position versus yaw errors and also increase the importance of eliminating errors at different times during the trial. For all values of t, we constrain the sum of the two weights to be 1.0 and solve for one variable, *ratio*; w_{pos} will be set to *ratio* and w_{yaw} will be set to $1.0 - ratio$. We further simplify the *eval* equation by discretizing *ratio* to have a value of 0.0 for $t > t_{max}$.

The experiments that are used to tune these two parameters for path following require the bicyclists to repeatedly track seven curves of different curvatures. Simulated annealing is used to automatically adjust the parameter values, t_{max} and *ratio*, to minimize the average of the distance and yaw errors that accumulated during the sequences. The results for the two systems are presented in table 2. The experiments used to tune these parameters for herding require the bicyclists to quickly shift 4.0 m to the right while preserving the riding direction (similar to a lane shift when driving down the highway). The characters are evaluated for their ability to minimize fork velocities, lateral velocities, incorrect riding direction, and errors relative to the goal position.

After all the COM trajectories of the candidate turning experiments have been evaluated for their distance and yaw errors, a single desired velocity is computed. The two navigation control actions, a and b, stored in the simulation LOD that generate COM trajectories with the smallest errors, $eval_a$ and $eval_b$, are combined with a weighted average to specify a new desired riding direction. The average is weighted based on the distance between each COM trajectory and the desired position at time t_{max}. The desired

Character	Path Following		Herding	
	t_{max}	*ratio*	t_{max}	*ratio*
Human Bicyclist	1.3	0.28	1.5	0.30
Alien Bicyclist	0.82	0.84	2.4	0.39

Table 2: The parameters used to define the weighting functions that evaluate COM trajectories in the path following and herding navigation control algorithm.

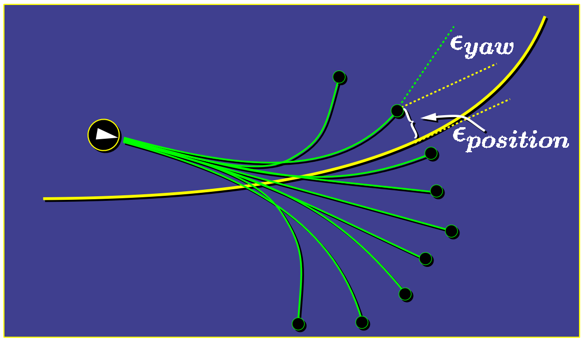

Figure 4: The evaluation of the simulation LOD trajectories with respect to following a path. The ten LOD trajectories are translated to the current position of the character and rotated to align with its facing direction. The trajectories are evaluated for their errors in position and yaw.

riding direction, $yaw_{desired}$, is multiplied by the nominal desired speed of each bicyclist to compute a desired velocity.

5. EXPERIMENTAL RESULTS

We compare the performance of navigation controllers that use simulation LODs to those that do not. The human and alien bicyclist characters are used in a pair of experiments to test an individual character's ability to follow a path and a group's ability to form a stable formation. In all of these experiments, the navigation controllers of the two bicyclists were called every 0.1 s to generate new desired velocities. At the beginning of each experiment, the bicyclists are initialized in a stable configuration with a speed of 12.0 m/s for the human bicyclist and 6.7 m/s for the alien bicyclist. The "simple" labels on the following graphs refer to a navigation controller like the one described in our earlier papers [2, 3], which does not utilize a simulation LOD.

5.1 Individual Path Tracking Experiment

In these experiments, seven curves of varying curvatures were used to explore the range of character turning abilities (figure 5). Figure 6 presents the comparison of a human bicyclist with and without simulation LOD and demonstrates that the human bicyclist without simulation LOD incurs a linearly increasing error as the magnitude of the corner increases. The navigation controller that uses simulation LOD, however, is better able to track the entire range of paths. The human bicyclist character has a relatively high COM and the character is not able to quickly lean into a corner. However, once the character has begun to turn, it is able to quickly lean into a more aggressive corner. The

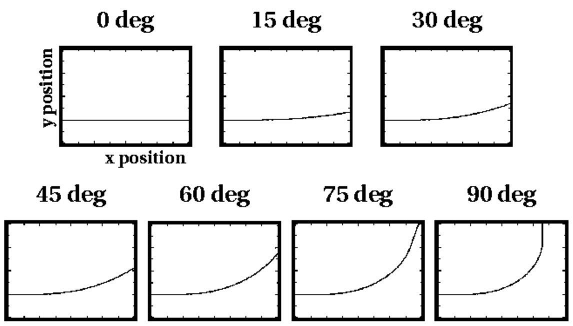

Figure 5: A set of seven curves used to evaluate the bicyclists' ability to follow a path. The lengths of these curves are scaled to equal the distance traveled by the human and alien bicyclists during a 7.0 s trial, 49.0 m long for the human bicyclist and 47.18 m for the alien. Additional straight segments are attached to the beginning and ending of the corners to provide continuity for derivative or look-ahead components.

simulation LOD provides the human bicyclist's navigation controller with the data that reveals this relationship and results in better performance.

Figure 7 presents a similar comparison for the alien bicyclist. Both navigation controllers allowed the alien character to complete the seven corners with much smaller errors than the human bicyclist perhaps because of the lower overall speed achievable by the alien bicyclist. Use of the simulation LOD decreased performance for paths with little curvature, but slightly increased performance for paths with high curvature.

5.2 Convergence to a Group Experiment

A second experiment tests the ability of a group of characters to readjust their positions relative to one another to find a stable configuration for the herd. The sixteen characters are initially scattered in random positions, facing in the same direction, within a 20 m square region. The herding navigation control algorithm computes desired positions for each character that attempts to bring them within 5 m of one another. The nominal riding direction and speed for the characters remain unchanged. The duration of the human and alien bicyclist experiments were the time required to form a stable group configuration, 120 s for the humans and 60 s for the aliens. Figure 8 demonstrates the configuration of a group of human bicyclists before and after the herding experiment. The two images have the same scale, so differences in spacing indicate the amount characters moved closer together.

The results from the human bicyclist herding experiment indicate that the navigation controller that uses simulation LOD can more quickly reach a stable formation (figure 9). Both groups of bicyclists aggressively steer to eliminate large initial position errors. The simulation LOD bicyclists, however, are able to turn more aggressively and therefore reduce the position errors more quickly. The simulation LOD is not perfect, however, and the bicyclists overshoot their mark at $t = 20$ s, which stalls their progress towards a stable formation. After recovering from the readjustment at 20 s, the

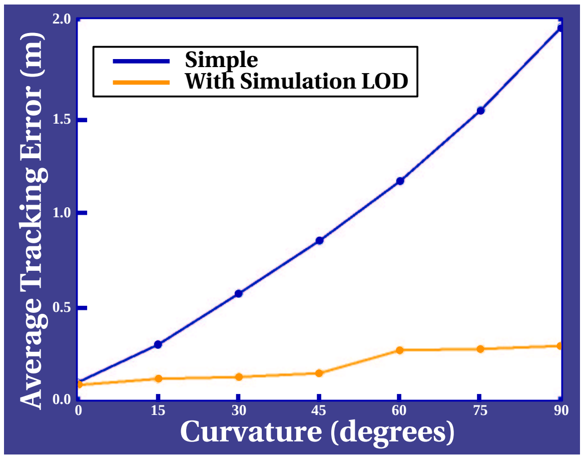

Figure 6: The results of the human bicyclist curve following experiments. The horizontal axis of the graph corresponds to the seven discrete curvatures of corners tested. The vertical axis specifies the average error between the character's COM and the path while completing the corner.

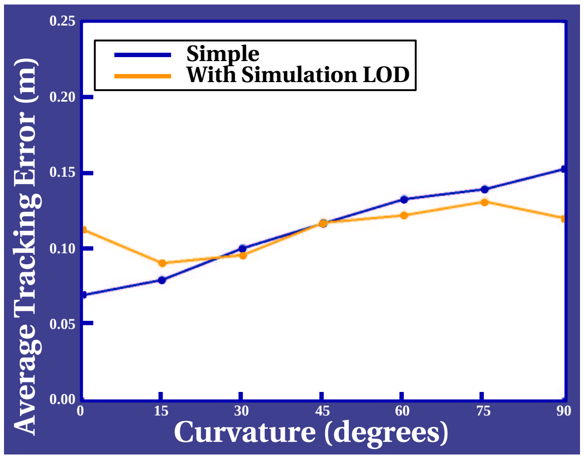

Figure 7: The results of the alien bicyclist curve following experiments.

simulation LOD bicyclists again demonstrate their ability to track more closely.

The results from the alien bicyclist herding experiment indicate a different sort of benefit of using simulation LOD (figure 10). Both the simple and the simulation LOD bicyclists have similar performance during the first 10 s of the experiment. During this period, the widely distributed characters are aggressively turning towards their desired positions. Both types of characters execute these maneuvers well. However, beyond the 10 s mark, the characters require more subtle navigation control actions to slowly settle into a stable configuration. The simple bicyclists demonstrate a large amount of oscillation in their position errors during this settling period.

These oscillations indicate an interesting relationship between one character's small overshoot errors and the overall performance of the group. The animation of the simple bicyclist herding experiment shows that the alien bicyclists

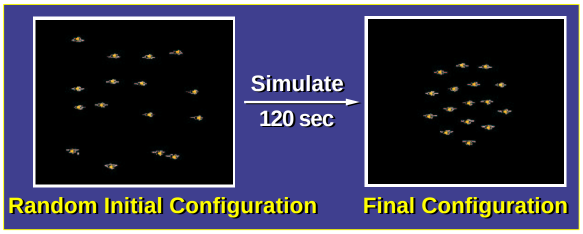

Figure 8: Sixteen human bicyclists are initialized in random positions within a 20 m square area. After 120 m, the bicyclists have stabilized in a nearly circular configuration.

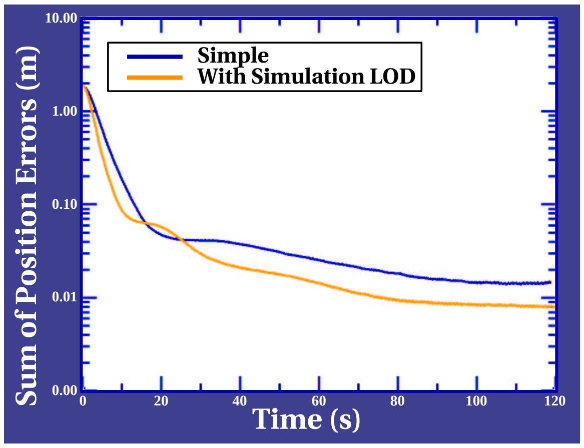

Figure 9: The results of the human bicyclist herding experiment. As time proceeds, we expect the position error to steadily drop and stabilize near zero.

frequently overshoot their desired position. After a character moves beyond its desired position, and during the period it attempts to correct the overshoot, neighboring characters adjust their desired positions away from the one that is out of position. Due to the speed with which the alien bicyclists can initiate and accomplish a change in riding direction, both the original violator and the reacting neighbors will simultaneously steer away from one another. Within moments, the bicyclists have overreacted to one another and instead created a position error by being too far apart.

This interaction explains the oscillations we see in these results. Our analysis of the character dynamics in this particular situation indicates that perhaps a simple delay would be sufficient to model a neighbor's overshoot. However, this delay would have to be adjusted dynamically to account for the variety of lean angles and turning velocities experienced by the bicyclists during execution. System dynamics are difficult to model and such tuning heuristics are less robust than the simulation LOD, which provides a better model of the alien bicyclist's mobility constraints, avoids the initial overshoot altogether, and is able to reduce these navigation control errors.

6. DISCUSSION

The frequency with which the navigation controller is called and the architecture of the mobility map are design decisions that greatly affect the performance of simulation LOD

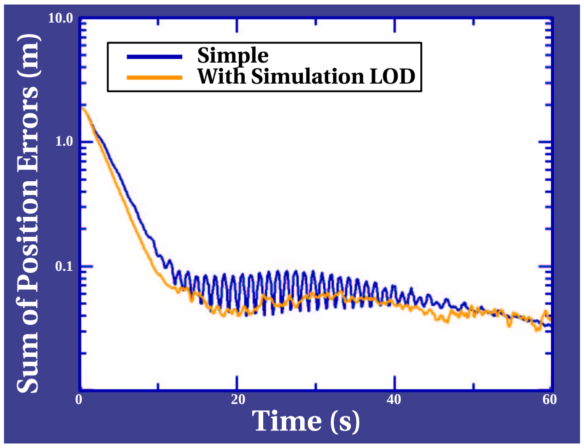

Figure 10: The results of the alien bicyclist herding experiment.

for navigation control. In these experiments, the navigation controller was executed every 0.1 s and adjustments to this value immediately affect the bicyclists' ability to follow a path and ride as a group. The mobility map we constructed is a very simple discretized representation of the fully simulated character. We consider the potential advantages of increasing the size and changing the data representation of the mobility map. We also consider the reusability of our simulation LOD method.

The navigation controller time step of 0.1 s balances different qualities of the simulation LOD to obtain good performance. Multiple time steps were evaluated and this value produced the best performance across the range of experiments. Fundamentally, the time step represents the duration each control action is executed and the frequency with which new control actions are selected. Longer time steps are successful when the navigation controller computes very good control actions that can be applied for a long time before causing undesirable errors. Shorter time steps are necessary when the computed control actions quickly result in task errors.

Due to the small number of stored initial states and control action results in the mobility map, it is not surprising that our selected navigation controller time step is short. Control actions computed by the navigation controller result from interpolating between nearest-neighbor matches, possibly resulting in significant position and θ_{yaw} errors. Interestingly, time steps shorter than 0.1 s result in decreasing performance. When a bicyclist attempts to turn left, it will first momentarily turn to the right. If the navigation control action is only used for a short duration, a left-turn action will be discontinued at a point when the bicyclist is still steering to the right. Subsequent corrective efforts will repeat this error, preventing the controller from breaking the cycle caused by the small time step. For these reasons, the navigation control time step must be carefully selected to preserve the quality of the generated actions.

The quality of the navigation control actions also relies heavily on the mobility map. In these experiments, the mobility map contains 1,000 trajectories, each 5.0 s long, that are indexed by 100 initial states and 10 tested control actions. Using this mobility map to predict future states of a bicyclist requires approximating the bicyclist's current state with one of the 100 initial states stored in the mobility map. Adding initial states to the mobility map reduces run-time approximation errors, but incurs off-line experimentation costs and storage costs. By using such data indexing methods as KD-trees, fast run-time retrieval of COM trajectories should be possible.

We are currently investigating automatic methods to determine the dimensions of the mobility map. In these bicyclist examples, we relied on domain knowledge to architect the mobility map and design off-line experiments. However, this expert intervention does not prevent the simulation LOD technique from being applied to other characters. As the dimensionality of the mobility map increases, the data creation and storage costs expand exponentially. We turn to recent computer graphics research regarding the generalization and reuse of motion capture data for inspiration. By using such techniques as principal components analysis and data clustering [1], we may be able to automatically collapse degrees of freedom while retargeting algorithms may permit the kinematic, data-driven animation [8] of these simplified components. We believe these alternative methods for animating characters will integrate nicely with our simulation LOD framework.

7. REFERENCES

[1] M. Brand and A. Hertzmann. Style machines. In *Proceedings of SIGGRAPH 2000*, pages 183–192, 2000.
[2] D. C. Brogan and J. K. Hodgins. Group behaviors for systems with significant dynamics. *Autonomous Robots*, 4:137–153, 1997.
[3] D. C. Brogan, R. A. Metoyer, and J. K. Hodgins. Dynamically simulated characters in virtual environments. *IEEE Computer Graphics & Applications*, 18(5):58–69, September/October 1998.
[4] D. A. Carlson and J. K. Hodgins. Simulation levels of detail for real-time animation. In *Graphics Interface*, 1997.
[5] S. Chenney, J. Ichnowski, and D. Forsyth. Dynamics modeling and culling. *IEEE Computer Graphics and Applications*, 19(2):79–87, March/April 1999.
[6] W. T. Dempster and G. R. L. Gaughran. Properties of body segments based on size and weight. *American Journal of Anatomy*, 120:33–54, 1965.
[7] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. *Proceedings of SIGGRAPH 2001*, pages 251–260, 2001.
[8] M. Gleicher. Rertagetting motion to new characters. In *Proceedings of SIGGRAPH 98*, pages 33–42, 1998.
[9] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of SIGGRAPH*, pages 9–20, July 1998.
[10] D. O'Brien, S. Fisher, and M. Lin. Automatic simplification of particle system dynamics. In *Computer Animation*, 2001.
[11] C. O'Sullivan and J. Dingliana. Collisions and perception. *ACM Transactions on Graphics*, 20(3), July 2001.
[12] P. Stone and M. Veloso. Team-partitioned, opaque-transition reinforcement learning. Technical report, Carnegie Mellon University, April 1998.
[13] K. H. Tan and M. A. Lewis. Virtual structures for high-precision cooperative mobile robotic control. In *IROS 1996*, November 1996.
[14] P. K. C. Wang. Navigation strategies for multiple autonomous robots moving in formation. In *Journal of Robotic Systems*, volume 8(2), pages 177–195, 1991.