A Hybrid Dynamical Systems Approach to Intelligent Low-Level Navigation

Eric Aaron[1] Harold Sun[2] Franjo Ivančić[1] Dimitris Metaxas[2]

[1] Department of Computer and Information Science, University of Pennsylvania
eaaron@graphics.cis.upenn.edu, ivancic@gradient.cis.upenn.edu

[2] CBIM Center, Division of Computer and Information Sciences, Rutgers University
hsuntn@yahoo.com, dnm@cs.rutgers.edu

Abstract

Animated characters may exhibit several kinds of dynamic intelligence when performing low-level navigation (i.e., navigation on a local perceptual scale): They decide among different modes of behavior, selectively discriminate entities in the world around them, perform obstacle avoidance, etc. In this paper, we present a hybrid dynamical system model of low-level navigation that accounts for the above-mentioned kinds of intelligence. In so doing, the model illustrates general ideas about how a hybrid systems perspective can influence and simplify such reactive/behavioral modeling for multi-agent systems. In addition, we directly employed our formal hybrid system model to generate animations that illustrate our navigation strategies. Overall, our results suggest that hierarchical hybrid systems may provide a natural framework for modeling elements of intelligent animated actors.

1 Introduction

Autonomous animated characters in computer games and other virtual worlds may possess several kinds of dynamic intelligence to enable reasonable, realistic navigation behavior. As a motivating example, consider an intelligent animated actor navigating in its virtual world. In one context, it continuously navigates near a long row of stationary obstacles on its way to a target (goal) position. It starts out uncomfortable, maintaining a large distance between itself and the obstacles. As time passes, however, it continuously grows more comfortable, which triggers discrete, instantaneous changes to its navigation: moving faster; allowing itself to get closer to obstacles; and generally becoming more aggressive in its action. In addition, as it travels, it distinguishes among different kinds of obstacles and among different actors around it. For instance, upon nearing a distant target, it keeps a sizable distance between itself and an odd-looking obstacle, but it passes close to a nearby friend.

Navigation systems such as the one underlying the motivating example above may be naturally described as *hybrid dynamical systems* (*hybrid systems*, for short), combinations of continuous and discrete dynamics. (We discuss hybrid systems in more detail in section 2.2.) The actor's position (and comfort level in its environment) may be described by continuous dynamics; changes between different navigation strategies may be described by discrete dynamics.

In this paper, we present a hybrid systems approach to low-level navigation for animated characters. It is a novel application of theoretical hybrid system models from both the hybrid systems [3] and animation perspectives.

As part of our presentation, we discuss general ideas for employing a hybrid systems framework to model the kind of intelligent low-level navigation described in the motivating example. Consider the diversity of dynamic intelligence demonstrated by the actor in that example. It autonomously reached its targets without colliding with obstacles, responding to moving obstacles in real time. It decided to change navigation behavior in response to a dynamically changing quantity, its comfort. It distinguished among entities around it, perhaps altering its course to reflect its subjective impressions of them. Notice further that all of this intelligence applies to *low-level* navigation responses, on the order of local perception and immediate spatio-temporal concerns. None of it is *high-level* intelligence, on the order of global pathfinding or complex deductive inference.

In addition to discussing general ideas for modeling intelligent navigation, we also present a specific, simple example hybrid system that incorporates them. Our system augments the agent steering and crowd simulation approach of [15, 16], retaining its reactive, scalable nature while expanding its behavioral intelligence. Our extensions include

two that are highlighted in the above motivating example: *selective repeller response*, which allows the actor to distinguish among various actors and obstacles around it; and *navigation mode switching*, which allows the actor to autonomously alter its general navigation behavior in response to its dynamically changing comfort. The hybrid systems framework is especially well suited for implementing navigation mode switching, which is inherently based on the interplay of continuous and discrete dynamics that hybrid system models explicitly emphasize.

We implemented our example navigation system as a hybrid system using the general-purpose hybrid system specification tool CHARON [7], concretely linking theoretical hybrid system models with practical navigation systems. In addition, we directly employed our CHARON implementation to generate animated worlds of targets, obstacles, and actors that demonstrate intelligent navigation.

2 Applying Hybrid System Theory to Multi-agent Animations

Systems with both continuous and discrete dynamics are not new in animation, but it is not always clear how these systems relate to well-understood hybrid system models. In contrast, we make a strong connection to existing hybrid system theory by using the hybrid system tool CHARON [7] to implement multi-agent animation systems. Our combination of rigorous theoretical foundations, dynamical system-oriented models, and the particular attributes we model distinguishes our approach from others in behavioral animation and behavioral robotics (e.g., [22, 26, 28]).

We base our animations primarily on the agent steering method presented in [15]. Below, we review the tools we employed to create our animations and discuss issues particular to implementing an animation system in CHARON.

2.1 A Dynamical System for Agent Steering

There have been many approaches to modeling and guiding the navigation behavior of autonomous agents. Sociophysical studies of human pedestrian behavior have resulted in *social force models* [17], which are more motivated by modeling crowds than by enabling fast computation of individuals' navigation. For individual entities, logicist, artificial intelligence-based techniques have been successfully used for cognitively empowered agents [21] and animated actors [14]; perception and dynamics-based techniques [10, 24, 28] are often more readily able to adapt to dynamic environments. Our particular approach to low-level agent navigation is based on the method in [15, 16], a scalable, adaptive approach to modeling multiple autonomous agents in dynamic virtual environments. Like treatments of similar issues in the field of behavioral robotics [20, 22], we consider only two-dimensional motion, although the mathematical foundations for three-dimensional navigation already exist [15].

Our animated worlds consist of three kinds of agents: *actors*, *targets* that represent actors' goals, and *obstacles* that actors attempt to avoid. There may be multiple actors, obstacles, and targets in an animation system. Further, obstacles and targets may be static and/or moving. These components provide a general conceptual palette that can be used to express a broad range of behaviors. For instance, an actor performing a multi-part task could be represented by its reaching a series of targets in sequence, each target corresponding to a component subtask.

At the core of the mathematics underlying our animated worlds are non-linear *attractor* and *repeller* functions that represent the targets and obstacles (respectively) in the system. Another non-linear system combines their weighted contributions in calculating an actor's angular velocity, dynamically adapting to real-time changes in the environment. Together, these non-linear systems generate natural-seeming motion, dynamically changing the heading angle of an actor in response to its dynamic environment, guiding it to avoid collisions with obstacles and other undesirable behaviors. The agent heading angle ϕ is computed by a non-linear dynamical system of the form:

$$\dot{\phi} = f(\phi, \mathbf{env}) = |w_{tar}| f_{tar} + |w_{obs}| f_{obs} + n \qquad (1)$$

The dynamical functions f_{tar} and f_{obs} are the attractor and repeller functions for the system, w_{tar} and w_{obs} are their respective weights on the agent, and n is a noise term to avoid local minima in the system.

This design reflects the *adaptive weighting* of obstacles and targets in the system: Entities exert different amounts of influence on an actor at any given moment, and the extent of these contributions varies in real time. For example, consider a case where an actor is very close to obstacles that block much of its direct path to its target (Figure 1). In such situations, the actor should temporarily disregard the target so that it won't be drawn into collisions with the obstacles. Later, when it is in less imminent danger of collision, the actor should once again consider the target. The adaptive weighting system captures such intuitions. In extreme contexts like the one just described, w_{tar} goes to zero. Similarly, in contexts where an actor should consider only a target (e.g., where there are no nearby obstacles), w_{obs} goes to zero. In non-extreme cases, the weights on obstacles and targets are non-zero, and they vary smoothly over time to reflect the changing relative positions of actors, obstacles, and targets.

The weights themselves are determined by computing

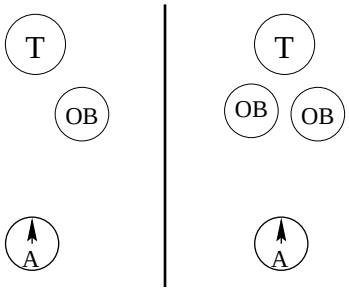

Figure 1. Two situations distinguished by adaptive weighting. On the left, the actor (A) should consider both the obstacle (OB) and the target (T) in determining its course. On the right, the actor should no longer consider the target; it should focus only on avoiding the obstacles. Our dynamical system for agent steering automatically distinguishes the two cases, changing the actor's behavior as appropriate.

the fixed points of the following non-linear system:

$$\begin{cases} \dot{w}_{tar} = \alpha_1 w_{tar}(1 - w_{tar}^2) - \gamma_{12} w_{tar} w_{obs}^2 + n \\ \dot{w}_{obs} = \alpha_2 w_{obs}(1 - w_{obs}^2) - \gamma_{21} w_{obs} w_{tar}^2 + n \end{cases} \quad (2)$$

The α and γ parameters are time-dependent functions designed to reflect conditions for the stability of the system. They ensure, for instance, that w_{tar} is zero when the actor should be temporarily disregarding a target, that w_{obs} is zero when the actor should be temporarily disregarding obstacles, etc. Many other parameters are also concealed in the terms presented above, but a discussion of the full mechanics of the adaptive weighting system is beyond the scope of this paper. We consider individual parameters only as needed in this paper; we refer readers to [15] for more technical information.

This is only an overview of one significant part of the agent steering system, but it gives a feel for the kind of mathematics involved. Further, it introduces the role parameters play in agent behavior, a notion to which we return later in this paper.

2.2 Hybrid Systems and CHARON

Hybrid systems occur frequently and naturally in many contexts, and they are studied by both computer scientists and control theorists [1, 2]. Past domains of application for hybrid system models include descriptions of biological processes [6], air-traffic management systems [27], and manufacturing systems [25]. From a general, intuitive perspective, any system characterized by discrete transitions between modes of continuous control is a hybrid system.

There are several different formal models for hybrid systems. Net-based models such as Constraint Nets [29], for instance, have been acknowledged in literature on cognitive agents. We focus in particular on automata-theoretic models such as hybrid automata [8, 12]. As a brief, non-technical introduction to this perspective, we consider a hybrid automaton as having: a set of *discrete states* called *control modes*; a *continuous state space* (a subset of $\mathbb{R}^n$ for some n); and descriptions of system evolution, with constraints both on continuous evolution within a control mode and on discrete transitions between control modes. A state of the overall system is a pair $(\textit{control mode}, \textit{continuous state})$. Research and analysis of hybrid automata underlies practical tools such as CHARON [7].

The architecture of a hybrid system in CHARON is expressed as *hierarchical agents*, a model conceptually similar to hierarchical hybrid automata. The key features of CHARON are:

Hierarchy. The building block for describing the system architecture is an *agent* that communicates with its environment via shared variables. The building block for describing flow of control inside an atomic agent is a *mode*. A mode is basically a hierarchical state machine, i.e., it may have submodes and transitions connecting them. CHARON allows *sharing* of modes so that the same mode definition can be instantiated in multiple contexts.

Discrete updates. In CHARON, discrete updates are specified by *guarded actions* labeling transitions connecting the modes. Actions may call externally defined Java functions to perform complex data manipulations.

Continuous updates. Some of the variables in CHARON can be declared *analog*, and they flow continuously during continuous updates that model passage of time. The evolution of analog variables can be constrained in three ways: *differential* constraints (e.g., by equations such as $\dot{x} = f(x, u)$), *algebraic* constraints (e.g., by equations such as $y = g(x, u)$), and *invariants* (e.g., $|x - y| \leq \varepsilon$) that limit the allowed durations of flows. Such constraints can be declared at different levels of the mode hierarchy.

Modular features of CHARON allow succinct and structured description of complex systems. Among other benefits, this modularity provides a natural-seeming structure for developing animation systems with multiple levels of behavior.

3 A Hybrid System for Intelligent Low-Level Navigation

We now present an example navigation system that demonstrates the kinds of intelligence underlying some low-level agent steering behavior. It is a substantial extension of the system summarized in section 2.1, but it retains the desirable qualities of its predecessor, including rigorous theoretical foundations, easy extensibility for higher-level navigation applications, and the scalability required for the computational burdens of multi-agent applications. In particular, we emphasize two significant extensions: *selective repeller response* and *navigation mode switching*.

Not coincidentally, we present our navigation system as a hybrid dynamical system. The hybrid systems perspective both motivated and simplified our low-level navigation modeling. In particular, it enabled us to formally specify navigation mode switching in a rigorous, system-theoretic framework.

3.1 Selective Repeller Response

In typical low-level navigation examples generated by the framework in [15], the "aggressiveness" of an actor is held constant. ("Aggressiveness" refers to the willingness of an actor to get close to repellers —obstacles and other actors— in its environment.) Thus, actors applied the same notion of "personal space" to other actors as to obstacles. Further, the framework makes no distinction within classes. Every obstacle is treated the same as every other; animated humans could not recognize that some obstacles were less noxious than others (e.g., flower beds vs. trash cans). Similarly, with respect to other actors, an actor could not get closer to one (a friend) than another (a stranger).

For our system, we extend the previous framework to allow an intelligent actor to respond *selectively* to the individual entities around it. Low-level navigation behavior may then reflect actors' subjective impressions of elements of the world around them.

In each actor, there is a parameter d_0 in the framework of [15] that encodes aggressiveness: Raising/lowering d_0 will increase/decrease the effect of a repeller on that actor. As a constant, d_0 applies equally to all repellers. To enable selective repeller response, we instead implement d_0 as a function that returns a value for each repeller. Thus, in terms of aggressiveness, each actor has its own relationships with every repeller in that world; a lookup function for d_0 encodes this in a straightforward way. This extension enables the selective responses described in the motivating example at the beginning of this paper, in which the moving actor opts to go closer to a stationary actor, a friend, than to a stationary obstacle. It also enables more complex relationships. Indeed, the function itself could evolve over time, representing changes in interpersonal closeness in a virtual world as it applies to low-level navigation. For the example animations in section 4.2, however, our d_0 is a static lookup function.

3.2 Navigation Mode Switching

An intelligent animated actor might not rely on the same dynamical navigation system for its entire path. Recall, for instance, the motivating example at the beginning of this paper. An actor might alter its basic navigation behavior based on a variety of factors: increased comfort in its environment; awareness that it is among friends; or even a simple recognition that it no longer requires complex obstacle avoidance to reach its goal. Following the framework in [15], each of these behaviors would be described by a different dynamical system. In our terminology, each dynamical system essentially defines a *mode* of navigation, and an actor might engage in *navigation mode switching* as it follows its course, "changing its mind" about appropriate low-level behavior much as it might about high-level behavior (e.g., global path planning).

We anticipate, in particular, that such changes could be real-time responses to real-time changes in the actor's dynamic virtual environment. This observation motivates our utilizing a hybrid systems framework for our navigation model; in it, we are able to specify transition conditions independent of pre-determined spatio-temporal locations. If we restricted ourselves to transitions only at pre-determined points in space or time, we would be effectively abandoning the real-time dynamic nature of low-level navigation.

To enable the kind of real-time mode switching described in our motivating example, we introduce a variable *comfort* intended to represent how generally comfortable an actor feels in its current local environment. In our model, an actor's comfort varies dynamically in real time, just like its position. Therefore, we give a differential equation form to describe the evolution of an actor's comfort:

$$\dot{c} = k_m \mathit{base}_m + f(\mathit{env}). \tag{3}$$

In (3), c is the variable for comfort, and k_m and base_m are real-valued parameters, both constant in a mode m but different in each mode. Parameter base_m represents the base rate of change in that mode, the fundamental way the actor's comfort evolves. Parameter k_m is a scaling factor; for our demonstrations in section 4.2 and on the supplementary website [5], we use high scaling factors in some modes to induce faster mode changes. We further simplify equation (3) by choosing a straightforward way to represent the dynamic influence of time and the world on an actor's comfort, making it such that $\dot{c}$ does not depend on c. There is no restriction on hybrid systems that imposes this, however,

and our general form can be readily extended to apply to more than just our simple example.

An actor's comfort is also influenced by its environment, as represented in equation (3) by the function $f(env)$. Function f could be a well defined function on a thoughtfully designed set of arguments (world attributes, motivations, etc.). We are more interested in demonstrating the flexibility and generality of the framework than in positing a particular world model, however, so we implement $f(env)$ as a random number generator, bounded appropriately for our values of k_m and $base_m$.

In our model, comfort varies dynamically with time, and an actor's behavior changes when its comfort reaches certain threshold levels. In particular, as an actor gets more comfortable in its environment, its velocity increases, it tends to become more aggressive (in the sense of section 3.1) with respect to obstacles and other actors around it, and it changes the way its comfort evolves. Each of these new behavior systems —marked by new velocity, new d_0 (either constant or function-valued d_0), and new parameters for the equation governing the evolution of c— is its own navigation mode. Thus, navigation mode switching encodes the behavior changes that occur upon comfort reaching certain thresholds. Mode transitions depend on comfort c, not necessarily on any pre-specified points in time or space. For example, transitions may occur upon an actor's reaching a target, but they are not constrained to do so.

Transitions in our model may also depend on the trend of the actor's comfort (i.e., $\dot{c}$) as well as the comfort value itself. For instance, when deciding which mode of obstacle avoidant behavior to employ for a particular portion of a navigation, an actor takes a certain transition if its comfort level is very high *or* if comfort is moderately high but trending higher at that time (i.e., $\dot{c} > 0$). This exemplifies how the hybrid systems perspective encourages a dynamically oriented approach to modeling some kinds of real-time decision making. Mode transition decisions may be made independent of time and space, dependent on both the zero'th-order value of comfort and its first-order trend.

3.3 System Architecture

The kind of navigation model featured in this paper is naturally described as a hybrid system: At its core are discrete transitions between continuous dynamical systems. Thus motivated, we implemented our example navigation system as a hierarchical hybrid system in CHARON. In this section, we describe its architecture.

Figure 2 shows the CHARON-agent architecture behind our model of an animated actor performing low-level navigation. CHARON agents (as discussed in section 2.2) are represented as rectangles. Note the hierarchy: The navigating actor (i.e., the outermost agent) has five component

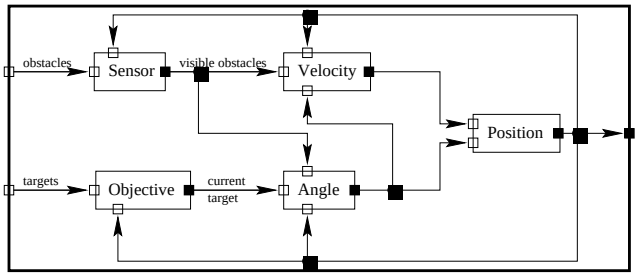

Figure 2. The CHARON-agent-level architecture of an actor performing low-level navigation. The outermost, darkest rectangle represents an actor. The rectangles contained within it (`Velocity`, `Angle`, etc.) are sub-agents.

sub-agents, including `Position` to determine its current position in a virtual world, `Sensor` to manage perception, `Objective` to determine its next target, and `Angle`, the dynamics of which we have described in this paper. Each of these CHARON sub-agents merits independent treatment; each embodies its own model of some aspect of navigation. For our implementation, we use a straightforward `Velocity` sub-agent, supplying a constant velocity that other sub-agents (e.g., `Angle`) can effectively change in real time. Our underlying architecture, however, allows for a more complex treatment of velocity.

The navigation mode switching described in section 3.2 occurs in CHARON sub-agent `Angle`, in which each navigation mode is straightforwardly represented as a CHARON mode. (Section 2.2 discusses the roles of modes and agents in CHARON.) As an actor becomes more comfortable, it switches to a new navigation mode. Aggressiveness parameter d_0, velocity, and the parameters of the dynamical system of evolution for comfort (see equation (3)) are changed to create a new notion of appropriate proximity to others that reflects that actor's new comfort. Comfort level may also decrease over the course of a navigation, so our system also includes some possible switches to lower-comfort modes. A schematic diagram of this straightforward transition system is available at the supplementary website [5].

We also include a mode-transition subsystem that emphasizes selective repeller response. When the actor reaches a certain point in its course, it enters this subsystem. The first navigation segment in this subsystem uses the standard, non-selective obstacle avoidance. After reaching a target, the actor then switches into one of two other possible modes: a simple, linear course (no obstacle avoidance); or selective obstacle avoidance. A schematic diagram that

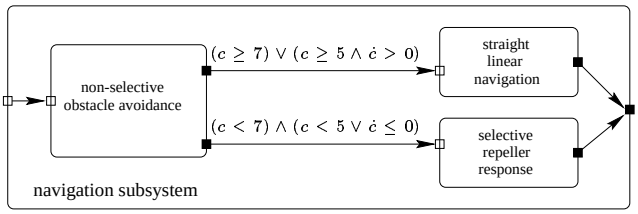

Figure 3. A subsystem that emphasizes selective repeller response. This subsystem may be embedded in a larger navigation system (and altered, as needed) as part of the `Angle` agent (see Figure 2). Note that transitions depend on both c (comfort) and $\dot{c}$.

shows transition conditions for this mode-switching subsystem is in Figure 3.

The particular system described in this section is merely a simple example, intended to straightforwardly illustrate both our general ideas on low-level navigation and the agent-oriented architecture supported by CHARON. Hierarchical CHARON agents correspond neatly to actors and other world entities, as well as to computational subcomponents of actors that merit independent treatment. CHARON modes correspond neatly to modes of behavior for these entities. By simultaneously integrating and distinguishing these conceptual levels, hierarchical hybrid system models can naturally represent abstractions that we consider when designing intelligent navigation systems. Furthermore, by adapting the system presented here, a more complete description of world attributes and influences (see, e.g., [9, 11, 18, 23]) could be incorporated into a navigation model.

4 Implementation Details and Experiments

4.1 Creating Animations from Hybrid System Models

We implemented our example navigation systems in CHARON using the key concepts noted in section 2.2. Navigating actors are implemented as CHARON agents (see section 3.3). Modes are created to represent continuous behaviors; particular continuous dynamics (e.g., the non-linear system described in section 2.1) are represented as differential or algebraic constraints of a form such as `diff {d(angle) = AngleFunction(angle,...)}`. If constraints are necessary to limit the time in a particular mode, they are represented as invariants such as `inv {Cond && !Cond2 && distance(x,y)<=distance(x,z)}`. Guarded transitions between modes are presented in a straightforward `trans from Mode1 to Mode2 when Cond do Effect` syntax; when the guard `Cond` is true, the transition is enabled, and if it is taken, statement `Effect` is executed along with the system's jump from `Mode1` to `Mode2`. In this way, the underlying continuous mathematics and relations between modes of behavior are explicitly represented in a CHARON program. Further, the modularity of agent-oriented CHARON code makes it easy to change one aspect of a system while leaving others intact. For example, it is straightforward to add new modes to our navigation mode switching systems without interfering with any of the existing ones.

CHARON also generates numerical simulations of hybrid systems, which we exploited in creating animations from our formal system specifications. We simply simulated our navigation systems in CHARON, then used a small translation routine (like a Perl script) to re-format the output of those simulations. The result of this translation was then used as input to an outside application, which created animations in which character action corresponded to the frame-by-frame data generated by CHARON. Except for small details (e.g., input format for the animation-generator), the procedure is straightforward. Issues of visualization (rendering, character models, camera motion, etc.) are independent of CHARON and may be performed using conventional methods.

Figures in section 4.2 contain images from our animations. In our scenes and experiments, actors (white mice) navigate in a virtual world of targets (usually blocks of cheese), obstacles (usually toys one might find on the floor), and other actors. (CHARON-generated animations, including those from which these Figures were taken, may be seen at the supplementary website for this paper [5].)

4.2 Multi-agent Animations

Figures 4–7 show frames from animations that demonstrate our approach to intelligent low-level navigation. An actor (white mouse), moving from the lower left to the upper right, begins a navigation with a comfort value of 0 and a uniformly applied d_0 value of 4. It starts out by moving around a series of obstacles (not shown in this paper — see supplementary website [5] for details), becoming more comfortable and more aggressive. The comfort and d_0 (aggression) values of the mouse are autonomously altered in accord with the navigation mode switching system in section 3.2; we, the designers of the animation, do not impose that those values be at any particular level at any particular place/time other than at the onset of the navigation.

Eventually, the white mouse arrives at a position where it begins a simple two-segment task: get the first/nearest block of cheese; then go get the more distant cheese. In

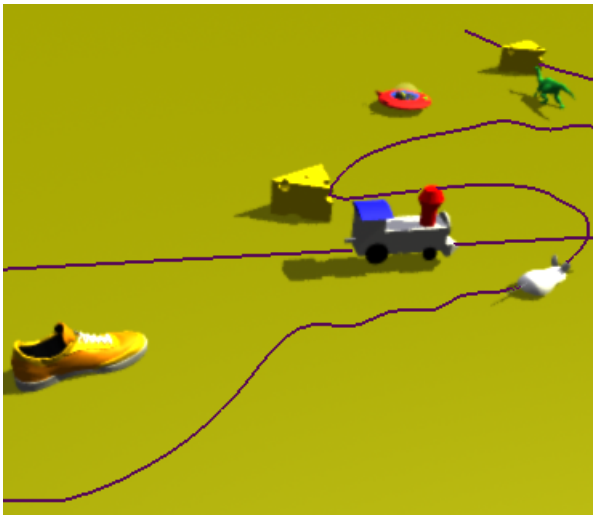

Figure 4. An actor (white mouse, on right side of image), having already swerved around a stationary obstacle (sneaker), now reacting to avoid a moving obstacle (train). The lines in the picture indicate the paths of the actor and moving obstacle before and after their current positions.

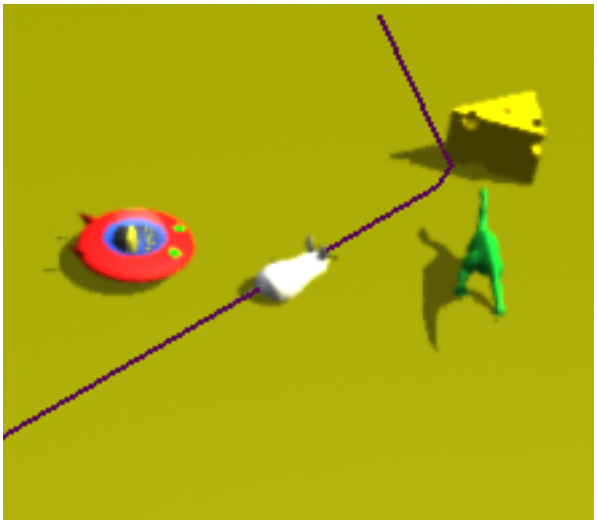

Figure 5. The actor (white mouse), having switched to simpler behavior. Its straight linear course takes it between two obstacles (a flying saucer and a dinosaur).

the first segment (Figure 4), the actor avoids a static obstacle (sneaker) and a moving obstacle (train) on the way to the first target cheese. This is a straightforward application of obstacle avoidance without selective repeller response or navigation mode switching. The mouse applies a uniform d_0 value of 2.5 to both obstacles.

Our protagonist mouse need not engage in complex evasive behavior in the second segment. It could reach the second target simply by traveling in a straight line. Or, it might instead opt for selective repeller response, depending on its comfort level. In our experiment, we implement this by having the mouse enter the subsystem presented in Figure 3 when it reaches the first target. Then, because the mouse's comfort is 6.1 and trending higher (its derivative at the time is approximately 0.075), it opts for straight linear navigation, as displayed in Figure 5. As part of this navigation mode switch (see section 3.2), its d_0 value drops to 0 —it no longer even considers obstacles, it just moves straight— and its velocity rises from 0.3 to 0.5 units per simulated second.

If it were somewhat less comfortable in its environment, however, the actor might take a different course. For instance, consider a scenario with two major differences from the one just described: The mouse's comfort is only 4.1 when it makes a navigation mode switch; and there is a toy mouse as an obstacle in place of the dinosaur in Figure 5. Our white mouse protagonist might not be quite comfortable enough to decide to forgo complex obstacle avoidance, but it might be comfortable enough to recognize the toy mouse on its right as non-threatening (while the flying saucer on its left remains troubling). As a result, it would dynamically adjust its course to pass asymmetrically between the obstacles, as shown in Figure 6. We model this by giving our protagonist a d_0 value of 1.2 as applied to the flying saucer and a d_0 value of 0.6 as applied to the toy mouse. These changes permit the white mouse to autonomously determine its asymmetric course.

If it instead decided to maintain its complex obstacle avoidant behavior, treating all repellers the same, it would take a much longer path, going around both obstacles. This option represents a kind of control experiment, a straightforward continuation of the standard obstacle avoidance that guided it up until the first target; Figure 7 is a frame from the resulting animation, showing how much farther the white mouse goes if it does not consider navigation mode switching or selective repeller response. (See [4] and [5] for further discussion and other animations.)

The website [5] of supplementary material contains animations that further demonstrate mode switching and selective repeller response, including one that demonstrates all the features in the system described in section 3.3 of this paper.

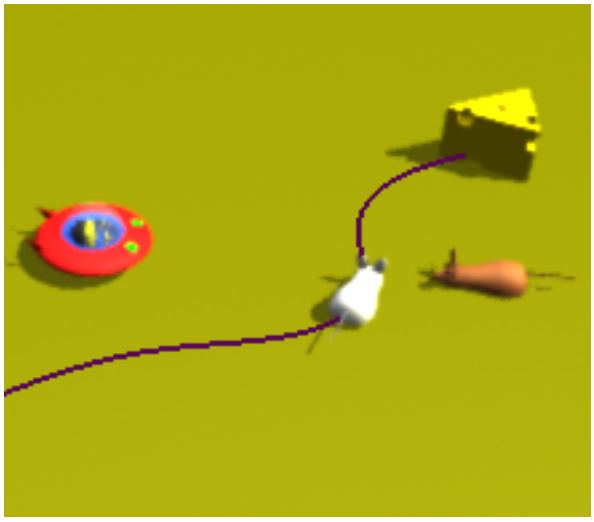

Figure 6. Contrast with Figure 5: The actor (white mouse), having switched to selective repeller response. It stays far from the flying saucer obstacle on its left, but it passes close by the friendly-looking obstacle (toy mouse) on its right when navigating around it.

Figure 7. Contrast with Figure 5: The actor, maintaining the complex, obstacle-avoidant behavior. It winds up taking a longer route to the target.

5 Notes on Practical Application of Hybrid System Theory

In this section, we present a few notes about practical issues that arose in our application of hybrid systems theory. Detailed discussions of these issues extend well beyond our current scope, but these brief treatments expose some motivation and context for the work in this paper.

5.1 Verification

One primary motivation behind our work is the eventual development of animation-oriented *formal verification tools*, e.g., programs that could mechanically verify aspects of animation systems. Such automated verification programs could be powerful design tools; animators could reason about more than just individual animations, they could reason about general systems that underlie entire classes of animations. For example, one could formally prove properties about whole classes of possible characters in a particular virtual world.

Verification is an extremely active area of research in the field of hybrid systems [2]. Our present work is, in part, a first exploration of animation from a hybrid systems perspective, creating a connection so that modal logics for hybrid systems might be employed to reason about animation. (A discussion of the logical frameworks established for reasoning about hybrid systems is beyond the scope of this paper; see [4] for details.) Indeed, related work with robots (e.g., [19]) suggests that a hybrid systems framework could be appropriate for reasoning about animations without undue disregard for the dynamic nature of animation systems.

5.2 Simulation

Our experience shows that CHARON simulations of formally specified animation systems often take longer than simulations of equivalent systems coded outside of CHARON [3]. Briefly, we believe the reasons for this are largely related to the differences in primary goals of the two disciplines, hybrid systems and animation. The hybrid systems side emphasizes analysis; for CHARON, accurate simulation is essential, and slow simulation is acceptable. On the animation side, speed is more important than extremely fine precision. Thus, although we know that our agent steering method runs in interactive time on Pentium II workstations due to a previous implementation, our CHARON simulations are sometimes significantly slower (an order of magnitude or more). We are currently exploring ways to speed up simulation without sacrificing the ability of CHARON to support system analysis.

5.3 Architecture

We have suggested that a hybrid automaton architecture is a flexible, modular framework for representing and organizing low-level navigation behaviors. Other research suggests that this flexibility extends even beyond the particular behaviors upon which we based our paper. For instance, in the robotics-oriented [13], approaching targets and avoiding obstacles are considered as two separate atomic behaviors which are joined together in a hybrid automaton model to create a single low-level navigation behavior functionally similar to the one we present in section 2.1. From a simulation standpoint, we do not see significant benefit in considering obstacle avoidance and target approach as separate behaviors for our animation work. From an architectural standpoint, however, the result in [13] supports our observation about the flexibility of the hybrid automaton model.

It seems that this flexibility would also make it straightforward to extend our current system with a hybrid automaton model of high-level navigation strategies. Our low-level navigation mechanism is essentially a control of heading angle, and it resides in the `Angle` agent in Figure 2 (section 3.3). Global, high-level navigation strategies, which would determine which target an actor would choose to go to next, would reside in the `Objective` agent. Due to the modularity of the underlying hybrid system model, it would be a simple exercise to integrate the two levels in a single actor, or to change behavior on one of the levels without affecting the other. It is not necessarily obvious, however, how to represent such high-level strategies in a hybrid automaton model that retains the reactive/behavioral flavor of our low-level navigation system. (See [3] for discussion.)

6 Conclusions and Extensions

Autonomous animated actors require complex intelligence to successfully navigate in animated virtual worlds. Indeed, even when considering only low-level (local) navigation, agents must decide among different modes of behavior, distinguish among entities in the world around them, etc., and then incorporate all this information to create realistic paths that reach targets and avoid obstacles. For a low-level navigation strategy to be effective for a multi-agent system, it must account for all these kinds of intelligence in a straightforward, scalable manner.

Employing a hybrid systems framework, we extended the crowd simulation and agent steering system of [15, 16] to accommodate many kinds of intelligence, including all those mentioned in the above paragraph. Our extensions included *navigation mode switching*, so actors could make substantial real-time changes in low-level navigation behavior in response to unpredictable world attributes. Further, we implemented *selective repeller response* to allow an actor's navigation to reflect its subjective impressions and perceptions of entities around it. We presented our extensions in a simple example system, a hierarchical hybrid system for multi-agent steering that displays these kinds of intelligence; we directly used this formalized hybrid system model to generate multi-agent animations. Our example system also demonstrates how a hybrid system-oriented perspective can influence and simplify some reactive/behavioral modeling, naturally expressing and emphasizing some dynamic aspects of decision-making for navigation. Our fundamental ideas may be readily extended or re-applied to other intelligent multi-agent navigation systems.

In our future work, we will continue exploring features provided by a hybrid systems framework. In particular, we hope to develop a hybrid system model of intelligent high-level navigation in CHARON; we could then embed this low-level navigation system into the high-level system (exploiting the modularity mentioned in section 2.2), creating a full system for agent navigation. In addition, we are investigating the potential for reasoning about animation systems using logics for hybrid systems, as discussed in [4].

Despite the relationship between hybrid system theory and animation systems, this natural interdisciplinary interface has not been well explored. Our current results suggest that hybrid systems can capture the interplay between continuous and discrete dynamics that naturally characterizes some intelligent behavior necessary for low-level agent navigation. We believe that further exploration will further extend our perspective on and vocabulary of multi-agent navigation and animation systems.

Acknowledgments

We thank Siome Goldenstein for his advice and technical assistance, and we thank Jan Allbeck and Norm Badler for helpful discussions. We also appreciate the many thoughtful comments made by reviewers of this paper. This research was supported in part by NSF grant NSF-SBR 8920230.

References

[1] *IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems*, 43(4), April 1998.

[2] *Proceedings of the IEEE, Special Issue: Hybrid Systems Theory & Applications*, 88(7), July 2000.

[3] E. Aaron, F. Ivančić, and D. Metaxas. Hybrid system models of navigation strategies for games and animations. In C. Tomlin and M. Greenstreet, editors, *Proceedings of Hybrid Systems : Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 7–20. Springer-Verlag, 2002.

[4] E. Aaron, D. Metaxas, F. Ivančić, and O. Sokolsky. A framework for reasoning about animation systems. In *Proceedings*

of the Third International Workshop on Intelligent Virtual Agents, volume 2190 of *Lecture Notes in Artificial Intelligence*, pages 47–60. Springer Verlag, 2001.

[5] E. Aaron, H. Sun, F. Ivančić, and S. Goldenstein. CHARON-generated animations and other supplementary material. Available at http://www.cis.upenn.edu/˜eaaron/CA02.html.

[6] R. Alur, C. Belta, F. Ivančić, V. Kumar, M. Mintz, G. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. In *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes In Computer Science*. Springer Verlag, April 2001.

[7] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee. Modular specification of hybrid systems in CHARON. In *Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*. Springer Verlag, 2000.

[8] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971–984, July 2000.

[9] N. Badler and J. Allbeck. Towards behavioral consistency in animated agents. In N. Magnenat-Thalmann and D. Thalmann, editors, *Deformable Avatars*, pages 191–205. Kluwer Academic Publishers, 2001.

[10] D. Brogan, R. Metoyer, and J. Hodgins. Dynamically simulated characters in virtual environments. *IEEE Computer Graphics and Applications*, 18(5):59–69, Sep/Oct 1998.

[11] A. Caicedo, J.-S. Monzani, and D. Thalmann. Toward lifelike agents: Integrating tasks, verbal communication and behavioural engines. *Virtual Reality*. To appear.

[12] J. Davoren and A. Nerode. Logics for hybrid systems. *Proceedings of the IEEE*, 88:985–1010, July 2000.

[13] M. Egerstedt. Behavior based robotics using hybrid automata. In N. Lynch and B. Krogh, editors, *Proceedings of Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 103–116. Springer-Verlag, 2000.

[14] J. Funge. *AI for Games and Animation*. A K Peters, 1999.

[15] S. Goldenstein, M. Karavelas, D. Metaxas, L. Guibas, E. Aaron, and A. Goswami. Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Computers And Graphics*, 25(6):983–998, 2001.

[16] S. Goldenstein, E. Large, and D. Metaxas. Dynamic autonomous agents: Game applications. In *Proceedings of Computer Animation 98*, 1998.

[17] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, 1995.

[18] C. Kline and B. Blumberg. The art and science of synthetic character design. In *Proceedings of the AISB 1999 Symposium on AI and Creativity in Entertainment and Visual Art*, 1999.

[19] T. J. Koo and S. Sastry. Bisimulation based hierarchical system architecture for single-agent multi-modal systems. In C. Tomlin and M. Greenstreet, editors, *Proceedings of Hybrid Systems : Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 281–293. Springer-Verlag, 2002.

[20] E. Large, H. Christensen, and R. Bajcsy. Scaling the dynamic approach to path planning and control: Competition among behavioral constraints. *International Journal of Robotics Research*, 18(1):37–58, 1999.

[21] H. Levesque and F. Pirri, editors. *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*. Springer, 1999.

[22] M. J. Matarić. Integration of representation into goal-driven behaviour based robots. *IEEE Trans on Robotics & Automation*, 8(3):304–312, 1992.

[23] S. Musse and D. Thalmann. Hierarchical model for real-time simulation of virtual human crowds. *IEEE Trans. on Visualization and Computer Graphics*, 7(2):152–164, 2001.

[24] H. Noser, O. Renault, D. Thalmann, and N. Thalmann. Navigation for digital actors based on synthetic vision, memory and learning. *Computers and Graphics*, 1995.

[25] D. Pepyne and C. Cassandras. Hybrid systems in manufacturing. *Proceedings of the IEEE*, 88:1108–1123, July 2000.

[26] C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Proceedings of SIGGRAPH '87*, volume 21, pages 25–34, 1987.

[27] C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management : A study in muti-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, Apr. 1998.

[28] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proc. of SIGGRAPH '94*, pages 43–50, 1994.

[29] Y. Zhang and A. Mackworth. Constraint nets: A semantic model for hybrid dynamic systems. *Theor. Computer Science*, 138(1):211–239, 1995.