The Visual Computer

Non-linear dynamical system approach to behavior modeling

Siome Goldenstein,
Edward Large,
Dimitris Metaxas

VAST Lab., Center for Human Modeling and Simulation, Computer and Information Science Department, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA, USA 19104-6389
e-mail: {siome,large,dnm}@graphics.cis.upenn.edu

We present a dynamic systems approach to modeling and generating low-level behaviors for autonomous agents. Such behaviors include real-time target tracking and obstacle avoidance in time-varying environments. The novelty of the method lies on the integration of distinct non-linear dynamic systems to model the agent's interaction with the environment. An *angular velocity control* dynamic system guides the agent's direction angle, while another dynamic system selects the environmental input that will be used in the control system. The agent interacts with the environment through its knowledge of the position of stationary and moving objects. In our system agents automatically avoid stationary and moving obstacles to reach the desired target(s). This approach allows us to prove the stability conditions that result in a principled methodology for the computation of the system's dynamic parameters. We present a variety of real-time simulations that illustrate the power of our approach.

Key words: Digital agents – Game animations – Low-level behaviors – Motion planning – Dynamical systems

The Visual Computer (1999) 15:349–364

1 Introduction

The importance of game and simulation applications grows everyday, as does the need for animated agents that operate autonomously in these environments. These agents must be able to exhibit certain low- and high-level behaviors and interact with the environment. Various approaches for modeling behavior and movement decisions have been investigated, among them the pioneering work of Reynolds (Reynolds 1987) and that of others (Haumann and Parent 1988; Wilhelms 1990; Renault et al. 1990; Bates et al. 1992; Noser and Thalmann 1993; Reynolds 1993; Ko et al. 1994; Tu and Terzopoulos 1994; Noser et al. 1995). These include, kinematic, dynamic, learning, and artificial intelligence-based approaches.

AI techniques for behavior generation, such as (Haumann and Parent 1988; Lethebridge and Ware 1989), generally require complex inferencing mechanisms. This may require considerable computational resources, raising the question of scalability of such systems, as the number of independent agents and behaviors grows, and each agent has a different set of goals and behavioral directives. In addition, agents must be able to interact with real-time moving objects that might either contribute to or compromise the final goal. Other approaches to this problem employ learning, perception, and dynamic techniques (Ridsdale 1990; Tu and Terzopouluos 1994; Noser et al. 1995; Grzeszczuk and Terzopouluos 1995), while computational geometry techniques have been employed in rather restricted environments (Wilfong 1988).

Dynamic system approaches to this problem have mostly been explored at the level of control theory (Cohen 1992; Liu et al. 1994; Kokkevis et al. 1995). This methodology restricts the type of behaviors that can be simulated since it does not cope well with dynamically changing environments and its computational complexity prohibits its use in real-time simulations. Others discuss the usefulness of different layers of control and behaviors to achieve better results (Kurlander and Ling 1995; Magnenat-Thalmann and Thalman 1995; Blumberg and Galyean 1995; Perlin and Goldberg 1996)

In this paper, we investigate and develop an alternative methodology that has its roots in behavior-based robotics (e.g., (Braitenberg 1989; Brooks 1991)) and is based on a novel way of combining differential equations exhibiting particular behaviors (Steinhage and Schöner 1997; Large et al. 1999). Using this type of approach, Schöner and colleagues (Schöner and Dose 1992; Schöner et al. 1996) have devel-

oped a dynamical system for robot path planning and control. In this system a set of behavioral variables, namely heading direction and velocity, define a state space in which the dynamics of robot behavior is described. Path planning is governed by a nonlinear dynamical system that generates a time course of the behavioral variables. The system dynamics are specified as a nonlinear vector field, while the task that the agent will execute depends upon the task constraints. Task constraints, such as obstacle avoidance and target tracking, are modeled as component forces which define attractors and repellers for the dynamical system. The individual constraint contributions are weighted and then added together into a single vector field, which determines the observed behavior. Next, a second dynamical system is used to compute these weights. This dynamical system, the task level system, operates at a faster time scale than the movement level (Large et al. 1999). Qualitatively different behaviors are modeled as fixed points of this dynamical system, and the environment determines the values of the parameters. As the perceptual information changes, parameters change, causing bifurcations in the task-level system. During the course of the simulation, one fixed point loses stability and another becomes stable, modeling the decision to cease executing one behavior and to execute another instead.

Here, we adapt the above methodology to model and simulate autonomous low-level dynamic behaviors which can be used in applications ranging from virtual environments to games. We use (see also our preliminary results, in (Goldenstein et al. 1998)) a set of time-varying differential equations that control the heading angle and forward speed of a given digital autonomous agent. Based on a principled combination of these equations we create a whole set of relatively complex low-level behaviors such as obstacle avoidance and target tracking. To avoid unstable fixed points in the differential equations[1] we add a Gaussian noise term in each equation. Using this system, decisions are made online and do not require any previous memory, training, or global planning. The set of targets and obstacles can change during the course of the simulation, since the agent is able to make smart local decisions based on its current global knowledge of the dynamic environment in which it is situated. For example, an agent will temporarily disregard a target if there is an unsurpassable moving or stationary obstacle immediately between it and the target. Like a human, it will first focus on avoiding the obstacle, and then refocus on the target.

In this paper we also prove an important result on the properties of the parameters of the task constraint system which determine the number of behaviors modeled. These properties are associated with the stability[2] of the task constraint system. We also provide an inductive proof that extends the above result to any number of dimensions in the task constraint system. This result allows the principled design of complex systems with large number of behaviors.

Our system allows single and multiple target tracking in the presence of multiple static and moving obstacles, and scales well with the number of behaviors. Obstacles are processed in a local fashion based on their relative location to the agent and the target. In our current implementation and without loss of generality the agents are memoryless and reactive in nature. Depending on the situation (emergence of new obstacles and/or targets) their movement can be discontinuous.

This paper is organized as follows. In Sects. 2 and 3 we present the dynamic formulation of the system. Section 4 describes the task constraint system responsible for the generation of the behaviors. We devise an inductive proof for the restrictions on this system's parameters (stability analysis) that will result in the desired behaviors. Finally, we present a series of real-time simulations involving complex interactions between the agents, the environment and the targets.

2 Movement dynamics and low-level behavior

Our methodology consists of the combination of two distinct dynamic systems to model the movement and behavior of each autonomous agent. The first system controls the movement of the agent. The state space of this system is two-dimensional, the first parameter represents the heading direction, while the other specifies its velocity (In a three-dimensional space there would be two heading angles and one forward velocity). The second system controls the

[1] In differential equation terminology a *fixed point* is a point where the derivative of the vector field is zero

[2] The *stability* is related to the local convergence properties of the system around a fixed point

agent's movement decision-making, i.e., its behavior. The state space of this system is the space of the agent's behaviors. The parameter values of the state vector components determine which elements of the environment (e.g., obstacles, targets) will be used in the calculation of the agent's movement and therefore behavior.

Each autonomous agent's movement is described in polar coordinates. It consists of a heading direction ϕ and a forward velocity v. The heading angle is controlled by a one-dimensional non-linear dynamical system, which consists of repellers placed in the subtended angle of the obstacles, and attractors in the subtended angles of targets (see Sect. 2.1). In our formulation, the heading speed is modified by the relative location of the obstacles (see Sect. 3).

Based on our formulation, an agent may ignore targets or obstacles, depending on the scene geometry around the agent at each time instance. The environment geometry is modeled based on another type of nonlinear dynamical system running at a faster time scale. This system outputs *weights* that linearly combine the different target and obstacle contributions as calculated by the first system. An important aspect of our methodology is that it scales linearly with the number of obstacles in the environment.

In the following sections we present the details of each of the two dynamical systems, the basic movement dynamics and the modeling of behaviors.

2.1 The basic movement dynamics

The first dynamical system models the control of the basic movement of each autonomous agent. The movement is defined by a 2D vector representing the agent's heading angle and forward speed.

The heading angle ϕ of a given agent is controlled by a dynamical system of the type:

$$\dot{\phi} = f(\mathbf{env}) \,, \tag{1}$$

where **env** is the vector of variables which model the environment (e.g., the geometry and position of the obstacles and targets).

According to our dynamical system formulation each element of the environment can attract or repel an agent. We will therefore use *attractors* to model targets and *repellers* to model objects that should be avoided. We begin our formulation by assuming two groups of influences on our agent. The first consists of the contribution from the target, the second from the sum of contributions from all obstacles. We then extend the formulations to account for multiple groups.

Using (1) an attractor is defined as

$$\dot{\phi} = f_{\text{tar}} = -a \sin(\phi - \psi) \,, \tag{2}$$

where ψ is the angle of the target's location relative to the agent's location and a is a constant parameter. Figure 1 shows a plot of (2), assuming that the target is in front of the agent ($\psi = 0$). The negative slope around the fixed point (in this case, the origin) is what gives the attractor property, forcing $\dot{\phi}$ to be positive if the target is to the right and negative if it is to the left of the current viewing direction of the agent.

We model different types of obstacles using repellers. A first simple approach would be to use a sinusoidal function, just like we did for the target, but with a positive slope instead. This would result in a positive slope necessary for the repelling effect. However the repelling effect would not necessarily be preserved if we linearly sum the contributions of multiple obstacles.

Consider the example in Fig. 2 with two distinct obstacles positioned symmetrically about the direction of the agent by an angle ψ. They are far enough apart so that the agent can simply move between them by proceeding forward without the risk of colliding. The obstacles' contribution to the agent's angle (through $\dot{\phi}$) is:

$$\begin{aligned} \dot{\phi} = f_{obs} &= a \sin(\phi - \psi) + a \sin(\phi + \psi) \\ &= a[\sin\phi\cos(-\psi) + \cos\phi\sin(-\psi) \\ &\quad + \sin\phi\cos\psi + \cos\phi\sin\psi \\ &= 2a\cos\psi\sin\phi = k'\sin\phi \,. \end{aligned} \tag{3}$$

From (2) and (3) we observe that this simple sum results in a repeller in the forward direction ϕ (with the amplitude controlled by ψ), certainly an undesirable effect.

In order to create a repeller other aspects of the environment geometry, such as the distance from the agent, should also be taken in account. In addition, we would like to control, based on parameters, properties of the repellers such as their spatial extent. Therefore, we use a model for a repeller which is a multiplication of three different functions, R_i, D_i, W_i, each modeling a different property

$$\dot{\phi} = f_{\text{obs}_i} = R_i \; W_i \; D_i \,. \tag{4}$$

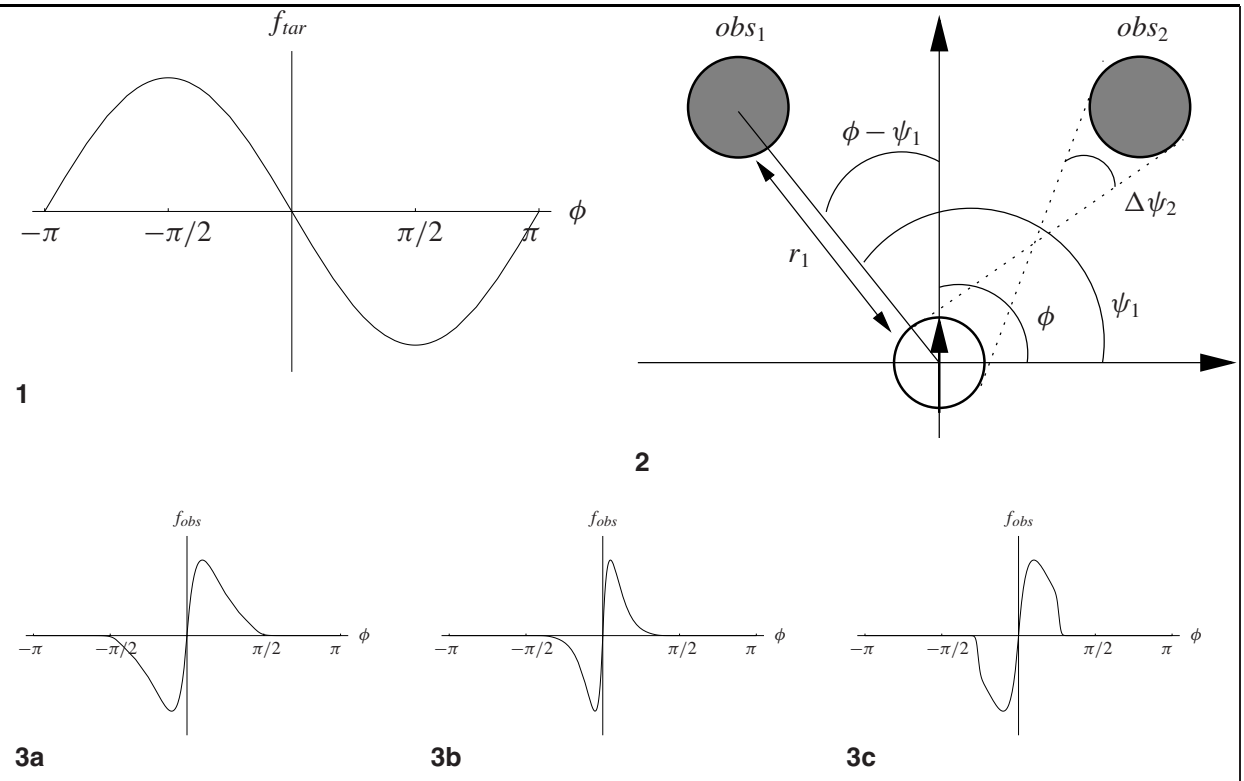

Fig. 1. Target function for the angle dynamical system for $\psi = 0$. An attractor (stable fixed point) at $\phi = 0$ and a repeller (unstable fixed point) at $\phi = \pi$

Fig. 2. Two obstacles in front of the agent

Fig. 3a–c. Repellers for three different parameters settings

Function R_i models a generic repeller, and is constructed as:

$$R_i = \frac{(\phi - \psi_i)}{\Delta\psi_i} e^{\left(1 - \left|\frac{\phi-\psi_i}{\Delta\psi_i}\right|\right)} , \tag{5}$$

where ψ_i is the angle of obstacle i with respect to the agent's direction, and $\Delta\psi_i$ is the subtended angle of the obstacle as seen by the agent.
The second function, W_i, is responsible for limiting the angular range of the repeller's influence in the environment and is defined as

$$W_i = \frac{1}{2}[\tanh(h_1(\cos(\phi - \psi_i) - \cos(2\Delta\psi_i + \sigma))) + 1] , \tag{6}$$

which models a window-shaped function and h_1 is responsible for the inclination of the window's sides and is modeled by

$$h_1 = 4/(\cos(2\Delta\psi) - \cos(2\Delta\psi + \delta)) . \tag{7}$$

Here δ is a safety margin constant.
The third and last function, D_i, models the influence of the obstacle in the environment by using the distance of the obstacle from the agent. It is modeled as

$$D_i = e^{-\frac{r_i}{d_0}} , \tag{8}$$

where r_i is the relative distance between the obstacle and the agent, and d_0 controls the strength of this influence as the distance changes.
The parameters ϕ_i, $\Delta\phi_i$ and r_i are graphically illustrated in Fig. 2.
Figure 3 shows three repellers obtained by using different values for the parameters (for all examples the

obstacle is directly ahead of the agent, i.e., $\psi = 0$). In Fig. 3a $\Delta\psi = \pi/10$, $r_i = 3$, $d_0 = 6$ and $\delta = 0.8$. Figure 3b shows a situation where we used the same parameters as in Fig. 3a, but changed $\Delta\psi$ to $\pi/20$. In Fig. 3c we used the same parameters as in Fig. 3a, but now a change was made to W_i (window-shaped function), by setting δ to 0.1.

The resulting effect on the agent's heading direction from all obstacles $i = 1, \ldots, n$, is the sum of the respective repellers

$$f_{\text{obs}} = \sum_{i=1}^{n} f_{\text{obs}_i} \; . \tag{9}$$

Based on the above design of each repeller (see (4)), we can use the simple sum of each individual obstacle repeller to combine multiple contributions (9). Let's assume again that the agent is facing two different obstacles located side by side (see Fig. 4). If the obstacles are too far apart, the agent should be able to pass between them, otherwise it will should go around them. This decision is taken automatically, as shown in Fig. 4. Figure 4a depicts the case where two obstacles are too close; Fig. 4b depicts the case where the distance between the obstacles is exactly equal to the size of the agent, a critical condition; and Fig. 4c depicts the case when the obstacles are far enough apart to allow the easy passage of the agent between them. In this simple case (no target and two obstacles) we have plotted (9) at the bottom of each figure as a function of ϕ. These functions show that the agent exhibits the correct behavior. Next, (9) needs a slight modification. In Fig. 4a we see that for the case where the agent is facing directly between the obstacles, $\dot{\phi} = 0$. This implies that there will be no change in the heading direction and eventually there will be a collision situation. If we look at the value of $\dot{\phi}$ slightly off the origin, we notice that the slope of the function is positive so the agent will repelled. We call this case an *unstable fixed point*. It is just like a ball situated in the crest of a hill – if placed exactly on the top is does not move, but if any slight perturbation occurs it will fall. To solve this we must add to the system a noise term n.

Based on the above we modeled the environment as a sum of a set of repellers (obstacles) and attractors (targets). The dynamic system that controls the heading angle ϕ of our agent is the weighted sum of these set of attractors and repellers with the addition of a noise term. Therefore, the final definition of the dynamical system controlling the heading angle in (1) is obtained as:

$$\dot{\phi} = f(\mathbf{env}) = |\boldsymbol{w}_{\text{tar}}| f_{\text{tar}} + |\boldsymbol{w}_{\text{obs}}| f_{\text{obs}} + n \; . \tag{10}$$

The weights $\boldsymbol{w}_{\text{tar}}$ and $\boldsymbol{w}_{\text{obs}}$ are intended to eliminate spurious attractors that can occur by a simple sum of the nonlinear functions modeling the various obstacles and targets in the environment. These *weights* are obtained through a constraint competition dynamical system which will be described in detail in Sect. 2.2 (see also (Large et al. 1999)). The weights are the essence of the low-level behavior modeling. Finally, as explained before, the Gaussian noise term n allows the system to escape from unstable fixed points.

Let's first ignore the influence of the weights (setting them to 1), and see the properties of our design in (10). In Fig. 5 we show the interaction between a target (attractor) and an obstacle (repeller). It is intuitively clear that the agent will have to go around the obstacle in order to arrive at the target without hitting the obstacle. In this case, the modeling of the agent's heading direction dynamics, $\dot{\phi}$, based on (10) is shown in the lower-right graph of Fig. 5. It is the summation of the target (upper-right graphic) and obstacle (middle-right graphic) functions. The presence of two final attractors, indicated by the two arrows in the lower-right graph, shows the two possible ways to get to the target and avoid the obstacle (going either to the left or to the right of the obstacle). Unfortunately, the simple sum (setting the weights to 1) will not always be sufficient. In Fig. 6 we extend the situation shown in Fig. 4a through the addition of an attractor located behind the two closely placed obstacles. The resulting function now has $\dot{\phi} = 0$ in the origin and a negative slope around it. This means that the result of this simple sum is an attractor in the forward direction. This implies that even with noise, the agent will proceed straight ahead and will try to go through the obstacles as if they were not there.

The use of weights whose values are determined based on the second dynamical system described below will result in the correct action of the dynamic system.

2.2 Constraint competition: modeling the weights

Individually, the attractors and repellers defined in Sect. 2.1 work in the desired way. Due to the nonlinearity in their definitions their simple sum might

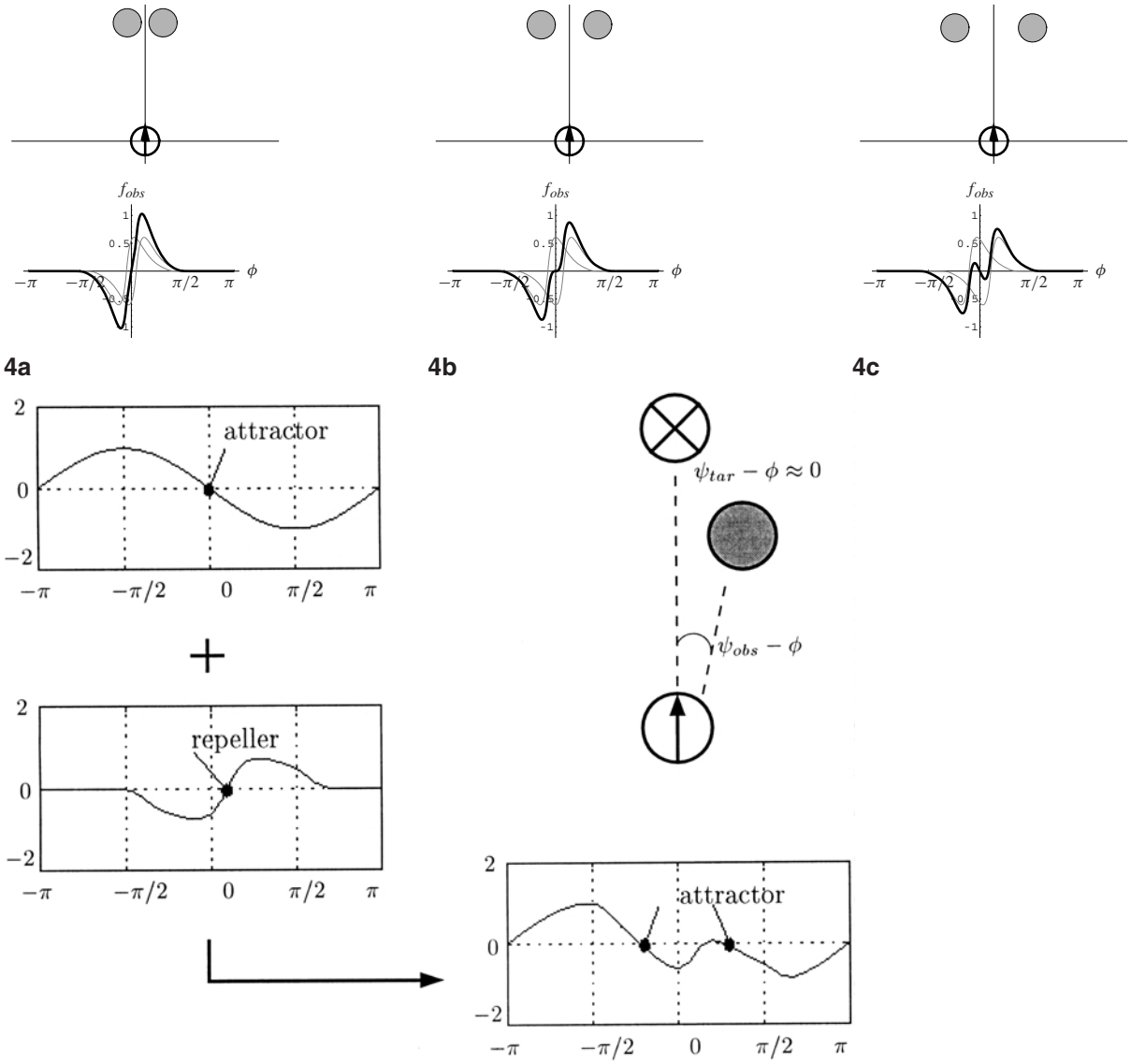

Fig. 4a–c. Interaction of repellers of two different obstacles
Fig. 5. Interaction between one attractor and one repeller

not always yield the expected result. In the first example (Fig. 5) the simple sum gave the correct result, but in the example shown in Fig. 6, the sum of one attractor with the two repellers results in an impossible path in between the two obstacles – they are too close to each other to allow the agent's passage.

To avoid this kind of problem, the composition of the attractors and repellers functions is obtained not by a simple sum, but through a weighted sum. The weights, $\boldsymbol{w}_i$, are the result of a second dynamical system, which runs at a faster time scale with respect to the dynamical system in (10). This second constraint

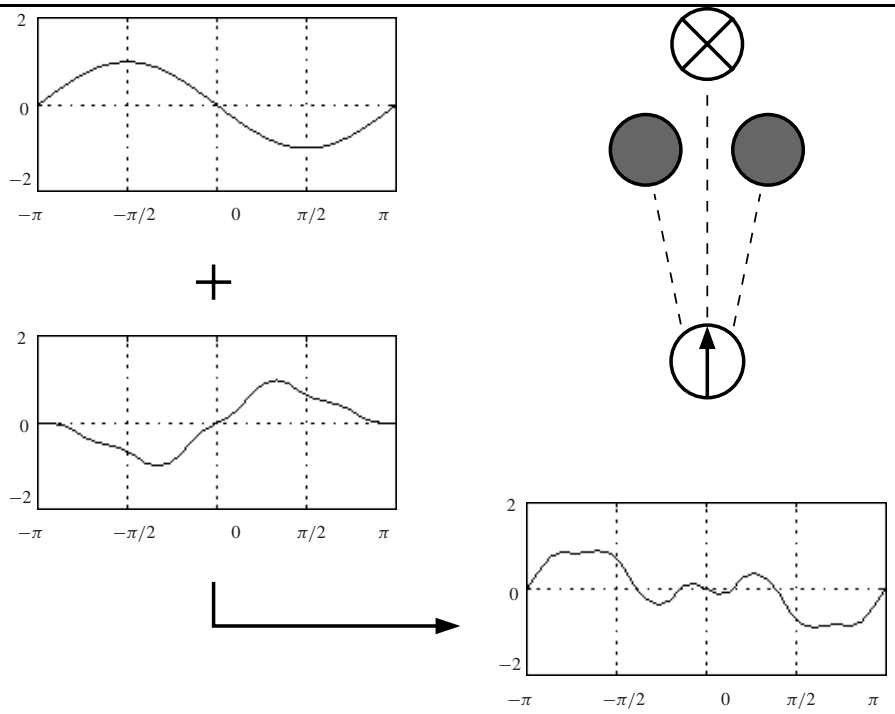

Fig. 6. Undesired attractor of the angle dynamical system

system is defined as

$$\dot{\boldsymbol{w}}_i = \alpha_i \boldsymbol{w}_i (1 - \boldsymbol{w}_i^2) - \sum_{j \neq i} \gamma_{i,j} \boldsymbol{w}_j^2 \boldsymbol{w}_i \ , \tag{11}$$

where in the simple case with only one kind of obstacle and a single target, the state space ($\boldsymbol{w}_i$) is bidimensional and consists of ($\boldsymbol{w}_{\text{tar}}$, $\boldsymbol{w}_{\text{obs}}$), as used in (10).

The parameters α_i and γ_{ij} are functions of the environment geometry at each time instant. They are called *competitive advantage* and *competitive interaction*, respectively. Together with the initial value of the state space they completely define the dynamical system.

If the constraint competition system runs at a much faster rate than the basic movement system, we can assume that the environment and thus α_i and γ_{ij} are fixed in (11). This is practically accomplished by letting the constraint system run until it stops (at a fixed point) for each step of the movement system. The α_i and γ_{ij} are only calculated once at the beginning of each step. This simplification allows us to precisely study the stability conditions.

The fixed points have the property that $\dot{\boldsymbol{w}} = 0$. Once the system arrives to one it does not leave. But sometimes the fixed point might be unstable in the presence of small perturbations, just like a ball on the crest of a hill. It is important to study how the system behaves in regions close to the fixed point. The fixed point is stable if the eigenvalues of the Jacobian ($J_{ij} = \partial \dot{\boldsymbol{w}}_i / \partial \boldsymbol{w}_j$) are all negative (this is the multidimensional generalization of the negative slope seen in Sect. 2).

The correct design of the parameter functions α_i and γ_{ji} will provide the desired low-level behavior of the agent as expressed in (10). Therefore, it is important to understand the stability of this system (for more details see (Perko 1991), and incorporate the geometry of the environment in the low-level behaviors through the α_i and γ_{ij}.

The details of the stability analysis for the two-dimensional case used in (10) is in Appendix A and follow closely the approach of (Large et al. 1999;

Table 1. Stability analysis

$\boldsymbol{w}_{\text{tar}}$	$\boldsymbol{w}_{\text{obs}}$	Stability
0	0	$\alpha_1, \alpha_2 < 0$
0	± 1	$\gamma_{12} > \alpha_1$ $\alpha_2 > 0$
± 1	0	$\gamma_{21} > \alpha_2$ $\alpha_1 > 0$
$\pm A_1$	$\pm A_2$	$\alpha_1, \alpha_2 > 0$ $\alpha_1 > \gamma_{12}$ $\alpha_2 > \gamma_{21}$ $\gamma_{12}\gamma_{21} < 0$ or $\gamma_{12}, \gamma_{21} > 0$

Schöner and Dose 1992). Table 1 summarizes these results.

The four distinct cases can be physically reinterpreted as turning off both target and obstacle contributions (when $\boldsymbol{w} = [0, 0]$, an obviously undesired solution), turning off either one of them ($\boldsymbol{w} = [0, \pm 1]$ and $\boldsymbol{w} = [\pm 1, 0]$), and an averaging solution, where both contributions are taken in account ($\boldsymbol{w} = [\pm A_1, \pm A_2]$).

The design of α_i and γ_{ij} depends on the desired properties of the final system. We want the obstacles contribution active in (10) unless there are no obstacles close enough to the agent. Target contributions should be deactivated when they are behind a repeller. For ease of design we will use positive values for the parameter functions α_i and γ_{ij}.

In Appendix B, we describe the design of the parameter functions α_i and γ_{ij} following again the approach from (Large et al. 1999; Schöner and Dose 1992).

3 Modeling the agent's velocity

In the previous sections we modeled the change in the agent's heading direction. In this section we model its forward movement. There is a series of different possibilities to accomplish this, but in this paper we will examine two of them. The first is the *Constant* velocity and the second is a *Constant time to contact* method.

The first option is obviously the easiest one, the agent always moves with a constant speed. This method can work well if the forward velocity is small enough so that the heading angle dynamics has enough time to react and avoid obstacles. Unfortunately in many situations it leads to disastrous results. Obstacles might not be static, changing the configurations is a faster way than the heading dynamical system can react only by itself. Therefore adaptive speeds are necessary in order to achieve a better behavior.

The *Constant time to contact* method (also used by Neven and Schöner (Neven and Schöner 1996)) seeks to solve this problem by making the speed change as a function of the distance between the agent and the closest visible obstacle. It is even possible to make a retreat (negative forward speed) if this distance is too small.

As the name suggest, this approach fixes the time that it would take the agent to reach the closest obstacle as if it were right in front of it:

$$v = \frac{r_{min} - d_1}{\tau_{2c}}\,, \tag{12}$$

In (12) τ_{2c} is the fixed time to contact, r_{min} is the distance between the agent and the closest target, and d_1 is a safety distance. With nearby obstacles in the vicinity, the agent will chose a smaller velocity, and for situations where the obstacles are far away, it will go faster. The value of d_1 also introduces a notion of personal space, in a sense that if the obstacle is too close to the agent it will try to retreat.

Based on our experiments we have found that *constant time to contact* method provides the most reasonable behaviors.

4 Expanding the constraint competition system by induction

In Sect. 2.2 we have seen the constraint competition system for a two-dimensional vector $\boldsymbol{w}$ of weights. In a general application, the classification of the contributions into only two groups (obstacles and targets) might be a rather restrictive description. In some applications it is conceivable to classify and treat separately different types of obstacles, or have different mutually exclusive targets.

We will use an inductive method to generalize our approach so that we can design multidimensional systems. Of course the base case (two dimensions) has already been examined in Sect. 2.2. Our induction hypothesis is that a k-dimensional system will have *at most* 2^k fixed points.

Let's suppose (by strong induction hypothesis) that we know how to solve/design all systems with di-

mensions $\leq k$. We then want to solve/design a $k+1$-dimensional system. As in Sect. 2.2 we will first find the *fixed points*, and then check their stability conditions.

4.1 The fixed points

The fixed points of the $k+1$-dimensional system can either have $\boldsymbol{w}_{k+1} = 0$ or $\boldsymbol{w}_{k+1} \neq 0$. We analyze separately the case where all other $\boldsymbol{w}_k$ are equal to zero.
There are three cases:

Case 1: $\boldsymbol{w} = [0, \cdots, 0, 0]^t$ and $\boldsymbol{w} = [0, \cdots, 0, 1]^t$. It is clear by inspection, just as in the two-dimensional case, that for the vectors $\boldsymbol{w} = [0, \cdots, 0, 0]^t$ and $\boldsymbol{w} = [0, \cdots, 0, 1]^t$, $\dot{\boldsymbol{w}} = \mathbf{0}$, so these are two of the fixed points.

Case 2: $\boldsymbol{w} = [A_1, \cdots, A_k, 0]^t$, where A_i can be zero. It is easy to see that this case reduces to a k-dimensional case. By the induction hypothesis we know that have at most 2^k fixed points.

Case 3: $\boldsymbol{w} = [A_1, \cdots, A_k, A_{k+1}]^t$ and $A_{k+1} \neq 0$. Again, this will hold in at most 2^k fixed points. If any of the A_i, $i \neq k+1$ is equal to zero, then it's existence conditions are known by induction (without loss of generality we could swap variables $\boldsymbol{w}_i$ with $\boldsymbol{w}_{k+1}$, and then we would have the case 2).
The fixed point in the case where all $\boldsymbol{w}_i \neq 0$, $i \in 1 \cdots k+1$ is

$$\dot{\boldsymbol{w}}_i = \boldsymbol{w}_i \left(\alpha_i - \alpha_i \boldsymbol{w}_i^2 - \sum_{j \neq i} \gamma_{i,j} \boldsymbol{w}_j^2 \right) . \quad (13)$$

We can write this as an linear system of equations for $\boldsymbol{w}_i^2$, that has the matrix form:

$$\begin{pmatrix} \alpha_1 & \gamma_{12} & \cdots & \gamma_{1(k+1)} \\ \gamma_{21} & \alpha_2 & \cdots & \gamma_{2(k+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{(k+1)1} & \gamma_{(k+1)2} & \cdots & \alpha_{k+1} \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_1^2 \\ \boldsymbol{w}_2^2 \\ \vdots \\ \boldsymbol{w}_{k+1}^2 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{k+1} \end{pmatrix} \quad (14)$$

or in a more compact form:

$$\mathbf{M}\mathbf{w}^2 = \bar{\alpha} . \quad (15)$$

From (15) we conclude that the $\boldsymbol{w}_i \neq 0$ fixed point is (when solvable),

$$\boldsymbol{w}_i = \sqrt{(\mathbf{M}^{-1}\bar{\alpha})_i} , \quad (16)$$

but only if $|\mathbf{M}| \neq 0$ (for the solution of the linear system), and all components of $\mathbf{M}^{-1}\bar{\alpha}$ are larger or equal to zero (because we are only interested in real values of $\boldsymbol{w}_i$). These are the existence conditions.

4.2 Stability of the fixed points

For the analysis of the stability of each of these candidate points we have to check the eigenvalues of the Jacobian matrix of (11) at each point.
The Jacobian is: (see (17))

$$J(\boldsymbol{w}) = \begin{pmatrix} \alpha_1(1-3\boldsymbol{w}_1^2) - \sum_{j \neq i} \gamma_{ij} \boldsymbol{w}_j^2 & -2\gamma_{12}\boldsymbol{w}_1\boldsymbol{w}_2 & \cdots & -2\gamma_{1(k+1)}\boldsymbol{w}_1\boldsymbol{w}_{k+1} \\ -2\gamma_{21}\boldsymbol{w}_2\boldsymbol{w}_1 & \alpha_2(1-3\boldsymbol{w}_2^2) - \sum_{j \neq i} \gamma_{ij} \boldsymbol{w}_j^2 & \cdots & -2\gamma_{2(k+1)}\boldsymbol{w}_2\boldsymbol{w}_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ -2\gamma_{(k+1)1}\boldsymbol{w}_{k+1}\boldsymbol{w}_1 & -2\gamma_{(k+1)1}\boldsymbol{w}_{k+1}\boldsymbol{w}_1 & \cdots & \alpha_{k+1}(1-3\boldsymbol{w}_{k+1}^2) - \sum_{j \neq i} \gamma_{ij} \boldsymbol{w}_j^2 \end{pmatrix} . \quad (17)$$

If $\boldsymbol{w} = \bar{\mathbf{0}}$ the Jacobian will have the format:

$$J(\boldsymbol{w}) = \begin{pmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \alpha_{k+1} \end{pmatrix} . \tag{18}$$

Therefore the zero vector case is just like the one in two dimensions. To be a *stable* point we have to require that $\alpha_i < 0, \forall i$.
For the case where $\boldsymbol{w} = [0, \cdots, 0, 1]^t$,

$$J(\boldsymbol{w}) = \begin{pmatrix} \alpha_1 - \gamma_{1(k+1)} & 0 & \cdots & 0 \\ 0 & \alpha_2 - \gamma_{2(k+1)} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & 0 \quad -2\alpha_{k+1} \end{pmatrix} . \tag{19}$$

This gives us $k+1$ stability conditions: $\gamma_{i(k+1)} > \alpha_i$, for $i = 1 \ldots k$, and $\alpha_{k+1} > 0$.
The case where $\boldsymbol{w}_k$ is equal to zero, but where $\boldsymbol{w}_i$ (for $i \neq k$) can be anything, the Jacobian will be

$$J(\boldsymbol{w}) = \begin{pmatrix} \boxed{} & 0 \\ 0 & \alpha_{k+1} - \sum_1^k \gamma_{(k+1)j} \boldsymbol{w}_j^2 \end{pmatrix} . \tag{20}$$

That means that the stability conditions will be exactly the ones for the sub-matrix (it has size k so by induction we know how to solve) plus the additional condition that

$$\alpha_{k+1} - \sum_1^k \gamma_{(k+1)j} \boldsymbol{w}_j^2 < 0 .$$

This same argument is valid if any other of the $\boldsymbol{w}_i$ is zero with $\boldsymbol{w}_{k+1} \neq 0$ (without loss of generality we can rearrange the order of the variables).
The only case left is when all $\boldsymbol{w}_i$ are different than zero. This case is by far the hardest, and has to be solved on a case by case basis, only if it is needed. All the eigenvalues of the Jacobian at the averaging point must be smaller than zero.

5 Summary of the algorithm

We start by defining the environment and it's geometry; which consists of agents, targets and obstacles. For each agent, target and obstacle we define its attributes, like size location and initial orientation.
The following pseudocode describes the flow of the algorithm:

```
1:  for each instant t do
2:    for each agent do
3:      Calculate ψ_tar and Δψ_tar (see Fig. 2);
4:      Use (2) to calculate f_tar and also ∂f_tar/∂φ;
5:      W_t = 0, f_obs = 0, Df_obs = 0;
6:      for each Obstacle i do
7:        Calculate ψ_obs_i and Δψ_obs_i;
8:        Using (5), (6), (8) and (4) calculate R_i,
          D_i, W_i, f_obs_i and ∂f_obs_i/∂φ;
9:        W_t = W_t + W_i, f_obs = f_obs + f_obs_i and
          Df_obs = Df_obs + ∂f_obs_i/∂φ;
10:     end for
11:     Calculate γ_12 (32), α_2 (33) and α_1 (34) (γ_21
        is a constant, here set to 0.05);
12:     Using initial value as w = [0, 0], simulate
        (21) until it stops at a fixed point;
13:     Use (10) to calculate φ̇;
14:     Use one of the velocity methods to calcu-
        late the forward velocity;
15:     Find next position and orientation (φ) of the
        agent through forward integration.
16:   end for
17: end for
```

6 Experimental results

Our system was implemented in C, using *lua* (Ierusalimschy et al. 1996, 1997; WebSite a), as an extensible embedded language to describe both the scene and the target(s)/agent(s) movement.
All the rendered animations we have obtained using our system can be viewed at the following web site: "`http://www.cis.upenn.edu/~siome/research/nonlin/`".
The following experiments explore and demonstrate the power of the system described in (10), which involves agents, targets and obstacles. In all the examples the constant a in (2) was set to 1 and the safety margin δ in (7) was set to 0.8. The Euler integration time step was 0.25 and all the simulations run in faster than real time.

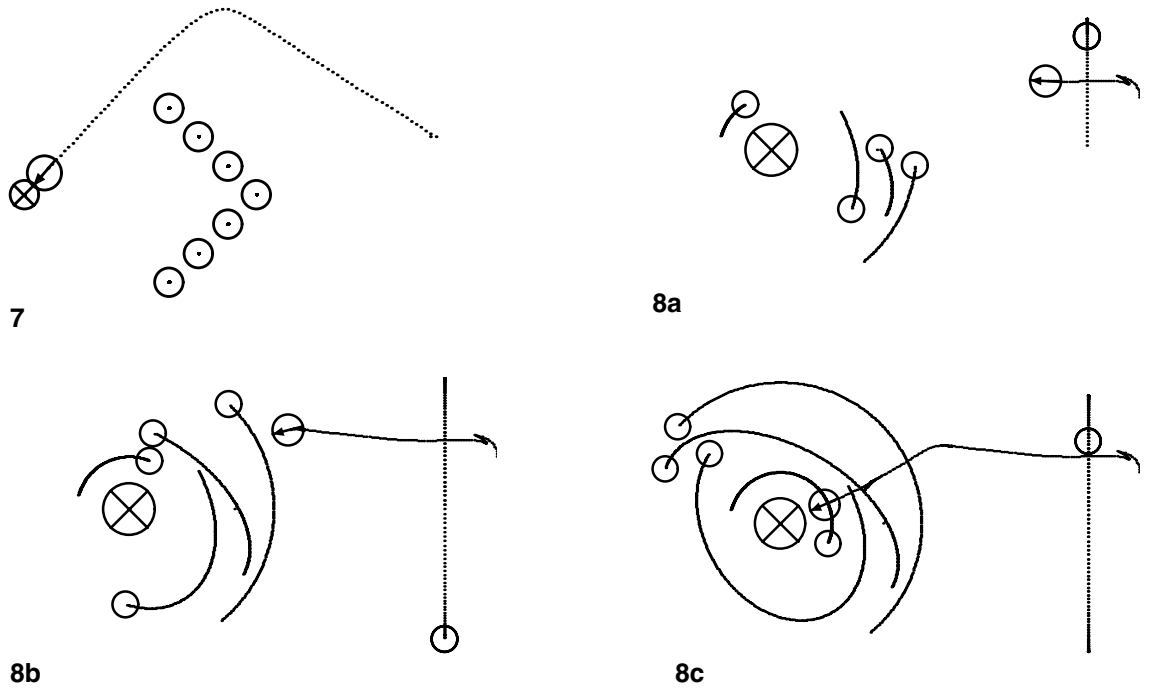

Fig. 7. First example: static target and obstacles. The *circle with an arrow* represents the agent, the *circle with the cross* is the target and the empty circles are obstacles. *Dotted lines* show the past trajectory of the moving objects. The same representation applies to the other figures

Fig. 8a–c. Second example, static target and moving obstacles

In the first experiment shown in Fig. 7 we used a single static target and a series of static obstacles between its location and the target's initial position. Note that in this case d_0 was 3.0. The agent has to deactivate the target's influence for most of the time in order to successfully avoid the obstacle barrier. In this case, the parameter d_0 has a very simple geometric interpretation, the clearance from the obstacles increases with its value. The value of d_0 changes the magnitude of the repeller in (4), but it is also used in the calculation of the parameters of α_1 and α_2 in (21).
In the second experiment, shown in Fig. 8, the scene is composed of one static target and multiple moving obstacles. The agent avoids collision by automatically changing direction and sometimes by a velocity reduction or even a complete stop. In this simulation d_0 was set to 2.0. In this example the velocity control system plays an important role in obstacle avoidance. The environment is composed of many moving obstacles close to each other, all with different trajectories and speeds. The value of d_0 has to be smaller than the one used in the first experiment (Fig. 7), this way the agent will try to reach the target (target's contribution is turned on on the task constraint system) even with more obstacles close by.
In the third experiment shown in Fig. 9, there is a group of static obstacles and a moving target. The agent successfully reaches the target and avoids the moving obstacles. In this case d_0 was set to 0.8 and the final velocity was the result of the method scaled by 0.8. This example is an illustration of this method's target tracking ability. In this case the value of d_0 had to be even smaller, so that the agent would be able to dodge closer around the static obstacles.
In the last experiment (Fig. 10), we illustrate the flexibility of our method by showing multiple moving and static targets together with moving and static obstacles. In this case the value of d_0 had to be small enough to allow the agent to pass in the spaces between the obstacle walls. In the snapshots taken here

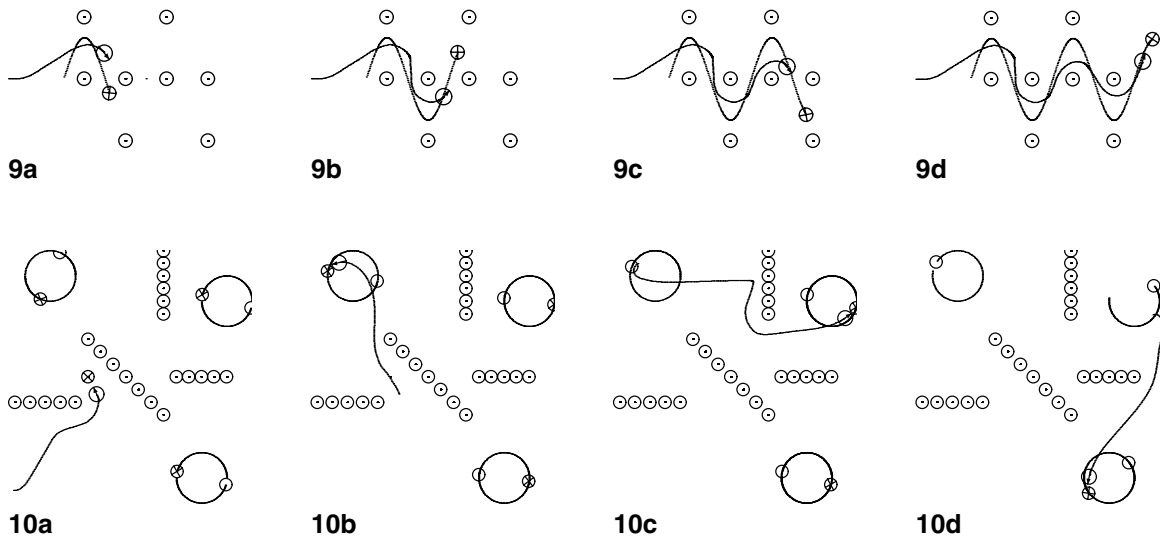

Fig. 9a–d. Third example, moving target and static obstacles

Fig. 10a–d. Fourth example, multiple moving/static targets and moving/static obstacles

d_0 was set to 1.0. Note in Fig. 10c that the target gets really close to the obstacle wall before it turns right to avoid it. This turning moment is exactly the change in the behavior, the moment where γ_{12} (see (32)) becomes larger than α_1 (see (34)). Although in this example there were multiple targets, only one (the closest) was taken in account at each time, this was done independently of the dynamical systems.

The animations on the web site corresponding to the above experiments were rendered using the rendering package Pov-Ray (WebSite b).

7 Conclusions

We have presented a technique to model autonomous agents exibiting low-level behaviors. Using a dynamical system approach we control the agent's heading direction and its velocity. We have demonstrated natural low-level agent behavior in environments with multiple targets and stationary/moving obstacles that is computationally efficient – all the experiments presented run in real time. We have paid particular attention to the setting of the few parameters and their geometrical meaning. Further we have shown how our approach can be extended to construct systems with a large number of behaviors.

This approach is appropriate as a basic low-level reactive behavior layer for an animation or virtual environment system. The increase of the number of obstacles does not affect the efficiency of the method, since for each extra obstacle only a new evaluation of f_{obs} is necessary, along with some other tests (like finding the closest obstacle for the forward velocity control). This means that the number of operations for an agent grows linearly with the number of objects affecting it.

The design of the constraint competition system parameter is not simple, but once it is done all that is needed to follow the method is to perform function evaluations, something that computers can do fast. We are currently working towards extending the above methodology to model more complex high-level behaviors, allow a different treatment for static obstacles in order to give the agent a more global knowlege of the environment. We also plan to parameterize the systems in order to achieve a family of controled reactive behaviors.

Appendix A.
Stability analysis of the obstacle and target system

In this appendix we analyse the stability for the two-dimensional case used in (10), where $\boldsymbol{w}_{\text{tar}} = \boldsymbol{w}_1$ and $\boldsymbol{w}_{\text{obs}} = \boldsymbol{w}_2$:

$$\begin{cases} \dot{\boldsymbol{w}}_1 = \alpha_1 \boldsymbol{w}_1 (1 - \boldsymbol{w}_1^2) - \gamma_{1,2} \boldsymbol{w}_1 \boldsymbol{w}_2^2 \\ \dot{\boldsymbol{w}}_2 = \alpha_2 \boldsymbol{w}_2 (1 - \boldsymbol{w}_2^2) - \gamma_{2,1} \boldsymbol{w}_2 \boldsymbol{w}_1^2 \end{cases} . \tag{21}$$

The candidate points to study are called *equilibrium* or *fixed points*, which are the ones where $\dot{\boldsymbol{w}}_1 = 0$ and $\dot{\boldsymbol{w}}_2 = 0$. Note that we can rewrite (21) as:

$$\begin{cases} \dot{\boldsymbol{w}}_1 = \boldsymbol{w}_1 (\alpha_1 - \alpha_1 \boldsymbol{w}_1^2 - \gamma_{1,2} \boldsymbol{w}_2^2) \\ \dot{\boldsymbol{w}}_2 = \boldsymbol{w}_2 (\alpha_2 - \alpha_2 \boldsymbol{w}_2^2 - \gamma_{2,1} \boldsymbol{w}_1^2) \end{cases} \tag{22}$$

and by simple inspection we can see that $\boldsymbol{w} = [0, 0]$, $\boldsymbol{w} = [0, \pm 1]$ and $\boldsymbol{w} = [\pm 1, 0]$ will result in $\dot{\boldsymbol{w}}_i = 0$. From (22), for a fixed point that has both $\boldsymbol{w}_i$ different than zero:

$$\begin{cases} \alpha_1 \boldsymbol{w}_1^2 + \gamma_{1,2} \boldsymbol{w}_2^2 = \alpha_1 \\ \alpha_2 \boldsymbol{w}_2^2 + \gamma_{2,1} \boldsymbol{w}_1^2 = \alpha_2 \end{cases} \tag{23}$$

or, in matrix notation,

$$\begin{pmatrix} \alpha_1 & \gamma_{12} \\ \gamma_{21} & \alpha_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_1^2 \\ \boldsymbol{w}_2^2 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} . \tag{24}$$

Now we can just solve the linear system for $\boldsymbol{w}_i^2$. This system can be solved when the determinant is non-zero ($\alpha_1 \alpha_2 \neq \gamma_{12} \gamma_{21}$) and when $\alpha_1 > \gamma_{12}$, $\alpha_2 > \gamma_{21}$, because we are only interested in real fixed points. The $\boldsymbol{w}_i$ are:

$$\begin{aligned} \boldsymbol{w}_1 = A_1 &= \pm \sqrt{\frac{\alpha_2 (\alpha_1 - \gamma_{12})}{\alpha_1 \alpha_2 - \gamma_{12} \gamma_{21}}} \\ \boldsymbol{w}_2 = A_2 &= \pm \sqrt{\frac{\alpha_1 (\alpha_2 - \gamma_{21})}{\alpha_1 \alpha_2 - \gamma_{12} \gamma_{21}}} \end{aligned} . \tag{25}$$

Now that we already know the fixed points ($[0, 0]$, $[\pm 1, 0]$, $[0, \pm 1]$, $[\pm A_1, \pm A_2]$) we have to check their stability. According to dynamical system theory, the point is considered stable when all the eigenvalues of the Jacobian have a negative real part. The Jacobian of (21) at a point $\boldsymbol{w}$ is:

$$J = \frac{\partial \dot{\boldsymbol{w}}}{\partial \boldsymbol{w}} = \begin{pmatrix} \alpha_1 (1 - 3\boldsymbol{w}_1^2) - \gamma_{12} \boldsymbol{w}_2^2 & -2\gamma_{12} \boldsymbol{w}_1 \boldsymbol{w}_2 \\ -2\gamma_{21} \boldsymbol{w}_2 \boldsymbol{w}_1 & \alpha_2 (1 - 3\boldsymbol{w}_2^2) - \gamma_{21} \boldsymbol{w}_1^2 \end{pmatrix} \tag{26}$$

For $\boldsymbol{w} = [0, 0]$, $\boldsymbol{w} = [\pm 1, 0]$ and $\boldsymbol{w} = [0, \pm 1]$ the Jacobian will be respectively

$$\begin{aligned} J_{(0,0)} = \begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} , \quad J_{(\pm 1,0)} = \begin{pmatrix} -2\alpha_1 & 0 \\ 0 & \alpha_2 - \gamma_{21} \end{pmatrix} , \\ J_{(0,\pm 1)} = \begin{pmatrix} \alpha_1 - \gamma_{12} & 0 \\ 0 & -2\alpha_2 \end{pmatrix} . \end{aligned} \tag{27}$$

The stability analysis for $\boldsymbol{w}_i \neq 0$ is more complex. The Jacobian will change with the change of sign (four solutions: $[\pm A_1, \pm A_2]$) because now it does not depend only on w_i^2, but also on the product $\boldsymbol{w}_i \boldsymbol{w}_j$, $i \neq j$. As an extra algebraic complication, these Jacobians (26) are not diagonal matrices.

By observing the system we can see that there are only two cases; one where $[+A_1, +A_2]$ or $[-A_1, -A_2]$, and another where $[-A_1, +A_2]$ or $[+A_1, -A_2]$

The eigenvalues for the Jacobian for the first case are

$$\begin{aligned} \lambda_1 &= \frac{1}{\det} \big(-m(d_1 + d_2) \\ &\quad - \sqrt{m^2 (d_1 + d_2)^2 + 4m d_1 d_2 \gamma_{12} \gamma_{21}} \big) \\ \lambda_2 &= \frac{1}{\det} \big(-m(d_1 + d_2) \\ &\quad + \sqrt{m^2 (d_1 + d_2)^2 + 4m d_1 d_2 \gamma_{12} \gamma_{21}} \big) , \end{aligned} \tag{28}$$

where $d_1 = \alpha_1 - \gamma_{12} > 0$, $d_2 = \alpha_2 - \gamma_{21} > 0$ and $m = \alpha_1 \alpha_2 > 0$ and $\det = \alpha_1 \alpha_2 - \gamma_{12} \gamma_{21} > 0$ (from the conditions for the existence of A_1 and A_2) .

After some long and careful manipulation, the extra stability conditions is $\gamma_{12} \gamma_{21} < 0$.

In the second case, the eigenvalues of the Jacobian are (29)

Again, after some manipulations it becomes clear that these eigenvalues are always negative if γ_{12}, $\gamma_{21} > 0$.

Remember that the case $(0, 0)$ is not desired (turning off of both contributions), so it should be unstable. Table 1 summarizes the results of the stability analysis from equation (11).

$$\lambda_1 = \frac{1}{\det}\left(-m(d_1+d_2) - \sqrt{m^2(d_1+d_2)^2 - 4m\det(m+\gamma_{12}\gamma_{21}-\alpha_2\gamma_{12}-\alpha_1\gamma_{21})}\right) \quad (29)$$
$$\lambda_2 = \frac{1}{\det}\left(-m(d_1+d_2) + \sqrt{m^2(d_1+d_2)^2 - 4m\det(m+\gamma_{12}\gamma_{21}-\alpha_2\gamma_{12}-\alpha_1\gamma_{21})}\right) .$$

Appendix B.
Parameter functions for obstacle and target system

The above considerations lead to the following design of the parameters functions α_i, γ_{ij}. First we design a fixed point detector for each of the target and obstacle functions, f_{tar} (see (2)) and f_{obs} (see (4)) respectively.
A fixed point detector for f_{tar} is defined as

$$P_{\text{tar}} = \text{sgn}\left(\frac{\partial f_{tar}}{\partial \phi}\right) e^{c_1|f_{\text{tar}}|} . \quad (30)$$

The first multiplication term (sgn function) will distinguish the attractor fixed point from the repeller fixed point of f_{tar} (cases when the target is just in front or is just behind the agent) by detecting the sign of the slope. The exponential term is one for fixed points (i.e., $f_{\text{tar}} = 0$) and is close to zero elsewhere (these values are controlled by the constant c_1).
For the obstacle case we can't use the exact same expression, since f_{obs} has a limited range of influence (due to the window function W_i (6)). There are several intervals in f_{obs} that have zero value but do not indicate the location of the obstacle (see Fig. 3). The solution to this uses the same design as in (6).

$$P_{\text{obs}} = \text{sgn}\left(\frac{\partial f_{obs}}{\partial \phi}\right) e^{c_1|f_{\text{obs}}|}\left(\sum_i W_i\right) . \quad (31)$$

Now, using P_{tar} and P_{obs}, we can design γ_{12}.

$$\gamma_{12} = \frac{e^{-c_2 P_{\text{tar}} P_{\text{obs}}}}{e^{c_2}} . \quad (32)$$

When $\gamma_{12} > \alpha_1$ (see Table 1) the deactivation of the target contribution ($\boldsymbol{w} = [0, \pm 1]$) is a stable solution. This definition results in a γ_{12} that is slightly larger than zero in most regions, but will have a narrow peak (c_2 controls how narrow) around the region where targets and obstacles are aligned.
The relation between γ_{21} and α_2 will switch between the stability of $\boldsymbol{w} = [\pm 1, 0]$ and $\boldsymbol{w} = [\pm A_1, \pm A_2]$. The only instant where we should turn off the obstacles contribution is when there are no obstacles close to the agent. The condition for it to happen is $\gamma_{21} > \alpha_2$, so we fix $\gamma_{21} = 0.05$, a low value. α_2 should be related to the distance of the obstacles to the target. The closer the obstacles are from the target, the larger α_2 should become. For this effect we use the functions D_i defined in (8), such that:

$$\alpha_2 = \tanh\left(\sum_i D_i\right) . \quad (33)$$

The hyperbolic tangent will smoothly limit the values of α_2 between 0 and 1.
We still have to select α_1. Remember that the relation between γ_{12} and α_1 will deactivate the target when $\gamma_{12} > \alpha_1$. To improve the collision avoidance, if there are close obstacles the target should be disregarded. To achieve this we decrease α_1 as α_2 increases, and we use the following equation:

$$\alpha_1 = 0.4(1-\alpha_2) . \quad (34)$$

Acknowledgements. The first author was supported by a Ph.D fellowship from CNPq, Brazil. This research is supported by NSF (IRI-97-01803,EIA98-09209), NASA (96-OLMSA-01-147), NSF-Career Award (IRI-9624604), AFOSR and ONR-YIP (N00014-97-1-0817).

References

1. Bates J, Loyall AB, Reilly WS (1992) An architecture for action, emotion, and socialbehavior. In: Proceedings of the Fourth European Workshop on Modeling Autonomous Agents in a multi Agents World, S. Martino al Cimino, Italy
2. Blumberg BM, Galyean TA (1995) Multi-level direction of autonomous creatures for real-time virtual environments. In: Proceedings of SIGGRAPH '95, pp. 47–54
3. Braitenberg V (1989) Vehicles. Experiments in Synthetic Psychology. MIT Press
4. Brooks RA (1991) Intelligence without reason. Technical Report 1293, Massachussetts Institute of Technology
5. Cohen MF (1992) Interactive spacetime control for animation. In: Computer Graphics (SIGGRAPH '92 Proceedings), 26, pp. 293–302
6. Goldenstein SK, Large E, Metaxas D (1998) Dynamic autonomous agents: Game applications. In: Proceedings of Computer Animation 98
7. Grzeszczuk R, Terzopoulos D (1995) Automated learning of Muscle-Actuated locomotion through control abstraction. In: SIGGRAPH 95 Conference Proceedings, pp. 63–70
8. Haumann DR, Parent RE (1988) The behavioral test-bed: Obtaining complex behavior from simple rules. The Visual Computer, 4(6):332–347
9. Ierusalimschy R, Figueiredo LH, Celes W (1996) Lua - an extensible extension language. Software: Practice & Experience, 26(6):635–652
10. Ierusalimschy R, Figueiredo LH, Celes W (1997) Reference manual of the programming language Lua 3.0
11. Ko H, Reich BD, Becket W, Badler NI (1994) Terrain reasoning for human locomotion. In: Proc. Computer Animation '94. IEEE Computer Society Press
12. Kokkevis E, Metaxas D, Badler NI (1995) Autonomous animation and control of four-legged animals. In: Proc. Graphics Interface'95, Quebec City, Quebec, Canada
13. Kurlander D, Ling DT (1995) Planning-based control of interface animation. In: Proc CHI95 Conf., pp. 472–479. ACM Press
14. Large EW, Christensen HI, Bajcsy R (1999) Scaling the dynamic approach to path planning and control: Competition among behavioral constraints. International Journal of Robotics Research, 18(1):37–58
15. Lethebridge TC, Ware C (1989) A simple heuristically-based method for expressive stimulus-response animation. Computers and Graphics, 13(3):297–303
16. Liu Z, Gortler SJ, Cohen MF (1994) Hierarchical spacetime control. In: Proceedings of SIGGRAPH '94, pp. 35–42. ACM Press
17. Magnenat-Thalman N, Thalmann D (1995) Digital actors for interactive television. Proc. IEEE (Special Issue in Digital Television, Part 2), 83(7):1022–1031
18. Neven H, Schöner G (1996) Dynamics parametrically controlled by image correlations organize robot navigation. Biological Cybernetics, pp. 293–307
19. Noser H, Thalmann D (1993) L-system-based behavioral animation. In: Proc. Pacific Graphics '93, pp. 133–146
20. Noser H, Renault O, Thalmann D, Magnenat Thalmann N (1995) Navigation for digital actors based on synthetic vision, memory and learning. Computer and Graphics, Switzerland
21. Perko L (1991) Differential Equations and Dynamical Systems. In: Texts in Applied Mathematics. Springer, Berlin Heidelberg New York
22. Perlin K, Goldberg A (1996) IMPROV: A system for scripting interactive actors in virtual worlds. In: SIGGRAPH 96 Conference Proceedings, pp. 205–216. ACM SIGGRAPH
23. Renault O, Thalmann NM, Thalmann D (1990) A vision-based approach to behavioural animation. The Journal of Visualization and Computer Animation, 1(1):18–21
24. Reynolds C (1987) Flocks, herds, and schools: A distributed behavioral model. In: Proc. SIGGRAPH '87, Computer Graphics, 21, pp. 25–34
25. Reynolds CW (1993) An evolved, vision-based behavioral model of coordinated group motion. In: Proc. 2nd International Conf. on Simulation of Adaptive Behavior, (ed), MIT Press
26. Ridsdale G (1990) Connectionist modelling of skill dynamics. Journal of Visualization and Computer Animation, 1(2):66–72
27. Schöner G, Dose M (1992) A dynamics systems approach to task level systems integration used to plan and control autonomous vehicle motion. Robotics and Autonomous Systems, 10:253–267
28. Schöner G, Dose M, Engels C (1996) Dynamics of behaviour: theory and applications for autonomous robot architectures. Robotics and Autonomous Systems, 16(2–4):213–246
29. Steinhage A, Schöner G (1997) The dynamic approach to autonomous robot navigation. In: Proceedings IEEE International Symposium on Industrial Electronics
30. Tu X, Terzopoulos D (1994) Artificial fishes: Physics, locomotion, perception, behavior. In: Proceedings of SIGGRAPH '94, pp. 43–50. ACM Press
31. WebSite, a URL "http://www.tecgraf.puc-rio.br/lua"
32. WebSite, b URL "http://www.povray.org"
33. Wilfong G (1988) Motion planning in the presence of movableobstacles. In: Proc. 4^{th} ACM Symp. Computational Geometry, pp. 179–288
34. Wilhelms J (1990) A "notion" for interactive behavioral animation control. IEEE Computer Graphics and Applications, 10(3):14–22

SIOME KLEIN GOLDENSTEIN studied Electronic Engineering at UFRJ and then received a MSc in Computer Science at PUC, both located in Rio de Janeiro – Brazil. Currently he is pursuing his PhD in Computer and Information Science Department at the University of Pennsylvania.

EDWARD W. LARGE is an Assistant Professor at the Center for Complex Systems at Florida Atlantic University. He received his PhD from the Ohio State University in 1994. He has held fellowships in Cognitive Science and Psychology at University of Pennsylvania, and in Artificial Intelligence at Toshiba's Research and Development Laboratory in Kawasaki, Japan. His research focuses on the dynamics of human perception, action, and cognition and the design of autonomous agents utilizing dynamical principles. His areas of specialization include music perception, auditory perception, and robotics.

DR. DIMITRIS METAXAS received his Diploma in Electrical Engineering from the National Technical University of Athens in 1986, his MSc in Computer Science from the University of Maryland in 1988, and his PhD degree in Computer Science from the University of Toronto in 1992. He joined the Department of Computer and Information Science, University of Pennsylvania as an Assistant Professor in September of 1992. Since January 1998 he has been an associate Professor in the same Department. Dr. Metaxas is the director of the VAST Lab (Vision, Analysis and Simulation Technologies Laboratory). He specializes in physics-based modeling techniques with applications to computer vision, graphics and medical image analysis and has published over 90 research papers in refereed journals and conferences. He has recently written a book titled "Physics-Based Deformable Models: Applications to Computer Vision, Graphics, and Medical Imaging", published by Kluwer Academic Publishers. He organized the first IEEE Workshop on Physics-based Modeling in Computer Vision, is an associate editor of GMIP and PAMI, is on the editorial board of Medical Image Analysis, and is a coeditor of the Computer Vision and Image Understanding 1997 Special Issue on Physics-based Modeling and Reasoning. In 1996, he received the U.S. National Science Foundation Career Award, in 1997, an Office of Naval Research Young Investigator Award. His research has received several best paper awards, including a 1998 Gold Medal in the Ninth World Congress on Medical Informatics.