Nonlinear Dynamic Systems for Autonomous Agents Navigation

Siome Goldenstein and **Dimitris Metaxas**
Computer and Info. Science Department
University of Pennsylvania
200 South 33rd Street
Philadelphia, PA 19104

Edward Large
Complex Systems and Brain Sciences
Florida Atlantic University
777 Glades Road, P.O. Box 3091
Boca Raton, FL 33431

Abstract

We present a method for the generation of real-time navigation of dynamic autonomous agents in game environments. The method is based on the use of dynamic systems theory which allows the development of sets of differential equations that exhibit distinct behaviors. The differential equations are a carefully designed set of attractor and repeller fields. Coupled together with a "behavioral" selection of the relevant contributions at each time instant are capable of exhibiting useful steering behaviors in complex environments. To avoid local minima, carefully designed noise terms are added. Using this approach we are able to demonstrate in real time behaviors such as single/multiple target tracking with stationary and moving obstacles.
Keywords: Digital Agents, Game Animation, Motion Planning, Dynamical Systems.

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 behaviors autonomously, without user intervention. Among the various methods higher levels of behavior, and movement decisions were investigated first in the pioneering work by Reynolds (Reynolds 1987), and then in work by others (Bates, Loyall, & Reilly 1992; Noser & Thalmann 1993; Reynolds 1993; Tu & Terzopoulos 1994; Noser *et al.* 1995). *AI* approaches (Lethebridge & C 1989; Funge, Tu, & Terzopoulos 1999) are capable of generating autonomous behavior, but typical such techniques require complex inferencing mechanisms. This may require considerable computational resources, raising the question of scaling up such systems as the number of independent agents grows, or when each agent has a completely different goal and behavioral directives. In addition to these factors, 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 (Grzeszczuk & Terzopoulos 1995).

In our dynamic system approach to this problem, the early methods were restricted to the generation of individual behaviors such as navigation toward a fixed goal. Recently, however, methods have been developed to allow an agent to arbitrate among a large number of potential behaviors, and to generate complex sequences of activity in a manner that is robust, yet flexible in the face of a changing environment (Steinhage & Schöner 1997; Large, Christensen, & Bajcsy" 1999; Goldenstein, Large, & Metaxas 1999). To achieve this result a second dynamical system is defined that operates in the space of task constraints. This dynamic approach, forces task constraints to compete for representation at the behavioral level. Thus, at any given time the behavioral vector field (and the observed behavior) comprises a subset of possible task constraints. The parameters of the dynamical system are chosen in such a way that the agent's behavior is appropriate to the current situation.

In this paper, we discuss an alternative methodology that has its roots in behavior-based robotics and is based on a novel way of combining differential equations exhibiting particular behaviors. According to this methodology, one defines a representation whose dimensions correspond to agent behavior. Using this type of approach, Schöner and colleagues have developed a dynamical system for robot path planning and control. In this system a set of behavioral variables, namely heading direction and velocity, defines a state space in which a dynamics of robot behavior is described (Schöner, Dose, & Engels 1996). 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 are modeled as component forces, defining attractors and repellers of the dynamical system. The individual constraint contributions are additively combined into a single vector field, which determines the observed behavior.

Here we adapt the above methodology to develop autonomous dynamic behaviors for games. In particular, we devise a set of time adaptive differential equations to rule 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 which are reactive in nature. 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 on-line 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 it is situated. An example of such a behavior is that the agent will temporarily disregard a target if there is an unsurpassable moving/stantionary obstacle immediately between them. It will then focus like a human would do to first avoid the obstacle and then refocus on the target.

Our system allows single/multiple target tracking in the presence of multiple static/moving obstacles. The design of the differential equations allows the tracking of targets whenever their position is within the visible cone of an agent requiring only the estimation of its current position. However, obstacles are processed in a local fashion based on their relative location to the agent and the target. Given our applications, in our current implementation our agents are memoryless, reactive in nature and depending on the situation (emergence of new obstacles and/or targets) their movement can be discontinuous.

In the following sections, we present previous related work in the area, the design of our system and the series of real-time experiments.

Movement Dynamics

In our methodology we combine 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. The second system controls the 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 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 (Section . In our formulation, the heading speed is modified by either the heading angle information or the relative location of the obstacles (Section).

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

In the following we present the details of each of each of the two dynamical systems.

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 models the environment (e.g., the geometry and position of the obstacles and targets) and we describe in detail below.

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 model an attractor as

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

where ψ is the angle of the target's location relative to the agent's location and a is a constant parameter.

In order to model complex environment obstacles, enemies or hazards are distinct entities. Fire-pits, for example, are clearly more dangerous than a large wall. Therefore the repeller definition should have enough parameters to model the different types of objects. We achieve this by defining a repeller to be the multiplication of three different functions, R_i, W_i, D_i, which result in being able to model the type of repeller, its distance to the agent and the extent of its influence to the environent. We therefore repeller as

$$f_{\mathrm{obs}_i} = R_i \ W_i \ D_i. \tag{3}$$

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

$$R_i = \frac{(\phi - \psi_i)}{\Delta\psi_i} e^{\left(1 - \frac{\phi - \psi_i}{\Delta\psi_i}\right)}, \tag{4}$$

where ψ_i is the angle of obstacle i and $\Delta\psi_i$ is the angle subtended by it.

[1] In differential equation terminology a fixed point is a point where the derivative is zero and acts like a trap resulting in the agent not to be able to move.

The second function, W_i, is responsible for limiting the angular range of the repeller's influence in the environment and is modeled as

$$W_i = \frac{1}{2}[\tanh(h_1(\cos(\phi - \psi_i) - \cos(2\Delta\psi_i + \sigma))) + 1], \quad (5)$$

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

$$h_1 = 4/(\cos(2\Delta\psi) - \cos(2\Delta\psi + \delta)). \quad (6)$$

Here δ is a "safety margin" constant.

The third and last function, D_i, models the influence of the obstacle to the environment by taking into account the distance of the obstacle from the agent and is modeled as

$$D_i = e^{-\frac{r_i}{d_0}} , \quad (7)$$

where r_i is the relative distance between them, and d_0 controls the strength of this influence as the distance changes.

The resulting influence on the agent from all obstacles $i = 1, \ldots, n$, is the sum of the respective repellers

$$f_{\text{obs}} = \sum_{i=1}^{n} f_{\text{obs}_i}. \quad (8)$$

Therefore, the definition of the dynamical system controlling the heading angle in (1) is obtained as:

$$\dot{\phi} = f(\mathbf{env}) == |w_{\text{tar}}| f_{\text{tar}} + |w_{\text{obs}}| f_{\text{obs}} + n. \quad (9)$$

The weights w_{tar} and w_{obs} are intended to eliminate spurious attractors that can occur by the direct summing of the nonlinear functions modeling the various obstacles and targets in the environement. These *weights* are obtained through a "constrain competition", the second dyanamical system mentioned previously and described in details in Section . They are the essence of the "low-level" behavior modeling. Finally, the noise term n is an extremely important factor. It allows the system to escape from unstable fixed points in the definition of (9) (e.g., the "center" of a repeller, where $\dot{\phi} = 0$, but any slight diplacement would make it escape from such a situation such as the situation of a ball situated on the crest of a hill).

All the above functions are carefully designed so that certain expected actions will appear in the final system. Let's first consider a simple example (Fig.), the result of a simple interaction between a target, an attractor, and an obstacle, a repeller. Let's also take the simple case that the location of the obstacle is close to the straight line between the agent and the target.

It is then clear that the agent will have to go around the obstacle in order to hit the target. In this case, the modeling of the agent's heading direction $\dot{\phi}$ based on (9) is shown in the lower right graph of Fig. . It is the composition of the target function (upper right graphic) and obstacle function (middle right graphic). The presence of two final attractors, indicated by the two arrows in the lower right graph, show the two possible obvious ways to get to the target and avoid the obstacle.

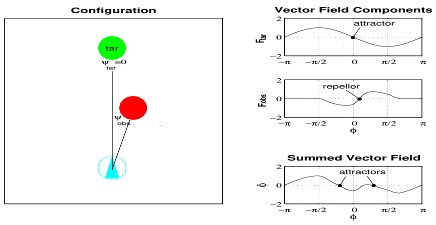
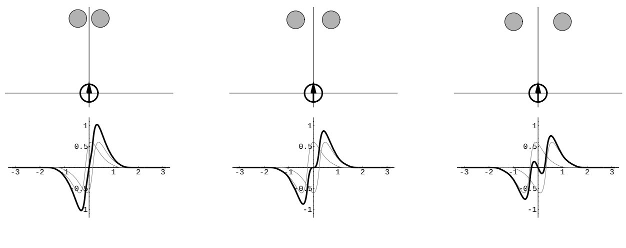


Figure 1: Attractor and Repeller interaction.

A second more complex example consists of the agent facing two different obstacles located side by side. If the obstacles are too far apart, the agent should be able to pass between them, otherwise it will have to go around them. This decision is taken automatically, as it can be seen in Fig. . Fig. 2(a) depicts the case where two obstacles are too close, Fig. 2(b) depicts the case where the distance between the obstacles is exactly equal to the size of the agent, a critical condition, and Fig. 2(c) depicts the case when the obstacles are far apart to allow the easy passage of the agent between them. For this simple case (no target and two obstacles) we have plotted at the bottom of each figure (9) as a function of the angle between the agent orientation and the y axis (assuming that the noise term n is zero). These functions clearly show that the dynamical system exhibits the correct behavior in terms of the value of the $\dot{\phi}$. For example in Fig. 2(a) $\dot{\phi} = 0$ depicts an unstable fixed point which would result in the agent trying to go through the obstacles. However, the insertion of a small amount of noise n will overcome this situation easily given the function diagram.



(a) (b) (c)

Figure 2: Interaction of repellers of two obstacles.

Constrain Competition

Individually, the attractors and repellers defined in section work well, but because of their non-linear characteristics their direct sum might not always yield the expected results. For instace, in the example shown in

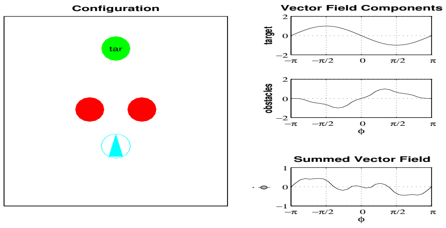


Figure 3: Attractor of the angle dynamical system.

Figure , the sum of one attractor with the two repellers propose an impossible path inbetween the two obstacles – they are too close from each other to allow the agent's passage.

To avoid this kind of problem, the composition of the attractors and repeller functions is not obtained by a direct sum, but though an weighted average by weights w_i. These weights are the result of the second dynamical system, which runs at a more refined time scale wrt the dynamical system in (9). This second system is modeled as

$$\dot{w}_i = \alpha_i w_i (1 - w_i^2) - \sum_{j \neq i} \gamma_{j,i} w_j^2 w_i, \qquad (10)$$

where in the simple case where only obstacles and targets are modeled, the state space (w_i) consists of $(w_{\text{tar}}, w_{\text{obs}})$, as used in (9).

This system is completely defined by the parameter functions α_i, termed *competitive advantage*, the parameter functions γ_{ji}, termed *competitive interaction*, and the initial value of its state space. At each time instant these parameters will be computed according to the geometry of the environment, and through (10) we obtain the weights to be used in (9). At a given time instance in the computation of (9), the computation of the weights based on (10) is not done in one step but in multiple steps. This is in order to ensure that the computed weights result in a stable fixed point of (10) as we will explain below. This explains why the whole system runs in multiple time scales – (10) is evaluated at a much faster rate compared to (9).

The correct design of the parameter functions α_i and γ_{ji} will provide the desired low level behaviors. 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. Table 1 shows the stability analysis for (10):

w_{tar}	w_{obs}	Stability	
0	0	Unstable	$\alpha_{\text{tar}}, \alpha_{\text{obs}} > 0$
0	± 1	Stable	$\gamma_{\text{obs,tar}} > \alpha_{\text{tar}}$
± 1	0	Stable	$\gamma_{\text{tar,obs}} > \alpha_{\text{obs}}$
$\pm A_{\text{tar}}$	$\pm A_{\text{obs}}$	Stable and	$\alpha_{\text{obs}} > \gamma_{\text{tar,obs}}$ $\alpha_{\text{tar}} > \gamma_{\text{obs,tar}}$

Table 1: Stability Analysis.

There are four distinct cases each one related to a different behavior. The first case, $(w_{\text{tar}}, w_{\text{obs}}) = (0, 0)$ leads to a situation where the target and the obstacle contributions in (9) are turned off. Obviously this case should be avoided, because the agent would move in an unpredictable way. To avoid this situation, both α_i should always be greater than zero.

The second case $(w_{\text{tar}}, w_{\text{obs}}) = (0, 1)$ occurs when the target's contribution is turned off (like in the case of Fig.). It is stable as long as $\gamma_{\text{obs,tar}} > \alpha_{\text{tar}}$.

The third case $(w_{\text{tar}}, w_{\text{obs}}) = (1, 0)$ happens when obstacles are ignored. This may occur, for example, when there are no obstacles near the target. This case is stable when $\gamma_{\text{tar,obs}} > \alpha_{\text{obs}}$.

The last case is when the values of both weights are nonzero, $(w_{\text{tar}}, w_{\text{obs}}) = (A_{\text{tar}}, A_{\text{obs}})$, also known as the "averaging" solution. The following two conditions have to be satisfied for this case to be stable $\alpha_{\text{obs}} > \gamma_{\text{tar,obs}}$ and $\alpha_{\text{tar}} > \gamma_{\text{obs,tar}}$. This is definetely a desirable situation.

It is important to note that conditions two and three are not mutually exclusive, and they can happen simultaneously. In this case we have a situation of *bistability*, where the stable condition that will prevail depends on the initial conditions. In this case there can be constant alternation between behaviors. A possible solution to avoid this problem is to give a "hysteresis" to the changing of the weights.

Based on the above, the design of α_i and γ_{ij} should create the different stable points according to the environment parameters. This process is described with details in (Large, Christensen, & Bajcsy" 1999), and the functions for this two-dimensional case are:

$$_{obs,tar} = \frac{e^{-c_2 P_{tar} P_{obs}}}{e^{c_2}} \qquad \gamma_{tar,obs} = 0.05$$

$$\alpha_{tar} = a_{tar} \qquad \alpha_{obs} = \tanh \sum_{i=1}^{n} D_i$$

where P_{tar} and P_{obs} are:

$$P_{tar} = \text{sgn}\left(\frac{dF_{tar}}{d\phi}\right) e^{c_1 |F_{tar}|} \qquad (11)$$

$$P_{obs} = W_{obs} \text{sgn}\left(\frac{dF_{obs}}{d\phi}\right) e^{c_1 |F_{obs}|} \qquad (12)$$

and also a_{tar} is such that whenever there is competition among targets and obstacles, targets will loose, but it will always be active if there is only a "background" noise. This is set here to be $0.4(1 - \alpha_{obs})$. D_i (7) is the function used in the distance contribution of each obstacle repeller, and their sum gives a good estimative of the concentration of obstacles around and near the agent.

Modeling the Agent's Velocity

Many different approaches can be used for modeling the forward velocity. A possible approach is to assign a constant value to the forward velocity. This approach has drawbacks in a real time changing environment: if an obstacle is suddenly in front of the agent, there might not be enough time for the agent to change direction and will result in a collision. A better approach is to have the agent move faster when there are no objects around and slower in a crowded area. The agent should also retreat when it is too close to an obstacle. An equation for the forward velocity that satisfies the above design criteria is the following

$$v = \frac{r_{min} - d_1}{t2c}, \tag{13}$$

where r_{min} is the distance to the closest obstacle, d_1 the *safety distance* and $t2c$ is the *time to contact*. This method basically applies a *constant time to contact* approach. If the closest obstacle is far then the forward velocity is large. Also if the closest obstacle is at a distance smaller than d_1, then the resulting forward velocity will be negative, meaning that the agent will retreat. Note that only obstacles in front of the agent should be considered for this calculation. We have used the above method in all our examples.

Experimental Results

The system was implemented in C, using *lua* (R. Ierusalimschy & Celes 1996) as an extensible embedded language to describe both the scene and the target(s)/agent(s) movement.

The constant a in (2) was set to 1 and the safety margin δ in (6) was set to 0.8. The Euler integration time step was 0.25 and all the simulations run in faster than real time. All experiments described below can be found on-line at *http://www.cis.upenn.edu/~siome/research/aaai2000*

In the first experiment we used a single static target and a series of static obstacles between it's location and the target's initial position. Note that in this case d_0 was 3.0.

In the second experiment the scene is composed of one static target and multiple moving obstacles. The agent avoids colision by changes of direction and sometimes by a velocity reduction or even a complete stop. In this simulation d_0 was set to 2.0.

In the third experiment 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.

In the last experiment we illustrate the flexibility of our method by showing multiple moving and static targets together with moving and static obstacles . The constant d_0 was set to 1.0.

In the videos all experiments appear rendered based on the use of the rendering package Pov-Ray.

Conclusions

We have presented a technique to model autonomous agents navigation for game environments. Using a dynamical system approach we control the agent's heading direction and its velocity. We have demonstrated natural low-level agent behavior in envirnments with multiple targets and stationary/moving obstacles.

There is a whole set of parameters to control the expected low-level behavior of the overall system. Unfortunately this set is not intuitive for an animator. We are currently working towards making the whole process of modeling a behavior both more high-level and "user-friendly" as well as flexible enough for different applications. New functions are being analyzed in order to achieve a significantly larger and more complex set of behaviors.

Acknowledgments

The first author was supported by a Ph.D fellowship from CNPq, Brazil. The second author was partially supported by an ONR YIP and an NSF Career Award.

References

Bates, J.; Loyall, A.; and Reilly, W. 1992. An architecture for action, emotion, and socialbehavior. In *Proceedings of the Fourth Europeans Workshop on Modeling Autonomous Agents in a multi Agents World*.

Funge, J.; Tu, X.; and Terzopoulos, D. 1999. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. *Proc. of SIGGRAPH 99* 29–38.

Goldenstein, S.; Large, E.; and Metaxas, D. 1999. Non-linear dynamical system apprach to behavior modeling. *The Visual Computer* 15:349–369.

Grzeszczuk, R., and Terzopoulos, D. 1995. Automated learning of Muscle-Actuated locomotion through control abstraction. In *Proc. of SIGGRAPH*.

Large, E.; Christensen, H.; and Bajcsy", R. 1999. Scaling the dynamic approach to path planning and control: Competition among behavioral constraints. *Int. Journal of Robotics Research* 18(1).

Lethebridge, T., and C, C. W. 1989. A simple heuristically-based method for expressive stimulus-response animation. *Computers and Graphics* 13(3).

Noser, H., and Thalmann, D. 1993. L-system-based behavioral animation. In *Proc. Pacific Graphics*.

Noser, H.; Renault, O.; Thalmann, D.; and Thalmann., N. 1995. Navigation for digital actors based on synthetic vision, memory and learning. *Computer and Graphics*.

Perko, L. 1991. *Differential Equations and Dynamical Systems*. Springer Verlag.

R. Ierusalimschy, L. F., and Celes, W. 1996. Lua - an extensible extension language. *Software: Practice & Experience* 26(6).

Reynolds, C. 1987. Flocks, herds, and schools: A distributed behavioral model. In *Proc. SIGGRAPH '87*.

Reynolds, C. 1993. An evolved, vision-based behavioral model of coordinated group motion. In *Proc. 2nd Int. Conf. on Simulation of Adaptive Behavior*.

Schöner, G.; Dose, M.; and Engels, C. 1996. Dynamics of behaviour: theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems* 16(2–4).

Steinhage, A., and Schöner, G. 1997. The dynamic approach to autonomous robot navigation. In *Proc. IEEE Int. Symposium on Industrial Electronics*.

Tu, X., and Terzopoulos, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proc. of SIGGRAPH*.