

Anticipation of Periodic Movements in Real Time 3D Environments

Vincent Labbé, Olivier Sigaud, and Philippe Codognet

poleia-LIP6, 8, rue du Capitaine Scott
75015 Paris France
{labbe,sigaud,codognet}@poleia.lip6.fr

Abstract. In this paper we present a method to anticipate periodic movements in a multi-agent reactive context, of which a typical example is the guard who patrols. Our system relies on a dynamic modeling of motion, based on a state anticipation method. This modeling is achieved with an incremental learning algorithm, dedicated to 3D real time environments. We analyze the performance of the method and demonstrate its usefulness to improve the credibility of pursuit and infiltration behaviors in front of patrolling agents.

1 Introduction

In a dynamic multi-agent environment, the reactive anticipation of movements of an opponent may be crucial to survive. In this paper, we focus on the anticipation of the motion of an agent who follows a well-defined periodic path. The guard who goes his rounds is a typical example. This guard is going to face another agent, for whom he will be either a prey or a predator. If the guard is a predator, it may induce several problems for the other agent: how to cut his trajectory without being seen, how to mislead him in another direction, etc. In the other case, the agent has to intercept the guard in what he thinks is the most favorable way for himself. Various criteria can be used, such as the quickest way, the one with the least hazard, etc. The method presented in this paper deals with this kind of matters. It is based on an incremental modeling of periodic movements carried out by a learning algorithm, dedicated to 3D real time environments.

Hereafter, we will use the verb “to patrol” in the meaning of “to patrol along a well-defined periodic path, while no major perturbation stops one’s behavior”.

Modeling this kind of patrol is interesting for video games, particularly in FPS¹ and strategic games. When a player commands several agents, it is interesting to be able to give them high level orders such as “watch a zone”, “infiltrate the enemy’s area” or “lay an ambush”. The “zone watching” behavior already exists in strategic games like StarCraft or Conflict Zone. But players miss infiltration or ambush behaviors

¹ “First Person Shooter”

orders. Our anticipation method allows one to get this kind of behaviors facing patrols.

This paper is organized as follows. In the next section we present related work regarding movement anticipation, especially in 3D real time surroundings. Then our method is explained in section 3. In section 4, our main results are presented and discussed. The following section discusses the possibility to apply our algorithm for the video game prey/predator context. In section 6, we discuss the benefits and limits of our method. Finally, we highlight the role of anticipation in our approach.

2 Background

A classification of anticipatory mechanisms is proposed in [1]. It describes four kinds of anticipation: implicit, payoff, sensorial and state-based. Our approach deals with the fourth category: state anticipation. To anticipate the movement of a patrolling agent, we build a model of the movement. This model allows the explicit simulation of the future motions of the agent. These predictions are directly used to improve escape or pursuit abilities and to obtain infiltration and ambush behaviors.

Christophe Meyer’s SAGACE method [4] provides another example of state anticipation. His system learns how to anticipate the actions of a human opponent in the context of repeated games with complete or incomplete information. It is based on two Learning Classifier Systems (LCSs) whose rules can evolve thanks to a Genetic Algorithm. The first one models the opponents’ behaviors, the other one plays depending on this model. These systems refine themselves during each game. Tested on the game ALESIA, with opponents using a fixed strategy, this method gives very good results because it determines an adapted strategy. However, it is designed to be used in “turn by turn” games. It seems that the SAGACE method is difficult to adapt to continuous real time games because LCSs scale poorly to continuous domains [2].

For continuous environments, Craig Reynolds developed algorithms to simulate motion behaviors in a visually credible way, notably of groups of animals, such as a fish shoal or a bird flock. These algorithms use simple and very quickly calculated mechanisms and which, combined, well adjusted and applied in a reactive way (about 30 times a sec) offer realistic motion results. A pursuit behavior endowed with a capacity of anticipation is proposed in [5]. It consists in leading the predator towards an estimation, at T steps of time ahead, of the prey’s position. To realize this prediction, the predator supposes that the prey’s speed will remain constant. The issue is to determine T . Reynolds suggests a simple method based on the distance between the prey and the predator, which is: $T = Dc$, where D is the prey/predator distance and c a parameter. This prediction is carried out at each time step, using the last observed velocity vector of the prey. This method does not pretend to be optimal, but improves the credibility of the pursuit behavior. Indeed, the extremely quick calculation does not reduce the agent’s reactivity, and it seems to pursue his prey in a more efficient manner.

However, such anticipation finds its limits in the periodic motion framework. As an example, if the prey is following a circular closed path, if it is going faster than the predator and the parameter c is not well adjusted, this one will move on a smaller circle inside the one followed by the prey, without ever reaching it (see fig 1).

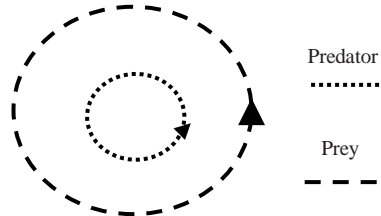


Fig. 1: A pathological case for Reynolds' pursuit behavior

Our method solves this kind of problem thanks to a more sophisticated model of the prey's motion than Reynolds' one (which is limited by the fact that the velocity vector is supposed constant). This model is learned in an incremental and reactive way and provides an estimation of a crossing point with the prey's trajectory without reducing the predator's reactivity.

Cyril Panatier [6] exposes another motion anticipation method, based on potential fields. The method is tested in a real time 3D environment with agents divided into two categories: friends and opponents. Each side has to push some pucks towards an opposed zone. Only one agent, the "adaptive" one, tries to interpret the others' movements in order to guess who his friends are. It assumes that the agent's velocity results in a combination of attraction/repulsion forces caused by other agents. To calculate the others' attraction/repulsion coefficients, the adaptive agent uses his surrounding perception, i.e. every agent's position, and his memory of the last perception and calculation. Thus it learns the potential field functions of every agent, in an incremental manner. Once the coefficients are calculated, the adaptive agent gets a global transition function letting him simulate the movement of each agent. In order to choose his next behavior it performs simulations of each agent motion. Then the selection is made by comparing the outcomes of each behavior.

This kind of anticipatory mechanism eventually endows one with infiltration or ambush abilities, but not facing a patrol. Indeed, by definition, the patrol's path is defined independently of the current behavior of other agents. Therefore the interpretation of a patrolling guard, by the mean of potential fields, would not result in a correct model.

In the video game Quake III, more centered on the prey/predator relationship, John Laird explains how his quakebot [7], based on Soar architecture, can anticipate some movements and other actions of his opponents. The bot starts by modeling its opponent from its observations. The model consists in observable variables like position, health, and current weapon, as well as non observable ones such as the current aim, which has to be guessed. Then he predicts the future behavior by simulating what he should do if he was in his opponent's situation with his own tactical knowledge. Thus Laird assumes that the enemy's goals and tactics are basically the same

as the quakebot's. Using simple rules to simulate his opponent's actions, the bot anticipates until he finds an interesting or too uncertain situation. The prediction is used to lay an ambush or to deny the enemy of a weapon or life bonus. This anticipatory mechanism, whose calculation takes a lot of time, is used only under some conditions in order not to reduce the bot's reactivity. Designed to anticipate human players' actions, this method seems unable to anticipate patrol movements. Indeed, in order to anticipate the guard's periodic motion, the bot should be able to consider the "patrol" mode like an internal state and above all, to model the path. As far as we know, this modeling capacity is not part of the quakebot's characteristics.

We propose to address this problem thanks to our periodic motion modeling method viable in a highly reactive 3D environment such as Quake III's.

3 Movement learning algorithm

Our algorithm is dedicated to a real time modeling of periodic motion in 3D worlds. The model performs a linear approximation of motion so as to anticipate the next position. The velocity is supposed constant on each segment, equal to the average of velocities observed between both extremities of the segment. When the agent accelerates, the anticipation happens to be wrong and the model is updated. In order to do so, edges are adapted permanently to fit to the motion points that present the strongest accelerations. This dynamical adaptation is made through experience, justifying the notion of learning. It is driven by comparisons between predictions and observations and implies a motion abstraction. Hereafter, we call "track" the motion model.

3.1 Model

The model itself consists of a circular chained list. The circularity of the list means that the motion is considered periodic a priori. The maximal size of the list is the only model parameter. List elements are called "edges", they are made of four components:

1. a position vector in 3D space
2. a distance: estimation of the distance between last and current edges.
3. a time: estimation of run time between last and current edges.
4. a radius: estimation of the maximal distance between prediction and observation on the segment.

Figure 2 illustrates the relationship between a 2D motion and the track. There are:

- The motion trajectory: undulant pull.
- The corresponding learned model: arrows linking A to B, B to C etc., edges and the grey circles that symbolize uncertainty radius of these edges.
- The position of the observed object at instant t: the star
- The prediction of the model for the same instant t: the dotted circle.

As shown by the orientation of the arrows, A is before B in the list. If A is the last point of the trace reached, then the motion prediction is made according to the [AB]

segment and the uncertainty radius in B. The prediction is made of a position and a radius representing an uncertainty. In other words, it points out that the object should be in a perimeter around an accurate point. This uncertainty can be interpreted like a motion abstraction carried out by the algorithm.

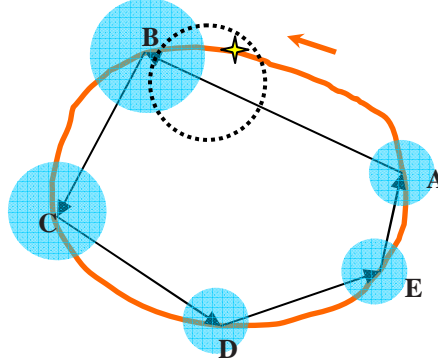


Fig. 2: example of track

On this scheme, the prediction is a success because the star is inside the dotted circle. Along time, the dotted circle moves from A to B where its radius will become the next edge's one, etc. Its velocity is constant on a segment: it represents an approximation of the average speed of the observed object on this segment.

3.2 Algorithm outline

At any time, the algorithm compares the position vector and observation date with the prediction realized thanks to the current model, and the model is updated according to the comparisons. When the prediction is wrong, a new edge corresponding to the current observation is added in the list. Then, in order to keep the model size constant, the algorithm must suppress an edge in a relevant way. The selected edge is the one whose suppression would involve a minimal loss of accuracy in prediction. The rest of the list is then updated in order to maintain the reliability of the model.

We define the model uncertainty as the average of the edges' radius weighted by the length of the corresponding segments. The model accuracy is the opposite of uncertainty. The selection of an edge is computed by anticipating the update of the list. Indeed, the uncertainty of the model grows during the update. Let A, B and C be three consecutive points of the track (see Fig. 3). Considering that B is selected then C is updated in the following way:

- $D_{AC} = D_{BC} + D_{AB}$
- $T_{AC} = T_{BC} + T_{AB}$
- The radius R_{t+1} is computed as:

$$R_{t+1} = \max(R_t, \|BH\|) \quad \text{where} \quad H = A + AH \quad \text{with} \quad AH = \frac{AC}{\|AC\|} \cdot \frac{T_{AB}}{T_{AB} + T_{BC}}$$

Where D_{AB} and D_{BC} are the distances from A to B and from B to C; T_{AB} and T_{BC} indicate the running time in the same way. H points to the position estimated when the real object is in B. At this moment, the gap between prediction and reality is theoretically² maximal on the segment [AC].

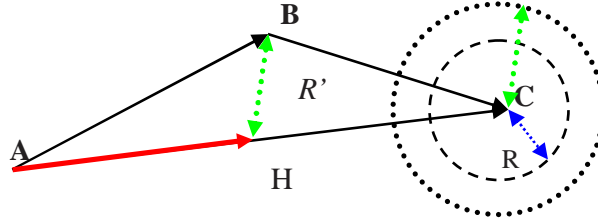


Fig. 3: uncertainty calculation

Another problem to solve is the discovery of the period. To know when the starting point is reached, the algorithm uses an elementary distance based on the average of the observed movements. If, for any reason (noise, sampling, etc.), the algorithm does not detect when the starting point is reached, then a dilation phenomenon appears, the track trying to model two or more periods as a single one. To deal with this problem, we have developed a mechanism that deletes the current first edge of the model when the position of a non-predicted observation is very close to another edge. This heuristic works quite well but can involve the inverse effect: the retraction phenomenon, when the trajectory has many intersection points with itself. The more intersection points, the more possibilities to meet a situation where the anti-dilation mechanism is triggered. It can result in a track reduced to a single segment to model the whole trajectory. The algorithm complexity in time and space is $O(n)$ where n is the maximal size of the list. We also designate this size as “model complexity”.

4 Experiments and results

We have tested our algorithm on several noisy periodic movements. The motion learning performance is evaluated according to two criteria linked with predictions:

- The track *reliability* is the right predictions rate, during one period.
- The *uncertainty*, as defined above.

² « theoretical » meaning in case where one starts exactly from A, what's not always right; so this is why we need to increase the radius.

Hereafter, we present results on two particular trajectories, in 8 and in W shapes. We define critical points as the points that present an outstanding acceleration. The intersection points of the trajectory with itself are also important. They make the discovery of period more difficult because of the mechanism used to solve the dilation problem. Thus the number of critical and intersection points determine the modeling difficulty. For these examples, the guard moves with a constant speed. Therefore the critical points correspond only to the curves' acute turns. Figure 4 shows the 8-trajectory endowed with 4 critical points (white squares in the initial curve picture) and the three tracks obtained after two periods. These tracks consist of 4, 12 and 24 edges. The circles symbolize the uncertainty radius of tracks' edges.

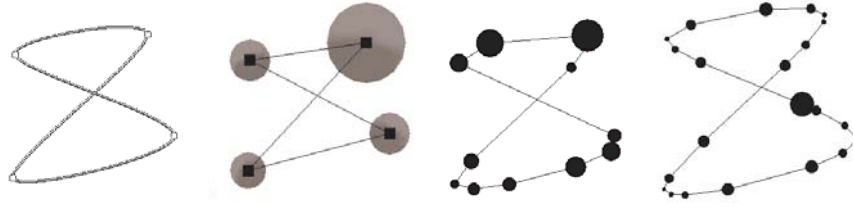


Figure 4: initial trajectory and models with 4, 12 and 24 edges

We can see that the edges are distributed around the critical points. Light curves contain only few edges. It illustrates the fact that the algorithm selects the trajectory points that present strong acceleration. Figure 5 presents the W-trajectory and some track obtained after two periods with 5, 10 and 20 edges models.

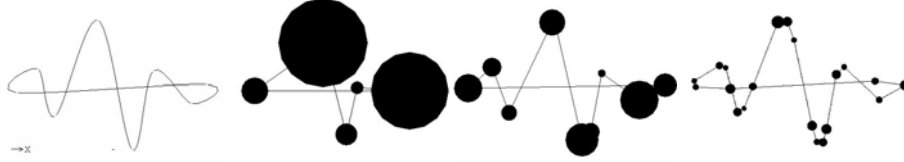


Figure 5: Initial trajectory and models with 5, 10 and 20 edges

This more complex trajectory has height critical points and five intersections. These pictures corroborate the trend brought to mind by the 8-trajectory: the more complex the model, the more accurate the predictions. The following graphs present the comparisons of uncertainty and reliability of three models of increasing complexity, along 10 periods regarding the two motions presented above. The values represent the averages from a sample of 10 tests.

Figure 6 presents the evolution of uncertainty:

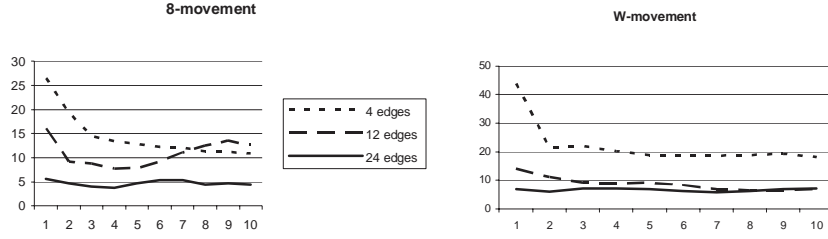


Fig. 6: uncertainty evolutions

The average uncertainty on predictions tends to increase with the trajectory complexity and to decrease with time and model complexity. After 3 periods, the uncertainty is relatively stationary. The curve that represents 12 edges model evolution on the 8-movement gets higher after the fifth period. This corresponds to the dilation phenomenon. Figure 7 deals with reliability evolution:

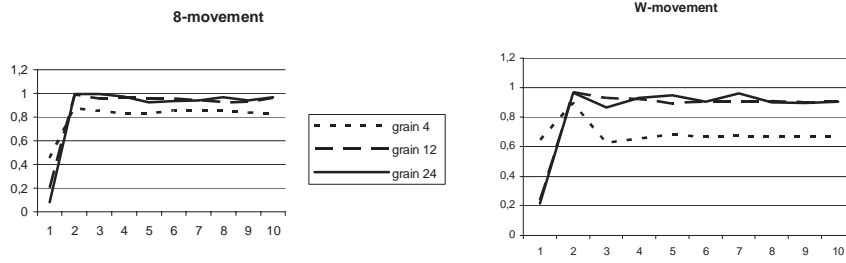


Fig. 7: reliability evolutions

We can see that the reliability of the predictions is almost stationary after the third period. Furthermore, the 4 edges model is not complex enough to allow a reliable anticipation of the W-movement.

5 Patrol anticipation application

As outlined in the beginning of this paper, our learning algorithm has been designed to anticipate patrol movements in a real time 3D environment, such as rounding guards in a video game surrounding. We can anticipate the guard's motions according to two main objectives: interception or avoidance.

5.1 Interception

Our interception method is based on the following principle: from the motion model of the guard, the agent decides to intercept him at the first edge that he can reach

before the guard by optimizing a particular criterion. The nature of this criterion determines the type of behavior. For instance:

- Minimizing interception time to obtain a pursuit behavior
- Maximizing position benefits to obtain an ambush behavior

The choice and computation of this criterion depend on the application, which is not the point of this paper. To check if the agent can reach a position before the guard, one must estimate the running times. The guard's running time is easy to estimate thanks to the track model corresponding to its trajectory. As far as the agent is concerned, it depends on his own model. From a simple calculation in function of the distance and average speed, the estimation remains dependent on the application and computation possibilities. In order to demonstrate the feasibility of our method, we have implemented a simple demonstration of a pursuit behavior in the context of a "toy" video game environment with a 3D real time surrounding.

5.2 Avoidance

The guard's avoidance behaviors are investigated from the point of view of infiltration, i.e. when an agent has to cross the guard's trajectory without being seen nor touched. We propose a simple anticipation method allowing infiltration behaviors.

We assume that the agent does not adapt his reactive steering parameters: once he starts moving, he will not stop anymore until he reaches his goal. The challenge is to start moving at the right moment to avoid being seen or touched. In order to determine this moment, the agent must simulate his movement as well as the guard's on several steps. From the current situation, he checks at each step if he is not in a critical situation. Critical situations can be collisions, inclusions in the guard's perception area, etc. If the agent fails, then he starts again until he finds a favorable moment to move. Here again, we have demonstrate this behavior in the same context as above.

6 Discussion

The anticipatory mechanism presented in this paper endows an agent with interception or avoidance capabilities in front of a patrolling guard without equivalent in the video game research literature. We have empirically demonstrated how this mechanism could be used, but we did not implement it yet in a actual video game. Actually, we must confess that there are not so many video game contexts in which our mechanism could be used. In most cases, the agents are designed to anticipate the human players' behavior [4] [7]. But the human player seldom behaves in a periodic way. The only favorable context is when a software agent is confronted to another software agent. This is the case, for instance, when the human player can give high level orders like "keep a zone" or "invade this building" and the agent must realize



the behavior on its own. Very few video games offer this possibility so far³ but we believe that this situation will change with the raise of interest in AI in the video games industry.

From a more technical point of view, our algorithm still suffers from some limitations. Given the dilation and retraction problems outlined in section 3, one needs to determine the horizon of the simulation, the number of steps and the complexity of the problem in a concrete way. In order to keep the algorithm reactive and to improve the behaviors significantly, we must find a good compromise between the resources used and the accuracy of predictions. One way to act on this compromise is to tune the model complexity. Experiments show that to obtain reliable and accurate predictions, the model's complexity must be sufficient in regard to the movements that must be anticipated. The more numerous the critical points and self-intersections are, the more complex the model of the periodic motion must be. As we have shown, this concern is not critical since the algorithm complexity is in $O(n)$. Another matter of discussion is the possibility to generalize the algorithm. An extension to the case where several guards patrol in the same area appears quite straightforward. On the contrary, its extension to non-periodic movements seems to be more difficult and should be studied in greater detail.


7 Conclusion

In this paper we have dealt with anticipatory mechanisms in two ways: on the one hand, anticipation is used to drive learning mechanisms, on the other hand, the learned model results in the possibility to anticipate periodic movements through mental simulations. Our system uses a dynamic modeling of movement, driven by an anticipatory mechanism. It is a clear case of state anticipation where the prediction error is used as a feedback signal to improve the model at each time step. Thanks to an incremental learning algorithm dedicated to real time 3D environments, it endows an agent, among other things, with infiltration and pursuit behaviors. Based on distances computation, the algorithm can easily be extended to N dimensions space. Though our system has proven efficient on toy problems, its applicability to an actual video game still needs to be demonstrated, which we hope to do in the near future.

References

1. Butz, M.V., Sigaud, O. and Gérard, P. (2003)  *Internal Models and Anticipations in Adaptive Learning Systems* In Butz et al. (Eds) LNCS 2684 :*Anticipatory Behavior in Adaptive Learning Systems* , pp 87-110,  © Springer Verlag.

³ StarCraft and Conflict Zone are notable exceptions.

2. Flacher, F. and Sigaud, O (2003).  Coordination spatiale émergente par champs de potentiel, *Numéro spécial de la revue TSI : Vie Artificielle*, A. Guillot et J.-A. Meyer (Eds), Hermès, 171-195.
3. Parenthoën, M., Tisseau, J. and Morineau, T., in *Rencontres Francophones de la Logique Floue et ses Applications (LFA'02)*, 219{226, Montpellier, France, 21-22 Octobre 2002.
4. Meyer, C., J.-G. Ganascia, and Jean-Daniel Zucker. (1997). *Modélisation de stratégies humaines par Apprentissage et Anticipation génétiques*. Journées Française de l'Apprentissage, JFA'97, Roscoff, France
5. Reynolds, C. W. (1999) Steering Behaviors For Autonomous Characters, in the proceedings of Game Developers Conference 1999, Miller Freeman Game Group, San Francisco, California, pages 763-782.
6. Panatier, C., Sanza, C., Duthen Y. *Adaptive Entity thanks to Behavioral Prediction*. in: *SAB'2000 From Animals to Animats, the 6th International Conference on the Simulation of Adaptive Behavior, Paris*. Meyer, ... , p. 295-303, 11 septembre 16 septembre 2000. Accès: <http://www-poleia.lip6.fr/ANIMATLAB/SAB2000/>
7. Laird, J. It Knows What You're Going to Do: Adding Anticipation to a Quakebot. in AAAI 2000 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment, March 2000: AAAI Technical Report SS-00-02.