HAL open science

AntReckoning: Dead Reckoning using Interest Modeling by Pheromones

Amir Yahyavi, Kévin Huguenin, Bettina Kemme

To cite this version:

Amir Yahyavi, Kévin Huguenin, Bettina Kemme. AntReckoning: Dead Reckoning using Interest Modeling by Pheromones. 10th ACM/IEEE International Workshop on Network and Systems Support for Games (NETGAMES), Oct 2011, Ottawa, ON, Canada. 10.1109/NetGames.2011.6080977 . inria-00616877

HAL Id: inria-00616877

https://inria.hal.science/inria-00616877

Submitted on 14 Mar 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

AntReckoning: Dead Reckoning using Interest Modeling by Pheromones

Amir Yahyavi
McGill University
Montréal, QC, Canada

Kévin Huguenin
McGill University
Montréal, QC, Canada

Bettina Kemme
McGill University
Montréal, QC, Canada

***Abstract*—In games, the goals and interests of players are key factors in their behavior. However, techniques used by networked games to cope with infrequent updates and message loss, such as dead reckoning, estimate a player's movements based on previous observations only. The estimations are typically done using dynamics of motion, taking only *inertia* and *external factors* (e.g., gravity, wind) into account while completely ignoring the player's goals (e.g., chasing other players or collecting objects).**

This paper proposes AntReckoning: a dead reckoning algorithm, inspired from ant colonies, which models the players' interests to predict their movements. AntReckoning incorporates a player's interest in specific locations, objects, and avatars in the equations of motion in the form of *attraction forces*. In practice, these points of interest generate *pheromones*, which fade and spread in the game world, and are a source of attraction.

Our simulations using mobility traces from World of Warcraft and Quake III show that AntReckoning improves the accuracy by up to 30% over traditional dead reckoning techniques.

I. Introduction

Interactive games have become very popular reaching an unprecedented scale, and therefore, forcing major on-line multi-player gaming platforms to develop a range of techniques to increase scalability. In interactive games, position update messages account for the largest portion of the network traffic [1], calling for techniques that predict player movements in order to reduce the update rate while keeping the error on player position low: the current position of an avatar is usually estimated from previous positions; only when the error is higher than a threshold a new position update is sent, thus reducing the update rate [2, 3]. Upon reception of a new update a convergence step is performed to hide the estimation errors from the player in rendering the motion [4]. Such techniques also help in coping with message loss or delay by extrapolating the new position when the new update is not received in time.

Dead reckoning estimates the position of an object from the equations of motion, based on previous observations, and has been successfully used in a number of areas including distributed simulations [2, 5] and games [3, 4]. While the performance of dead reckoning, in its current form, is good for vehicles moving smoothly [6], it may degrade in games where players, driven by their short-term environment-related goals, make frequent and sudden changes to their movements.

A typical example of this is a wounded player, moving in a given direction, having both an attacker shooting at him and a health pack in his vision field: he tends to maintain the same motion (because of inertia), while trying to move towards the health pack and to evade from the attacker. As games generally have relaxed physical rules sudden drastic changes in movements (e.g., rapid U-turn) can occur. These unpredictable changes may dramatically reduce the performance of dead reckoning as it only takes mechanics into account.

Motivated by this example, we argue that key factors in an avatar's motion are not only inertia but also the objectives of the game as well as entities in his vicinity that we refer to as points of interest. Following this line of reasoning, we propose AntReckoning. To the best of our knowledge it is the first interest-based approach to dead reckoning. The main concepts involved in AntReckoning are as follows:

- Each entity is assigned a given attractiveness leading to the generation of pheromones that fade and spread in the world;
- Pheromones in the vicinity of an avatar exert attraction on it. Attraction is integrated in the equations of motion, under the form of forces, to estimate its future position.

The main contributions of AntReckoning are: (1) to incorporate players' interests into the equations of motion used for dead reckoning, and (2) to use pheromones to model such interests, taking temporal and spatial aspects into account. Moreover, pheromones offer a practical solution to the decentralized implementation of interest-based dead reckoning.

We evaluated AntReckoning using mobility traces from World of Warcraft and Quake III. Our simulation results show that AntReckoning consistently outperforms dead reckoning for a carefully chosen set of parameters, and can improve the average accuracy of the estimation by up to 30% over traditional dead reckoning. Since AntReckoning involves several parameters we identify the key ones and perform a preliminary sensitivity analysis to evaluate their respective impact on the accuracy. We also discuss solutions to set game-related parameters, such as attractiveness of objects and avatars.

II. Background

In multi-player games, players control their avatars which evolve in a virtual space called the *game world*. Clients regularly exchange the states of their avatars, including their position. They do so directly or through servers. The objective of the game is to accomplish missions such as going to a given location, collecting objects, or killing other entities.

In networked games, dead reckoning exploits information contained in the last state updates to extrapolate the time-dependent future state of entities. Dead reckoning enables less

frequent exchange of state updates and also helps coping with update message loss. A typical dead reckoning problem is the estimation of the position of a moving entity, which is required for rendering the virtual world at the clients, between two successive updates. In this situation, the key variables are the kinematic ones: the last position $\mathbf{x}_t$ and velocity $\mathbf{v}_t = \dot{\mathbf{x}}_t$ of the entity and possibly its acceleration $\mathbf{a}_t = \ddot{\mathbf{x}}_t$ (as defined by the *IEEE Standard for Distributed Interactive Simulation* [7]), where t represents time. Extra information that helps estimate the forces the entity is subject to can also be included.

The study of the trajectory of objects rely on the dynamic equations of motion, and more specifically, on Newton's second law which links the *acceleration* of an object, its *mass* and the *forces* it is subject to: $\mathbf{a}_t = \frac{1}{m}\sum \mathbf{f}$. When a closed form expression of the (sum of the) forces is known, the ordinary differential equation characterizing the trajectory of the object can be obtained from this relation, and formally or numerically solved. The future position of the object is thus fully determined by its initial position and velocity.

In practice, a polynomial approximation derived from Taylor series expansion of the position, as a function of time, is used to predict the position in a near future. For instance, the second-order polynomial predictor is given by:

$$\mathbf{x}_{t+\delta t} = \mathbf{x}_t + \mathbf{v}_t\ \delta t + \frac{1}{2}\mathbf{a}_t\ \delta t^2 \tag{1}$$

Note that such predictors are accurate only for small values of δt (compared to the rate at which players move and change their direction). It has been shown that using derivatives of orders higher than two usually results in a negligible improvement in the prediction [8, 9]. As a result, the use of first and second order derivatives is usually preferred and estimating the velocity and acceleration and sending them with the current position is sufficient for *short-term* dead reckoning.

Estimating velocity and acceleration is commonly done from previous observation using exponential moving average (EMA). In short, EMA estimates the velocity by a weighted sum of its current value, namely the difference between the current position $\mathbf{x}_t$ and the last position $\mathbf{x}_{t-\delta t}$ divided by the time interval, and the last estimation:

$$\mathbf{v}_t = \alpha_v\ \frac{\mathbf{x}_t - \mathbf{x}_{t-\delta t}}{\delta t} + (1-\alpha_v)\ \mathbf{v}_{t-\delta t} \tag{2}$$

$$\mathbf{a}_t = \alpha_a\ \frac{\mathbf{v}_t - \mathbf{v}_{t-\delta t}}{\delta t} + (1-\alpha_a)\ \mathbf{a}_{t-\delta t} \tag{3}$$

Such an approach, taking into account previous estimations, has proven to have a beneficial smoothing effect [10].

III. Motivation and design rationale

When using a first order predictor, the velocity of the avatars is usually estimated from short term history of their states, thus taking into account only their *inertia* in the prediction. To increase the accuracy of dead reckoning, one needs to incorporate the second derivative (i.e., the acceleration) and estimate it from the forces the avatar is and will be subject to.

Estimating the forces an avatar is subject to is a difficult problem since it depends on the player's decisions: For instance, a player can suddenly accelerate to have his avatar in the game world chase another avatar. The intuition behind this reasoning is that a player is more likely to follow another player or to go to pick up a game item (e.g., a weapon) rather than just continuing on its current path. Key here is the fact that players' moves are usually driven by specific goals and interests that are themselves related to features of the virtual world. Indeed, in games such as World of Warcraft, players are interested in specific locations, certain objects, and other avatars, referred to as points of interest (POI).

More specifically, an interest-based dead reckoning algorithm should handle the following situations: (1) Game objects attract players specially if they are valuable or when the player is in urgent need of them. For example if a player comes across a powerful weapon he will most likely move towards it and pick it up. Similarly, players running out of ammo or wounded would pick up ammunitions or health packs. Attraction therefore depends on both the objects of interest and the moving avatar. (2) Players are attracted by the avatars they are chasing or they want to trade with, and repulsed by the avatars that are chasing them. (3) Interesting and popular locations (e.g., top of a hill, corner, *etc.*) in the game, namely hotspots, are a source of attraction. Such attraction points can be inferred from the history of the movements of all players.

In order to use the framework of dynamics while considering game strategy for predicting avatar movement, we incorporate player interest in the second order predictor in the form of attraction forces. The intensity of the attraction forces exerted by POIs on a given avatar depends on their relative *attractiveness* given the current state of the avatar and can be determined or learned in most games.

IV. The AntReckoning algorithm

In AntReckoning, points of interest are treated as ants that generate pheromones modeling their relative attractiveness. Pheromones are chemicals that exert attraction forces on players, integrated in a second order predictor. They spread in the game world, and fade over time, therefore taking geometrical and temporal aspects into account. Throughout this section, we use the example depicted in Figure 1 to illustrate the different mechanisms involved in AntReckoning. Table I summarizes the parameters of AntReckoning together with a brief description and the value used in the evaluation. We discuss how to tune these parameters in Section V.

Model Consider a game evolving in discrete event loops called *frames*. In each frame each player needs to know the positions of other avatars, which he receives through position updates. Consider player Q who seeks to estimate the position $\mathbf{x}_{t+\delta t}$ of the avatar of player P in frame $t+\delta t$ while the last update received contains the position $\mathbf{x}_t$ (and possibly the estimated velocity $\mathbf{v}_t$ and acceleration $\mathbf{a}_t$) of P in frame t.

Dead reckoning AntReckoning makes use of a second order predictor where the second order term is a weighted sum of the acceleration of the avatar and the attraction forces. The estimated position therefore writes:

$$\mathbf{x}_{t+\delta t} = \mathbf{x}_t + \mathbf{v}_t\ \delta t + \frac{1}{2}\left(\alpha\ \frac{1}{m}\ \mathbf{F}_t + (1-\alpha)\ \mathbf{a}_t\right)\ \delta t^2\ , \tag{4}$$

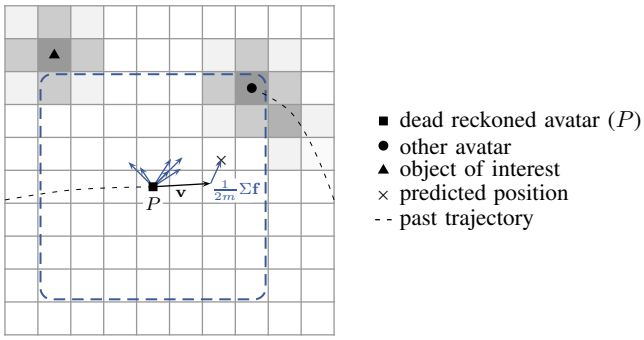

Fig. 1. Overview of AntReckoning: the game world is divided into cells using a regular square grid. Each cell contains a certain amount of pheromone reflected in grayscale. To estimate the current position of P, one adds (1) his velocity, estimated from his trajectory, and (2) the sum of the attraction forces generated by cells inside a square region around him, to the position of P in the last frame. Attraction forces are directed towards the attracting cells and their intensity is proportional to the amount of pheromone they contain.

where δt is the number of frames elapsed since the last position update, $\mathbf{F}_t$ is the sum of the attraction forces exerted by pheromones on P and other forces (e.g., gravity), and $\mathbf{v}_t$ (resp. $\mathbf{a}_t$) is the estimated velocity (resp. acceleration) of P. In AntReckoning, the estimation of velocity and acceleration is performed from previous observations using EMA as described in §II with parameters α_v and α_a respectively.

Figure 1 illustrates the estimation of the position of P for the next frame using the current velocity alone (i.e., $\alpha_v = 1$) and attraction forces alone (i.e., $\alpha = 1$).

Pheromones As common to most games, AntReckoning assumes a game world divided into non overlapping *cells*, e.g., Delaunay triangulation, Voronoi tessellation, binary space partitioning, or regular grids (e.g., square grid in Figure 1) typically used for tasks such as path finding, collision detection, or graphical rendering. The management of pheromones and the computation of attraction forces exerted by them is performed at the granularity of a cell: for each avatar P for which Q performs dead reckoning, Q computes the concentration of pheromone (represented in grayscale in Figure 1) in each cell and computes and sums the corresponding attraction forces. For the sake of scalability, only the cells in a limited region around P, called the attraction region and denoted by $\mathcal{R}$, are considered, e.g., a fixed-size square represented with dashed lines in Figure 1. Since pheromones spread, even points of interest outside $\mathcal{R}$ are taken into account.

For each P for which player Q has to perform dead reckoning, the concentration of pheromone inside a cell that is part of the attraction region $\mathcal{R}$ of P is calculated as follows:

- **Generation**: in each frame, each point of interest within a cell, be it an avatar or an object, generates a given amount of pheromone related to its attractiveness to P. Attractiveness is a function of the characteristics of the object and the current state of the considered avatar (as in the wounded player example). This amount is added to the concentration of the cell. The maximum concentration of a cell can be capped to limit the attractiveness of any single cell.
- **Evaporation**: in order to limit in time the attraction of previous positions of points of interest, pheromones fade in time, meaning that their concentration is decreased at the beginning of each frame. Exponential decays, i.e., removing a fixed percentage of the old pheromones at the beginning of each frame, have been successfully used in previous work on ant colonies (e.g., Max-Min ant colonies [11]). Beyond its simplicity and its effectiveness, such an evaporation model ensures that the total number of pheromones in the game world does not grow to infinity over time.
- **Dissemination**: since pheromones spread, the concentration of pheromone in neighboring cells are mutually dependent. At the beginning of each frame (after the evaporation step), a given amount of pheromone is removed from each cell and evenly dispatched to its neighboring cells. The size and shape of this neighborhood depend respectively on the speed of pheromones and the world map (e.g., wall, hills, *etc.*).

These phenomena are captured by the following recursive expression of the concentration of pheromone in a cell, for a given player P, at frame t:

$$\mathrm{ph}_t(\mathrm{cell}) = \overbrace{\varepsilon\ \mathrm{ph}_{t-\delta t}(\mathrm{cell})}^{\text{evaporation}} + \overbrace{\sum_{\mathrm{entity}\in\mathrm{cell}} \mathrm{attractiveness}(\mathrm{entity}, P)}^{\text{generation}} + \underbrace{\sum_{\mathrm{c}\in\mathcal{N}(\mathrm{cell})} \frac{\varepsilon\cdot\gamma}{|\mathcal{N}(\mathrm{c})|}\mathrm{ph}_{t-\delta t}(\mathrm{c})}_{\text{incoming dissemination}} - \underbrace{\varepsilon\cdot\gamma\ \mathrm{ph}_{t-\delta t}(\mathrm{cell})}_{\text{outgoing dissemination}}\ , \quad (5)$$

where ε is the evaporation factor, γ is the dissemination factor, and $\mathcal{N}(\cdot)$ is the set of a cell's neighboring cells. The attractiveness of a player to itself is set to zero. These phenomena can be observed in Figure 1 around the trajectory of a moving avatar: some pheromones remain and some spread around its previous positions; all pheromones fade over time.

To better understand the evolution of the concentration of pheromone described by Equation (5), consider the central cell c in the simplified example depicted in Figure 2. Cell c contains an object (depicted with a triangle), its neighborhood is composed of the four adjacent cells, and its current concentration of pheromone is 32. Assuming an evaporation factor ε of 0.5 the concentration is first reduced to 16. Considering a dissemination factor γ of 0.5, another 8 pheromones are then removed and evenly dispatched to the four neighboring cells. As a result of pheromone dissemination from the neighboring cells, cell c receives a total of $1+4+2+4=11$ incoming pheromones. Finally the pheromones generated by the object in c, say 5, are added to its concentration yielding a total of $16-8+11+5=24$ pheromones in cell c at the next frame.

To further improve AntReckoning's scalability, concentrations of pheromone lower than a given threshold are ignored, since their attraction power is negligible.

Attraction In physics, attraction forces between two bodies are generally directed along the line connecting them and their intensity is a decreasing function of the distance between them. In the case of spring attraction, the force is inversely proportional to the distance between the two bodies while it is inversely proportional to the square of the distance for gravitational and electromagnetic attractions. In AntReckoning, the

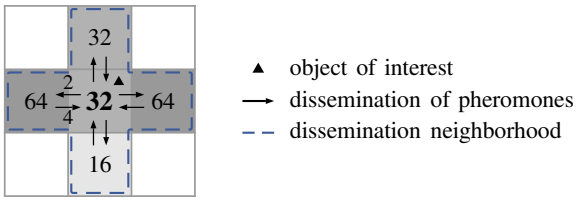

Fig. 2. Illustrative example of the evolution of the concentration of pheromone inside a cell with an evaporation factor of $\varepsilon = 0.5$ and a dissemination factor of $\gamma = 0.5$: half of the pheromones are removed due to evaporation and half of the remaining pheromones are evenly dispatched to the four neighboring cells.

attraction force exerted by a cell on an avatar is directed along the line connecting the position of the avatar, i.e., $\mathbf{x}_t$, to the center of the cell. The intensity of the attraction force is proportional to the concentration of pheromone in the cell divided by the distance raised to a certain power:

$$\|\mathbf{f}_t(\text{cell}, \mathbf{x}_t)\| = \frac{\text{ph}_t(\text{cell})}{d(\text{cell}, \mathbf{x}_t)^k} \ , \tag{6}$$

where k is a parameter of the system. Attraction forces of various intensities originating from P and directed towards cells containing pheromones can be observed in Figure 1.

Throughout this section, we considered solely players attracted by objects and other players, through pheromones. However, repulsion, e.g., of players by one another as described in the motivation section (§III), can easily be incorporated in the force model of AntReckoning: making repulsive objects (e.g., time bomb) or avatars (e.g., an attacker) generate pheromones with negative values would result in repulsive forces moving P away from them in the predictions.

TABLE I
IMPORTANT PARAMETERS IN ANTRECKONING.

Parameter	Description	Value
δ	# of frames since last position update	variable
α	weighting coef. acceleration v.s. forces	1
α_v	weighting coef. in EMA of velocity	0.8
α_a	weighting coef. in EMA of acceleration	0.8
$\mathcal{R}$	region for attraction	variable
ε	evaporation factor	variable
γ	dissemination factor	0
k	decreasing power of attraction forces	2
attractiveness($\cdot$)	attractiveness of avatars and objects	variable

V. PARAMETRIZATION AND IMPLEMENTATION

In here we discuss practical considerations about AntReckoning, more specifically the tuning of its parameters and its implementation in a decentralized setting.

A. Parametrization

Defining parameters of AntReckoning are as follows:

Attractiveness of points of interest can be divided into two categories: (1) game objects and (2) players:

- **Game objects:** Most games are able to define the attractiveness of the game objects based on their value or power in the game. In addition, the attractiveness of objects can be defined as a function of key factors in the player's current state, estimated from the analysis of game play traces. For instance, one can experimentally estimate the probability that a player with a given health level having a health pack in his sight picks it up within the next δt seconds. This estimation can then be used to define the attractiveness of a health pack as a function of the current health of an avatar.
- **Players:** Attractiveness of players for one another can be based on their recent interactions, e.g., trading or fighting, and built into the equations as follows: the attractiveness of a player Q to a player P is a function of the time $t_{P,P'}$ elapsed since their last interaction. An exponentially decreasing factor (i.e., $e^{-t_{P,P'}}$) takes the pace of game into account. The type of interaction (e.g., shooting or being shot) determines the sign of attractiveness, i.e., attraction or repulsion. Other factors such as the items a player is carrying, e.g., flag, can be incorporated in his attractiveness.

Mass modulates the effect of attraction forces on avatars. Avatars with higher masses are less subject to attraction than others. Different mass levels can be used to capture the relative attraction of avatars by objects: for instance, heroes may move only to achieve important goals and should therefore be assigned a high mass while regular units which move to achieve secondary goals (e.g., collect resources) should be assigned small masses. Note that the scale of mass is proportional to that of attractiveness: doubling the mass of all avatars is equivalent to halving the attractiveness of all objects.

Attraction region & cell size The size of the attraction region, in game world distance unit, and the size of each cell affects the accuracy of predictions. Larger attraction regions take into account farther objects and smaller cells compute the direction of attraction forces at a finer granularity, thus providing better accuracy. This, however, comes at the cost of computational and memory overheads. More precisely, the computational and space complexities of dead reckoning the position are proportional to the number of cells inside the attraction region. Note that since attraction decreases rapidly with the distance, increasing the size of the attraction region beyond a certain point may bring only a negligible improvement.

In games where players see the world through the eyes of their avatars, the attractiveness of objects should be weighted based on their relative positions to the avatar. The rationale is that players would be more attracted by objects they can actually see, which depends on their vision field and on the world map (e.g., walls). Also, taking into account the game map for dissemination allows to prevent avatars from being attracted by objects they cannot reach, e.g., behind an obstacle.

B. Implementation

Players have access to information about other avatars and objects in their vicinity in order to render the game world. This information is received from the server, other players, or can be extracted from maps. Therefore, if P is in the vicinity of Q, Q has access to information about the points of interest in a limited vicinity around P. For reasonably sized attraction regions, Q has access to all the necessary information and history to run AntReckoning at no additional network cost.

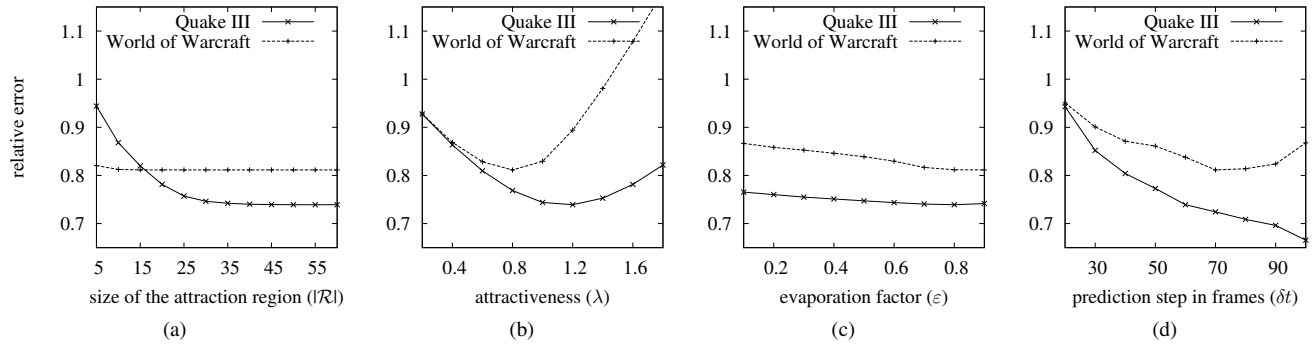

Fig. 3. Sensitivity analysis of AntReckoning around point $(|\mathcal{R}|, \lambda, \varepsilon, \delta) = (60, 1.2, 0.8, 60)$ for Quake III and $(60, 0.8, 0.9, 70)$ for World of Warcraft.

Because it uses cells and pheromones instead of maintaining the precise current and past positions of points of interests, AntReckoning provides a lightweight implementation of interest-based dead reckoning. For each avatar for which dead reckoning is performed, AntReckoning uses a single fixed-size data structure, i.e., the map of pheromones around the avatar, that maintains the amount of pheromone in each cell. The map captures the entire temporal and geographical information within a fixed amount of memory, independent from the number of players and from time. By including the amount of pheromone a cell contains in the partition structure provided by the game, and by performing the pheromone generation step during the iteration over the entities required by the game rendering, the overhead of AntReckoning would remain low.

The complexity of AntReckoning can be optimized in two ways: (1) If a player performs dead reckoning for himself and sends the prediction to other players, he can use a single map of pheromones, thus significantly reducing the complexity. Also, since he has access to more information about the points of interest around his avatar and its current state, the computation is more accurate; (2) In case where players also perform dead reckoning for other players, the complexity can be reduced by making the attractiveness a function of only the points of interest, regardless of the state of the dead reckoned avatar, as then only one pheromone map would be needed.

VI. Evaluation

The goal of the evaluation is two-fold: (1) perform preliminary experiments comparing AntReckoning to traditional dead reckoning in order to validate our approach and estimate the potential gains it conveys; (2) perform a sensitivity analysis of AntReckoning to identify its key parameters and evaluate their individual impact on the performance.

We evaluated AntReckoning using traces collected from Quake III and World of Warcraft. The first consists of the positions of the 48 players and entities (e.g., ammunitions) in the game world, in `q3dm1` and `q3dm17` maps, for each frame as well as the map information (e.g., walls). The second consists of sparse position information about more than 200 players in the Wintergrasp region, and has been obtained from [12].

The methodology used for the evaluation was as follows. We divided time in frames and set the players' positions for each frame as their last position updates in the corresponding time interval. We estimate the position of players δt frames ahead with both traditional second-order dead reckoning, i.e., based on both the estimated velocity and acceleration, and AntReckoning, and compare their performance with respect to the relative error, knowing the actual position of the players at this frame. More specifically, we compute the Euclidean distance between the estimation and the actual position, i.e., the error, and look at the ratio between the errors of AntReckoning and traditional dead reckoning. A value of 0.8 for this metric means that AntReckoning decreases the estimation error by 20% over traditional dead reckoning. That is, values smaller than 1 denote an improvement in accuracy.

For the sake of simplicity, we used a basic version of AntReckoning: only players generate pheromones (since no information about the objects were available in the World of Warcraft trace) and all players generate the same amount λ of positive pheromones regardless of the state of the avatar for which dead reckoning is performed. Consequently a single pheromone map is used for all avatars. We used a square region of attraction and took into account pheromone evaporation. However, we did not consider pheromones dissemination: only cells players go through contain pheromones.

For our sensitivity analysis, we considered the following parameters: (1) the diameter of the region of attraction, denoted by $|\mathcal{R}|$; (2) the attractiveness of players for each other, denoted by λ; (3) the evaporation factor, denoted by ε; (4) the duration of the prediction step, denoted δt. Other parameters were fixed to the values specified in Table I. The results from our trace-driven evaluation are compiled in Figure 3.

The effect of the size of the attraction region is shown in Figure 3(a). By increasing the size of the attraction region, a larger number of attraction forces are taken into account, therefore, improving the quality of predictions. However, increasing the size beyond a certain point results in negligible performance gains, confirming our intuition. This allows to perform efficient interest-based dead reckoning at no network cost with limited information locally available at each player and with limited computational and memory overhead.

Figure 3(b) demonstrates the effect of attractiveness on the performance of dead reckoning: while taking attraction forces into account improves the accuracy, over-estimating their influence and disregarding inertia decreases the accuracy.

Figure 3(c) shows that evaporation has little effect on the

performance of the algorithm: players' actions are mostly driven by other avatars' recent positions. This is most likely due to the fact that Quake III is a fast paced game. However, in World of Warcraft where the pace of the game is slower and players are more motivated by long term goals, evaporation has a higher impact: larger values of ε, which characterize a slower evaporation, significantly increase the performance.

Human interactions are an order of magnitude slower than game events. Therefore, in very short periods of time (e.g. 50 ms), avatars movements are mostly a function of their inertia. However, in longer periods (e.g., one second) they follow players' interests. This fact is illustrated in Figure 3(d) where AntReckoning performs better when prediction is done farther into the future. Given that many protocols (e.g., Donnybrook) rely on dead reckoning to increase the delay between sending updates to up to seconds, AntReckoning will be of use.

Our experiments on traces from World of Warcraft showed a slightly lower improvement over dead reckoning, i.e., up to 20%. In addition, the optimal set of parameters was different from Quake III showing the importance of fine tuning AntReckoning parameters based on the game. In world of Warcraft, players are motivated by longer term objectives and interactions happen at a lower pace, highlighting the effects of the evaporation factor and the length of the prediction step.

VII. Related work

A number of techniques improving the performance of dead reckoning have been proposed. In [13] the threshold above which a new update is sent is adapted to the precision requirements, determined by the relative position of the entities. In [14], the metric used to evaluate the prediction error takes temporal aspects into account in order to optimize the *perceived* inconsistency. These techniques are orthogonal to our approach and could be used together with AntReckoning to increase the performance of dead reckoning.

In [5] neural networks are trained and used, instead of instantaneous estimates, to predict changes of the entity velocity. ARIVU [15] predicts the next player *actions* in mobile games, using historical data, to determine if the wireless interface can be put into sleep mode, thus saving energy. These techniques could be incorporated in AntReckoning to help estimating velocity and acceleration, and player actions respectively.

Ant colony optimization has been successfully used to solve a wide range of problems, e.g., the traveling salesman problem [11, 16]. AntReckoning is inspired from these techniques, e.g., the concepts of evaporation, spreading, *etc.*, however it is, to the best of our knowledge, the first application of pheromones to interest modeling and dead reckoning.

Beyond predicting avatar position, the interest model of AntReckoning, e.g., attractive pheromones, can be used to make convergence, path-finding (e.g., A^*), and artificial intelligence [4, 17] algorithms exhibit human-like behavior.

Acknowledgements

Amir Yahyavi was funded by NSERC Strategic Grant STPGP/350626-2007. Kévin Huguenin was partially funded by a scholarship offered by University of Rennes I.

VIII. Conclusion

This paper proposed AntReckoning, an interest-based algorithm to improve dead reckoning. AntReckoning models a player's interest in other players and game objects in the form of attraction forces exerted by pheromones at a low computational and memory overhead. Preliminary results on Quake III and World of Warcraft have shown improvements up to 30% in accuracy over traditional dead reckoning, even in basic settings, demonstrating a great potential. Future work includes measuring bandwidth saved by AntReckoning using threshold-based dead reckoning as well as refining the execution of the algorithm with the notions of repulsion, map features, and vision. We also plan to develop techniques to parametrize AntReckoning with offline trace-driven analysis and online learning as well as applying it to convergence algorithms.

References

[1] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-Peer Support for Massively Multiplayer Games," in *INFOCOM*, 2004.

[2] R. Fujimoto, *Parallel and Distributed Simulation Systems*. Wiley Interscience, 2000.

[3] S. Aggarwal, H. Banavar, A. Khandelwal, S. Mukherjee, and S. Rangarajan, "Accuracy in Dead-Reckoning Based Distributed Multi-Player Games," in *NETGAMES*, 2004.

[4] C. Murphy, *Game Engine Gems 2*. AK Peters/CRC Press, 2011, ch. Believable Dead Reckoning for Networked Games, pp. 308–326.

[5] A. McCoy, T. Ward, S. McLoone, and D. Delaney, "Multistep-Ahead Neural-Network Predictors for Network Traffic Reduction in Distributed Interactive Applications," *ACM TOMACS*, vol. 17, pp. 1–30, 2007.

[6] Z. Berman and J. Powell, "The Role of Dead Reckoning and Inertial Sensors in Future General Aviation Navigation," in *PLANS*, 1998.

[7] *IEEE Standard for Distributed Interactive Simulation – Application Protocols*, IEEE Standard 1278.1-1995 Std., 1996.

[8] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation*. Addison-Wesley, 1999.

[9] L. Pantel and L. Wolf, "On the Suitability of Dead Reckoning Schemes for Games," in *NETGAMES*, 2002.

[10] R. Brown, *Smoothing, Forecasting and Prediction of Discrete Time Series*. Dover Publications, 2004.

[11] T. Stützle and H. Hoos, "Max-Min Ant System," *Future Generation Computer Systems*, vol. 16, pp. 889–914, 2000.

[12] J. Miller and J. Crowcroft, "The Near-Term Feasibility of P2P MMOG's," in *NETGAMES*, 2010.

[13] W. Cai, F. Lee, and L. Chen, "An Auto-Adaptive Dead Reckoning Algorithm for Distributed Interactive Simulation," in *PADS*, 1999.

[14] D. Roberts, R. Aspin, D. Marshall, S. Mcloone, D. Delaney, and T. Ward, "Bounding Inconsistency Using a Novel Threshold Metric for Dead Reckoning Update Packet Generation," *Simulation*, vol. 84, no. 5, pp. 239–256, 2008.

[15] B. Anand, K. Thirugnanam, L. Long, D.-D. Pham, A. Ananda, R. Balan, and M. Chan, "ARIVU: Power-Aware Middleware for Multiplayer Mobile Games," in *NETGAMES*, 2010.

[16] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents," *IEEE TSMC*, vol. 26, pp. 29–41, 1996.

[17] J. Bai, D. Seah, J. Yong, and B. Leong, "Offloading AI for Peer-to-Peer Games with Dead Reckoning," in *IPTPS*, 2009.