Multimedia Systems
DOI 10.1007/s00530-012-0275-z

Interest Modeling in Games: The Case of Dead Reckoning

Amir Yahyavi · Kévin Huguenin · Bettina Kemme

Published on-line: July 2012

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 mainly on previous observations. The estimations are typically made by using dynamics of motion, taking only *inertia* and some *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 spread and fade in the game world, and are a source of attraction.

To motivate and validate our approach we collected traces from Quake III. We conducted specific experiments that demonstrate the effect of game-related goals, map features, objects, and other players on the mobility of avatars. Our simulations using traces from Quake III and World of Warcraft show that AntReckoning improves the accuracy by up to 44% over traditional dead reckoning techniques and can decrease the upload bandwidth by up to 32%.

Keywords Interest Modelling; Dead reckoning; Multi-Player Online Games; Ant Colonies.

1 Introduction

Interactive games have become very popular over the last decades reaching an unprecedented scale, and therefore, forcing major on-line multi-player gaming platforms to develop a range of techniques to increase their scalability. In interactive games, position update messages account for the largest portion of the network traffic [21] raising bandwidth issues. This calls for techniques that accurately predict player movements in order to reduce the update rate, while keeping the error on player position low.

Traditionally, the current position of an avatar is estimated from previous positions. Only when the error is higher than a threshold a new position update is sent, thus reducing the update rate [1, 13]. Upon reception of a new update, a convergence step is performed to hide the estimation errors from the player in rendering the motion [26]. Such techniques also help cope with message loss by extrapolating the new position when the new update is not received.

Dead reckoning estimates the position of an object from the equations of motion, based on previous observations. It has been successfully used in a number of areas including distributed simulations [13, 24], games [1, 26] and aviation. Although the performance of dead reckoning, in its current form, is good enough for vehicles moving smoothly [5], it may degrade to an unacceptable degree 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, with both an attacker shooting at

This article is a revised and extended version of an article that appeared in the Proceedings of the 10th ACM/IEEE International Workshop on Network and Systems Support for Games (NETGAMES 2011) [34].

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.

A. Yahyavi (✉) · B. Kemme
McGill University, School of Computer Science, Montreal, Canada.
E-mail: amir.yahyavi@cs.mcgill.ca

K. Huguenin
EPFL, School of Computer and Communication Sciences, Lausanne, Switzerland

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 evade from the attacker. As games generally have relaxed physical rules, sudden drastic changes in movements (e.g., U-turn) can occur. These unpredictable changes dramatically reduce the performance of dead reckoning as it only takes mechanics into account.

Inspired by this example, we argue that key factors in an avatar's motion are not only inertia but also the player's interests, specifically the objectives of the game, as well as entities in his vicinity that we call points of interest. Following this line of reasoning, we propose *AntReckoning*. To the best of our knowledge it is the first approach to use interest modeling for 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 spread in the game world and fade over time;
- 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 the avatar's future position.

The main contributions of AntReckoning are (1) to incorporate player interest into the equation of motion used for dead reckoning, and (2) to provide a framework for interest modeling and to use pheromones which take temporal and spatial aspects of players' interest into account. In addition, pheromones offer a practical solution to the decentralized implementation of interest based dead reckoning. We motivate by showing the usefulness of our interest based dead reckoning by using traces from Quake III. We demonstrate that game objects and map features have a measurable impact on the player mobility patterns and, not surprisingly, so do player interactions.

In addition, we evaluate our AntReckoning algorithm by using traces of World of Warcraft and Quake III, providing a basis to fine tune AntReckoning's game and player-specific parameters. Our simulation results show that AntReckoning, if properly configured for the game, consistently outperforms dead reckoning, and improves the average accuracy of the estimation by up to 44% over traditional dead reckoning. As AntReckoning involves several parameters, we identify the key ones and perform a thorough sensitivity analysis to evaluate their respective effect on the accuracy. We also discuss solutions to set game-related parameters, such as the attractiveness of objects, and practical implementation aspects.

The rest of the paper is organized as follows. Section 2 briefly introduces the reader to multi-player online games and to the basics of mechanics underlying dead reckoning techniques. Section 3 reports on preliminary experiments demonstrating the effect of game-related goals, map features, objects, and other players on the mobility of avatars. Section 4 presents AntReckoning and introduces the key mechanisms and parameters involved. Section 5 discusses the technical implementation details of AntReckoning and gives insight on the fine tuning of its parameter through experimentation. Section 6 reports on the sensitivity analysis and performance evaluation of AntReckoning, based on Quake III and World of Warcraft mobility traces. Section 8 surveys related work and Section 9 concludes the paper.

2 Background

In multi-player games, players control their avatars which inhabit in a virtual space called the *game world*. The so-called game world contains a number of features, including hills and buildings, and is populated by objects (e.g., weapons and health packs) and avatars (see Figure 1). Clients regularly exchange the states of their avatars, including their position. The update dissemination to clients can be done directly among players or via game 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. The applications of dead reckoning can typically be divided into two categories: (1) enabling less frequent exchange of state updates by issuing an update only when the prediction error is higher than an acceptable threshold (this is called threshold-based dead reckoning); (2) helping cope with loss or jitter when frequent update messages are sent at a fixed rate. Therefore, 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 variables: the entity's last position $\mathbf{x}_t$, its velocity $\mathbf{v}_t = \dot{\mathbf{x}}_t$, and possibly its acceleration $\mathbf{a}_t = \ddot{\mathbf{x}}_t$ (as defined by the *IEEE Standard for Distributed Interactive Simulation* [18]), where t represents time. Extra information, that helps estimate the forces the entity is subjected to, can also be included. For objects extended in three-dimensional space, the kinetic state includes orientation and angular velocity as well (and possibly the angular acceleration).

The study of the trajectory of objects relies on the dynamic equations of motion, and more specifically, on Newton's second law, which links the *acceleration* $\mathbf{a}_t$ of an object, its *mass* m and the *forces* $\mathbf{f}$ it is subject to:

$$\mathbf{a}_t = \frac{1}{m} \sum \mathbf{f} \,. \tag{1}$$

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. When the mass is constant and in the case where the forces are determined by the entity's position and external factors such as wind and gravity, the future position of the object is fully determined by its initial position and velocity

In practice, a polynomial approximation derived from the 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

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

Note that such predictors are accurate only for small values of δt (compared to the speed 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 [27, 29]. 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 observations using an exponential moving average (EMA). In short, EMA estimates the velocity by a weighted sum of its current *instantaneous* value, specifically the difference between the current position $\mathbf{x}_t$ and the last position $\mathbf{x}_{t-\delta t}$ divided by the time interval δt, and the last estimation:

$$\mathbf{v}_t = \alpha_v\,\frac{\mathbf{x}_t - \mathbf{x}_{t-\delta t}}{\delta t} + (1-\alpha_v)\,\mathbf{v}_{t-\delta t} \tag{3}$$

$$\mathbf{a}_t = \alpha_a\,\frac{\mathbf{v}_t - \mathbf{v}_{t-\delta t}}{\delta t} + (1-\alpha_a)\,\mathbf{a}_{t-\delta t} \tag{4}$$

In other terms, the estimate of the velocity is a weighted sum of the current and previous values of the instantaneous velocity. The weights of the previous values decrease exponentially with time. Such an approach has proved to have a beneficial smoothing effect [7].

3 Motivation and Design Rationale

When using a first order predictor, the velocity of the avatars is usually estimated from the 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 estimate the forces the avatar is and will be subject to and incorporate them in the second derivative (i.e., the acceleration in the equations of motion).

Estimating the forces an avatar is subject to is a difficult task 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

Fig. 1 Screenshots of a Quake III game in the `q3dm01` map. Objects, avatars and map features in the vicinity of an avatar play an important role in the decision made by the player and thus affect the way he moves.

than just continuing on its current path. The 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, namely points of interest (POI).

To support this claim a number of experiments were done using Quake III traces in `q3dm01` map. The `q3dm01` map is spread over a single level and it is composed of two main rooms connected by two crossing corridors. A third corridor on the south of the southernmost room leads to a cave (dead-end). Objects are disseminated on the map, including a powerful weapon in the center of each room and a body armor in the cave. All objects disappear when picked up and reappear at the same location a few seconds later. When killed, players respawn at one of the so-called *respawn spots*. Figure 2 depicts the map with the different objects it contains and the respawn spots.

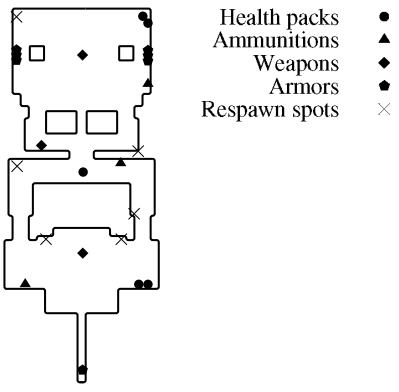

Fig. 2 The `q3dm01` map from Quake III. The map is spread over a single level and it is composed of two main rooms connected by two crossing corridors. A third corridor leads to a cave (dead-end).

We conducted two sets of experiments. The first set of experiments examines the effect of *game items* (weapons,

ammunitions, *etc.*) and *map features* (walls, corridors, corners, *etc.*) on the players' behavior, whereas the latter examines the effects of *interactions* between players on their behavior. Both are crucial to model player's interest and should be taken into account in a behavior prediction algorithm.

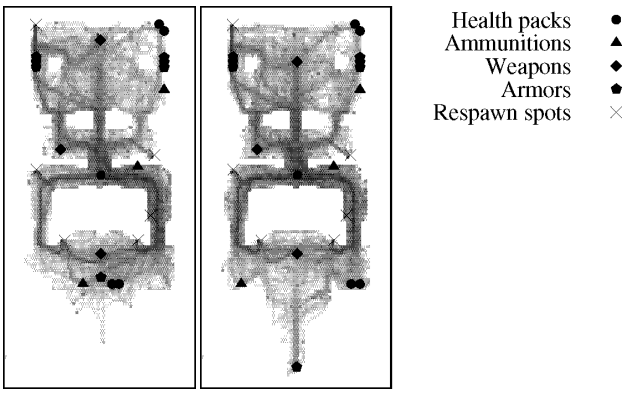

Fig. 3 Presence of players in the `q3dm01` map from Quake III with different entity positions (logarithmic grayscale colormap). The presence data was extracted from a 16-player game trace of 10 minutes.

Game Items and Map Features: In order to demonstrate the effect of players' interest in game items, two experiments were conducted in which the locations of interesting items (e.g., weapons, health packs, ammunitions) was changed. Figure 3 depicts the concentration of player movements in the game world, where darker regions are regions more populated by the players during the gaming session. Based on Figure 3, a number of observations can be made:

- Some places and paths on the map are popular due to their strategic advantage in the game, regardless of the location of the game items. These spots are advantageous because they provide a better cover, vantage points, or shortcuts in the game due to the map design.
- Changing the location of popular game items results in dramatic changes in the popularity of different regions in the game world. In Figure 3, players' presence in lower map parts (i.e., two lower corners of the southernmost room and the corridor leading to the cave) is almost diminished after removing interesting items (i.e., ammunitions, health packs and the body armor). This shows that player's interest, and by extension their behavior and their mobility, is affected by game items: when game items' locations are changed, new *hotspots* emerge and some regions become less populated.
- Popular player paths change as the game item locations are changed. This is due to the fact that players tend to choose the shortest and safest paths to items of interest. As a result, when item locations change, players' movement patterns also change. Examples of such path changes can be observed in different regions in Figure 3: The lower right and left corridors are equally attractive when the left corner of the southernmost room contains ammunitions; However when all the items in the second room are gathered in the centers, the right corridor is preferred.

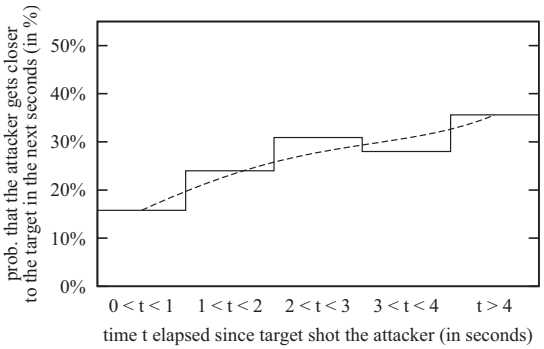

Fig. 4 Effect of player interactions on their behavior. When engaged in a fight, players tend to get away from each other (escaping or shooting while moving backward). However, when not being shot at back, players moves towards their target more often.

Player Interactions: Player interactions also affect players in the game world. In order to evaluate their effect, the following measurements were performed: When an interaction, in which player P shoots player Q, occurs at time t, we record the position of both players, as well as the time elapsed since Q last shot P, referred to *interaction recency*. We then look at the difference between the distance between P and Q's positions at time t and the distance between P's position at time $t+\delta t$ and Q's position at time t. If this difference is positive, it means that P tried to get closer to Q while shooting at him. We aggregate these points by time bins of one second and compute, for each bin, the probability that P tries to get closer to Q. Results are depicted in Figure 4 as a function of time elapsed since the interaction happened. We observe that players in a firefight usually get away from each other as they want to get out of each others line of fire. However, they are less likely to do so when the player they are shooting at is not shooting back at them. The rationale is that by getting close to a target that is not responding they are able to better target them. However, when being shot back by the other player, players tend to move to a more defensive stance. This is typical of most first-person shooter death match games. In other types of games, however, player interactions might have different effects. For example, friends playing in teams completing a mission together will not run away from each other and may move together to carry on different game tasks.

To sum up, a successful interest-modeling algorithm should handle the following situations: (1) Game items attract players. This is especially true 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 ammunitions or who are wounded would pick up ammunitions or health packs. Attraction therefore depends on both the objects of interest and the state of 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. These can be determined using interaction history between players, their teams, their social network (i.e. friendship relations) and the nature of the game they are playing. (3) Interesting and popular locations (e.g., top of a hill, corners, *etc.*) in the game, namely hotspots, are sources of attraction. Such attraction points can be inferred from the history of the movements of all players and/or the map design. (4) Players are repulsed by some game locations and map items, e.g. locations where players are endangered, or become under attack by map items such as automatic game-controlled towers.

In order to use the traditional 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/repulsion forces. The intensity of the forces exerted by POIs on the avatars depends on their *attractiveness* and can be determined or learned.

4 The AntReckoning Algorithm

AntReckoning implements a scalable algorithm based on *pheromones* to model players' interests in a lightweight and efficient way. This model is used to improve the accuracy of position predictions made by the dead reckoning algorithm.

In AntReckoning, points of interest are treated as ants that generate pheromones modeling their relative attractiveness. Pheromones are chemicals (which concentration is coded by a floating point number) that exert attraction forces on players, integrated in the second order predictor. They spread in the game world, and fade over time, therefore capturing the geometrical and temporal aspects of interest. Throughout this section, we use the example depicted in Figure 5 to illustrate the different mechanisms involved in AntReckoning. Table 1 (located on page 7 at the end of this section) summarizes AntReckoning's parameters together with a brief description and the values used in the evaluation. We discuss how to tune these parameters in Section 5.

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.

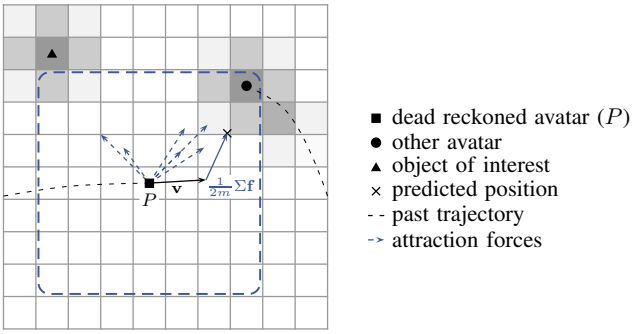

Fig. 5 Overview of AntReckoning: the game world is divided into cells by using a regular square grid. Each cell contains a certain amount of pheromone, represented here in grayscale. To estimate the current position of P, one adds (1) his velocity, estimated from his past trajectory, and (2) the sum of the attraction/repulsion forces (divided by his mass) generated by cells inside a square region around him (called attraction region), 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.

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:

$$\mathbf{x}_{t+\delta t} = \mathbf{x}_t + \mathbf{v}_t\,\delta t + \frac{1}{2}\left(\alpha\,\frac{1}{m}\,\mathbf{F}_t + (1-\alpha)\,\mathbf{a}_t\right)\,\delta t^2\ , \quad (5)$$

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 by using EMA, as described in Section 2, with parameters α_v and α_a.

Figure 5 illustrates the estimation of the position of P for the next frame using the current instantaneous velocity alone (i.e., $\alpha_v = 1$ in Equation 3) and attraction forces alone (i.e., $\alpha = 1$ in Equation 5). For a player P and for each cell in its attraction region with non-zero pheromone values, attraction forces on the player are computed. Each attraction force (i.e., dashed vectors), corresponds to the direction of the cell, the amount of pheromone in that cell, and the distance to the cell. These attraction vectors are then summed up and added to other physical forces the player is subject to. The final force vector then creates acceleration that is added to the current speed of the player: $\mathbf{v}$.

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 5) typically used for tasks such as path finding, collision detection, or graphical rendering. We denote by C the size of a cell in game world unit. 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 ($P = Q$ is also possible as will be described later), Q computes the concentration of pheromone (represented in grayscale in Figure 5) 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 5 (Note, however, that other shapes may be considered for the attraction region, such as a cone reflecting the player's vision field as discussed in Section 5.) As 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 possibly 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 ($\mathrm{ph}_{\max}$) to limit the attractiveness of any single cell at a given frame.
- **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 [31]). Beyond its simplicity and its effectiveness, such an evaporation model ensures that the total amount of pheromones in the game world does not grow to infinity over time. Using an evaporation factor of 1 (i.e., pheromones do not fade) gives a pheromone map similar to the presence map depicted in Figure 3, which captures popular locations and paths but disregards the dynamic environment of the game. In case objects always respawn on the same location, such a map would also capture object attraction. However, it would do so even if the object has not respawn yet. Using an evaporation factor of 0 (i.e., pheromones entirely fade in one frame), on the other hand, captures only the interest in surrounding objects and players at time t.
- **Dissemination**: As 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 simultaneously removed from each cell and evenly dispatched to its neighboring cells. The size and shape of this neighborhood depend respectively on the predetermined speed of pheromones dissemination and the game world topology (e.g., wall, hills, *etc*.). For example, obstacles may block the dissemination of pheromones to avoid attraction to unreachable areas.

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:

$$\mathrm{ph}_t(\mathrm{cell}) = \overbrace{\varepsilon\ \mathrm{ph}_{t-\delta t}(\mathrm{cell})}^{\text{evaporation}} + \overbrace{\sum_{\mathrm{entity}\in\mathrm{cell}} \mathrm{attractiveness}(\mathrm{entity}, P)}^{\text{generation}} + \underbrace{\sum_{\mathrm{c}\in\mathcal{N}(\mathrm{cell})} \frac{\varepsilon\cdot\gamma}{|\mathcal{N}(\mathrm{c})|}\mathrm{ph}_{t-\delta t}(\mathrm{c})}_{\text{incoming dissemination}} - \underbrace{\varepsilon\cdot\gamma\ \mathrm{ph}_{t-\delta t}(\mathrm{cell})}_{\text{outgoing dissemination}}, \quad (6)$$

where ε is the evaporation factor (percentage of pheromones that remain after evaporation), γ is the dissemination factor (percentage of pheromones that spread in the neighboring cells), 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 5 around the trajectory of a moving avatar: some pheromones remain and some spread around its previous positions; all pheromones fade.

To better understand the evolution of the concentration of pheromone described by Equation (6), consider the central cell c in the simplified example depicted in Figure 6. Cell c contains an object (depicted with a triangle), its neighborhood $\mathcal{N}$ 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 (i.e., 2 pheromones each). 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.

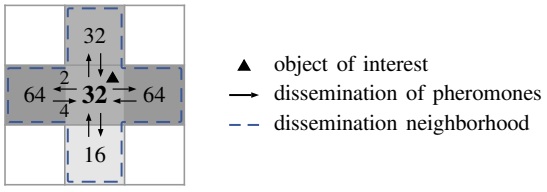

Fig. 6 Illustrative example of the evolution of the concentration of pheromone inside a cell with an evaporation factor of $\varepsilon = 0.5$ and a dissemination factor of $\gamma = 0.5$: half of the pheromones are removed due to evaporation and half of the remaining pheromones are evenly dispatched to the four neighboring cells. In addition, the object of interest lying in the cell generates pheromones and incoming pheromones disseminate from the neighboring cells.

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

The recursive equation 6 is linear. One can compute the equation separately and then sum up the pheromone maps corresponding to each static point of interest. More specifically, for static objects, the dissemination/evaporation pattern converges in time. Its limit depends only on ε and γ and can be determined formally or estimated by simulations. The results can be incorporated into the game or the map to simplify pheromone generation. Interestingly enough, the limit has a spatially limited support: Dissemination remains localized, which limits the complexity of the computation. The pheromone map corresponding to a set of static points of interest is the sum of the respective dissemination patterns centered on the points of interest and multiplied by their attractiveness.

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 intensity of the force is inversely proportional to the distance between the two bodies. However, in gravitational and electromagnetic attractions the force is inversely proportional to the square of the distance. In AntReckoning, the attraction force exerted by a cell on an avatar is directed along the line that connects 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:

$$\|\mathbf{f}_t(\text{cell}, \mathbf{x}_t)\| = \frac{\text{ph}_t(\text{cell})}{d(\text{cell}, \mathbf{x}_t)^k} \ , \tag{7}$$

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 5.

Throughout this section, we considered solely players *attracted* by objects and other players. However, *repulsion* of players by one another, as described in the motivation section (Section 3, Figure 4), can easily be incorporated into 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.

Post-Processing In order to improve the predictions and make them consistent with the game physiques, a number of post-processing checks are performed. In these checks the predicted position is corrected to take into account the game physics and the game map as well as other limiting factors: As the speed of avatars is bounded, the predicted position should remain within a distance $v_{\max} \cdot \delta t$ of the last known position. The current speed of the player can also be taken into account to ensure that the attraction forces do not cause a fast change in a player movement when the player is standing still. Similarly, a reachability map is used to correct the final prediction for the player's position to take into account the topology of the map (i.e., a player cannot move beyond a wall but may be attracted to a player that has just moved there). Figure 7a illustrates a basic correction technique to ensure that the predicted position is consistent with the speed limitations (we used this technique in our experiments): the circle represents the acceptable range of a player's movement. If the predicted position falls outside the area of possible movement or if the predicted path crosses an obstacle, then the predicted direction is preserved but the predicted position is changed to be the intersection between the path and the boundaries or the obstacle, if any. Another possible approach is to use a path finding algorithm to move around the obstacles towards the destination as illustrated in Figure 7b. In case a player is standing still we ignore attraction forces completely or greatly reduce their influence. If the attraction forces produce a very large vector while the speed vector is relatively small, we limit the size of the final vector to a factor of the speed vector size.

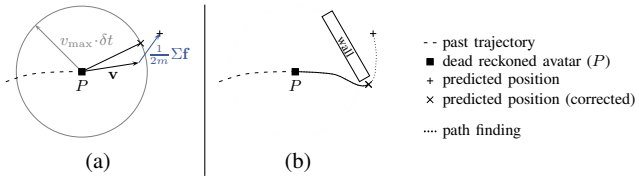

Fig. 7 Prediction correction: The predicted position of player P, based on his position and on external forces, violates the game physics, here (a) the speed limitation and (b) the map reachability. It is corrected as follows: for (a), the predicted direction is preserved but the distance is adjusted to the maximum possible value $v_{\max} \cdot \delta t$. For (b), a path-finding algorithm is used to determine the path from the current position of the avatar to the predicted position. If the predicted path is longer than $v_{\max} \cdot \delta t$, the predicted position is placed at a distance $v_{\max} \cdot \delta t$ to the current position of the avatar, along the path.

Table 1 Important parameters in AntReckoning.

Parameter	Description	Value
δ	# of frames since last position update	variable
α	weight coef. acceleration v.s. forces	0.5
α_v	weight coef. in EMA of velocity	0.8
α_a	weight coef. in EMA of acceleration	0.8
$\mathcal{R}$	region for attraction	variable
ε	evaporation fact. (% remaining)	variable
γ	dissemination fact. (% disseminating)	variable
k	decreasing power of attraction forces	2
attractiveness$(\cdot)$	attractiveness of avatars and objects	variable
ρ	base amount pheromone generated	40
$\text{ph}_{\max}$	maximum pheromone in a cell	100
C	cell size	variable
η	prediction threshold	variable

5 Parametrization and Implementation

We discuss practical considerations about AntReckoning, more specifically the tuning of its parameters and its implementation in a decentralized setting with partial knowledge.

5.1 Parametrization

The defining parameters of AntReckoning are as follows:

Attractiveness of points of interest can be divided into three categories: (1) game objects (2) players and (3) locations.

- **Game objects:** Most games are able to define the attractiveness of the objects based on their value or power. 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. The amount of pheromones generated by an item then depends on its attractiveness. For instance, in our experiments we defined the attractiveness of health packs in Quake III as a function of the avatar's health. We did so in our implementation by experimentally estimating 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.
- **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. In Quake III, given the results from Figure 4 a linearly decreasing factor takes the time of the interaction and the pace of the game into account. This means the player generates less negative pheromone for their targets if they have not shot back. The type of interaction (e.g., shooting or being shot at, trading, chatting, *etc.*) as well as their team can determine 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 into the attraction as well. In our experiments, we consider the interaction history between players to determine the sign and the amount of pheromone generated by players.
- **Locations:** Some locations in the game offer an advantageous position for players. These regions are not only popular because of the game items that exist in them but are attractive themselves. Examples of such locations are the top of a hill, which provides a good vantage point, or behind a wall that provides good cover. In order to improve the accuracy of predictions, these locations are assigned a pheromone generator that attracts, or in case of unpopular locations, repulses players by generating pheromones. Hotspots can be determined by either the game designers or through a trace analysis and heat maps, for example by looking at the number of players passing through a given location (as in Figure 3) or the number of kills/deaths that occurred at a given location to determine popular and unpopular regions. These measurements can be limited to the past few minutes of the games to capture the dynamic nature of hotspots.

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—, whereas regular units that move to achieve secondary goals (e.g., collect resources) should be assigned small masses. In addition, slow characters are assigned a higher mass whereas faster characters receive a lower mass to take into account their movement capabilities in the way attraction and repulsion forces affect their speed. 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. Given that in Quake (and other similar games) most players are equal in their capabilities, we assign the same mass to all players in our implementation. However, in games like Warcraft players are assigned levels and experience points and different units have different moving and fighting capabilities, thus helping to assign different masses to them.

Region Sizes: The size of the attraction region, in game world distance units, 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. Note that as attraction decreases rapidly with the distance, increasing the size of the attraction region beyond a certain point may bring only a negligible improvement. Therefore, in case of a large area of interest, the size of the attraction region can be limited based on the $\mathrm{ph}_{\max}$ that can be left on a single cell, and how pheromones force is reduced as a function distance which is modeled by the parameter k. We experiment with different region sizes in our evaluation and k is set to 2 (see Table 1).

Vision field: In games where players see the world through the eyes of their avatars, the players' vision field should be taken into account in the attraction region. 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). The vision field of players' depends on the viewing vector (i.e., a spherical cone with angle 45$^{\mathrm{o}}$ around the player's viewing vector in Quake III) and on the visibility information typically stored in the game map files (`bsp` files in Quake III). The region of attraction can be extended beyond the vision field to cope with players or objects suddenly entering a player's vision field. Our imple-

mentation of AntReckoning takes the players' vision fields into account and we experiment with this parameter.

Finally, taking into account the game map for dissemination and attraction prevents avatars from being attracted by objects they cannot reach, e.g., behind an obstacle.

5.2 Discussion & Implementation

In order to produce pheromone maps, a player needs to be aware of the players and objects in its vicinity. This is addressed by *interest management*.

Interest management handles which game objects (and their corresponding updates) should be received by the player, based on his location in the game world. This information is necessary to render the game world and are received from the server, other players, or extracted from the map. In some games (e.g., Quake III) the game world is relatively small and the area of interest can be the whole or all the visible portions of the game world. The player receives updates about all (potentially) visible players and objects in the game. In such a case, AntReckoning can easily calculate the necessary pheromone maps for the game. In games like World of Warcraft where the player receives new game objects as he explores the game world, a pheromone map for the player's area of interest is created. Items are taken into account and start generating pheromones as soon as they enter the area of interest. Over time, evaporation and dissemination remove the effect of the items no longer present in the area of interest. As a result, maintaining pheromone maps in both types of games comes at no additional network cost and is possible with the locally stored information.

Distributed implementation As discussed in Section 2, we divide the use of dead reckoning into two categories and discuss the implementation details of each form: (1) akin to normal dead reckoning, all players perform dead reckoning for all other players in their area of interest. In this case, all players, including the owner of the avatar, run the exact same algorithm for calculating pheromones with the exact same information. Through this, every player can detect when other players' prediction errors become higher than the acceptable threshold and send an update. (2) players perform dead reckoning only for themselves and include their prediction (a prediction vector) in the update sent to other players. In this case, other players do not have to perform dead reckoning and the player sends an update when he detects that the prediction he sent now has an error above the threshold. Note that AntReckoning can work both in centralized and distributed (Peer-to-Peer such as Donnybrook [6]) settings. In a centralized architecture, the server is a primary copy holder for all players' avatars and performs all the required task. The rest of algorithm is the same as in a distributed architecture.

There exist three possible implementations of AntReckoning. Two implementations are for the first type of AntReckoning in which players all perform dead reckoning. One of the two uses a higher number of pheromone maps and yields better accuracy than the other. The second type of AntReckoning has an implementation which yields better accuracy than both implementations of the first type.

1. Two possible implementations exist for the *first* type of dead reckoning: (1) maintain a different pheromone map for each player in the game world taking their interaction history, vision field, status, *etc.* into account. This scheme provides accurate predictions. However, it requires access to the interaction history of other players and information about their area of interest typical of games where updates are sent to everyone or area of interest is large enough (e.g., AOI is at least twice the size of attraction region). It also has a high overhead as a different pheromone map has to be maintained for each player. (2) maintain a single pheromone map for the area of interest. In this case, interaction history and similar information are ignored: only general information about points of interest is used to generate pheromone maps. Thus, the cost of generating pheromone maps is greatly reduced as only one map is maintained. But this comes at the price of lower accuracy in the predictions.
2. The *second* type of dead reckoning yields better accuracy and lower overhead in comparison to *both* implementations of the first type. In this scheme the player only maintains one pheromone map for itself, taking the interaction history, vision field, mass, and its status into account. Since the player has access to much more information about himself than other players, the pheromone map can be more accurately calculated. In addition, as only one pheromone map is maintained, the computational and memory overheads are minimal. The player will include his prediction vector in the updates sent to others.

Memory Overhead: The only memory overhead of AntReckoning is the maintenance of the pheromone maps. However, it can effectively be reduced to maintaining a single pheromone map and only for the player's area of interest, meaning the size of the pheromone map would be small. In addition, it is possible to use the built in tessellation mechanisms in games to maintain the pheromone map. In this case, the only overhead is for maintaining a single floating point number for each already existing cell. As the size of the pheromone map is a function of the size of area of interest and the size of each individual cell ($|\mathcal{R}|/\mathrm{C}$), it does not grow as the size of the game world becomes larger, and only requires a fixed amount of memory independent from the number of players and from time.

Computational Overhead: The computational overhead of AntReckoning is limited to pheromone processing and computing the attraction forces. During the generation of pheromones one needs to go through all the points of interest inside the area of interest. However, most games already perform such a loop of the items in each frame and execute the *think* function for each item. The think function is used for updating an item's status in the game [6]. Therefore, the only overhead is the generation of the corresponding pheromone amount for the item during the execution of its think function. Calculating the attraction forces is a function of number of cells inside the attraction region and does not grow with the number of items inside the area of interest or the size of the map.

Note that an alternative interest-based dead reckoning algorithm in which the precise current and past positions of all items are stored and taken into account for calculating the attraction forces, is also possible. However, such an approach would result in substantial memory and computational overhead. One of the main benefits of using pheromone-based interest modeling is that it enables us to keep a history of movements and interactions inside the game world in a low overhead manner.

6 Evaluation

The goal of the evaluation is two-fold: (1) Compare AntReckoning to traditional dead reckoning in order to validate our approach and estimate the gains it conveys. We compare with respect to the accuracy of the extrapolation when updates are sent at a fixed rate and with respect to the bandwidth usage when position updates are sent only when the prediction error grows above a given threshold; (2) Conduct a sensitivity analysis of AntReckoning to identify its key parameters and evaluate their individual effects on the performance.

6.1 Experimental setup

We evaluated AntReckoning by using traces collected from Quake III and World of Warcraft. The first consists of 16 players involved in a 10-minute death match in the `q3dm01` map. All the 16 players remained connected during the entire time of the game. The trace includes players' positions, items in their possession (e.g., weapons and ammunitions), their state (e.g., health, armor, speed, viewing angle), and their interactions (e.g., shooting, killing) in each frame. In addition, it contains information about items available in the game world, in each frame, and the players' item pickups. Map information (e.g., walls) is extracted from the map file (i.e., the `bsp` file). The second trace, from World of Warcraft, contains sparse position information about more than 200 players in the Wintergrasp region, and was obtained from [25]. The methodology used for the evaluation is as

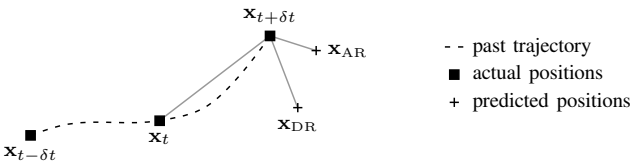

Fig. 8 Illustration of the metric used in the evaluation of AntReckoning: the relative improvement over traditional dead reckoning is defined as the ratio of the prediction errors $d(\mathbf{x}_{\mathrm{AR}}, \mathbf{x}_{t+\delta t})/d(\mathbf{x}_{\mathrm{DR}}, \mathbf{x}_{t+\delta t})$, where d denotes the Euclidean distance.

follows. We divide time in frames and set the players' positions for each frame as their last position update in the corresponding time interval. We predict 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. First, we 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 prediction and the actual position, i.e., the error, and look at the ratio between the prediction error of AntReckoning and that of traditional dead reckoning (as illustrated in Figure 8). A value of 0.8 for this metric means that AntReckoning decreases the prediction error by 20% over traditional dead reckoning. That is, values smaller than 1 denote an improvement in accuracy. Next, we implement a threshold-based dead reckoning algorithm, where players send position updates only when the prediction error (i.e., the Euclidean distance between the predicted position and the actual position) grows beyond a given threshold. We compute and compare the respective upload bandwidth consumptions of dead reckoning and AntReckoning.

The implementation of AntReckoning for Quake III is as follows. We focus on predicting the movements of a single player inside the game. The pheromone map is generated based on the game world as observed by this player. Players generate the same initial amount (ρ) of negative pheromones for other players inside the game and game items generate the same initial amount of positive pheromones for other players. The amounts of pheromone generated is then modulated by the players interaction recency as discussed in Section 3.

For World of Warcraft, we use a basic version of AntReckoning since no information about the objects and interactions were available in the trace: only players generate pheromones and all players generate the same amount ρ of positive pheromone regardless of the state of the avatar for which dead reckoning is performed. Consequently a single pheromone map can be used for all avatars. We use positive pheromones because players in World of Warcraft tend

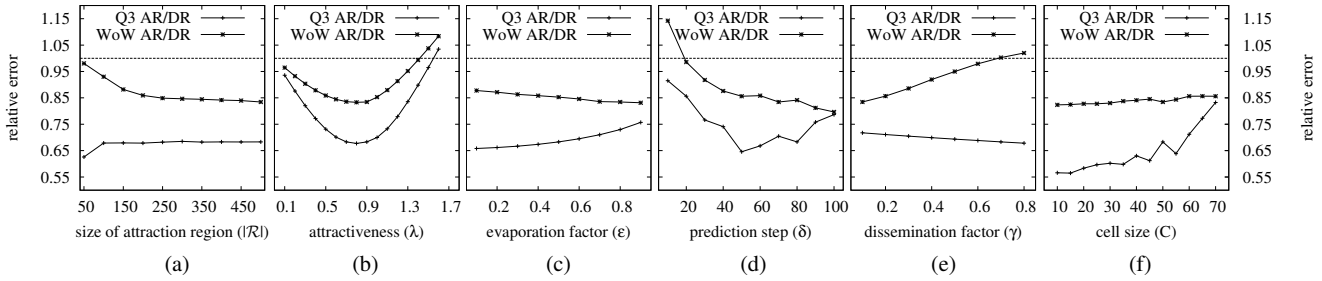

Fig. 9 Experimental sensitivity analysis of AntReckoning using a 16-player trace from Quake III in the `q3dm01` map and a movement trace from World of Warcraft in the Wintergrasp region. A frame corresponds to 50 ms in Quake III and 300 ms in World of Warcraft. The analysis was performed around the point $|\mathcal{R}| = 500$, $\lambda = 0.9$, $\varepsilon = 0.5$, $\delta = 80$, $\gamma = 0.7$, $\mathrm{C} = 50$ in Quake III and $|\mathcal{R}| = 500$, $\lambda = 0.9$, $\varepsilon = 0.8$, $\delta = 70$, $\gamma = 0.1$, $\mathrm{C} = 50$ in World of Warcraft.

to get closer to each other in most cases, be they fighting or trading. We assign the same mass value to all players in both games: In Quake III we do so because each player controls a single avatar and all avatars have roughly the same characteristics; In World of Warcraft we do so because of the lack of information on the avatars' characteristics.

6.2 Sensitivity analysis

We conduct our sensitivity analysis along the following parameters: (1) the diameter of the region of attraction, denoted by $|\mathcal{R}|$; (2) the reference attractiveness of players and objects, denoted by λ; (3) the evaporation factor, denoted by ε, note that this is the percentage of pheromone left after evaporation in each frame; (4) the duration of the prediction step, denoted by δt. (5) the pheromone dissemination factor, denoted by γ, which captures the percentage of pheromone that spread to neighboring cells; (6) size of the cells, denoted by C (For comparison, in Quake III, the entire `q3dm1` map is of size $\sim$1400$\times$2700 game units and an avatar occupies a floor area of 30$\times$30 game units. Translated into real-world length units, assuming that a person occupies a floor area of 0.5$\times$0.5 square meters, a cell of size 50$\times$50 corresponds to $\sim$0.8$\times$0.8 square meters). Other parameters were fixed to the values specified in Table 1 (page 7). The experimental results for the evaluation of Quake III and World of Warcraft are compiled in Figure 9, with respect to the prediction accuracy, when position updates are sent at a fixed rate. The results in this figure are with post processing and visibility turned off to better highlight the effects of different parameters. The graphs show only the relative error: AntReckoning v.s. traditional dead reckoning. Another interesting number is the absolute error of traditional dead reckoning normalized by the distance covered by the avatar between times t and $t + \delta t$, i.e., $d(\mathbf{x}_{\mathrm{DR}}, \mathbf{x}_{t+\delta t})/d(\mathbf{x}_t, \mathbf{x}_{t+\delta t})$, using the notations from Figure 8. This number only depends on the prediction steps δ: it increases with δ and is generally around 46-110% in our experiments, where an error above 100% means worse prediction than standing still. Therefore, it is both significant and relevant to improve the performance by up to 44% over traditional dead reckoning.

Before studying the graphs, the differences between these two games should be highlighted. In Quake III player movements and interactions happen at a very fast pace. Players' are mostly interested in and affected by other players in their close vicinity. The game world is relatively small and players mostly have short time goals and their interactions are mostly repulsive (e.g., shooting at each other). In World of Warcraft, players have more attractive interactions (e.g., trade, chat, *etc.*), the game world is very large, and can be very sparse. Players are interested in a larger region and have long term goals. The pace of the game is relatively slow.

The effect of the size of the attraction region is shown in Figure 9a. In World of Warcraft, by increasing the size of the attraction region, a larger number of attraction forces from farther regions 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. In Quake III however, players are interested only in POIs in their close vicinity and taking distant cells into account slightly decreases the performance. The limited effect of distant cells enables efficient interest-based dead reckoning: by limiting the attraction region to the area of interest (or a part of it), pheromone maps can be computed at no network cost (i.e., with limited information available locally at each player) and with limited computational and memory overhead.

Figure 9b demonstrates the effect of attractiveness: although taking attraction forces into account improves the accuracy, over-estimating their influence and disregarding inertia decreases the accuracy. This demonstrates that the current speed and movement of the player should indeed be taken into account along with the attraction forces. This is specially acute for short-term predictions as players are mostly influenced by their inertia, as explained below.

Figure 9c shows the effect of evaporation: given the fast pace of Quake III and short term goals of players, leaving a

higher percentage of pheromone at each frame decreases the quality of predictions. The evaporation factor controls how much history of the game is taken into account and should reflect the pace of the game: lower percentage of pheromone should be kept for fast paced games. As expected, in World of Warcraft, where the pace of the game is slower and players are more motivated by long term goals, evaporation has a different impact: larger values of ε, which characterize a slower evaporation, 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), an avatar's movements are mostly a function of his inertia. However, in longer periods (e.g., one second) they follow players' interests. This fact is illustrated in Figure 9d where AntReckoning performs better when prediction is done further 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. In fast paced games, these improvements might slightly decrease for very long prediction periods as players movements become more affected by interactions, given the rate of interactions, in the future that are not taken into account in the current attraction forces. In slower games (i.e., World of Warcraft) predicting further into the future is possible.

Figure 9e demonstrates the effect of the dissemination factor on the quality of predictions. In World of Warcraft given the size of the game world, cells and sparsity of the players, using larger dissemination factors result in less accurate attraction forces being calculated (i.e., attraction vectors not directly pointing towards the point of interest) and lower quality of predictions. In Quake III given the large choice of attraction region and cell size, and the length of the prediction, higher dissemination factors can slightly help in predicting further into the future.

The effect of the cell size is demonstrated in Figure 9f. In general, smaller grid cells improve the accuracy of the predictions since they provide a higher accuracy in calculating the direction of the attraction forces. In Quake III, where players interact in smaller game worlds, and are interested in a small region around them, small cells can greatly improve the quality of predictions. In World of Warcraft, however, where the game world is large and players are sparse, the use of smaller cells conveys only small improvements. Note that using smaller cells comes at the price of higher computational overheads (i.e., $\frac{|\mathcal{R}|}{C}$). The performance gain must therefore be balanced with the overheads.

It should be noted that the optimal sets of parameter values were quite different for Quake III and World of Warcraft showing the importance of fine tuning AntReckoning parameters based on the game characteristics. In World of Warcraft, players are motivated by longer term objectives and interactions happen at a lower pace, highlighting the effects of the evaporation, and dissemination factors, and the length of the prediction step. The game world is large, therefore, highlighting the effects of the attraction region while undermining the effects of the cell size. In Quake III, on the other hand, due to the smaller size of the world, cell size can play an important role while a larger attraction region provides negligible gains. Evaporation and dissemination factors should also be fine tuned according to the fast pace of the game and the size of game world. In both cases the attraction factor should be fine tuned to take the prediction length into account: shorter prediction lengths require smaller attraction factors. In addition, we suspect in games such as world of Warcraft, use of different mass values based on the character type can further help improve the accuracy of predictions.

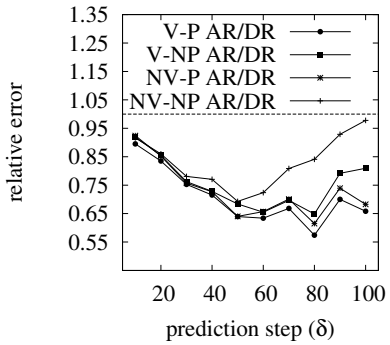

Fig. 10 Effect of visibility and post-processing on AntReckoning's performance. The tag V (resp. P) in the legend of the graph means that visibility (resp. post-processing) is used and NV (resp. NP) means it is ignored. The analysis was performed around the non-optimal point $|\mathcal{R}| = 500, \lambda = 1.1, \varepsilon = 0.6, \gamma = 0.6, \mathrm{C} = 50$ in Quake III to better show the effects of post-processing and visibility.

Post processing can further improve quality of prediction in some cases by up to 15-20% and help stabilize the performance of AntReckoning. Similarly, vision filtering can increase the quality by 10-15%. This is shown in Figure 10 where the effect of taking visibility and post-processing into account is shown as a function of the prediction duration (δ). We chose a non-optimal set of parameters to show that post-processing can correct the prediction errors, for example, when the attraction forces are too large due to bad parameter selection. It also improves the performance further for longer periods of prediction as error correction becomes more important for longer periods of time. Similarly, taking visibility into account is more important for capturing longer term goals.

6.3 Performance evaluation

We now look at the bandwidth savings provided by AntReckoning in a threshold-based approach. In short, the threshold-based approach operates as follows (see Algorithm 1 for a pseudo-code version): each player predicts the

Input:
prediction error threshold: η
last time a position was sent: t_0
actual position (at time t): $\mathbf{x}_t$
$t_0 \leftarrow -\infty$;
at each frame (time t) **do**
  $\hat{\mathbf{x}}_t \leftarrow \text{prediction}(\mathbf{x}_{t_0}, t)$;
  if $\|\mathbf{x}_t - \hat{\mathbf{x}}_t\| > \eta$ **then**
    send($\mathbf{x}_t$);
    $t_0 \leftarrow t$;
  end
end

Algorithm 1: Threshold-based dead reckoning: a player predicts the position of his avatar for the current frame, based on the last sent position update, sending an update only if the prediction error is larger than a threshold η.

position of his avatar and that of the other avatars. Since players run the same algorithm, they can detect when the distance between their predicted position and the actual position of their avatar is higher than the threshold, and therefore, send a position update message. The bandwidth savings are calculated based on the number of updates sent, for a given threshold, by traditional dead reckoning and by AntReckoning, in several sample Quake III gaming sessions. Results are depicted in Figure 11 (the prediction error threshold η is given in game world distance units). It can be observed that higher thresholds are typically considered when a higher prediction length is chosen, and in this situation, AntReckoning conveys significant bandwidth savings; up to 32%.

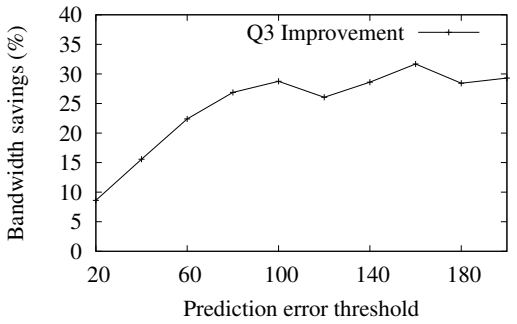

Fig. 11 Performance evaluation of AntReckoning with respect to upload bandwidth consumption in a threshold-based approach. Units are in game world distance.

Note that while for lower thresholds the percentage of improvements is smaller, each percentage improvement translates into a higher number of updates not being sent given the fact that at lower thresholds (and prediction lengths) updates are sent at a much higher rate.

While we do not provide a user study of how better predictions improve the quality of game play for players, the effects of these prediction errors, as the inconsistencies perceived by players, have been well studied [8, 10, 27, 28]. Therefore, AntReckoning can be used to both improve the network requirements of games as well as the quality of game play. In addition, it is demonstrated that different types of games can benefit from AntReckoning: even in games such as Quake III where sudden changes in movements are common, large improvements can be gained. While we achieved improvements up to 20% for World of Warcraft with information currently available, showing that even a simple implementation with a single pheromone map for all players can provide improvements over dead reckoning, we suspect that given enough information about the game world, games similar to World of Warcraft can benefit more from AntReckoning in comparison to fast paced games due to their characteristics and higher predictability of players actions.

7 Discussion

Choosing the threshold: The threshold is a parameter of the algorithm and is used as follows: as soon as the error grows beyond the threshold, an update is sent. In practice, the prediction error threshold is chosen as a trade-off between bandwidth and accuracy of prediction: it should be small enough so that the motion of avatars remains smooth and that the games-play remains fair (e.g., missing the target because the avatar is not displayed at its actual position). Such subjective aspects can only be evaluated through experiments coupled with user studies.

A number of techniques improving the performance of dead reckoning by choosing better thresholds have been proposed. In [8] the threshold above which a new update is sent is adapted to the precision requirements, determined by the relative position of the entities. In [28], the metric used to evaluate the prediction error takes temporal aspects into account in order to optimize the *perceived* inconsistency. Chen and Zarki [10] provide an objective evaluation framework for Quality of Experience (QoE) in Games. QoE takes not only the precision but three basic perceptions into account: *responsiveness*, *precision*, and *fairness*. These techniques are orthogonal to our approach and could be used together with AntReckoning to increase the performance of dead reckoning with respect to the quality of experience.

Cheating: Multi-player online games are vulnerable to cheating. Specific techniques, including dead reckoning, have been successfully exploited for cheating. In particular, with dead reckoning a player can (1) *actively* drop position updates or predictions and send incorrect predictions to confuse/blind other players and (2) *passively* exploit the prediction to gain an unfair advantage.

With *suppress-correct cheat* a player purposefully drops a number of consecutive updates and then sends an in-

valid update before being considered a dead peer. Similarly, players can implement a *consistency cheat* where they send different predictions to different players. Many methods have been proposed to address these general types of cheats [19, 33]. For instance, one can verify updates based on the maximum speed of players and the game world physics [14], make all players send hash codes of their updates, namely *commitments*, before sending their actual updates [4], and compare *a posteriori* predictions against the actual positions using (trusted) servers [14], referees [32], or dynamic proxies-players [17]. Detected misbehavior can then feed reputation systems [20].

Position predictions can also be used to unduly assist the players in the game. For instance, aim bots can automatically adjust the players' aiming directions, based on other avatars predicted positions, to increase the chances of hitting the target. Similarly, predictions can be used to implement a *map hack* attack, i.e., displaying objects/avatars the player is not supposed to see (e.g., behind walls). Such aids can be detected using player behavior analysis tools [22]. Note that such cheats apply to normal dead reckoning as well. In fact, in most cases AntReckoning does not create new opportunities to cheat. However, in AntReckoning, the fact that predictions implicitly give an insight on the players' interests provides an additional opportunity to cheat. For example, a prediction which indicates that an avatar will move towards a health pack in the next few seconds can possibly show that this avatar has a low health. This opportunity is however relatively difficult to exploit as the final prediction is the summation of *many* interest-based attractions forces, e.g., a weapon behind the health pack or another player shooting at the player.

Finally, the calculations involved in AntReckoning can be secured by tools such as PunkBuster [12] and Valve Anti Cheat (VAC) [30], which prevent the players from tampering with the game code, and accountable systems [15].

8 Related Work

A number of techniques improving the performance of dead reckoning have been proposed. In [8] the threshold above which a new update is sent is adapted to the precision requirements, determined by the relative position of the entities. In [28], 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 [24] neural networks are trained and used, instead of instantaneous estimates, to predict changes of the entity velocity. ARIVU [2] 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 us estimate 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, 31]. AntReckoning is inspired from these techniques, e.g., the concepts of evaporation, spreading, *etc.* However, to the best of our knowledge, this is 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 [3, 26] algorithms exhibit human-like behavior. The information contained in pheromone maps can be used to choose between popular and unpopular paths in the path-finding algorithm for players. The AI players can use pheromone maps to direct them towards popular parts of the map and to choose paths typically used by players.

While most dead reckoning algorithms rely on linear movements between predicted positions, pheromone maps in AntReckoning can be used to generate non linear and hence more realistic movements between positions. In [6] guidable AIs are used to move between position, hence providing non linear movements. This approach can further benefit from AntReckoning by guiding the AI to follow pheromone paths in order to move between two consecutive position updates.

Close to our work, a significant body of work has been devoted to the analysis of human mobility (e.g., [16, 23]) in real life, with numerous applications to urban planning and delay tolerant networking. In this context, several studies have demonstrated a strong influence of the locations of points of interest (e.g., specific businesses) on human mobility [35]. These results have been further leveraged for pedestrian mobility predictions [9]. A similar approach to AntReckoning can be exploited to predict pedestrian movements, where other people, buildings, and shops are treated as points of interest that generate pheromone.

9 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. Our results on Quake III and World of Warcraft have shown improvements over traditional dead reckoning: up to 44% in accuracy for a fixed update rate and up to 32% with respect to bandwidth for a bounded prediction error, even in basic settings, demonstrating a great potential. In addition, while the focus of this paper was on its applications in dead-reckoning in multi-player

games, the proposed algorithm can be applied to many other areas such as convergence algorithms. Future work includes developing techniques to parametrize AntReckoning with offline trace-driven analysis and online learning as well as applying it to convergence algorithms. We also plan to extend our work to pedestrian mobility prediction.

References

1. Aggarwal S, Banavar H, Khandelwal A, Mukherjee S, Rangarajan S (2004) Accuracy in Dead-Reckoning Based Distributed Multi-Player Games. In: Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES), ACM
2. Anand B, Thirugnanam K, Long L, Pham DD, Ananda A, Balan R, Chan M (2010) ARIVU: Power-Aware Middleware for Multiplayer Mobile Games. In: Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES), IEEE
3. Bai J, Seah D, Yong J, Leong B (2009) Offloading AI for Peer-to-Peer Games with Dead Reckoning. In: Proc. of Int. Workshop on Peer-to-Peer Systems (IPTPS)
4. Baughman N, Liberatore M, Levine B (2007) Cheat-Proof Playout for Centralized and Peer-to-Peer Gaming. IEEE/ACM Transactions on Networking 15:1–13
5. Berman Z, Powell J (1998) The Role of Dead Reckoning and Inertial Sensors in Future General Aviation Navigation. In: PLANS'98: Proceedings of the IEEE Position Location and Navigation Symposium, IEEE
6. Bharambe A, Douceur J, Lorch J, Moscibroda T, Pang J, Seshan S, Zhuang X (2008) Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games. In: Proc. of ACM Int. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), ACM
7. Brown R (2004) Smoothing, Forecasting and Prediction of Discrete Time Series. Dover Publications
8. Cai W, Lee F, Chen L (1999) An Auto-Adaptive Dead Reckoning Algorithm for Distributed Interactive Simulation. In: Proc. of Int. Workshop on Principles of Advanced & Distributed Simulation (PADS), ACM
9. Calabrese F, Di Lorenzo G, Ratti C (2010) Human Mobility Prediction Based on Individual and Collective Geographical Preferences. In: ITSC'10: Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems, IEEE
10. Chen P, El Zarki M (2011) Perceptual View Inconsistency: An Objective Evaluation Framework for Online Game Quality of Experience (QoE). In: Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES)
11. Dorigo M, Maniezzo V, Colorni A (1996) Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics 26:29–41
12. Even Balance Inc (2000) PunkBuster. `http://www.evenbalance.com/`, Visited March 1st, 2012
13. Fujimoto R (2000) Parallel and Distributed Simulation Systems. Wiley Interscience
14. Goodman J, Verbrugge C (2008) A Peer Auditing Scheme for Cheat Elimination in MMOGs. In: Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES)
15. Haeberlen A, Aditya P, Rodrigues R, Druschel P (2010) Accountable Virtual Machines. In: Proc. of Int. Conf. on Operating Systems Design & Implementation (OSDI), pp 1–16
16. Harri J, Filali F, Bonnet C (2009) Mobility Models for Vehicular Ad Hoc Networks: A Survey and Taxonomy. IEEE Communications Surveys Tutorials 11:19–41
17. Huguenin K, Yahyavi A, Kemme B (2011) Cheat Detection and Prevention in P2P MMOGs. Tech. Rep. SOCS-TR-2011.5, McGill
18. IEEESTD96 (1996) IEEE Standard for Distributed Interactive Simulation – Application Protocols. 10.1109/IEEESTD.1996.80831
19. Kabus P, Terpstra W, Cilia M, Buchmann A (2005) Addressing Cheating in Distributed MMOGs. In: Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES)
20. Kaiser E, Feng WC (2009) PlayerRating: A Reputation System for Multiplayer Online Games. In: Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES), pp 8:1–8:6
21. Knutsson B, Lu H, Xu W, Hopkins B (2004) Peer-to-Peer Support for Massively Multiplayer Games. In: Proc. of IEEE Int. Conf. on Computer Communications (INFOCOM), IEEE
22. Laurens P, Paige R, Brooke P, Chivers H (2007) A Novel Approach to the Detection of Cheating in Multiplayer Online Games. In: Proc. of Int. Conf. on Engineering of Complex Computer Systems (ICECCS), DOI 10.1109/ICECCS.2007.11
23. Lee K, Hong S, Kim SJ, Rhee I, Chong S (2009) SLAW: A Mobility Model for Human Walks. In: Proc. of IEEE Int. Conf. on Computer Communications (INFOCOM), IEEE
24. McCoy A, Ward T, McLoone S, Delaney D (2007) Multistep-Ahead Neural-Network Predictors for Network Traffic Reduction in Distributed Interactive Applications. ACM Transactions on Modeling and Computer Simulation 17:1–30
25. Miller J, Crowcroft J (2010) The Near-Term Feasibility of P2P MMOG's. In: Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES), IEEE
26. Murphy C (2011) Game Engine Gems 2, AK Peters/CRC Press, chap Believable Dead Reckoning for Networked Games, pp 308–326
27. Pantel L, Wolf L (2002) On the Suitability of Dead Reckoning Schemes for Games. In: Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES), ACM
28. Roberts D, Aspin R, Marshall D, Mcloone S, Delaney D, Ward T (2008) Bounding Inconsistency Using a Novel Threshold Metric for Dead Reckoning Update Packet Generation. Simulation 84(5):239–256
29. Singhal S, Zyda M (1999) Networked Virtual Environments: Design and Implementation. Addison-Wesley
30. Steam (2002) Valve Anti-Cheat System (VAC). `https://support.steampowered.com/kb_article.php?p_faqid=370`, Visited March 1st, 2012
31. Stützle T, Hoos H (2000) Max-Min Ant System. Future Generation Computer Systems 16:889–914
32. Webb S, Soh S, Lau W (2007) RACS: A Referee Anti-Cheat Scheme for P2P Gaming. In: Proc. of Int. Workshop on Network & Operating Systems Support for Digital audio & Video (NOSSDAV)
33. Webb SD, Soh S (2007) Cheating in Networked Computer Games: A Review. In: Proc. of Int. Conf. on Digital Interactive Media in Entertainment & Arts (DIMEA), pp 105–112
34. Yahyavi A, Huguenin K, Kemme B (2011) AntReckoning: A Pheromone-Based Dead Reckoning Algorithm for Games. In: Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES)
35. Zheng Y, Zhang L, Xie X, Ma WY (2009) Mining Interesting Locations and Travel Sequences from GPS Trajectories. In: Proc. of Int. World Wide Web Conferences (WWW), ACM