Comparing Dead Reckoning Algorithms for Distributed Car Simulations

Youfu Chen
Multi-plAtform Game Innovation Centre
Interdisciplinary Graduate School
Nanyang Technological University
Singapore
ch0002fu@ntu.edu.sg

Elvis S. Liu*
Southern University of Science and Technology
Shenzhen, China
esyliu@sustc.edu.cn

ABSTRACT

Dead reckoning is an important technique used in distributed virtual environments (DVEs) to mitigate the bandwidth consumption of frequent state updates and the negative effects of network latency. This paper proposes a novel dead reckoning approach for common DVE applications such as multiplayer online games. Unlike traditional dead reckoning approaches that estimate the movements of remote entities with pure kinematic models, the new approach performs extrapolations with the considerations of environmental factors and human behaviours. We have performed experiments, based on a distributed car simulator, to compare the the new approach with representative existing dead reckoning approaches. The results show that the new approach gives more accurate predictions with an acceptable overhead.

CCS CONCEPTS

• **Information systems → Massively multiplayer online games**;
• **Computing methodologies → Distributed simulation**;

KEYWORDS

Dead Reckoning, Distributed Virtual Environments, Multiplayer Online Games, Distributed Simulation

ACM Reference Format:
Youfu Chen and Elvis S. Liu. 2018. Comparing Dead Reckoning Algorithms for Distributed Car Simulations. In *SIGSIM-PADS -18 : 2018 SIGSIM Principles of Advanced Discrete Simulation, May 23–25, 2018, Rome, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3200921.3200939

1 INTRODUCTION

A distributed virtual environment (DVE) is a virtual reality simulation that allows users to interact in real-time even though they are at geographically different locations. DVE has been studied extensively for more than three decades. Historically, it can be traced back to the SIMNET project [4] developed by the US Department of Defense and the Multi-User Dungeon (MUD) game [1] developed by two students of the University of Essex. In recent years, large-scale DVEs have become a major trend in entertainment applications, mainly due to the enormous popularity of multiplayer online games (MOGs) [12]. These applications aim to support thousands of participants, which imposes a high load on the DVE networks.

Much of the research effort in DVE has been focused on addressing the requirement of scalability. As DVEs become more data intensive and more latency sensitive, providing scalable data distribution services is crucial for their successful deployment. A simple approach for DVE data distribution would be to have each participant (or host) regularly broadcast the entity states that it maintains. Apparently, as the scale of DVE grows, this approach would consume significant network resources. To address this problem, two techniques have been developed to reduce the bandwidth consumption. One is referred to as interest management [6], which filters irrelevant data transmissions on the DVE network. The other is referred to as dead reckoning (DR), which was introduced by SIMNET and its successor Distributed Interactive Simulation (DIS) [8] to reduce the frequency of state updates.

The basic idea of DR is simple: instead of receiving entity updates at every time step, the participants use a kinematic model (referred to as dead reckoning model (DRM)) to predict the movement of remote entities based on their last known kinematic states. In this way, the participants are able to simulate the entities' movement for a period of time without receiving any state update, and thus the bandwidth consumption can be greatly reduced. In addition, the participants also maintain DRMs for their own entities and constantly monitor the deviation between the DRMs and their corresponding actual models. When the deviation exceeds a predefined threshold, the participant would broadcast the actual state of the corresponding entity to all participants, a process commonly referred to as 'rollback', in order to correct their DRMs. Obviously, the efficiency of DR depends strongly on the prediction accuracy of the DRM. If corrections are broadcast frequently, a significant bandwidth consumption would be induced on the network. In addition, frequent corrections would also cause a large number of 'lag' effects on the participants' screen. Even with the use of smoothing algorithms [5], the illusion of presence would be seriously affected. Therefore, designing an accurate prediction model is an important requirement for DR based DVEs.

In this paper, we focus on improving the prediction accuracy of the DRM. We propose a path-assisted DR (PADR) approach, which performs extrapolations with the considerations of environmental

*Corresponding author

factors and human behaviours. Our approach is designed specifically for DVEs that contain predefined paths. We also present a quantitative study to compare the prediction accuracy of the proposed approach with representative existing DR approaches. Simulation results, based on The Open Racing Car Simulator (TORCS) [2], show that the proposed approach gives more accurate extrapolations than the existing approaches with an acceptable overhead.

2 BACKGROUND AND RELATED WORK

The traditional DRM is basically of two types—first-order and second-order, which indicate the order of kinematic equations used for movement prediction. Let s denote the predicted position of an entity, s_0 denote its position in the last update, v denote the velocity, a denote the acceleration, and Δt denote the elapsed time since the last update, the kinematic equations are defined as:

$$s = s_0 + v\Delta t \tag{1}$$

$$s = s_0 + vt + \frac{1}{2}a\Delta t \tag{2}$$

Apart from the traditional DIS-based DR approach [5, 8], many improved DR algorithms have been proposed in the literature. Generally, these improved approaches focus on a major design requirement—performance, which is measured by the reduction of network bandwidth consumption. In addition, computational efficiency is also an important requirement to consider when designing an efficient DR algorithm.

Position History-Based Dead Reckoning (PHBDR) [11] is one of the early variants of the traditional DR algorithm. It predicts the future movement of an entity based on a typical curve-fitting approach. This method uses three latest reported positions sent by the remote host as the basis of extrapolation. In other words, its prediction is independent of the entity's velocity and acceleration. As a result, its correction packet size is smaller than other DR approaches since it contains only the position information.

Another approach, the auto-adaptive DR [3], adjusts the threshold value based on the user interaction with remote entities. Drawing on the concept of interest management [6], area of interest (AOI) of entities are used to determine the threshold dynamically. The algorithm assigns different thresholds to remote entities at different ranges. Therefore, entities of less interest are updated less frequently, which reduces the number of packets.

To avoid extra network consumption yielded by unpredictable movements, the pre-reckoning approach [15] issues an update packet immediately when unpredictable inputs are detected. It introduces a new threshold—angle of embrace—to indicate whether a movement is unpredictable (sharp turn) or not. By utilising this threshold, the number of rollbacks caused by unpredictable movements can be reduced.

To exploit the advantages of various DR algorithms, adaptive classifier system-based dead reckoning [14] relies on a set of rules (such as sensor ranges) to determine which DR algorithm to be adopted at run time. Once these rules are defined by the system designer, they can be discovered automatically by the DVE system.

Some prior work focuses on amelioration of the threshold value used in DR algorithms. [7] defines a metric called 'physics-consistency-cost', which takes into account the consequences caused by rollbacks. Moreover, [10] proposes a time-space threshold and a hybrid scheme in place of the traditional spatial threshold that does not consider time delay.

The approaches reviewed so far can be classified as *mechanical* DR algorithms, since their DRMs are developed based on mechanics. In this paper, we propose a novel DR algorithm called PADR, which improves prediction accuracy by considering human behaviours. Our approach can achieve a great reduction of bandwidth consumption at the cost of a small computational overhead. We have also performed a number of experiments to compare the performance of the proposed approach and the representative DR approaches reviewed in this section, based on a distributed car simulation.

3 PATH-ASSISTED DEAD RECKONING ALGORITHM

Paths are common elements in the DVEs. A Path is usually a road or track laid down by the DVE developers, which allows participants to travel from one meaningful location to another. The participants, particularly those new to the DVE, tend to make their journey along some common paths. This phenomenon can be frequently observed in motorways, narrow areas, and unfamiliar areas.

The key component of the PADR algorithm is a heuristic movement prediction model. Its basic idea is simple: participants do not move randomly in the virtual space. If they move on a path, their next position will very likely be on the same path. Therefore, instead of predicting their next position with a pure kinematic model, PADR directly places them on a predetermined position along a path or an interpolated point between two predetermined positions along a path. On the other hand, if they are not on the path, then the traditional DR prediction model will be applied. In this way, even if the assumption (that the participants tend to follow common paths) is wrong, large prediction deviation can be avoided and the prediction accuracy of the algorithm will be equivalent to the traditional DR algorithm.

The PADR algorithm is comprised of three components, namely path structure, path mapping, and state prediction. They are described in details in the following three subsections.

3.1 Path Structure

In most of the DVE implementations, paths are explicitly modelled by distinctive colours, textures, or functions. For example, on the outskirts of a town, paths are coloured in brown distinguishing from the grassland nearby; or in a race track, driving on asphalt surfaces is much faster than on the sandy ones. Moreover, some natural or artificial facilities are deemed to be part of a path, such as tunnels, bridges, or streets. To model these paths in a simulation, we use a sequence of predefined points to represent them. A path P_i (for $i = 1, 2, \ldots, M$, where M is the number of paths) is defined as a sequence of sampled points along an actual path. The element of the sampling sequence is denoted by

$$\alpha_{ik} = (x_{ik}, y_{ik}.z_{ik}), \quad k = 1, 2, \ldots, N_i \tag{3}$$

where N_i is the number of sampling points on P_i.

3.2 Path Mapping

Path mapping is a process to determine if an entity moves on a path. This process is executed when a prediction violates the entity's actual path (i.e. the deviation between the actual and predicted positions is larger than a predefined threshold). If the mapping process determines that the entity is on-path, the path-assisted model will be used as the prediction model. Otherwise, traditional DRM will be used. The prediction model will remain unchanged until another violation occurs.

However, comparing the entity's position with all sampling points in a regular manner is a computationally intensive process, which can be referred to as the brute-force approach. The time complexity of this approach is $O(\sum_{i=1}^{M} N_i)$, and the space complexity is $O(\sum_{i=1}^{M} N_i)$. Obviously, a more efficient approach should be employed.

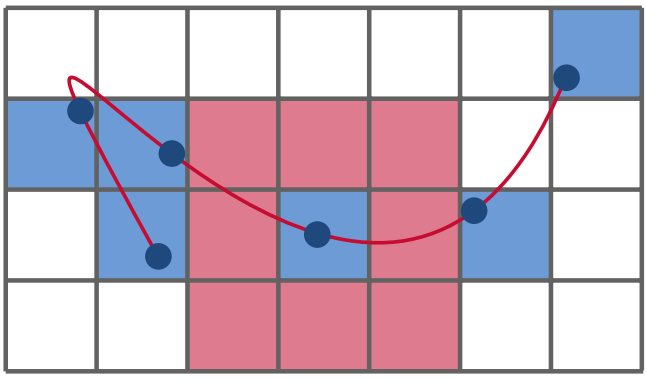


Figure 1: Path mapping

The PADR algorithm employs a spatial hashing method to speed up this process, which is illustrated by the example shown in Figure 1. In the figure, the virtual space is partitioned into uniform grids. The red curve represents an entity's trajectory and the blue dots on the trajectory are sampling points. The grid that a sampling point resides in is coloured in blue to indicates its rough location. All neighbour grids of a blue grid are coloured in red. A blue grid and its neighbouring red grids form a *detection zone* of the corresponding sampling point. If an entity moves into a detection zone and its distance to the sampling point is smaller than a predefined threshold, the entity is considered near the sampling point.

In the initialisation stage of the algorithm, every grid in a detection zone is stored in a hash map H as a key k, and its corresponding value v is the sampling point. In a certain case, a grid may become a neighbour of two sampling points, such that a key k' that belongs to a sampling point v' is identical to a key k that belongs to sampling point v, where $k' = k$. If this is the case, we need to map the grid to its nearest sampling point. Specifically, if the distance between the centre of k and v' is smaller than the distance between the centre of k and v, then the old key-value pair (k, v) will be replaced with the new pair (k, v') in the hash map.

During runtime, the mapping process is very efficient. For any entity, the algorithm first determines which grid it resides in and uses the grid as a key k to retrieve its nearest sampled point, such that $v = H(k)$. The entity is considered off-path if the value v is not found in the hash map. If v exists and the distance between the entity and v is smaller than the threshold, the entity is considered on-path (i.e. near the sampling point v). The general case time complexity of the mapping process for each entity is $O(1)$, which is scalable for large-scale DVEs. The experimental evaluation of the mapping process is presented in Section 5.

3.3 State Prediction

State prediction is an important component of the PADR algorithm since its strength is dependent on the accuracy of its predictions. As discussed previously, if the new heuristic model can indeed increase the prediction accuracy, the number of rollbacks and the bandwidth consumption can be reduced.

The prediction process is described as follows. If the PADR algorithm determines that a participant-controlled entity is on a path and locates its nearest sampling point, in the next time step, PADR will estimate its new position based on its kinematic states as well as the path information. Both the first and second order kinematic models can be incorporated into the PADR algorithm. The algorithm first calculates the distance the entity could travel between two successive time steps. Since it is on a path, the algorithm would assume that it always moves along the path and towards the next sampling point. As discussed previously, this assumption may not be always true. The algorithm needs to regularly monitor the deviation between the entity's actual and predicted positions. Once the deviation exceeds a threshold, the algorithm would issue a rollback, which essentially *resets* the entity's position to the actual one.

It is worth to note that the size of update packet of PADR is smaller than that of the traditional DR approach. Assume that position, velocity, and acceleration, are of the same size of $3u$ in the three-dimensional space. Assume further that the sampling point index has a size of u. When an entity is on a path, the PADR algorithm may issue an $5u$ *on-path* update packet, which consists of two sampling point indices and the entity's velocity. On the other hand, when the entity is not on a path, traditional DRM would be employed. The *off-path* packet contains the entity's position and velocity for the first order model, and the position, velocity, and acceleration for the second order model, which has a size of $6u$ and $9u$, respectively. In summary, the packet size of traditional DR is 1.2 times (first order) or 1.8 times (second order) of the size of the on-path packet.

Given an on-path probability $\mathbb{P}$, the expected packet size of PADR can be calculated by

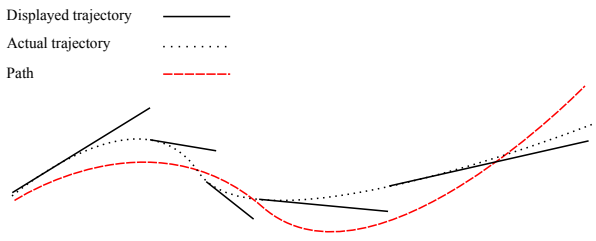
$$\bar{S}_{PADR1} = \left[5\mathbb{P}(I) + 6\mathbb{P}(I')\right] u \tag{4}$$

for the first order model, and by

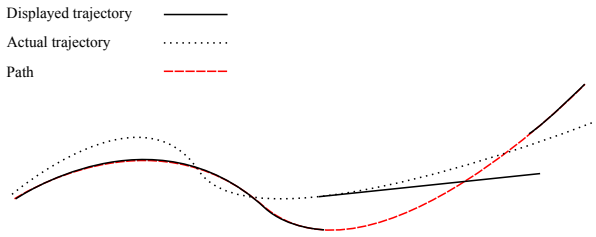
$$\bar{S}_{PADR2} = \left[5\mathbb{P}(I) + 9\mathbb{P}(I')\right] u \tag{5}$$

for the second order model, where I is the event that the participants move along the paths and I' is the event that the participants do not move along the paths.

Figure 2 illustrates two example trajectories of the PADR and traditional DR algorithms. In the figure, the paths, the actual trajectories of the entity, and the displayed trajectories of the entity are indicated by red dashed curves, black dotted curves, and black solid



(a) Trajectory of Traditional DR



(b) Trajectory of PADR

Figure 2: Trajectories of two DR algorithms

curves, respectively. Moreover, no convergence is applied, which implies that once the entity violates the threshold, it would be directly placed on the actual position. We can see from this example that, since the actual trajectory approximates the path, the PADR algorithm violates the threshold less frequently than the traditional DR algorithm, and therefore it issues fewer rollbacks. Consequently, the displayed trajectory of the traditional DR algorithm appears to be more intermittent, where the displayed trajectory of the PADR algorithm appears to be more continuous.

Quantitative evaluations of the two algorithms and other representative DR algorithms are presented in the Section 5.

4 IMPLEMENTATION AND SIMULATION

We have implemented and performed our experiments in TORCS, which is an open source simulator under GNU General Public License (GPL) [13]. TORCS is widely used for realistic 3D car simulations. Its screenshot is shown in Figure 3. In our experiments, each experimenter chose a vehicle to 'drive' in the simulator and compete with other non-human-controlled vehicles. Each simulation consists of three laps but the experimenter were not required to finish them. In order to compare the performance of different DR approaches in a complex and realistic environment, the road track map 'CG Track 3' provided by TORCS was chosen to be the test scenario. This map contains various road surfaces that the vehicles may be 'driven' on. For example, in a grass surface the drivers are difficult to steer and brake, while in a sand surface they are difficult to steer and accelerate. The movements of each of the experimenters (referred to as 'trace') were recorded and were analysed at the end of the simulations.

Figure 3: A screenshot of TORCS

We implemented seven DR algorithms in TORCS, namely PHBDR, the first and second order of pre-reckoning, the first and second order of traditional DR, and the first and second order of the proposed PADR algorithm. The subjects of comparison are the representative DR algorithms reviewed in Section 2. All simulations were performed using the same hardware configurations. Furthermore, the unit of time in TORCS is 0.002 seconds, which implies that the kinematic states of the human-controlled vehicles were updated every 0.002 seconds in the simulations.

5 EXPERIMENTAL RESULTS AND EVALUATION

Three metrics were adopted to evaluate the DR algorithms. The first metric is the prediction accuracy of the DRM, which is measured in terms of the number of rollbacks. As discussed previously in Section 1, the performance of DR depends strongly on the prediction accuracy of the DRM. If rollbacks are issued frequently, a significant bandwidth consumption would be induced and the participants might experience serious lag effects on the client side.

The second and most important metric is bandwidth consumption, which refers to the term $M \times B$ in the resource equation described in Section 2. It reflects how well a DR algorithm can alleviate the network bandwidth consumption in a DVE. We calculated the packet size by using the method described in Section 3.3 and measured the number of packets sent during the simulations. For PHBDR, the rollback packets contain only the position of an entity; therefore its packet size is $3u$. The rollback packets of pre-reckoning and traditional DR contain a typical kinematic state of an entity, which has a size of $6u$ and $9u$ for the first and second order models, respectively.

The third metric is computational efficiency, which is an important consideration when employing DR algorithms. Due to the fact that DVEs are often used for real-time applications, using a fast DR algorithm is crucial for maintaining responsiveness as well as enhancing system scalability. The computational efficiency is measured by the elapsed CPU clock ticks to run the DR algorithms.

5.1 Simulation Map

Although the map shown on the top-right corner of Figure 3 is two-dimensional, the actual map used for the simulations is three-dimensional. Figure 4 illustrates the shape of the actual map in 3D. Compared to the 2D map, the actual track goes up and down on the z-axis, which is a typical scenario in a 3D car simulation. Figure 5 illustrates one of the trajectories generated by a participant-controlled vehicle (for three laps). We can see that the trajectories closely approximate the actual track, except in some turns. This is normal as human participants were not familiar with the track and therefore could not always turn the vehicle smoothly.

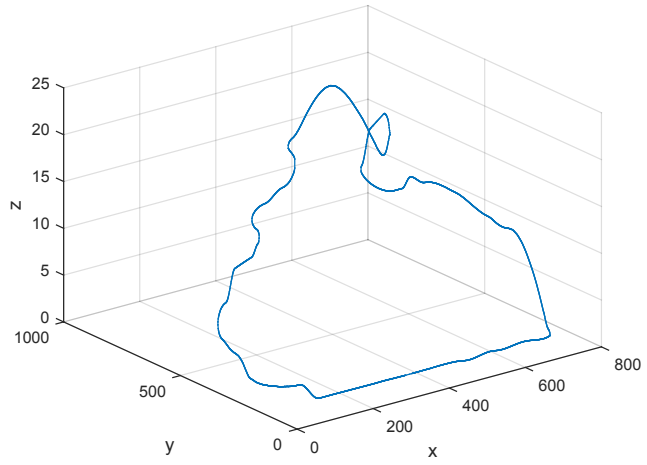


Figure 4: Path shape

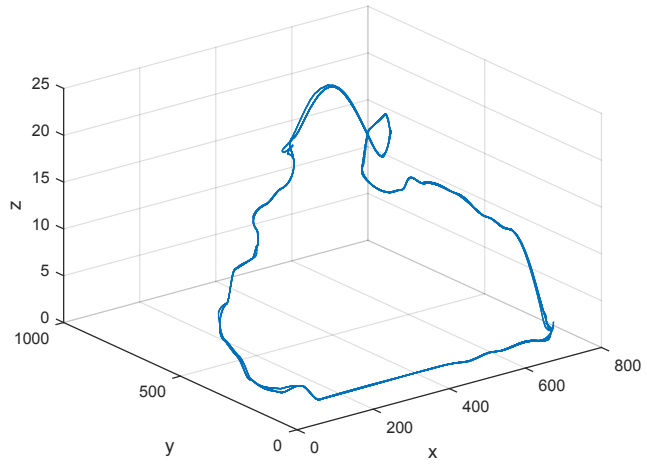


Figure 5: A collected actual trajectory

5.2 Prediction Accuracy

Figure 6 compares the seven algorithms in terms of the number of rollbacks. The first observation is that, conforming to the study in [9], all second order algorithms incurred less rollbacks than their corresponding first order versions. This suggests that the second order DRMs are always more accurate than the first order models.

Secondly, we can see that the PHBDR approach issued the most rollback packets in the experiment. Although its DRM adopts the angle of embrace concept to avoid sharp turns, it still could not keep up with the actual path. Therefore, it can be concluded that using curve fitting based on position history is insufficient to accurately predict the future path of participant-controlled vehicles.

Another noticeable information is that the prediction accuracy of pre-reckoning algorithms is almost the same as the traditional DR algorithms in both the first and second order models. In fact, the lines representing the results of the two models are overlapped on the line graph. One reason for this finding might be due to the small number of sharp turns in the simulation scenario (see Figure 4), which in turn could not demonstrate the strength of the pre-reckoning approach.

The most important finding in this experiment is the performance of the proposed PADR approach. We can see from Figure 6 that when threshold value is small, pre-reckoning and traditional DR issue slightly fewer rollbacks than PADR; however when the threshold value increases, PADR would issue fewer rollbacks than them. This is due to the fact that although the track looks like a line in Figure 5, it actually has a certain minimum width. The PADR model is able to put the extrapolated vehicle on the right track, but the prediction may not follow the exact path of the participant-controlled vehicles. When the threshold is small, the DR system would become very strict on the prediction results. It would determine that many of the PADR predictions are incorrect even though they actually follow the track. On the other hand, when the threshold increases (note that: to approximately the half-width of the track), most of the PADR predictions that follow the track would be determined as correct predictions, which leads to significantly fewer rollbacks than all other DR algorithms. According to our simulation results, the second order PADR algorithm issued at least 26% fewer rollbacks than the second best approach—the second order pre-reckoning algorithm. This suggests that the proposed PADR approach has the best prediction accuracy when the threshold approximates or is larger than the half-width of the path.

5.3 Bandwidth Consumption

Figure 7 shows the results of the second set of experiments, which compare the bandwidth consumption (in terms of total packet size) of the seven DR algorithms. The packet sizes are calculated based on the assumptions described in Section 3.3 and their unit is u. The results clearly show that the proposed PADR approach consumes less bandwidth than all other approaches.

Among the subjects of comparison, PHBDR has the best performance. Since it sent the most rollbacks in the previous set of experiments, this finding is somewhat unexpected. However, consider the fact that PHBDR's rollback packet contains only the position of the entity, which is half or one-third of the size of other approaches, its bandwidth consumption should certainly be much lower than other approaches.

Similarly, although the second order DRMs have better prediction precision (see Figure 6), they are more bandwidth-consuming than their corresponding first order version. This suggests that the

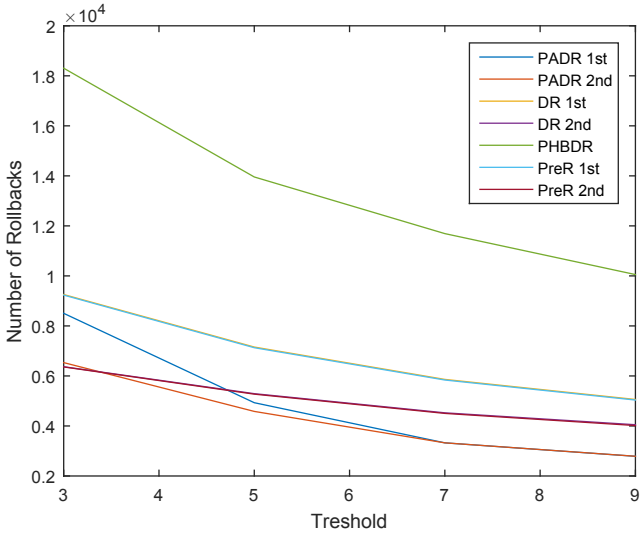


Figure 6: Prediction accuracy

number of rollbacks is not the only consideration for minimising the bandwidth consumption, but the packet size is also an important factor.

It should also be noted that the bandwidth consumption of the pre-reckoning algorithms is almost the same as the traditional DR algorithms in both the first and second order models. Similar to the previous set of experiments, the lines representing the two algorithms are overlapped on the line graph. This confirms our expectation since they have the same packet size and issued approximately the same number of rollbacks.

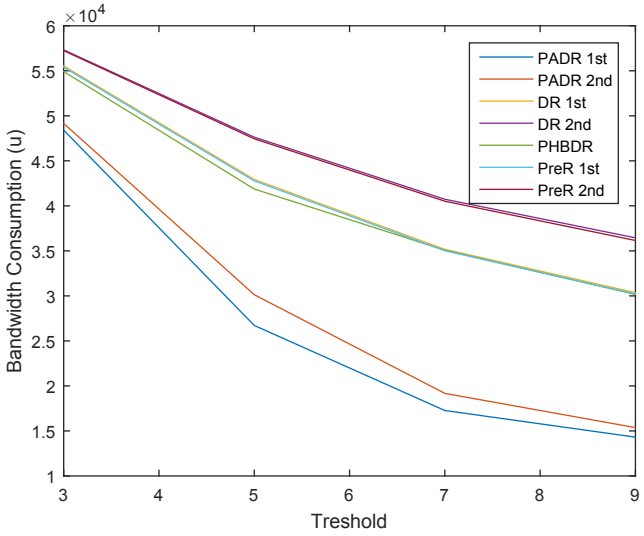


Figure 7: Bandwidth consumption

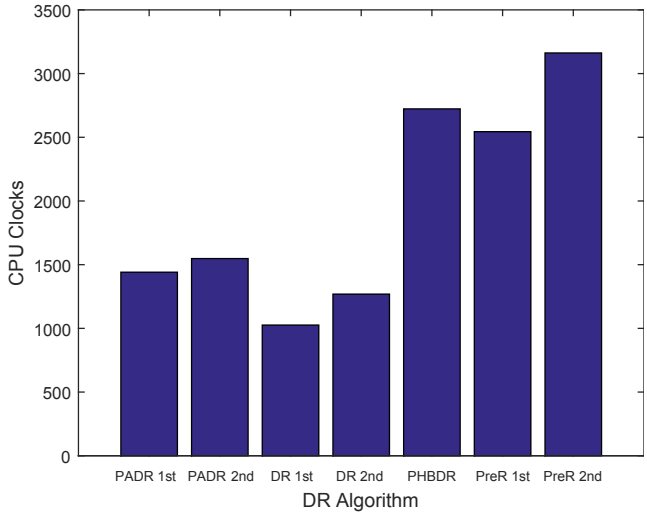


Figure 8: Computational cost

5.4 Computational Efficiency

Figure 8 shows the computational cost of the seven DR algorithms with a constant threshold value of 10. The cost is measured by the number of clock ticks the CPU spends running these algorithms. We found that PHBDR and pre-reckoning algorithms have a very high computational cost. According to our algorithm profiling, this is caused by the computation of the angle of embrace.

Furthermore, the results show that the two traditional DR algorithms are the fastest among the tested approaches. This is due to the fact that their calculations are much simpler than the others. The PADR algorithms are only slightly slower than the traditional DR algorithms. If we take the previous observations into account, it can be concluded that the PADR approaches significantly reduce the number of rollbacks and bandwidth consumption at the cost of a small computational overhead.

6 CONCLUSIONS

DR in DVE is a movement prediction technique, which is designed to reduce bandwidth consumption and alleviate the lag effects caused by network latency. Most of the existing DR approaches in the literature perform prediction based on kinematic models. This paper proposes a new DR approach called PADR, which performs extrapolations with the considerations of environmental factors and human behaviours. The new approach is particularly suitable for DVEs that contains paths such as distributed car simulations. We have implemented and evaluated seven DR algorithms on a widely used 3D car simulator—TORCS. Experimental results have demonstrated that the proposed PADR approach significantly reduces the number of rollbacks and bandwidth consumption at the cost of a small computational overhead.

Our future work concerns two main directions. First, we will evaluate PADR and other DR algorithms based on the movement data obtained from a real-life massively multiplayer online role-playing game (MMORPG). User behaviours in a MMORPG are much different from that of a car simulation. The participants usually

spread out in the virtual space and do not follow a single path. Therefore, the DR algorithms are expected to perform incorrect predictions more often in a MMORPG than in a car simulation. Second, we will develop a dynamic prediction model, which generates path structures during runtime. Instead of using a predefined path, the new model will create and remove paths according to the participants' movement patterns and performs prediction based on the commonness of the trajectories.

ACKNOWLEDGMENTS

The research reported in this paper is financially supported by the Tier 1 Academic Research Fund (AcRF) under project Number RG136/14.

REFERENCES

[1] Richard Allan Bartle. 2003. *Designing Virtual Worlds.* New Riders Games.
[2] Wymann Bernhard. 2001. torcs News. http://torcs.sourceforge.net/. (2001).
[3] Wentong Cai, Francis Lee, and Lian Chen. 1999. An auto-adaptive dead reckoning algorithm for distributed interactive simulation. In *Proceedings of the thirteenth workshop on Parallel and distributed simulation.* IEEE Computer Society, 82–89.
[4] J. Calvin, A. Dickens, R. Gaines, P. Metzger, D. Miller, and D. Owen. 1993. The SIMNET Virtual World Architecture. In *Proceedings of the IEEE Virtual Reality Annual International Symposium.* 450–455.
[5] R.M. Fujimoto. 2000. *Parallel and distributed simulation systems.* Wiley.
[6] Elvis S Liu and Georgios K Theodoropoulos. 2014. Interest management for distributed virtual environments: A survey. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 51.
[7] Seamus C McLoone, Patrick J Walsh, and Tomas E Ward. 2012. An enhanced dead reckoning model for physics-aware multiplayer computer games. In *Distributed Simulation and Real Time Applications (DS-RT), 2012 IEEE/ACM 16th International Symposium on.* IEEE, 111–117.
[8] David L. Neyland. 1997. *Virtual Combat: A Guide to Distributed Interactive Simulation.* Stackpole Books.
[9] Lothar Pantel and Lars C Wolf. 2002. On the suitability of dead reckoning schemes for games. In *Proceedings of the 1st workshop on Network and system support for games.* ACM, 79–84.
[10] Dave Roberts, Rob Aspin, Damien Marshall, Seamus Mcloone, Declan Delaney, and Tomas Ward. 2008. Bounding Inconsistency Using a Novel Threshold Metric for Dead Reckoning Update Packet Generation. *Simulation* 84, 5 (May 2008), 239–256.
[11] Sandeep K Singhal and David R Cheriton. 1994. *Using a position history-based protocol for distributed object visualization.* Technical Report. DTIC Document.
[12] Jouni Smed, Timo Kaukoranta, and Harri Hakonen. 2002. *A Review on Networking and Multiplayer Computer Games.* Technical Report 454. Turku Centre for Computer Science.
[13] Richard Stallman et al. 1991. Gnu general public license. *Free Software Foundation, Inc., Tech. Rep* (1991).
[14] Samir Torki, Patrice Torguet, and Cédric Sanza. 2007. Adaptive Classifier System-Based Dead Reckoning.. In *Proceedings of the 10th Immersive Projection Technology Workshop at the 13th Eurographics Symposium on Virtual Environments (IPT/EGVE 2007).* 101–108.
[15] Xiaoyu Zhang, Denis Gračanin, and Thomas P Duncan. 2004. Evaluation of a pre-reckoning algorithm for distributed virtual environments. In *Parallel and Distributed Systems, 2004. ICPADS 2004. Proceedings. Tenth International Conference on.* IEEE, 445–452.