유효갱신시간에 기반한 가변 데드레코닝 알고리즘

유석종, 정혜원°, 최윤철
연세대학교 컴퓨터과학과

An Adaptive Dead Reckoning Algorithm using Update Lifetime

SeokJong Yu, HyeWon Jung, YoonChul Choy
Dept. of Computer Science, Yonsei Univ.

Abstract

This paper proposes a new, adaptive Dead Reckoning model, called Dynamic Dead Reckoning, for Distributed Interactive Simulation and humanoid avatar systems. The proposed model can overcome the weak points of traditional Dead Reckoning caused by a fixed threshold and strong dependency on rotation event. This paper introduces new criteria for update message filtering, named as Update lifetime. The Dynamic Dead Reckoning keeps the balance between extrapolation fidelity and filtering performance by two component models, Variable Threshold Mechanism and Rotation Event Model. The experimental results show that the proposed model can lower the increment rate of update traffic to the increase of rotation frequency without any significant loss of accuracy.

1. Introduction

Participating in interactive, multi-player network games on the Internet is becoming more common activity. But, in spite of rapid development of computer and the Internet technologies, there still exist many obstacles to the achievement of Internet-based collaboration. In the development of such distributed collaborative systems, the key issues are the securing of consistency and scalability. For continuous sharing of an environment, state update messages must be exchanged on the network among client hosts. The amount of state update message increases in proportion to the number of participant. For example, where n clients share a virtual environment with each periodically publishing events m times per second, the system must process the propagation of $m*(n-1)$ messages per second [1].

State update message filtering is a process to reduce message traffic and to improve scalability of system, and is one of the most important issues in Distributed Interactive Simulation (DIS), and various networked applications. *Spatial Partitioning schemes* [2, 3, 4], such as Hexagonal cell, Locale, and AOI, provide primary filtering methods to restrict the region of message propagation by certain spatial models, and to reduce the amount of update traffic.

Dead Reckoning is a representative method using *Event Sampling scheme*. It provides an active filtering algorithm that aims at sharing of the movement of entity [5]. Previous research works on Dead Reckoning have been focused on the evaluation of extrapolation equations [6] and the performance investigation of DR mechanism [7].

Traditional Dead Reckoning is efficient for straightly moving vehicles of DIS, but it has a limitation as follows. The fixed threshold mechanism of Dead Reckoning has a trade-off between update traffic and extrapolation accuracy, which means, it does not guarantee both sides at the same time. The performance of Dead Reckoning is entirely dependent on rotation events of entity. Especially, in the movements of high rotation frequency such as humanoid avatars make, it is hard for Dead Reckoning to achieve a good performance. This paper describes an adaptive Dead reckoning algorithm as a solution to the weak points of traditional Dead Reckoning.

2. Lifetime of state update

2.1 Traditional Dead Reckoning

Dead reckoning is a message reduction method based on a path estimation algorithm. The state of an entity is updated when a distance error between two paths is larger than given threshold. Path extrapolation is performed by following equations.

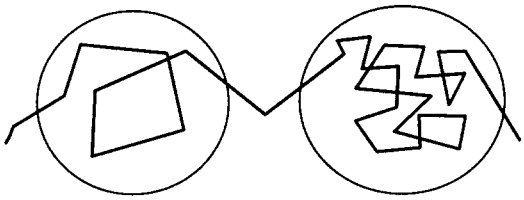
$$X_t = X_{t'} + V_{t'}\tau, \quad Y_t = Y_{t'} + V_{t'}\tau, \quad Z_t = Z_{t'} + V_{t'}\tau \quad (1)$$

$(X_{t'}, Y_{t'}, Z_{t'})$ and $V_{t'}$ represent respectively the position and velocity of an entity as found in the update message of the last state. τ is an elapsed time since the last update. The above formulas are used to extrapolate the position of the entity at time $t = t' + \tau$.

2.2 Update lifetime

Lifetime of state update (*Update lifetime*) can be defined as a valid duration time of state update, or the time interval between two consecutive updates. Fig. 1 describes two typical situations of different *Update lifetimes*. There is a big difference in the rotation frequency of the paths A and B,

assuming that the moving velocity is equal. Intuitively, it is found that average *Update Lifetime* of path A is longer than that of the path B. The path B of short *Update Lifetime* of path consumes more network resource for the same time interval, but nevertheless the importance of each state update to the whole path is lower than the path A. The principle of filtering using Update lifetime is to save network resources consumed in the intervals composed of shorter update lifetime than a specified level.



A. Low rotation frequency B. high rotation frequency

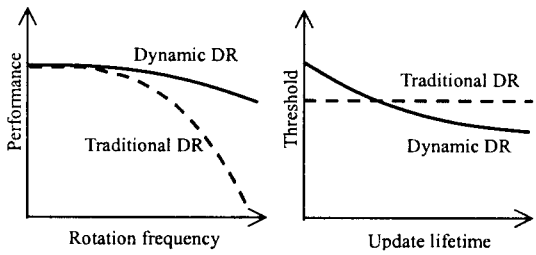
Fig. 1 Two different lifetimes of entity state update

2.3 Rotation frequency

How frequent and how big an entity rotates determines the number of state update. *Update lifetime* is ruled by rotation behaviors of entity, especially, by *Rotation frequency*. It is defined as the number of rotations during a specified time unit. The path B can cause an additional message processing load as well as the over-consumption of network resource.

3. Dynamic Dead Reckoning algorithm

The ultimate aim of the proposed DDR is the adaptive control of update traffic based on the lifetime of state update. The filtering performance of DR rapidly falls in inverse proportion to the increase of rotation frequency for the innate characteristics of filtering mechanism. DDR can overcome this weak point of traditional DR by two proposed models: *Variable Threshold Mechanism* (VTM) and *Rotation Event Model* (REM).



(a) Performance to rotation frequency
(b) Variable threshold mechanism

Fig. 2 Comparison of two Dead Reckonings

3.1 Variable Threshold Mechanism

Fig. 2 (b) shows the threshold strategy of VTM. It uses a threshold range, instead of a fixed threshold value, which provides a mechanism to search an optimal threshold within the threshold range based on *Update lifetime*. The threshold range is specified by given minimum and maximum values. In the intervals of short update lifetime, filtering performance is increased by assigning a larger threshold than default one. On the contrary, in the intervals of long update lifetime, a smaller threshold improves extrapolation accuracy.

3.2 Rotation Event Model

REM plays roles of the measurement and analysis of rotation events, especially, *Rotation interval*, which is the control factor to determine *Update lifetime*. When there exists any necessity in reassigning new threshold, REM transfers related information to VTM. The major tasks of REM are summarized as follows:

- REM stores certain amount of recent rotation events in *Event Queue*.
- REM evaluates *Update lifetime* as the time interval between rotation events.
- REM differentially imposes an additional penalty on *Update lifetime* according to the recentness of events.
- REM determines a new threshold with reference to rotation frequency history.
- REM executes the loop of above tasks every time a new rotation event occurs.

● *Event Queue* and *Event Queue time*

REM provides *two-level Event Queues (EQ)*, *Local EQ* and *Global EQ*, which are temporary time buffers for storing recent events and average update interval, respectively. *Event Queue time* is a limitation time required for each event to have to stay in *EQ*. The events elapsed more than *EQ* time are deleted from *EQ*.

● *Rotation Interval*

Rotation Interval (*RI*) is the temporal interval between two consecutive rotation events in *EQ* (Fig. 3). *RI* can be used as a substitute for *Update lifetime* since the time rotation occurs is equal to the time entity state is updated in case of the ideal condition that threshold is 0, and all the events are shared. When *i-th* rotation event, R_i, occurs, *Rotation Interval* is defined as

$$RI_i = | t(R_i) - t(R_{i-1}) |$$
$$(t(R_i) > t(R_{i-1}), i=1, 2, \ldots, n) \quad (2)$$

where $t(R_i)$ is the time that R_i occurs.

● *Average Rotation Interval*

To judge recent rotation pattern, *Average Rotation Interval* (*ARI*) is measured. *ARI* is calculated with *RIs* of current *Local EQ*, and it is in inverse proportion to rotation frequency. *ARI* is 0 when there exists no event for recent *EQ time*.

$$ARI_k = (RI_k + RI_{k-1} + \ldots + RI_{k-Nk+2}) / (N_k - 1)$$
$$(k = 1, 2, \ldots, l) \quad (3)$$

where N_k and l are the number of rotation events and the number of ARIs, respectively.

● *Rotation Penalty*

In equation (4), all the *RIs* of *Local EQ* uniformly participate in the calculation of ARI. But there exist a difference between late and early *RIs* in the influences on state updating. *Rotation Penalty* is a penalty differently imposed on the events of *EQ* in a way to shorten *RIs* since the rotations lately occurred become a heavier burden on message processing than earlier ones. *Rotation penalty* makes use of the fact that smaller *ARI* gets a bigger threshold. Each *RI* is resized as much as the weight value of the Sub-Interval to which it belongs (Fig. 3). *Rotation Penalty* is obtained as follows:

$$ARI_k^{RP} = (\sum RI_i\, w_j) / (N_k - 1) \quad (i=k, k-1, \ldots, k-N_k+2,\ N_k>1) \quad (4)$$

$$w_j = w_1 - (j-1) \times dw / (m-1) \quad (t(SI_{j-1}) < t(R_i) \le t(SI_j)) \quad (5)$$

$$dw = |w_1 - w_m| \quad (w_1 > w_m) \quad (6)$$

$$t(SI_j) = \text{current time} - \Delta t_{local} \times (m-j)/m \quad (j=0, \ldots, m) \quad (7)$$

where m is the number of Sub-Intervals of local EQ, w_i is Rotation Penalty for RIs of SI_i, and Δt_{local} is Local EQ time.

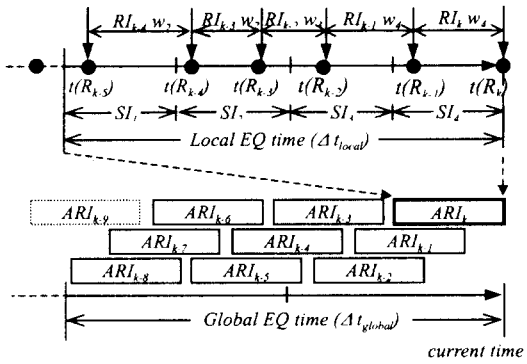


Fig. 3 RI and ARI

● *Recent Rotation Frequency*

For more exact comparison of current *ARI* with the past *ARIs*, *Global EQ* stores the past *ARIs* occurred for Global EQ time. *Recent Rotation Frequency (RRF)* is obtained by the comparison of *ARIs* of *Global EQ*. *RRF* is the extent of current rotation frequency compared with the rotation interval history. *RRF* is calculated as follows:

$$RRF_k = 1 - (ARI_k^{RP}{}_{current} - ARI_k^{RP}{}_{min}) / (ARI_k^{RP}{}_{max} - ARI_k^{RP}{}_{min}) \quad (ARI_k^{RP}{}_{max} > ARI_k^{RP}{}_{min}) \quad (8)$$

where $ARI^{RP}{}_{current}$ is current ARI of Local EQ, and $ARI^{RP}{}_{max}$ and $ARI^{RP}{}_{min}$ are the largest and the smallest ones among the ARIs of Global EQ, respectively.

● *Dynamic Threshold*

Dynamic Threshold (DT) is the threshold used in DDR, and current *DT* is valid only until a new rotation event occurs.

$$DT^{RI}{}_k \begin{cases} default\ threshold & (if\ RRF\ is\ not\ defined) \\ thd_{min} + RRF_k \times (thd_{max} - thd_{min}) & (elsewhere) \end{cases} \quad (9)$$

where thd_{max} and thd_{min} are specified values

Cases	*RRF*	*DT*
$ARI_{max} <= ARI_{min}$	not defined	default
$ARI_{current} == ARI_{max}$	1	thd_{max}
$ARI_{current} == ARI_{min}$	0	thd_{min}

Table 1. Relationship among *ARI, RRF,* and *DT*

4. Experimental Results

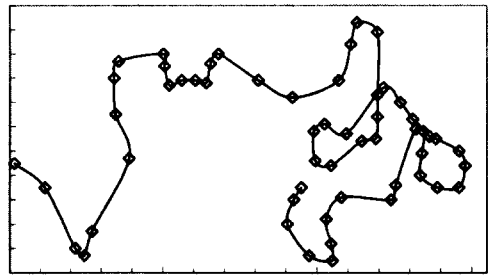
To evaluate the performance of Dynamic Dead Reckoning, statistics, such as update traffic, extrapolation errors, and trajectories of moving paths, were measured.

Parameters	Initial values
Local EQ time (second)	30
Global EQ time (second)	200
Number of avatars (entity)	100
Threshold (meter)	2, 6, 10 (TDR) 2 ~ 10 (DDR)

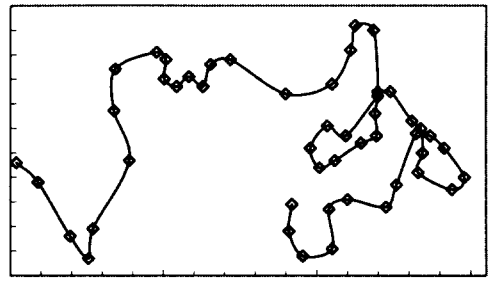
Table 2. Experimental parameters

4.1 Tracing of moving paths

To offer comparison of performance visually, the moving paths were traced as shown in Fig. 4. A square shape in the paths symbolizes the position of rotation. In TDR path, the number of update increases in sharp curve intervals, compared with DDR paths.



(a) Traditional DR (updates = 55)



(b) Dynamic DR (updates = 49)

Fig. 4 Comparison of two filtered paths

4.2 Update message measurement

To evaluate the performance of DDR, update traffic was measured with changing the combination ratio of two kinds of RI, Short-RI (1~8 sec) and Long-RI (11~18 sec). Table 3 lists major experimental results.

Parameters	TDR	DDR
Average update traffic	713.8	659.1
Average traffic increment rate	6%	3%
Average extrapolation error	0.5776 meter	0.6518 meter

Table 3. Principal experimental results

According to the results, it is found that DDR reduced the increment rates of traffic to the increase of *Rotation frequency*, which is the control factor to filtering performance. The curve of DDR is less sharp than that of mean of TDRs when the ratio of Short-RI increased (Fig. 5). This implies the fact that DDR is effective on the movement of high rotation frequency.

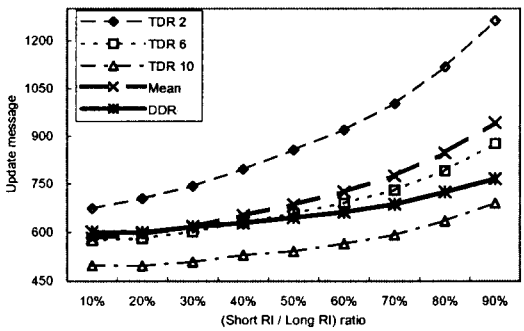


Fig. 5 Comparison of update traffic

4.3 Extrapolation accuracy

For the comparison of extrapolation accuracy to *Rotation Interval*, distance errors between real path and dead reckoned path per state update were measured (Fig. 6). Average error ratio of average errors of DDR to TDR was 112.8% (Table 3), and the difference was 0.0742 meter. In other word, the accuracy of DDR algorithm to TDR was 87.2%. This shows that DDR guarantees the accuracy of filtering algorithm close to that of TDR.

5. Conclusion

The most important contributions of the proposed Dynamic Dead Reckoning are as follows. One is the introduction of new filtering criteria, *Update lifetime*, which is a control factor to performance. The other is the suggestion of *Variable Threshold Mechanism* and *Rotation Event Model* that can overcome the weak point of traditional DR, and dynamically maintain the balance between fidelity and performance of path extrapolation. According to experimental results, compared with traditional DR, Dynamic DR reduced the increment rate of update traffic to the movement of high rotation frequency. They also show that Dynamic DR achieved better performance without any significant loss of accuracy. The proposed model will be helpful in improving the performance when it is applied to humanoid avatar-based cyberspaces. In the future work, we will study a better adaptable method to incorporate more parameters into filtering algorithm.

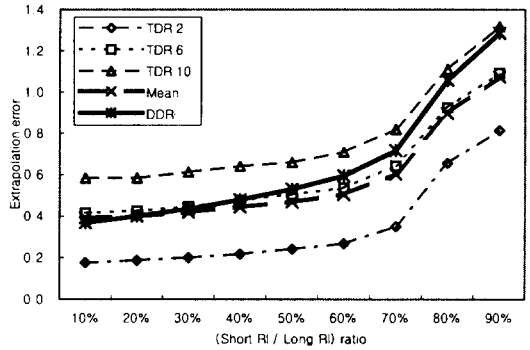


Fig. 6 Comparison of extrapolation error

References

[1] Funkhouser, T. A. RING: A client-server System for multi-user Virtual Environments. Computer Graphics (1995 SIGGRAPH Symposium on Interactive 3D Graphics), Monterey, CA, April 1995, pp. 85-92.

[2] Barrus, J.W., Waters, R.C., and Anderson, D.B. Locales and Beacons: Efficient and Precise Support for Large Multi-user Virtual Environments. Mitsubishi Electric Information Technology Center America. (see http://www.merl.com/reports/TR95016a).

[3] Hagsand, O. DIVE- A platform for multi-user virtual environments. IEEE Multimedia Vol. 3, No. 1, pp. 30-39, 1996.

[4] Macedonia, M., Pratt, D., and Zyda, M. NPSNET: A network software architecture for large scale virtual environments. Presence. Vol. 3, No. 4, pp. 265-287, Fall 1994.

[5] Bassiouni, Mostafa A. and Chiu, Ming-Hsing. "Performance and reliability analysis of Relevance filtering for scalable distributed interactive simulation", ACM Transactions on Modeling and Computer Simulation, Vol. 7, No. 3, July 1997, Pages 293-331.

[6] Lin, Kuo-Chi. Dead Reckoning and Distributed Interactive Simulation. In Proc. of SPIE conference (AeroScene'95), Orlando Florida, April 1995.

[7] Durbach, Corention and Fourneau, Jean-Michel. Performance Evaluation of a Dead Reckoning Mechanism. In Proc. of IEEE 2nd Int. Workshop on Distributed Interactive Simulation and Real-time Applications, pp. 23-29, July 1998.