STATISTICAL DETERMINATION OF HYBRID THRESHOLD PARAMETERS FOR ENTITY STATE UPDATE MECHANISMS IN DISTRIBUTED INTERACTIVE APPLICATIONS

Damien Marshall, Seamus McLoone, Declan Delaney, Tomas Ward

°National University of Ireland Maynooth
Maynooth,
Co. Kildare,
Ireland

† Contact: damien.marshall,seamus.mcloone@eeng.nuim.ie

Abstract

Collaboration within a Distributed Interactive Application (DIA) requires that a high level of consistency be maintained between remote hosts. However, this can require large amounts of network resources, which can negatively affect the scalability of the application, and also increase network latency. Predictive models, such as dead reckoning, provide a sufficient level of consistency, whilst reducing network requirements. Dead reckoning traditionally uses a spatial error threshold metric to operate. In previous work, it was shown how the use of the spatial threshold could result in potentially unbounded local absolute inconsistency. To remedy this, a novel time-space threshold was proposed, that placed bounds on local absolute inconsistency. However, use of the time-space threshold could result in unacceptably large spatial inconsistency. A hybrid approach that combined both error threshold measures was then shown to place bounds on both levels of inconsistency. However, choosing suitable threshold values for use within the hybrid scheme has been problematic, as no direct comparisons can be made between the two threshold metrics. In this paper, a novel comparison scheme is proposed. Under this approach, an error threshold look-up table is generated, based on entity speed and equivalent inconsistency measures. Using this look-up table, it is shown how the performance of comparable thresholds is equal on average, from the point of view of network packet generation. These error thresholds are then employed in a hybrid threshold scheme, which is shown to improve overall consistency in comparison to the previous solution of simply using numerically equal threshold values.

1. Introduction

One of the key factors in a Distributed Interactive Application (DIA) is the maintenance of a sufficient level of consistency between remote hosts involved in the application. Consistency is the degree to which remote views of the same environment agree, and is required to be of a high standard for collaboration between remote hosts to be successful [1].

However, maintaining consistency is expensive and can negatively impact on the scalability of the application. Indeed poorly applied consistency maintenance techniques can, paradoxically, decrease remote consistency. A common method for reducing the performance requirements for Distributed Interactive Applications, thus improving scalability and reducing latency, is the use of predictive schemes such as the Hybrid Strategy Model and Dead Reckoning [2,3]. Dead reckoning is one the most widely used predictive techniques. It uses a short-term predictive model based on entity dynamics to reduce network traffic. Under the dead reckoning scheme, an update is transmitted between hosts containing entity dynamics information. The remote hosts then use this data to model the behaviour of the entity in the short term. This decouples the simulation participants, to an extent, allowing the simulation to proceed independently between update messages which are always subject to some latency. A further update is not transmitted until the modelled and actual path deviate by a specific error threshold value.

Traditionally, this threshold value is the spatial difference between the actual and modelled position. In previous work, however, it was demonstrated that the conventional approach of using a purely spatial threshold for dead reckoning gives bounded spatial inconsistency but potentially unbounded absolute inconsistency [4]. Absolute inconsistency takes into account the duration of the spatial inconsistency [5]. A novel threshold metric to replace the existing dead reckoning spatial error threshold was proposed. This threshold, known as the time-space error threshold, uses local absolute inconsistency measures in determining when an update is required, and was shown to result in bounded local absolute inconsistency but unacceptably large spatial inconsistency. A hybrid threshold combining both a spatial error threshold and the time-space error threshold metric results in both bounded spatial inconsistency and bounded absolute inconsistency.

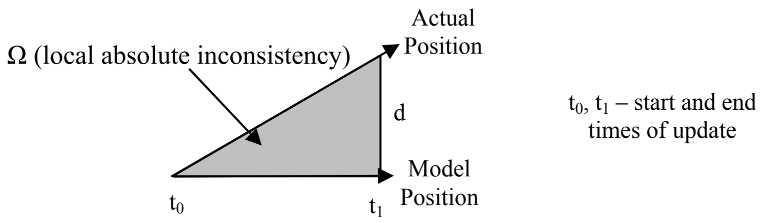
However, choosing error threshold values for use within the hybrid scheme has been problematic to date. As both error thresholds use different measures of inconsistency to operate, no direct comparison can be made between them, meaning that their relative performance cannot be judged, and more importantly, suitable thresholds for use within the hybrid scheme cannot be accurately chosen.

In this work, a novel approach that allows spatial and time-space error thresholds to be compared is proposed. Using average inconsistency measures and modelled entity speed, an error threshold look-up table can be generated. Suitable time-space error thresholds can then be chosen using a specific spatial error threshold.

The rest of the paper is structured as follows. In Section 2, a brief background to the hybrid threshold metric is given. Section 3 examines the impact of entity speed on the spatial and time-space threshold metrics. Section 4 describes how the average inconsistency values arising from variable speed values are used to generate an error threshold look-up table. Results using comparable error thresholds are presented in both single and hybrid cases in Section 5. The paper finishes with some conclusions and details of future work in Section 6.

2. The Hybrid Threshold Metric

For the benefit of the reader, the key operation mechanisms of the hybrid threshold scheme will now be summarised. In a Distributed Interactive Application, one of the most common measures of inconsistency is spatial inconsistency. This is simply the distance between the actual position of a virtual entity on the local host and the position on the remote host. Under the dead reckoning scheme, an update is only transmitted when a specific level of spatial inconsistency is reached, represented by *d* in Figure 1. It should be noted that the actual position path can take the form of complex paths. Here, we represent the path as linear for illustrative purposes.



Ω (local absolute inconsistency)
Actual Position
d
Model Position
t_0
t_1
t_0, t_1 – start and end times of update

Figure 1. The actual and modelled path using dead reckoning and a spatial threshold metric.

However, the spatial threshold metric does not take the duration of the inconsistency into account. Zhou et al. provide a method of measuring inconsistency in both the time and space domain [5]. This measure takes into account the duration of the spatial error, reasoning that the duration that the error lasts for, as well as the size of the error, is important to the end user. The authors use this metric to measure the absolute inconsistency arising from a Distributed Interactive Application simulation before execution time. The inconsistency measure is given in Equation 1.

$$\Omega = \begin{cases} 0, & \text{if } |\Delta(t)| < \varepsilon \\ \int_{t_0}^{t_0+\tau} |\Delta(t)|\,dt, & \text{if } |\Delta(t)| \geq \varepsilon \end{cases} \quad (1)$$

Here,

Δ is the difference between the position of a local object and its remote replication;
$\Delta(t)$ is the above difference over a duration t;
t_o is the start time of the inconsistency and τ is the duration;
ε is the minimum perceivable distance.

Basically, this measure is the area underneath the curve given by spatial inconsistency over time, given by Ω in Figure 1. In previous work it was demonstrated, using Zhou's metric, that local absolute inconsistency is potentially unbounded when a spatial threshold is employed [4]. To rectify this, a new dead reckoning threshold metric, known as the time-space threshold metric, was proposed. This metric employs Zhou's measure of absolute inconsistency in determining when a dead reckoning update should be transmitted. It was shown that the use of this metric places a bound on local absolute inconsistency but could result in unbounded spatial inconsistency.

To remedy this, a hybrid method, which combines both the spatial and time-space metrics into a single metric was then proposed, and was shown to place a bound on both levels of inconsistency [4]. However, a problem then arises in choosing suitable time-space error threshold values for use in the hybrid scheme. The previously suggested solution of using two error thresholds of equal value is not sufficient, as there is no clear reasoning why any particular pair of values should be chosen.

Methods of choosing spatial error thresholds based on environment characteristics have already been proposed in other work. In this paper, we propose that by first choosing a spatial threshold, a comparable time-space error threshold can then be found by comparing average spatial and absolute inconsistency measures. Such thresholds can subsequently be used within a hybrid threshold metric. Other work has already demonstrated different that dead reckoning error threshold schemes can be compared using average inconsistency measures. Lee et al. use an area of interest management scheme to vary the dead reckoning error threshold based on distance[6]. As entities within the environment become closer, the error threshold is reduced so as to aid closely coupled interaction. In the work by Cai et al. , the spatial error threshold is adapted based on the number of rotations made by the entity in its short-term history [7]. The more rotations made, the higher the threshold, and vice versa. The pre-reckoning algorithm from Duncan et al. employs a variable spatial threshold that detects the likelihood of the error threshold being breached within the near future, and sends a packet based on this knowledge [8].

However, as will now be demonstrated in the next section, the impact of entity speed on the inconsistency arising from use of both error metrics must first be considered before any comparisons can be made.

3. Impact of Entity Speed

Consider the example scenario presented in Figure 2. At time t_0, a dead reckoning update is transmitted, so the error between the actual and modelled position is zero. Subsequently, the actual entity position moves away from the modelled path. In this case, the time-space threshold metric is employed. If an entity speed of v_1 is used, another update is transmitted when the local absolute inconsistency, Ω, reaches the error threshold at time t_1 i.e. when the area under the graph given by spatial inconsistency and time is equal to a predefined error threshold value. The spatial inconsistency at this point is given by d_1.

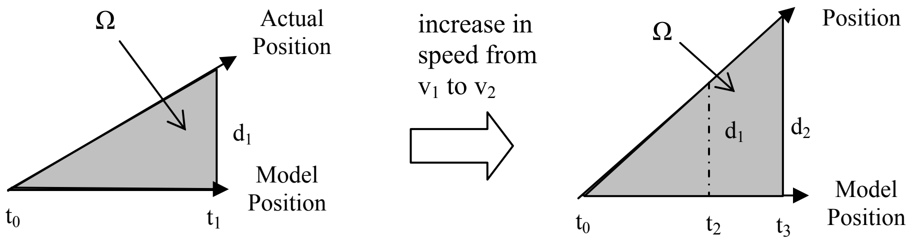


Figure 2. Speed impacts on the resultant spatial inconsistency when a time-space error metric is employed.

If speed is increased from v_1 to v_2, the time taken to reach the spatial inconsistency, d_1, clearly decreases to t_2. As the time-space error metric takes the duration of the inconsistency into account, the time-space threshold is now reached at a later time, t_3, at which point a spatial inconsistency of d_2 has arisen. Note, that the value of local absolute inconsistency, Ω, remains the same for both cases. Conversely, if speed is decreased, the spatial inconsistency is also decreased, while Ω once again remains the same.

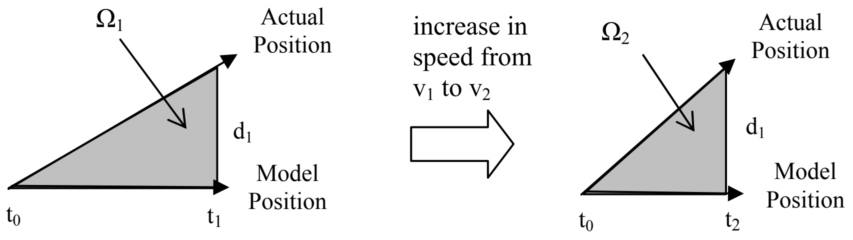


Figure 3. Speed impacts on the resultant absolute inconsistency when a spatial error metric is employed.

Consider now the scenario when a spatial threshold is employed as shown in Figure 3. When the entity speed is v_1, an update is sent at time t_1 as the spatial inconsistency, d_1, has exceeded a predefined error threshold. Local absolute inconsistency is measured as Ω_1. If the speed increases to v_2, the update is now transmitted at an earlier time of t_2. The spatial inconsistency remains the same. However, the reduced time taken to reach the spatial error threshold in this case results in a smaller absolute inconsistency measure, Ω_2. Conversely, with a decrease in speed, the time between updates increases, meaning that d_1 is not reached until a later point in time, causing an increase in local absolute inconsistency.

The impact of this analysis is that inconsistency measures change depending on the underlying entity speed when dead reckoning is employed. This means that for any comparisons made between the spatial and time-space threshold metrics, modelled entity speed must also be considered.

4. Experimentation

We now propose a scheme that uses entity speed information and average inconsistency measures when choosing suitable threshold values for use in the hybrid threshold metric. Under this approach, the average local spatial inconsistency and local absolute inconsistency arising from the use of both spatial and time-space error metrics are calculated for multiple error threshold values. This procedure is then repeated for varying entity speed values. Error thresholds that give the same average inconsistency measures are deemed to be equivalent.

The result of this process is an error threshold look-up table. Using knowledge about the virtual environment, for example entity size, a suitable spatial error threshold can be initially chosen. This is then employed, along with information about entity speed, to index into the look-up table and choose a suitable time-space error threshold.

To investigate this approach, four separate test courses were developed in the Torque game engine [9]. Several different tracks are considered in order to illustrate the effectiveness of the proposed method. A plan view of each is given in Figure 4. The games engine was extended to include full logging of player position (x, y and z coordinates). Six participants of varying age, sex and virtual environment experience were then enlisted to navigate each test course for between 4 and 8 times.

Next, a BOT, or computer controlled entity, used this behavioural data to emulate the behaviour of each participant within each environment, while using dead reckoning with either a spatial or time-space error metric. Entity speed was varied per simulation. The three speeds chosen were 25 Torque Game Units/s (TGU/s), 12.5 TGU/s and 8.33 TGU/s, referred to as full, half and third speed, respectively.

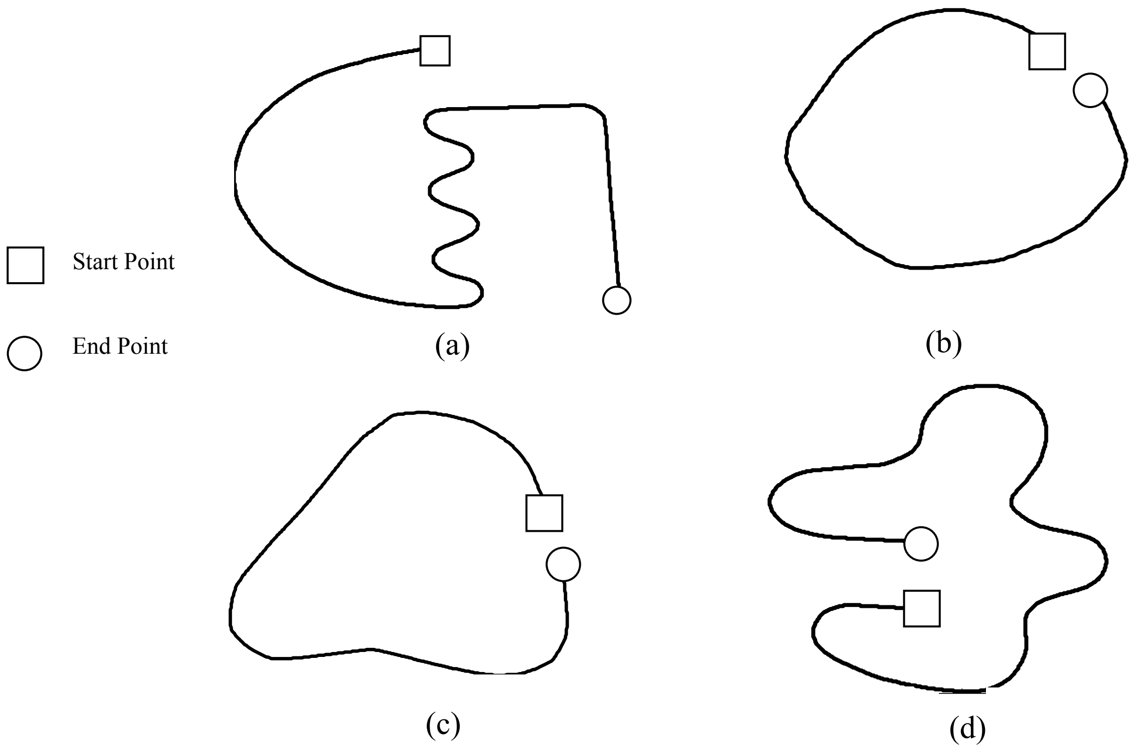


Figure 4. Plan view of (a) test course 1 (b) test course 2 (c) test course 3 (d) test course 4

During each simulation, the number of networks packets generated, the average spatial inconsistency per second, and the average absolute inconsistency per update was recorded. The latter is determined per update, as this measure is dependant on the spatial inconsistency arising during the time period since the last update was transmitted. In the next section, the data collected during these simulations is presented, and analysis of the data is given.

5. Results and Analysis

Figure 5 and 6 show the local average inconsistency measured for test course 1 and test course 3 for all user trajectories. The upper plots in both figures shows the average local spatial inconsistency per second, while the lower plots shows average local absolute inconsistency per update. Similar results were obtained for test course 2 and test course 4, but are not included here due to space restrictions.

It can be seen that, in general, the average spatial inconsistency is seen to grow linearly as the spatial error threshold increases for both example courses. The small fluctuations in the data can be attributed to the varying behaviour of different users within each environment. For example, a user that behaves erratically, moving towards and away from the dead reckoning model, will result in a slightly different error measure than a user who moves away in a constant manner from the model path. As spatial inconsistency is not affected by entity speed when a spatial threshold is employed, data is only presented for a single speed value.

Examining the average spatial inconsistency arising from the use of the time-space error threshold, the impact of speed is clearly evident. In the upper plots of both Figure 5 and Figure 6, it can be seen that as the entity speed increases, the resultant average spatial inconsistency also increases. This is consistent with our analysis in Section 3.

In the lower plots of average local absolute inconsistency, the linear relationship between the average local absolute inconsistency value and the time-space error threshold is evident. Intuitively, it would be expected that the average local absolute inconsistency value would be exactly equal to the error threshold. However this is not the case, as an extra network packet is generated when the entity enters the environment.

Therefore, an extra packet is always used in calculating the average value, resulting in a slightly lower average value. The impact of speed is again evident in both these figures, with respect to the average absolute inconsistency obtained when using a spatial threshold metric. Lower speeds result in higher local absolute inconsistency than higher speeds. Again, this supports our analysis in Section 3, and highlights the need to consider entity speed when comparing the two threshold metrics.

5.1 Locating comparable threshold values

Using the average inconsistency measures shown in both Figures 5 and 6, comparable spatial and time-space error thresholds can be chosen. Consider, for example, the case where a space error threshold of 8 TGU is chosen as a suitable error threshold for test course 3. From the upper plot in Figure 6, it can be seen that the average spatial inconsistency per second in this case is 2.9 game units. Extending a line across the plot at this point, the time-space error thresholds that yield the equivalent average spatial inconsistency per second are 4.4 TGU seconds, 9.3 TGU seconds and 14.8 TGU seconds for full, half and third speed respectively.

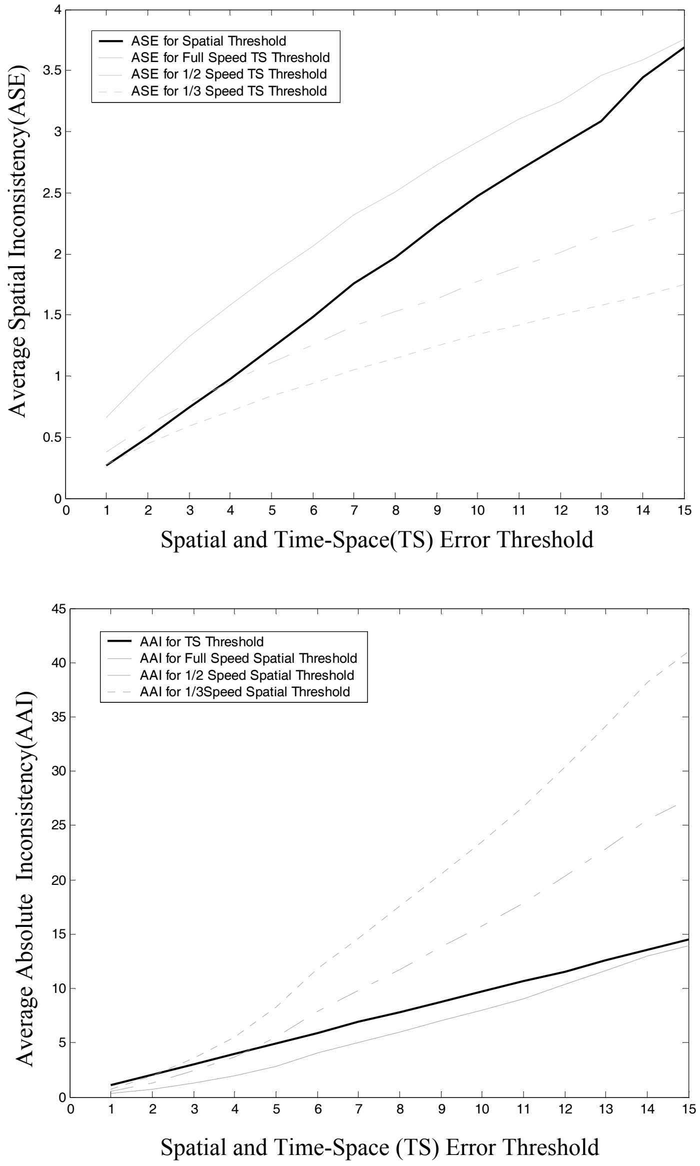


Figure 5. The upper plot shows the average spatial inconsistency for test course 1. The lower plot shows the average absolute inconsistency for the same course.

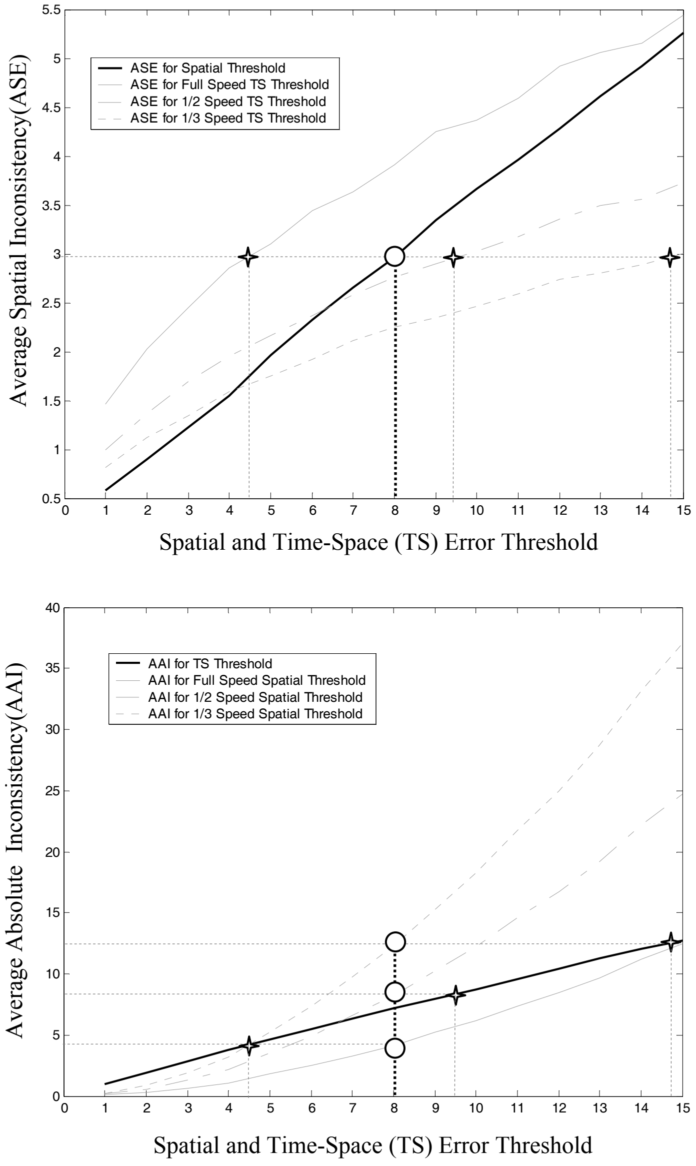


Figure 6. The upper plot shows the average spatial inconsistency for test course 3. The lower plot shows the average absolute inconsistency for the same course. Equivalent threshold values are chosen based on equivalent average inconsistency measures.

To verify these values, the chosen example time-space error thresholds were located on the lower plot, and the average absolute inconsistency at each point was found. The spatial threshold values that yield the same average absolute inconsistency were then established for full, half and third speed values. It is clearly evident from the lower plot in Figure 6 that the equivalent spatial threshold in each of the three cases is 8 TGU, which is the original example spatial error threshold chosen in the previous paragraph.

Test Number	**Spatial Threshold**	**Course 1**			**Course 2**			**Course 3**			**Course 4**		
		Speed			**Speed**			**Speed**			**Speed**		
		Full	**Half**	**Third**	**Full**	**Half**	**Third**	**Full**	**Half**	**Third**	**Full**	**Half**	**Third**
Test 1	4	1.9	4.2	6.3	1.1	2.6	3.9	1.1	2.6	4.3	2.8	5.3	8.4
Test 2	6	3.6	7.5	11.8	2.8	7.3	11.6	2.6	5.7	9.2	5.1	10.8	15.5
Test 3	8	5.7	11.8	19	4.4	9.3	14.8	4	8.5	14.1	8	17.2	25.5

Table 1. Spatial thresholds and their equivalent time-space threshold.

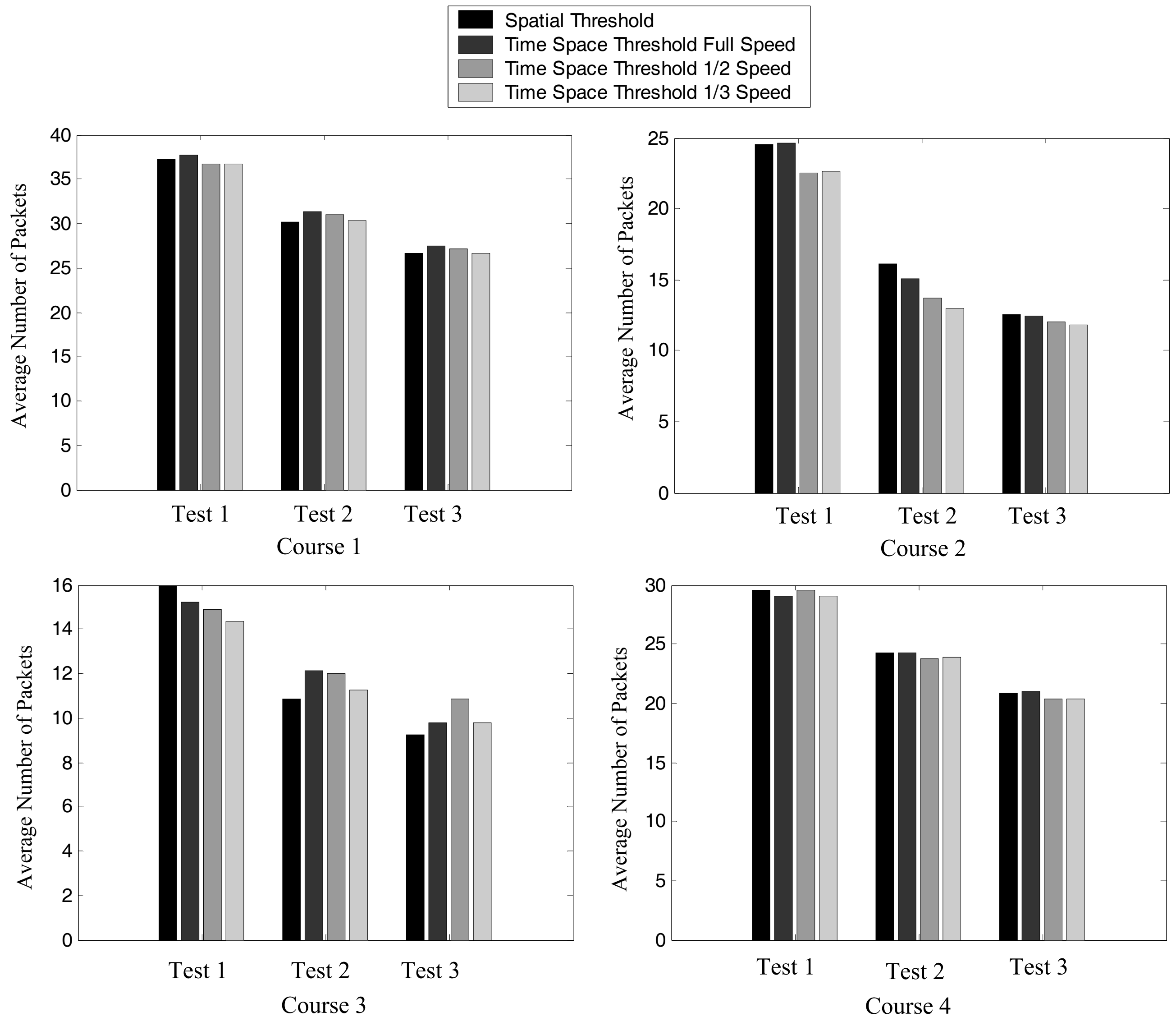


Figure 7. Average network packets generated for comparable spatial and time-space thresholds for the four test course.

5.2 The error threshold look-up table

The comparison process outlined in Section 5.1 is then repeated for varying spatial threshold values and varying speed values for each course. The result is an error threshold look-up table. Using this table, a virtual environment designer can first choose an appropriate spatial threshold. Depending on the entity speed a suitable time-space error threshold can then be chosen. An error threshold look-up table for the four test courses discussed in this work is shown in Table 1. Here, suitable time-space error threshold values are chosen for spatial error threshold of 4, 6 and 8 TGU.

The data presented in Table 1 was then employed in a series of dead reckoning simulations, in order to investigate the performance of the comparable thresholds from the point of view of network packet generation. The results are presented in Figure 7. In each case, the average number of packets recorded for all user trajectories within each environment is shown. Each test number corresponds to a test number in Table 1. So, for example, in course 1, for full speed, a time-space error threshold of 1.9 TGU seconds is chosen for a spatial threshold of 4 TGU, and for half speed, a time-space error threshold of 4.2 TGU seconds is used.

In all cases in Figure 7, it is interesting to note that for equivalent spatial and time-space thresholds, the number of packets generated, on average, is approximately equal. Although this seems unusual at first, it makes sense upon closer inspection of the data, as the average inconsistency that arises for both thresholds is equivalent.

Initially, this result may appear insignificant, as it provides little guidance on choosing one threshold metric over another, when they both give the same number of packets and inconsistency on average. However, as detailed in the next section, it becomes important for choosing suitable error thresholds for use in the hybrid threshold scheme.

5.3 Hybrid Threshold Scheme

In previous work, time-space thresholds for use in the hybrid threshold scheme were arbitrarily set equal to the spatial error threshold. However, as shown in the previous section, this is not necessarily the best approach, as the same error threshold values usually do not yield the same average inconsistency. We will now examine the difference between using the old approach and the novel approach proposed in the previous section.

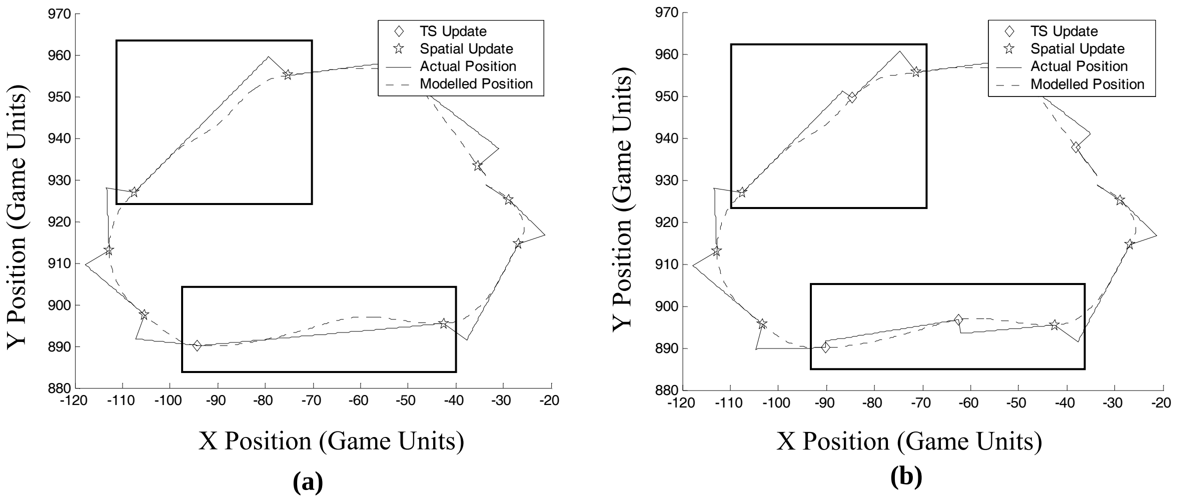


Figure 8 (a) Packets generated when numerically equivalent spatial and time-space thresholds are employed. **(b)** Packets generated for the hybrid threshold using comparable error thresholds from our proposed approach.

Figure 8(a) shows a typical user trajectory and modelled path when a numerically equal time-space and spatial error threshold of 6 is employed for test course 3. Figure 8(b) shows the same user trajectory. In this case, however, a time-space threshold of 5.6 TGU seconds is used, based on the error threshold look-up table shown in Table 1. Speed is set to the full value of 25 TGU/s in both cases.

Comparing both plots, the benefits of our proposed technique is evident. In the two regions highlighted in Figure 8(a), it can be seen that a spatial inconsistency has been allowed to continue for an extended duration. Even though the hybrid threshold is being employed, this situation is not identified, as the time-space threshold has been chosen arbitrarily, and is not suitable for the test course or entity speed. The same two regions are again highlighted in Figure 8(b). As a suitable threshold has now been chosen, the extended absolute inconsistency evident in Figure 8(a) is quickly identified, and an update packet is transmitted to rectify the inconsistency.

Further analysis are outlined in Table 2. Here, the average spatial inconsistency per second, the average absolute inconsistency per update, and the number of update packets are shown for the trajectory featured in Figure 8, but for multiple spatial threshold values. The case where a numerically equivalent threshold is chosen is shown, along with the case where the error threshold look up table in Table 1 is used.

	Avg. Spatial Inconsistency	**Avg. Absolute Inconsistency**	**Number of updates**
Spatial Threshold =4 Time- Space Threshold = 4	1.7880 TGU	1.2315 TGU seconds	19
Spatial Threshold =4 Time- Space Threshold = 1.1	1.4680 TGU	0.9460 TGU seconds	20
Spatial Threshold =6 Time- Space Threshold = 6	2.4603 TGU	2.5371 TGU seconds	13
Spatial Threshold =6 Time- Space Threshold = 2.6	2.1974 TGU	1.9497 TGU seconds	15
Spatial Threshold =8 Time- Space Threshold = 8	3.2430 TGU	3.7253 TGU seconds	12
Spatial Threshold =8 Time- Space Threshold = 4	2.8950 TGU	3.0672 TGU seconds	13

Table 2. Table of the average spatial and absolute inconsistency arising from using the hybrid threshold metric with differing threshold values for course 3. The number of network updates is also shown in each case.

Examining the data in Table 2, it can be seen that as the spatial error threshold value increases, the number of network packets decreases, with the effect of increasing both average inconsistency measures. Compared with the numerical equivalent threshold values, the values chosen using the error threshold look-up table results in both a lower average spatial inconsistency and lower average absolute inconsistency. However, it should be noted that the number of packets in each case increases. Clearly, our proposed method results in better consistency, but at a slight increase in packets. Moreover, the underlying threshold values now have some relevant meaning. A key issue now arises. On one hand, an increase in packets can improve consistency by reducing modelling error. However, on the other hand, an increase in packets can lower consistency by increasing network traffic. Clearly a compromise is required between the number of packets transmitted, and the overall consistency obtained. This is the basis of future research.

6. Conclusions and Future Work

In this paper, measures of average inconsistency were investigated in order to obtain suitable thresholds for use in the hybrid threshold scheme.

It was demonstrated that entity speed impacts directly on the measured inconsistency when both a spatial and time-space error thresholds are employed. An increase in speed results in an increase in spatial inconsistency when a time-space threshold is employed while a decrease in speed results in a decrease in spatial inconsistency. The scenario is reversed when a spatial threshold is employed, with an increase in speed causing an increase in absolute inconsistency, and vice versa.

This fact means that any comparisons between spatial and time-space error metrics need to also take entity speed into account. With this in mind, a novel scheme was proposed, that allows comparison of threshold metrics. Under this approach, a spatial error threshold is first chosen based on characteristics of the virtual environment. Combining this information with knowledge about modelled entity speed and course information, a time-space error threshold with equivalent average spatial inconsistency is then chosen. This process is repeated for varying entity speed and spatial threshold value. The result is an error-threshold look-up table. A virtual environment designer can then pick a suitable time-space error threshold, based on a spatial error threshold value and the entity speed.

It was shown how equivalent error thresholds chosen using this process generated an equal number of network packets on average. These error thresholds were then employed in a hybrid threshold scheme, and compared to the previous approach of simply using numerically equivalent values. It was demonstrated that this approach allows a tighter consistency to be maintained within a Distributed Interactive Application, as the two error threshold values are better related. However, there is also a resulting increase in the number of update packets required. Future work will investigate the tradeoff between the number of update packets transmitted across the network, and the overall value of consistency obtained.

7. Acknowledgements

This work is supported by Science Foundation Ireland and Enterprise Ireland under grant number IRCSET/SC/04/CS0289.

8. References

[1] Delaney, D., T. Ward and S. McLoone. "On consistency and network latency in distributed interactive applications: A survey - Part I." *Presence: Teleoperators and Virtual Environments* 15(2),April 2006
[2] IEEE. *IEEE Standard for Distributed Interactive Simulation - Application Protocols IEEE Std 1278.1-1995* IEEE. (1995)
[3] Delaney, D., T. Ward and S. Mc Loone "Reducing Update Packets in Distributed Interactive Applications using a Hybrid Model". *16th International Conference on Parallel and Distributed Computing Systems, August 13-15*, Reno, USA,pp. 417-422,2003
[4] Roberts, D., D. Marshall, S. McLoone, D. Delaney, T. Ward and R. Aspin "Exploring the use of local inconsistency measures as thresholds for dead reckoning update packet generation". *Distributed Simulation and Real Time Applications*, Montreal, Canada,pp. 195 - 202,2005
[5] Zhou, S., W. Cai, F. B. S. Lee and S. J. Turner. "Time-space Consistency in Large Scale Distributed Virtual Environment." *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 14(1):pp. 31 - 47,2004
[6] Lee, B.-S., W. Cai, S. J. Turner and L. Chen. "Adaptive Dead Reckoning algorithms for distributed Interactive Simulation." *International Journal of Simulation* 1(1-2):pp. 21-34,1999
[7] Cai, W., F. B. S. Lee and L. Chen "An Auto-adaptive Dead Reckoning Algorithm for Distributed Interactive Simulation". *Proceedings of the 13th Workshop on Parallel and Distributed Simulation, May 1-4*, Atlanta, Georgia,pp. 82-89,1999
[8] Duncan, T. P. and D. Gracanin "Pre-reckoning algorithm for distributed virtual environments". *Winter Simulation Conference*, New Orleans, Louisiana,pp. 1086-1093,2003
[9] Marshall, D., A. McCoy, D. Delaney, S. McLoone and T. Ward "A Realistic Distributed Interactive Application Testbed for Static and Dynamic Entity State Data Acquisition". *IEE Irish Signals and Systems Conference*, Belfast, Ireland,pp. 83-88.,2004