Evaluation of the Motion-Aware Adaptive Dead Reckoning Technique Under Different Network Latencies Applied in Multiplayer Games

Luis Fernando Kawabata de Almeida, Alan Salvany Felinto
Computer Science Department
Universidade Estadual de Londrina, UEL
Londrina, Brazil
kawabata_pc@hotmail.com, alan@uel.br

***Abstract*—Multiplayer games hold virtual worlds which connect dozens of players in the same session, in which these human players experience network latency in networked interactions negatively. A number of predictive techniques were developed to deliver the best experience for the players. With the growing numbers of players and the complexity of the worlds, the limitations of such techniques, like network delay, consistency, responsivity and bandwidth cost, become evident. The main technique used nowadays is called Dead Reckoning (DR) and was firstly presented decades ago. Based on it a variety of authors proposed improvements to the prediction method, culminating on Kharitonov's proposal of the Motion-Aware Adaptive Dead Reckoning (MAADR) technique. The author's evaluation did not consider latency, an important factor that affects the consistency (i.e. veracity) of the information and the Quality of Experience for the players. The proposal of this paper is to evaluate both the MAADR and DR prediction techniques in 4 situations ranging from simple to complex movement patterns, with 0 to 300ms of network delay. The results show that the MAADR performance is superior when compared to the classic algorithm for each situation and each different latency evaluated. It shows that the classic DR presents a large decay in precision within medium and high latencies and it is not ideal in situations with great intolerances to network delay. Because of the obtained results, it is recommended the usage of the MAADR technique when medium and high latencies are expected or when there is an intolerance for the loss of the Quality of Experience.**

***Keywords*-Dead Reckoning; DR; latency; MAADR; predictive techniques; virtual environments; video games; online games; Unity3D**

I. Introduction

Electronic games are ludic activities where the user acts and is conditioned by the rules of the virtual world, resulting in intriguing and fun experiences [1]. The games market is increasingly growing, moving 116 billion dollars worldwide in 2017, with an estimate of moving 143.5 billions in 2020 [2].

Initially, games were made with the only perspective of single users experiencing the game. With technological advances in networking, the developers gained the perspective of connecting players for multiplayer adventures. Such feature has grown to the point of the creation of the Electronic Sports (eSports) [3] category for multiplayer competitive games.

The connection among players makes the sharing of a great amount of information of the shared world plausible. To make these scenarios possible, a variety of techniques were proposed, with which the continuous growth of the shared information got their limitations evidenced. Thus, there is a necessity of more studies in these areas to diminish the consequences of these limitations. [4]

Online games are naturally derived from Distributed Virtual Environments (DVEs) [5], being usually built with a client-server architecture. Objects, or entities, in the game are called local if they belong to the player who is running the application, or remote if they do not. The purpose of the remote entities is to replicate the behaviour of their respective local entities [6].

There are two main lines of research that handle the connectivity and network bandwidth consumption factors in DVEs. The first strand identifies and addresses the quantity of information problem. The main line of researches that address this problem approach it with the idea of managing areas of interest, being widely used on DVEs.

The second strand, that manages the interesting information, is approached by a range of predictive techniques that must balance the network consumption and the consistency for the local player. The predictive techniques inside this strand prioritize either the consistency for the remote objects, or the responsivity for the local objects. Nonetheless, both must maintain good consistency on remote objects.

A classic predictive technique is called Dead Reckoning (DR). It manages to focus the responsiveness while maintaining good levels of consistency and being simple. Some examples of big commercial games that used the DR are Quake [7] and Half life [8]. The wide use of the technique made several authors propose iterations for the DR with the purpose to increase its accuracy [6].

The DR iterations have good results, but the authors do not consider latency on their evaluations. Latency is a great factor for consistency, since higher latency causes major errors on the predictive techniques. Thus, the presence network delay may cause an undesirable low Quality of Experience [9].

Studying the iterations found on the literature Kharitonov *et al* [10] proposed a technique called Motion-Aware Adaptive Dead Reckoning (MAADR). The author evaluated his technique not considering latency and obtained great results. Aggarwal [6] showed good results of his proposal comparing it with the DR technique under latencies, showing that the classic DR has a problem in such circumstances. Thus, an evaluation of the MAADR technique compared with the DR technique under latencies should be done.

The proposal of this paper is to analyze and evaluate the performance of the classic DR technique with time stamps compared with the MAADR on situations with medium and high latencies. 4 situations were created, inspired on evaluations from other authors on the literature, that test both simple and complex movements.

In Section II we report the papers found in the literature about predictive techniques, from the DR technique until the MAADR technique. The Section III details the implementation made for the proposed evaluation and its setup. The Section IV details the test situations created and the analysis of the evaluation for each situation, discussing them. Based on the results and the discussion, the Section V concludes the paper.

II. Related Work

A DVE is defined as a technology created for the interaction of several players connected to an 2D/3D virtual environment through the internet. These environments simulate immersive, complex and interactive experiences, while connecting a number of participants from different physical places [5].

DVEs provide great experiences, but have big challenges. The main proposals on the literature, that address the management of information problem, have several characteristics that influence them: Consistency, responsiveness, network delay, network bandwidth consumption, among others. The object positions are notorious for being the main variable focused by predictive techniques because of their predictability in term of physics.

The proposals on literature can be categorized as either those that prioritize consistency or responsiveness. Some genres of games have a major sensibility to one of the categories: Real-Time Strategy (RTS) Games have a major sensibility to consistency, while the First Person Shooter (FPS) games have a major sensibility to responsivity.

The consistency on this context is relative to the information present on remote entities when compared with the information on the local entities. One classic example that exposes the problem that may arise upon the presence of inconsistency is a dead player shooting and possibly killing another player. To manage distributed information in a consistent way is one of the main challenges of a DVE [11].

A few important contributions that focused the consistency must be mentioned: The Lockstep Protocol, proposed by Levine *et al* [12] ; Pipe-lined Lockstep proposed by lee *et al* [13] ; The Local-Lag technique proposed by Mauve *et al* [14] ; The Trailing State Synchronization (TSS) technique by Cronin *et al* [15] and the Progressive Slowdown technique proposed by Shen *et al* [16].

A low delay on the computing of the player actions define high responsiveness. Even though some genres of games have a necessity of one over the other, on eSports of any genre, the responsiveness is a characteristic with a great focus. The main challenge when focusing the responsivity is to maintain good levels of consistency.

The classic proposal called Dead Reckoning focus mainly on responsivity, while maintaining good levels of consistency. It is a predictive and optimistic solution, with the main purpose of decreasing the number of position update messages needed for a good consistency. Thinking on its limitations, several authors proposed iterations to make it better.

Aggarwal proposed one of the first iterations over the DR technique, the Globally Synchronized Dead Reckoning (GSDR). With his evaluation, he was the only author to evaluate and propose a technique that deals with latency. On the GSDR, the insertion of time stamps on the messages is done, along with the synchronization of the clocks of both server and client.

The DR technique implements an optimistic predictive model, with which the local entity can decrease the update rate by setting it to a fix value and still maintain a good consistency. The model is optimistic because it expects the prediction to be correct. If it fails convergence techniques are used to correct the trajectory. The use of the predictive model results in the decrease of bandwidth cost, in decrease of network delay and in network decongestion [10].

Simplistic convergence techniques can be jarring for the player and cause problems for Quality of Experience [9]. The most used technique is the interpolation convergence, which causes a smooth transition from the incorrect trajectory towards the correct trajectory. The interpolation takes a predetermined time in seconds where in the end, the remote object is performing the correct trajectory. [4]

The DR technique can be divided into two parts: The condition to when send update messages and the predictive expressions. One, the other or both are modified on the iterations from the literature. The classic model fix the update rate and has simple expressions for the prediction.

Cai **et al** [17] proposed the technique called Auto-Adaptive Dead Reckoning (ADR). The adaptiveness of the technique comes with the idea of decreasing the update rate for the remote objects that are further away from the local player's viewport. Since objects not shown to the player can have higher inconsistencies without hurting the Quality of Experience (QoE). Cai was the first to adapt the update rate of the technique. The evaluation from the author for the ADR test the performance without considering latency.

Aiming to explore the expression used on the ADR technique, Cai [17] and Lee [18] proposed different expressions to be used, focusing on abrupt movements caused by the corrections present on the ADR. Good results were found but no latency was considered.

Duncan *et al* [19] focused on the adaptability with the Prereckoning technique. He was the first to use movement information to adapt the update rate. The adaptability of the technique comes from the tracking of the acceleration to preemptively send an update message. If the local object starts to accelerate, the remote object will predict wrongly, thus a preemptive message should be sent. The evaluation resulted in a good increase of the precision but with no considerations with latency.

The balance between the two focused characteristics is essential, where low update rates lead to great error on the prediction model and high update rates cause inefficient use of network bandwidth, even though the higher precision [10].

Kharitonov [10] describes an ideal Dead Reckoning algorithm as one that balance the update rate without consuming too much network bandwidth and that do not lead the prediction model to have great errors. Thus, maintaining good levels of consistency, responsiveness, and precision. Based on this definition, the Kharitonov proposed a technique that modifies both the update condition and the predictive expressions.

Kharitonov called his algorithm Motion-Aware Adaptive Dead Reckoning (MAADR). It is motion-aware for having a complex predictive model, that uses velocity, acceleration and circular motion. And it is adaptive for using the complex model not only on the remote object, but also on the local object, where the preemptive sending of update messages is applied when changes on movement patterns are identified. The update messages are sent either preemptively as described before or if the time limit threshold is met. The complex movement model then is used on the remote object to predict the correct trajectory to traverse based on the information received. This technique increases significantly the performance of the predictions but entails a higher cost of network bandwidth and processing for the expressions. If the current trajectory being performed is found to be wrong on the remote object, a correction routine is triggered, in which a convergence technique is performed.

Kharitonov concludes showing that the MAADR technique has a great improvement over the DR technique. He also concludes that the update rate of 2Hz is the ideal value to maintain consistency without a great cost for network bandwidth. In his evaluation no latency was present, which could further argument the use of the MAADR in the place of the classic DR.

III. Implementation

To best implement the proposed evaluation, the Unity3D engine [20] was chosen for being a professional tool used by a large number of developers to make games. UNet [21] is a networking framework present inside Unity that has the capability of delaying messages received by the application. An implementation of the MAADR technique [10] and of the classic DR, both with time stamps on the update messages, were submitted to 4 situations to best evaluate their performance under medium and high latencies. On each situation the error of both techniques was measured with the Euclidean distance between the current position on the remote object and the correct position from the local object every 0.02 seconds.

A. Networking

Using the network framework UNet [21], a server-client architecture was created, in which the clients must send the actions of the players to the server and the server must then calculate the result and validate them. The server must then send all the updated information back to the clients, keeping the highest possible consistency.

The ideal DR technique must balance consistency and network consumption [10], where a high update rate causes a high network cost, even though it increases the precision of the technique. The evaluation proposed on this paper aims to measure the ability of the techniques in maintaining the precisions of the predictions within higher latencies. The main variables varied are the latency and the motion patterns. Since the condition for sending update messages in neither of the techniques consider the present latency, the network consumption is maintained constant for each situation and algorithm pair along all latencies tested. Therefore, even though the network consumption is

an important variable to balance on DR techniques, the evaluation of this paper is focused only on the precision of the techniques in the presence of medium and high latencies.

B. MAADR and DR implementations

The DR technique is defined by the message sending condition and the prediction expressions. The first is defined by a fixed time limit threshold of 2Hz (0.5 seconds), which is the update threshold found by Kharitonov that best balance network consumption and precision. The second is a simple predictive model with a unique linear constant velocity expression. The MAADR technique is firstly defined the presence of a change on the motion patterns, which results on a preemptive message sending, or the reaching of the time limit threshold (2Hz, 0.5 seconds). And is secondly defined by a complex prediction model with 3 motion patterns: Stationary movement; Constant velocity and Accelerated velocity movement. For better control in the evaluation this model is divided into 2 modules: Linear and Angular movement kernels, in which both have the same expressions but with the change from linear to angular variables. The prediction with 2 modules is the same as with 1 module, thus, the use of 2 modules do not interfere with the prediction nor the results of the experiment.

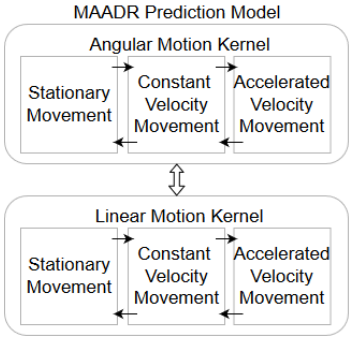

Figure 1. Motion patterns implemented in two modules, adapted from Kharitonov [10] for easier control.

The figure 1 shows the two kernels used in the implementation for the evaluation of the MAADR technique.

For each motion pattern, there is an expression related to it. The equations 1, 2, and 3 shows respectively the expressions for the following motions patterns: Stationary, Constant velocity movement and Accelerated movement.

$$S_f = S_i \tag{1}$$

Where S_f is the final position and S_i is the initial position.

$$S = S_0 + \Delta T \times V \tag{2}$$

Where S is the final position, S_0 is the initial position, ΔT variation of time and V is the constant velocity.

$$S = S_0 + \Delta T \times V_0 + \frac{A \times \Delta T^2}{2} \tag{3}$$

Where S is the final position, S_0 is the initial position, ΔT variation of time, V_0 is the initial velocity and A is the constant acceleration.

In the moment where the remote object receives an update message and verifies that an error occurred the correction routine starts to interpolate the wrong trajectory with the correct predicted trajectory. The equations 4, 5, and 6 are used in the interpolation convergence routine.

$$S_j = S_j 0 + \Delta T \times V_0 + \frac{A \times \Delta T^2}{2} \tag{4}$$

Where S_j is the correct final position of the local object, $S_j 0$ is the correct initial position of the local object, ΔT is the variation of time, V_0 is the initial velocity and A is the acceleration

$$S_f = S_f 0 + \Delta T \times V_0 + \frac{A \times \Delta T^2}{2} \tag{5}$$

Where S_f is the wrong final position of the remote object, $S_f 0$ is the wrong initial position of the remote object, ΔT is the variation of time, V_0 is the initial velocity and A is the acceleration.

$$I_p = S_f + (S_j - S_f) \times T_i \tag{6}$$

Where I_p is the final interpolated position, S_f and S_j are the final wrong and correct positions respectively for the equations 4 and 5 and T_i represents the percentage of the interpolation, it varies from 0 to 1.

IV. Analysis and Results

For the evaluation of the addressed techniques, 4 situations were proposed. The situations were designed to evaluate both of the techniques using a range from simple to complex motion patterns. To perform the situations, a system implemented to read inputs from a file and move the local characters was used in order to maintain consistency and precision for the evaluation.

The first situation is a simple straight line with a small acceleration period; The second situation evaluates the presence of both a linear acceleration and an angular movement sequentially, resulting in a circular motion; The third situation stresses both algorithms with a constant linear acceleration in parallel with

abrupt changes in the angular velocity, resulting in a ZigZag movement and the fourth situation proposes a random movement, resulting in a case with a greater number of abrupt changes, closer to real cases.

The update rate is set to 2Hz, as described before. The interval between 0 and 300 simulated latency, in milliseconds, was chosen to address both the minimum possible and a high value (300ms) of simulated latency. In such high latency the Quality of Experience gets precarious and the loss of players is imminent [9]. The following values were tested on each situation for each technique: 0, 50, 100, 150, 200, 250 and 300 milliseconds. The fixed interval of 50 milliseconds allows for a deep analyzes of the consequences of the presence of latency, without an overflow of redundant information.

A. Situation 1: Line with acceleration

The first situation is the simplest but yet quite common on several genres of games. It is composed of an acceleration period, followed by a constant velocity period. This movement pattern can be found not only on racing games, where the player's vehicle must move on a straight line, with or without acceleration. But also in game genres such as FPS, where the player must move in a line to a location where the battle will take place, or Real Strategy Games (RTS) where the player sends his units to a position that only needs the movement of a straight line. The main proposition of this situation is to evaluate not only the ability of both the techniques to be precise on this simple case, but also to measure the accuracy decreasing magnitude upon the presence of delay in such a case. This situation was inspired by the first linear motion case used in the evaluation by Aggarwal [6].

Figure 2. Situation 1: Straight line movement.

The figure 2 illustrates the trajectory made by a remote object with no simulated latency. The movement is in the vertical orientation, from the bottom to the top. There is a total of 2 marks on the figure that represent the changes of the motion patterns, either on the angular or on the linear motion kernels.

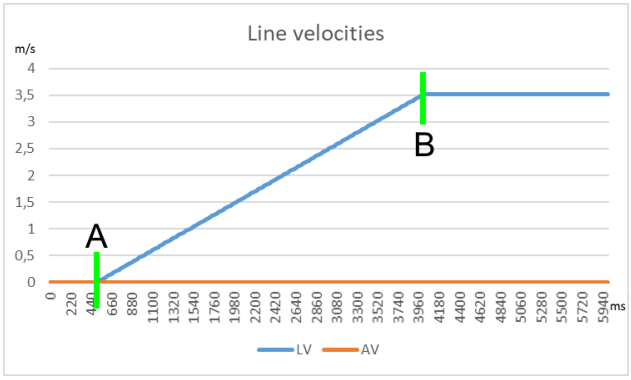

Figure 3. Speed profiles for Situation 1: Line movement

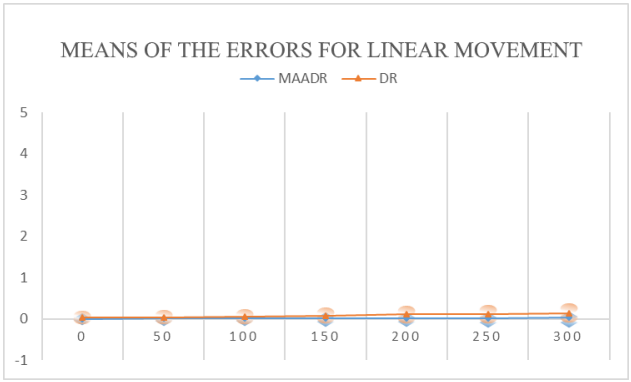

Figure 4. Situation 1: Means of the errors across latencies.

The A mark represent the beginning of the period of acceleration and the B mark represent the end of it.

The figure 3 shows the velocities graph for the Situation 1. The same marks that are present on the figure 2 are present on the figure 3 and with the same meanings. The angular velocity is kept at 0 radians per seconds along all the situation.

The figure 4 illustrates a chart with the means of the errors for each technique across each of the latencies evaluated shown on the X-Axis. The diamond points on the line represent the means of the errors of the

Table I. Table of the average of errors and standard deviations in the situation 1, with 2Hz of update threshold along the latencies evaluated.

Simulated Latency	MAADR	DR
0ms	**3.65E-3 ± 4.62E-3**	3.71E-2 ± 3.34E-2
50ms	**7.94E-3 ± 9.50E-3**	4.47E-2 ± 4.62E-2
100ms	**1.21E-2 ± 1.13E-2**	5.90E-2 ± 5.31E-2
150ms	**1.48E-2 ± 1.74E-2**	8.63E-2 ± 7.34E-2
200ms	**1.91E-2 ± 1.86E-2**	1.11E-1 ± 8.85E-2
250ms	**2.48E-2 ± 1.92E-2**	1.24E-1 ± 1.00E-1
300ms	**3.21E-2 ± 2.25E-2**	1.48E-1 ± 1.18E-1

MAADR technique with the diamond points above and bellow of these points illustrating the standard deviation for the MAADR. The triangle points represent the means of the errors for the DR technique while the oval points represent the standard deviations for the DR. The Table I iterates on the analysis of the results presenting the values of the means of the errors and standard deviations for each technique and each latency, with the results of the technique with lower errors and lower standard deviations as bolded text. Both visualizations of the result show that the MAADR have greater precision both in lower and higher latencies. Although the situation 1 represents a simple complexity of movement by players, it is an movement pattern that has a great probability of happening on games.

B. Situation 2: Circular movement

The situation 2 is supposed to evaluate the presence of both linear and angular velocity sequentially, representing a circular motion. It starts with a period with acceleration, followed by a period with constant linear and angular velocity. The angular velocity results in 2 full circles being performed by the character that then moves with a linear velocity towards the end of the situation. The main genre of games that covers this movement pattern is the racing genre, where on several curves this pattern will be present. On other genres this pattern may appear, but less often. This situation was prepared to evaluate both techniques under closed circular paths, such as the first scenario tested by Kharitonov [10], and the influence of latency over this type of behaviour.

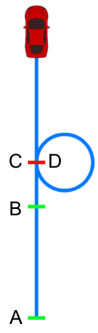

Figure 5. Situation 2: Circular movement.

The figure 5 illustrates the trajectory made by the remote object. The Trajectory is orientated from the bottom to the top of the figure. There is a total of 4 marks present, which represent changes in the motion patterns either from the linear motion kernel or the angular motion kernel, similarly to the previous

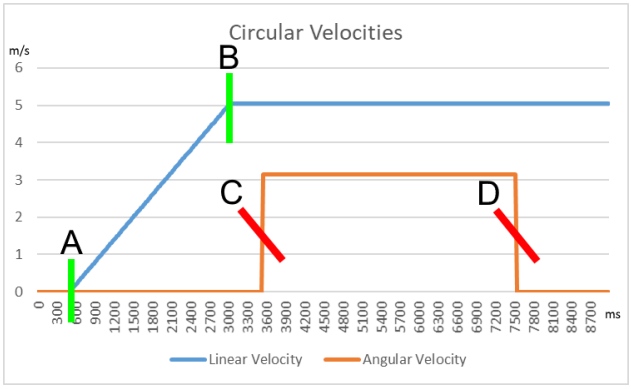

Figure 6. Speed profiles for Situation 2: Circular Movement.

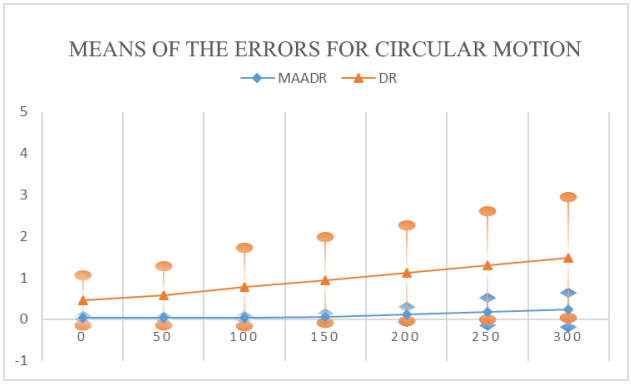

Figure 7. Situation 2: Means of the errors across latencies.

situation. The A mark represents a presence of linear acceleration; The B mark represents an absent linear acceleration; The C mark represents the beginning of the circular motion, where a constant angular velocity, that is represented in radians per second, is set and the D mark represents the end of the circular motion, where the angular velocity is set to 0.

The figure 6 illustrates how both the linear and the angular velocities magnitudes behave along the situation. There are 4 marks on the graph that represent the same marks on the figure 5. With the figure 6, all the changes in the motion patterns can be seen with more clarity.

Table II. Table of means of errors and standard deviations on the situation 2, with an update rate of 2Hz.

Simulated Latency	MAADR	DR
0ms	**3.85E-2 ± 3.02E-2**	4.56E-1 ± 6.02E-1
50ms	**2.75E-2 ± 3.14E-2**	5.72E-1 ± 7.10E-1
100ms	**3.47E-2 ± 4.24E-2**	7.79E-1 ± 9.39E-1
150ms	**5.17E-2 ± 7.76E-2**	9.49E-1 ± 1.04
200ms	**1.14E-1 ± 1.85E-1**	1.12 ± 1.16
250ms	**1.84E-1 ± 3.26E-1**	1.31 ± 1.31
300ms	**2.28E-1 ± 4.15E-1**	1.49 ± 1.45

On the figure 7 a presentation of the means of the errors for both techniques under the shown latencies in the X-Axis can be seen. The diamond points on the line present the means of the errors for the MAADR technique, with the diamond points bellow and above these points representing the standard deviations for the MAADR. The triangle points on the line represent the means of errors for the DR technique, with the oval points indicating the standard deviations for the DR. The results from the figure 7 show a great increase on the error of the DR technique upon higher latencies, while the MAADR algorithm maintains a more consistent and lower means of errors.

The lower precision of the DR compared to the MAADR on circular movements was previously shown by Kharitonov [10] when the proposal of the MAADR technique was made. Therefore the worse result of the DR technique is not unprecedented, but the great proportion in which the DR has a greater error compared to the MAADR shows the importance of the previously not made evaluation.

The Table II shows the means of the errors for each technique and each latency tested, agregated with the standard deviation of each test and with bolded texts for the best technique for the given latency. The results on the table corroborate with the conclusion discussed above on the figure analysis, where the greater the latency, greater is the difference between the error of the two techniques.

C. *Situation 3: ZigZag movement*

The situation 3 was created to stress the techniques with a mixture of the complex elements present on the previous situations. A linear acceleration is present within all the scope of the situation and the presence of angular velocities that abruptly changes from one value to its opposite, making an ZigZag movement. The abruptly changing pattern that can be seen in this situation can be found on several games genres such as racing games, appearing where the track demands an zigzag movement. Other genres have this movement pattern, although with less defined intervals between the abrupt changes in motion. On FPS games the player may choose to do this pattern of movement when trying to dodge attacks from other players, for an example. This situation was inspired by the second and third scenarios created by Kharitonov [10], where there is the evaluation of both techniques under abrupt changes of direction, and how they adapt to constantly changing motion patterns.

The figure 8 shows the trajectory made by the remote object for the situation 5. The trajectory is orientated from bottom to top, with a ZigZag movement to the right. There are 10 marks that represent the changes in

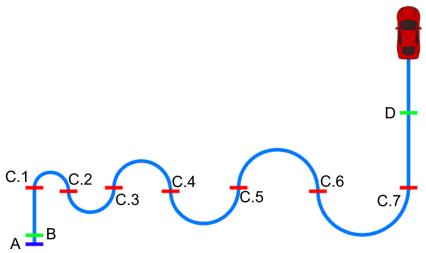

Figure 8. Situation 3: Zigzag movement

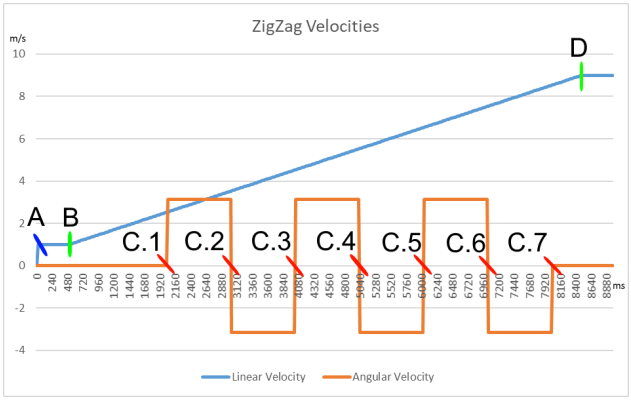

Figure 9. Speed profiles for Situation 3: Zigzag Movement.

motion patterns. On the A mark, the linear velocity is set to a small constant value; On the B mark, the linear acceleration is set to a constant value; The C.1 mark represents the beginning of the ZigZag movement with a small constant linear acceleration. The marks C.2 through C.6 alternates the signal of the angular velocity being set, making the object zigzag through the situation; On the mark C.7 the angular velocity is set to 0 and on the mark D the linear acceleration is set to 0.

The results of the variation of the linear and angular velocities can be seen on the figure 9, in which a graph is shown with the values for the linear and angular velocities throughout the situation. The same marks present on the figure 8 are also present on the figure 9.

Table III. Table with the means of the errors and standard deviations for the situation 3, with an update rate of 2Hz.

Simulated Latency	MAADR	DR
0ms	**5.91E-2 ± 5.63E-2**	6.02E-1 ± 6.73E-1
50ms	**7.89E-2 ± 9.95E-2**	7.57E-1 ± 7.93E-1
100ms	**1.05E-1 ± 1.67E-1**	9.95E-1 ± 9.61E-1
150ms	**3.40E-1 ± 4.99E-1**	1.25 ± 1.11
200ms	**7.09E-1 ± 7.98E-1**	1.49 ± 1.25
250ms	**9.24E-1 ± 1.01**	1.77 ± 1.44
300ms	**1.23 ± 1.23**	2.03 ± 1.59

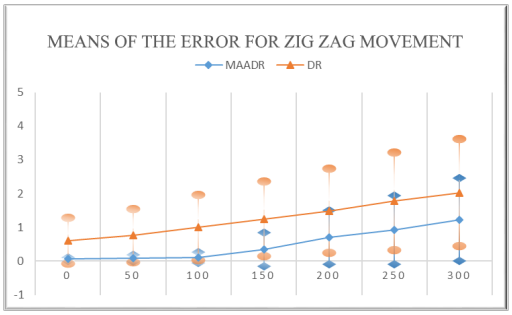

Figure 10. Situation 3: Means of errors chart across latencies.

The figure 10 shows the means of the errors for each technique, MAADR and DR, along the latency X Axis for the situation 3. The diamond points on the line illustrate the means of the errors for the MAADR and the diamond points bellow and above these points indicate the standard deviations. The triangle points indicate the means of errors for the DR technique while the oval points illustrate the standard deviation for each mean of error of the DR algorithm. For this situation, the idea of not only having linear acceleration and circular motion is expanded to create abrupt changed in the magnitude of the angular velocity, making a ZigZag movement and thus, a complex movement both for the DR and the MAADR algorithms.

The complexity of the situation reflects the results, in which even the MAADR suffers in precision if the lantecy is high. Nonetheless, the precision of the DR technique has a higher magnitude for its increasing means of error and standard deviations, making the MAADR the better of the two for this complex case.

The Table III shows, with the same format as previous tables, the means of the errors and the standard deviations for both techniques under all evaluated latencies, with bolded text for the best algorithm for the evaluated latency. The results on the table further corroborate with the conclusion thus far.

D. Situation 4: Random movement

The final situation 4, inspired by the evaluations from Aggarwal [6], implements random movements in order to stress both techniques and be closer to real situations. The randomness of this situation was achieved with the use of the pseudo random number generator present on the Unity API. The generator was applied to generate a series of inputs, where the parameters are set to be able to generate a linear velocity from 0 to 5 m/s, 0 to 2 m/s^2 of linear acceleration, -4.18 to 4.18 rad/s of angular velocity, -1.57 to 1.57 rad/s^2 of angular acceleration and a time between inputs of 0.1 seconds to 1 second. The result of this random generation is the presence of several abrupt changes in the motion pattern, causing the situation to be more noisy. Because of the parameters used, the main genre of this situation is the racing genre, although it also can be found on FPSs because of the movement of players that constantly change with abrupt events.

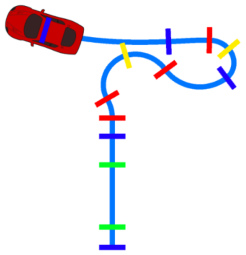

Figure 11. Situation 4: Random movement

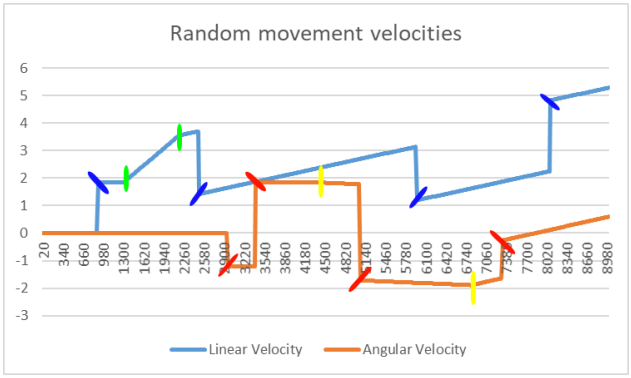

Figure 12. Speed profiles for Situation 4: Random Movement.

The figure 11 shows the trajectory made by the remote object for the situation 4. The trajectory is orientated from bottom to top, with a Random movement. There are 14 marks that represent the changes in motion patterns. Each mark represent an abrupt change caused by the random input randomly generated.

The speed profiles for this situation can be seen on the figure 12, in which a graph is shown with the values for the linear and angular velocities throughout the situation. The same marks present on the figure 11 are also present on the figure 12 to better expose the implications of the abrupt changes on the actual velocities.

The figure 13 shows the means of the errors for each technique, MAADR and DR, along the latency X Axis for the situation 4. The diamond points on the line represent the means of the errors and the diamond points above and bellow these points illustrate the standard deviations for each mean of error of the MAADR technique. The triangle points represent the

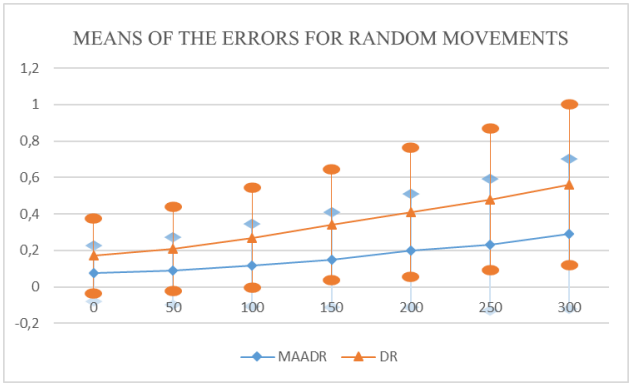

Figure 13. Situation 4: Means of errors chart across latencies.

Table IV. Table with the means of the errors and standard deviations for the situation 4, with an update rate of 2Hz.

Simulated Latency	MAADR	DR
0ms	**7.43E-2 ± 1.54E-1**	1.70E-1 ± 2.05E-1
50ms	**8.83E-2 ± 1.85E-1**	2.10E-1 ± 2.31E-1
100ms	**1.20E-1 ± 2.29E-1**	2.69E-1 ± 2.75E-1
150ms	**1.50E-1 ± 2.62E-1**	3.40E-1 ± 3.03E-1
200ms	**1.99E-1 ± 3.11E-1**	4.10E-1 ± 3.53E-1
250ms	**2.31E-1 ± 3.60E-1**	4.80E-1 ± 3.89E-1
300ms	**2.91E-1 ± 4.11E-1**	5.61E-1 ± 4.41E-1

means of errors while the oval points represent the standard deviations for the DR technique. On this situation the exploration of a random situation with abrupt changes on the movement pattern was given, resulting in not only a complex situation but also an evaluation closer to real cases.

The abrupt characteristic of this situation is present on the final results. With a general view, the MAADR presented a better overall precision than the DR technique, as expected. The tendency of a higher proportion of the error on the side of the DR can also be seen. The presence of a great number of abrupt changes affected the peaks in errors not only of the DR, but also the MAADR, making the standard deviations of the MAADR much closer to the DR when compared to other situations.

The Table IV exposes the means of the errors and the standard deviations for the techniques under the evaluated latencies. The bold texts represent the best algorithm for the evaluated latency. The greater magnitude of the error present on the DR technique can cause a loss on the QoE proportional to it, making the MAADR more suited to maintain a good QoE under noisy situations.

E. Discussion

Each situation was proposed with the intent to test the simple and the complex movements that can be made by the player, inspired by real movement patterns and inspired by the evaluations of key authors from literature. In all situations, the MAADR kept a better precision than the DR technique. This was expected since on the Kharitonovs evaluation [10] the conclusion was exactly the same. The key evaluation and results from this paper are the behaviour from both techniques under medium and high latencies. The results show that both techniques lose precision as latency gets higher, but the DR technique loses precision with a much higher magnitude than the MAADR and a high proportion.

Analyzing the standard deviation for each set of errors in each situation it is noticeable that the DR reaches much higher peaks of errors on the remote model. The high peaks of error, summed with the higher errors overall and the decrease of the precision under medium and high latencies, makes the usage of the DR not advisable if medium or high latencies are expected.

Henderson *et al* [9] concluded that at the latency of 100, the delay in the actions is noticeable to some players, at the latency of 200, practically all players notice the delay and at 300 milliseconds of latency, the loss of QoE is inevitable. The DR technique is largely used because of its simplicity and the fact that it is good enough to solve the problem. But if medium and high latencies are expected or unwanted, the greater imprecision of the DR technique in these cases become evident and alarming. The usage of the MAADR, specially in these cases, is recommended to maintain adequate levels of QoE, considering the higher costs of network bandwidth and processing present on the MAADR compared with the DR technique.

V. Conclusion

The main contribution of this paper is the experimental analysis of the performance of the DR and MAADR techniques, applied in multiplayer games, under a relevant interval of latencies. The situations tested cover from a responsive and consistent case with no latency or delay (0ms) to great levels of latency (300ms), which causes a great loss in the consistency and Quality of Service and makes the loss of players imminent. The obtained results show that the DR algorithm loses a great magnitude from its performance compared to the MAADR algorithm, in which the latter can maintain good levels of Quality of Experience under medium and high latencies. Under low latencies, the error difference between the two techniques is less relevant, which explains the usage of the DR as the main technique since its conception for DVEs. However in medium and high latencies (100-300ms) the proportion in which the DR increases its error and the higher magnitude overall of the error makes

the shared information less consistent and increases considerably the loss in QoE. For these reasons, the usage of the MAADR in situations where medium or high latencies are present or there is an unwanted loss of QoE, is recommended. The adoption of the MAADR in place of the DR would provide an increase of the QoS and an increase on the interest in the game.

ACKNOWLEDGEMENTS

The authors would like to thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) for the financial support that allowed the development and publication of this paper.

REFERENCES

[1] P. Schuytema, "Design de games: uma abordagem prática," Cengage Learning, 2008.
[2] Newzoo, "2016-2020 global games market," unpublished.
[3] M. G. Wagner, "On the Scientific Relevance of eSports," Int. Conf. on internet computing, 2006, p. 437-442.
[4] C. Savery , T. C. N. Graham, "Timelines: simplifying the programming of lag compensation for the next generation of networked games," Vol. 19, Multimedia Systems, 2013, p. 271–287.
[5] Y. Chen, E. S. Liu, "A path-assisted dead reckoning algorithm for distributed virtual environments," 19th ed., IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DSRT), IEEE, 2015, p. 108–111.
[6] S. Aggarwal, H. Banavar, A. Khandelwal, S. Mukherjee, S. Rangarajan, "Accuracy in dead-reckoning based distributed multi-player games," 3rd ed., ACM SIGCOMM workshop on Network and system support for games, ACM, 2004, p. 161–165.
[7] F. Sanglard, "Quake engine code review : Prediction," Link, 2009, Accessed: 2018.
[8] Y. W. Bernier, "Latency compensating methods in client/server in-game protocol design and optimization," Link; 2001, Accessed: 2018.
[9] T. Henderson, S. Bhatti, "Networked games: A qos-sensitive application for qos-insensitive users?," Proceedings of the ACM SIG COMM Workshop on Revisiting IP QoS: What Have We Learned, Why Do We Care?, 2003, p. 141–147.
[10] V. Y. Kharitonov, "Motion-aware adaptive dead reckoning algorithm for collaborative virtual environments," Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry, ACM, 2012, p. 255–261.
[11] S. Singhal, M. Zyda, "Networked virtual environments: design and implementation," ACM Press/Addison-Wesley Publishing Co., 1999.
[12] B. N. Levine, N. E. Baughman, "Cheat-proof playout for centralized and distributed online games.," Vol. 1. IEEE, INFOCOM, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings, IEEE, 2001.
[13] H. Lee, E. Kozlowski, S. Lenker, S. Jamin, "Synchronization and cheat-proofing protocol for real-time mulit-player games," Proceedings of 1st Workshop on Entertainment Computing, Makuhari, Japan. 2002.
[14] M. Mauve, J. Vogel, V. Hilt, W. Effelsberg, "Local-lag and timewarp: providing consistency for replicated continuous applications," IEEE Transactions on Multimedia 2004, p. 47–57.
[15] E. Cronin, B. Filstrup, A. R. Kurc, S. Jamin, "An efficient synchronization mechanism for mirrored game architectures," Proceedings of the 1st workshop on Network and system support for games, ACM, 2002, p. 67–73.
[16] H. Shen, S. Zhou, "Achieving critical consistency through progressive slowdown in highly interactive multi-player online games," IEEE 17th Iternational Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, 2013, p. 332–337.
[17] W. Cai, F. B. Lee, L. Chen, "An auto-adaptive dead reckoning algorithm for distributed interactive simulation," Proceedings of the 13th Workshop on Parallel and Distributed Simulation, IEEE, 1999, p. 82–89.
[18] B. STEPHEN, J. T. L. Chen, "Adaptive dead reckoning algorithms for distributed interactive simulation," School Of Computer Engineering Nanyang Technological University Singapore, 2000.
[19] T. P. Duncan, D. Gracanin, "Algorithms and analyses: pre-reckoning algorithm for distributed virtual environments." Proceedings of the 35th conference on Winter simulation: driving innovation, Winter Simulation Conference, 2003, p. 1086–1093.
[20] Unity game engine, 2018 Link, Accessed: 2018.
[21] Networking framework for unity, 2018 Link, Accessed: 2018.