Performance of Dead Reckoning Algorithms Across Technology Eras

Will Oliver, Peter Ryan and Peter Ross
Defence Science & Technology Group
506 Lorimer St, Fishermans Bend, Victoria 3207, Australia
william.oliver@dst.defence.gov.au
peter.ryan@dst.defence.gov.au
peter.ross@dst.defence.gov.au

Keywords:
Distributed Interactive Simulation; dead reckoning algorithms, Moore's Law

ABSTRACT: *Dead reckoning (DR) is employed in Advanced Distributed Simulation to reduce the need to continually update a simulated entity's state information. The IEEE 1278.1 Distributed Interactive Simulation (DIS) protocol standard defines a set of 9 algorithms for entity position and orientation dead reckoning. Early work on the performance of these algorithms was reported by researchers Towers and Hines sponsored by the then Advanced Research Projects Agency to assess the functionality of a distributed warfighting environment with real and simulated entities. These authors also fortunately included C source code and sample results that were used to regenerate the 1994 model results. This code and these results can be used as benchmarks for assessing DR algorithm performance from 1994 to the present era where CPU speed is far greater. Results show that DR performance in even a simple modern processor such as a Raspberry Pi is far superior to the most advanced systems of 1994. Relative performance of the standard algorithms also shows similar trends across technology eras with the algorithms that use orientation always running far slower. Estimates of processor time can also be determined theoretically from counting the number of floating point operations that each algorithm processes. These results are relevant to the approach that will be adopted for the next generation of DIS and other simulation protocols such as the Real Time Platform Reference Object Model.*

1. Introduction

Distributed Interactive Simulation (DIS) is a networking protocol standard that provides a method of communicating entity state and other information such as voice communications, radar and sonar emissions through Protocol Data Units (PDUs). These PDUs consist of data packets that are broadcast over the simulation network. The latest standard IEEE-1278.1-2012, also known as DIS version 7, was released in 2012 [1]. The Simulation Interoperability Standards Organization (SISO) is in the process of updating DIS to a proposed version 8 [2].

A key feature of DIS is its use of a technique known as dead reckoning to limit the rate at which simulated entities need to update their entity attributes such as position and velocity. DIS also uses a heartbeat mechanism to maintain the continuity of entities within a simulation exercise.

Early work on the performance of these algorithms was reported by Towers and Hines and was sponsored by the then Advanced Research Projects Agency to assess the functionality of a distributed warfighting environment with real and simulated entities [3]. These authors also fortunately included C source code and sample results that were used to regenerate the published 1994 model results. This code and these results can be used as benchmarks for assessing dead reckoning algorithm performance from 1994 to the present era where CPU speed is far greater.

2. Dead Reckoning in DIS

In DIS, each application maintains an internal model of the entities it is simulating and also dead reckoned models of these. When an entity's position or orientation has changed from the dead reckoned model by more than a set threshold value, the application issues a new Entity State PDU to inform the other participating applications of its position and orientation.

DIS provides a standard set of 9 dead reckoning algorithms although only a few are generally used. The first of these assumes a static entity, numbers 2 – 5 use world-centered coordinates with the origin at the centre of the earth, while numbers 6 – 9 employ body-centered coordinates. Each algorithm is designated by three elements:

- The first indicates whether the model specifies rotation as either fixed (F) or rotating (R).
- The second specifies dead reckoning rates to be held constant as either rate of position (P) or rate of velocity (V). This also identifies the algorithm as being first (P) or second (V) order with respect to position.
- The third specifies the coordinate system to be used with the dead reckoning algorithm as either world coordinates (W) or body axis coordinates (B).

DIS version 8 proposes to replace the geocentric coordinate system with a geodetic coordinate system. This has implications for dead reckoning where it is suggested that a single algorithm could suffice for most situations [4]. This would save bandwidth by reducing coordinate conversion between geocentric and geodetic coordinate systems.

2.1 Floating Point Operations

The number of floating point operations (FLOPS) for each algorithm was counted with the results shown in Table 1. This demonstrates that the algorithms (3, 4, 7, 8) which include orientation generally have more FLOPs than those that assume fixed orientation (1, 6, 9) and would thus require more processing time. Algorithm 9 still requires calculation of rotation matrices (section 4.2) and has 115 FLOPS.

Table 1. Number of operations for different dead reckoning algorithms (using Tower and Hines code [3])

Algorithm	Model	No of Floating Point Operations
1	Static	15

2	FPW	47
3	RPW	110
4	RVW	99
5	FVW	43
6	FPB	50
7	RPB	125
8	RVB	177
9	FVB	115

However execution times do not generally scale linearly with the number of FLOPS since the different types of floating point operation take different times and the ratios of these are processor dependent. Divide and multiply operations are more complex and thus slower than add and subtract operations, while cosine and sine functions (that are used in the dead reckoning algorithms) are slower still. Thus the number of FLOPS per algorithm can only be used for a qualitative estimate of performance.

Limare [5] carried out some tests of integer and floating point arithmetic for different processors including Intel and ARM for various precisions. He found that floating point products are generally twice as slow as sums and that division is even slower. The Intel processors perform better than the ARM processors used in a Raspberry Pi.

2.2 Processing Times and Metric for Dead Reckoning Performance

The C code listing from Towers and Hines was provided with their paper [3]. This was converted to source code using Adobe's Optical Character Recognition feature and then debugged to remove transcription errors. The reconstituted C code was then validated using the two test cases provided in the paper. No further changes were made to the code.

Linux, and other POSIX[1] based operating systems, provide inbuilt utilities to determine processor time for specific operations. These include: *clock()* and *clock_gettime()*. The *clock* function returns an approximation of processor time used. However, it is only accurate to microseconds. Thus to use *clock* to determine processing time, many repetitions of the DR algorithm must be performed.

The *clock_gettime* function returns a C structure with two components: (a) the clock time in seconds, and (b) the clock time in nanoseconds after the second. This is a more accurate way to determine processing time. Towers and Hines provided results for two test cases with different initial conditions for DRA 4 and DRA 8. The method applied to determine processing time for each dead reckoning algorithm was:

i. Create an array of 1000 identical PDUs for the first test case
ii. Measure the elapsed time to perform dead reckoning for the array of PDUs for the first test case using the initial values from [3]
iii. Repeat the measurement for the second test case with initial values from [3]
iv. Average processor time over the 1000 PDUs and both test cases to get the mean value of processing time.

In all cases the free GNU Compiler Collection (GCC) system was used to compile and run the source code. Several Linux variants (Ubuntu, Debian, and Raspbian) and Cygwin[2] were employed as the test environments.

A Dead Reckoning Metric was then defined as the average of the inverse of the sum of processing times for all 9 algorithms normalised to that of the slowest system designated as system 1 (the Motorola 60840 VME Processor):

$$Metric\ for\ system\ i = \sum_{j=1}^{j=9} T_j^1 \Big/ \sum_{j=1}^{j=9} T_j^i$$

This metric will be used to rate the performance of different systems by providing a single figure of merit.

[1] The Portable Operating System Inteface (POSIX) is described by IEEE Std 1003.1
[2] Cygwin is a Unix-like environment that runs under Windows

3. Results

Measurements of dead reckoning performance were made on modern systems and compared with the 1994 results from Towers and Hines. These are discussed in the following sections.

3.1 Results from 1994

In their 1994 paper, Towers and Hines carried out tests on DRA 4 (second order; world coordinates; roll, pitch and yaw rotation) and DRA 8 (second order; entity coordinates; roll, pitch, and yaw rotation) for various systems. Detailed results were provided for a set of initial values for these two algorithms dead reckoned over an 0.5 s timestep while general performance data was provided for all 9 algorithms using the DIS 2.0.3 draft standard dated 1993 [6]. Articulation parameters were not included in the entity dead reckoning.

The systems on which the dead reckoning algorithms were tested by these authors [3] are listed in Table 2.

Table 2. Systems tested by Towers and Hines [3]

System	Released	Features
Motorola 60840 VME Processor[3]	1990	used in early personal computers;40 MHz; floating point unit; same features as Intel 80486
SPARC Station 10[4]	1992	40 MHz workstation
SGI Indigo 2[5]	1993	workstation with 100 Mhz and 150 Mhz variants

The initial results from their 1994 paper are reproduced in Figure 1. However in the original study no details were provided as to how these results were determined so their accuracy is unknown. Results are plotted on a semi-logarithmic scale. Interestingly the curves are nearly parallel on this scale with the slowest algorithms 3, 4, 7, and 8 being the most complicated in terms of operations to be performed. Algorithm 1 has non-zero processing time since CPU time is required for checking the dead reckoning field, updating the time, and checking for articulation parameters.

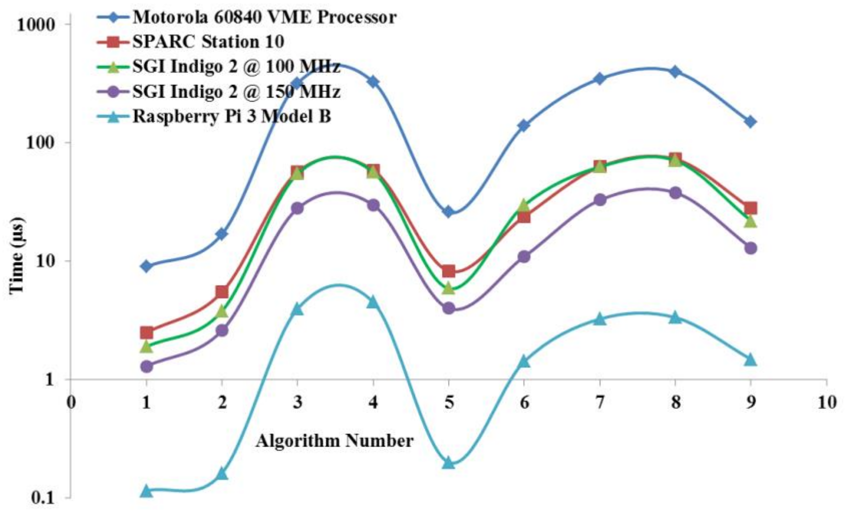

Figure 1. Comparison of performance of processors for the standard set of dead reckoning algorithms from Towers and Hines [3]

[3] https://en.wikipedia.org/wiki/Motorola_Single_Board_Computers
[4] https://en.wikipedia.org/wiki/SPARCstation_10
[5] https://en.wikipedia.org/wiki/SGI_Indigo

From Table 1, it can be seen that the algorithms that include orientation have up to double the number of FLOPS. This is consistent with Figure 1 and is expected since these algorithms require calculation of Euler angles to manage orientation. The parallel nature of the curves also shows that the ratios of processor times for the different systems are nearly identical and also provides confidence in the methodology used to determine processor time.

The results from a Raspberry Pi 3 Model B running the Raspbian operating system are also included. Interestingly, this single board computer easily outperforms the fastest systems on which the original 1994 tests were made. For algorithm 4, for example, the Raspberry Pi takes only about 4.5 ms whereas the SGI Indigo 2 running at 150 MHz requires 30 ms and the Motorola VME 330 ms. The general shape of the graph for the Raspberry Pi is also similar to those of the older systems.

3.2 Systems Tested

Table 3 lists all the systems on which tests were performed. These include single board computer systems, workstations and modern PCs.

Table 3. Systems tested in present report

System	Date	Features	Operating System
Raspberry Pi 1 model B: BCM2835 (RPI1)	2012	700 MHz ARM1176JZF-S processor - ARMv6 architecture	Linux (Raspbian)
Raspberry Pi 2 model B (RPI2)	2014	BCM2836 with a 900 MHz 32-bit quad-core ARM v7 architecture	Linux (Raspbian)
Raspberry Pi 3 model B (RPI3)	2016	BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARMv8-A architecture (note there were 2 Raspberry Pi 3 model B systems tested)	Linux (Raspbian)
Dell Inspiron 13 Notebook (PC1)	2015	Intel i7-5500, 2.4 GHz	Windows 10 (Cygwin)
Dell Desktop (PC2)	2013	Intel i7-3770; 3.7 GHz	Windows 7 (Cygwin)
Dell Latitude 7280 (PC3)	2017	Intel i7 7600U. Dual core operating at 3.9 GHz.	Linux (Ubuntu)
Creator Ci20	2015	Dual-core 1.2 GHz MIPS32 processor	Linux (Debian)
Sun Blade 100 workstation	2001	UltraSPARC IIe500 MHz	Linux (Debian)
G4 PowerMac	1999	PPC G4 400 MHz	Linux (Ubuntu)

Figure 2 shows results from several modern systems. Two Dell computers PC1 and PC2 run Windows 7 and 10 natively and the tests were done using the Linux environment Cygwin. The other, PC3, is a 2017 Dell PC running Ubuntu. Interestingly their performances are nearly identical across the range of algorithms however they are far more powerful than the Raspberry Pi system.

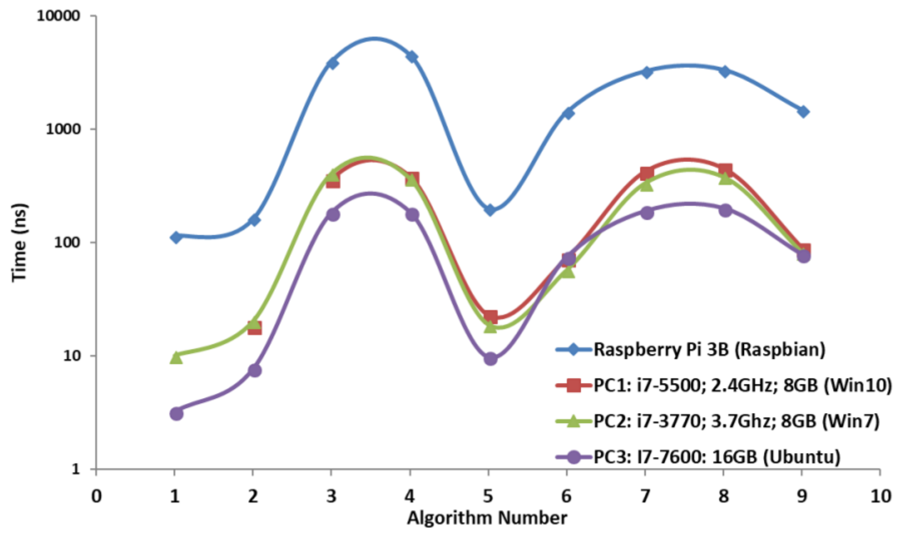

Figure 2. Comparison of performance of modern processors for the standard set of dead reckoning algorithms

3.3 Modern Single Board Systems

As part of this investigation, several single board systems were tested including four Raspberry Pi systems (running Raspbian) and one Creator Ci20 system. These results are shown in Figure 3. The Sun Blade system results are also included for comparison.

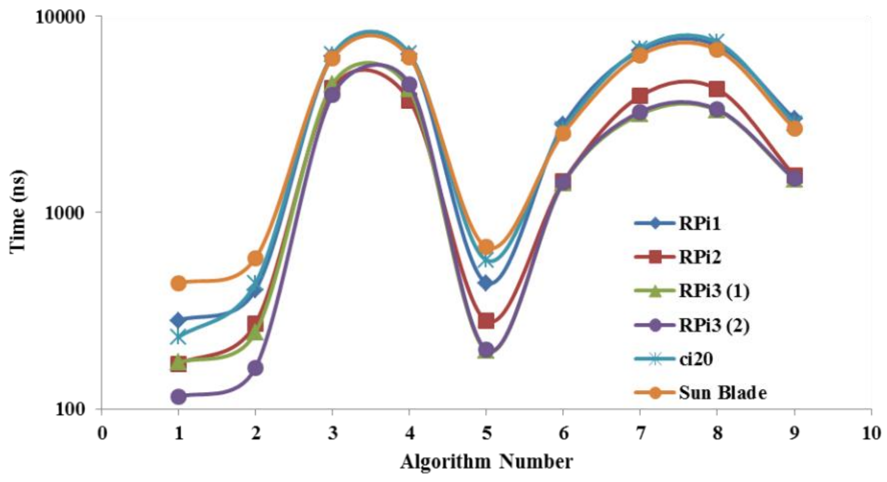

Figure 3. Comparison of performance of single board systems for the standard set of dead reckoning algorithms using the defined metric and a semi-log scale

Interestingly there is not a huge difference in performance among the four Raspberry Pi systems although the RPi2 and RPi3 have more powerful ARM V7 and V8 processors respectively than the initial RPi1 that has the older ARM V6. The MIPS Creator Ci20 is a Linux/Android development system that comes preloaded with Debian Linux and has a MIPS 32 processor [6]. This system has similar performance to the RPi1 and is roughly twice as slow as the RPi3. As before the general shapes of the curves are similar on the logarithmic scale. The performance of the Raspberry Pi machines, as noted previously, is far superior to that of the 1994 systems

[6] https://en.wikipedia.org/wiki/Imagination_Creator

enabling these systems to run real time simulations and games. The Sun Blade system (2001) has similar performance to the Ci20 system (2018).

4. Analysis of Dead Reckoning Performance

4.1 Dead Reckoning Metric

Figure 4 shows the dead reckoning metric for all the systems tested, ranging from 1.0 for the VME processor to nearly 2000 for modern PC systems running Linux and Windows variants. The systems are sorted in order of dead reckoning metric value, not age. The four 1994 systems perform worse than a modern Raspberry Pi while the G4 Power Mac (1999) and Sun Blade (2001) have similar performance to the MIPS ci20 (2018) single board system. The highest metric of 1865 is measured for the Linux PC with an Intel i7 7600U dual-core system operating at 3.9 GHz. This demonstrates the vast increase in computing power over the past quarter century.

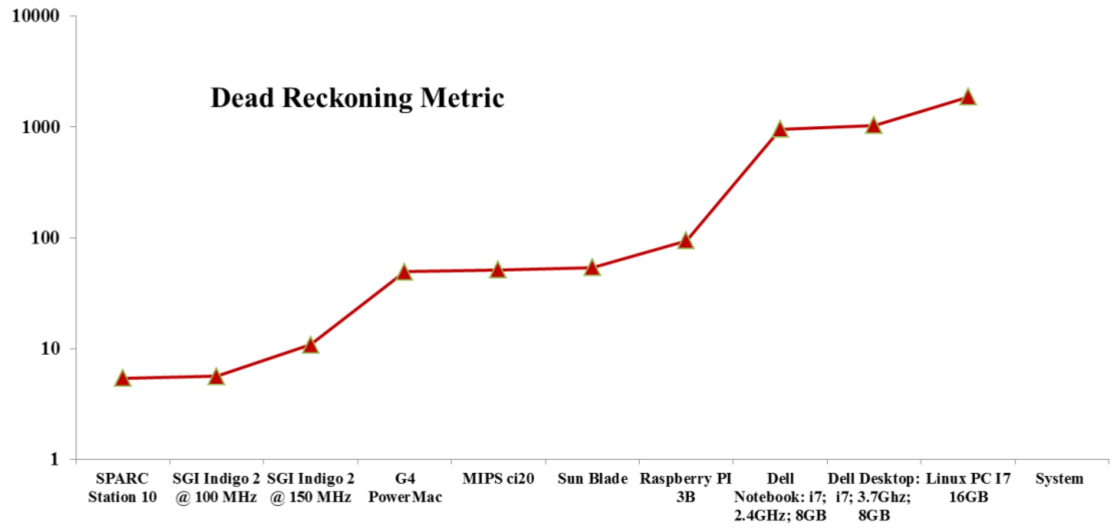

Figure 4. Comparison of performance of computer systems for the standard set of dead reckoning algorithms using the defined metric and a semi-log scale (note x-axis denotes computer systems rather than years)

Gordon Moore observed that the number of transistors in a dense integrated circuit doubled roughly every 18 month – 2 years leading to the eponymous 'Moore's Law' [7]. Thus computing power should also double across an 18 month to 2 year period if it also scales linearly with number of transistors. Over 24 years there will be roughly 12 - 16 generations leading to an increase in the range: 4000 - 65000. This is greater than the observed factor of 1800 but computing performance does not necessarily increase linearly with number of transistors. Further, many different systems with different architectures are being compared rather than a single processor family over the 24 year period and there are other factors that influence performance such as clock speed and cache memory.

4.2 Relative Perfomance of Specific Algorithms

It has been noted previously that the relative performances of the different DRAs are similar across all systems discussed. A quantitative comparison of relative performance is shown in Figure 5 for the ratios of measured times of (1) DRA 4/DRA 8, (2) DRA 4/DRA 5, and (3) DRA 8/DRA 9. DRA 4 and DRA 8 were selected since they are the most complex world-centered and body- centered algorithm respectively while DRA 5 and DRA 9 are the same algorithms with no rotation included.

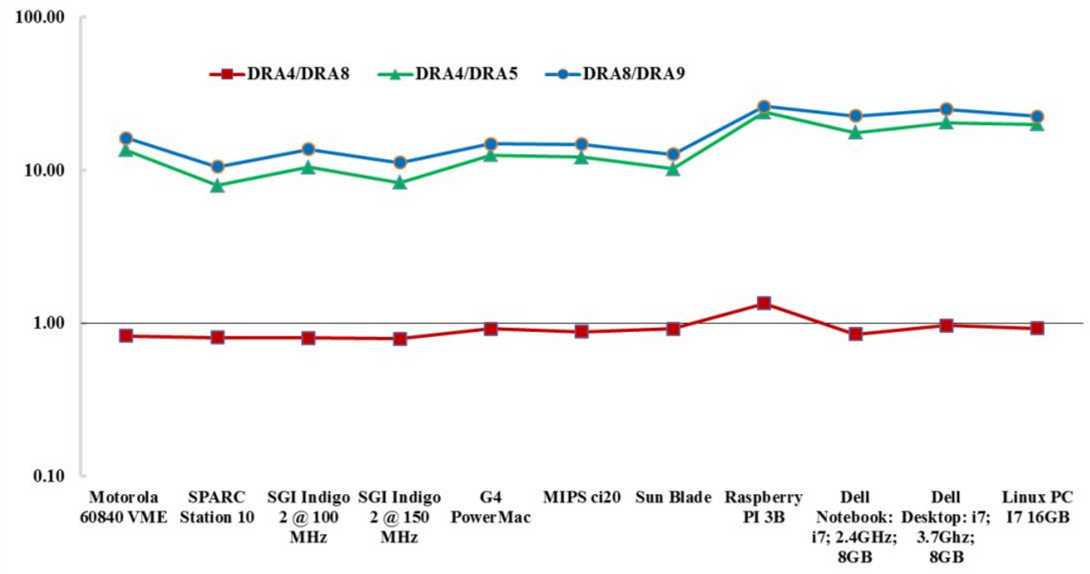

Figure 5. Comparison of relative performance of all systems for three key dead reckoning algorithms

DRA 4 and DRA 8 have similar performance for all systems with mean ratio 0.84. However the mean value for DRA 4/DRA 5 is 12.2 while that for DRA 8/DRA 9 is only 2.8 demonstrating that DRA 9 is far slower than DRA 5. This reflects that DRA 5 is a simple quadratic [1]:

$$P = P_0 + V_0 \Delta t + \frac{1}{2} A_0 \nabla t^2$$

where P is the dead-reckoned position after time Δt, and P_0, V_0, A_0 are the initial position, velocity, and acceleration respectively. In contrast, DRA 9 requires calculation of several rotation matrices to determine the dead-reckoned position [1]:

$$P = P_0 + [R_0]^{-1}_{w \to b}([R1]V_b + [R2]A_b)$$

where R_0 is the entity initial world to body orientation matrix, V_b, A_b are the body referenced velocity and acceleration respectively and *R1* and *R2* are as defined in Annex E of [1].

This is also broadly consistent with the FLOPS count in Table 1 where DRA 4 has 99, DRA 5 43, DRA 8 177 and DRA 9 115 FLOPS respectively. It would be expected that DRA 9 would run far slower than DRA 5 from these data.

4.3 Implications for DIS Version 8

Murray proposed a reduced set of dead reckoning algorithms for DIS Version 8 based on using the Geodetic Coordinate system [4]. For moving entities, a new single body-referenced algorithm, DRA 10, would be implemented as the Geodetic equivalent of DRA 8 for moving entities. Circular turns are handled more accurately using body-referenced coordinates since the circular turning path is derived from angular velocity rather than acceleration. DRA 1 would still be required for static entities while life forms and hovering helicopters that can rotate but do not always point in their direction of motion may require other algorithms. Tests showed that the body-referenced dead reckoning algorithm gives superior performance in geodetic coordinates than in geocentric for both straight line and circular trajectories.

Considering the above results and analysis, DRA 8 has the greatest number of floating point operations (177) of the standard algorithm set, but performs only marginally worse than the equivalent world-referenced DRA 4. However, the highest performing system here (PC3) requires only about 200 ns for this computation and future systems will perform faster. These results can be used to benchmark proposed algorithms for later versions of

DIS.

5. Discussion

Dead reckoning has long been considered to be a significant user of computing resources in simulation exercises. However the vast increase in performance over the past quarter century due to Moore's Law has resulted in dead reckoning processing times being reduced by a factor of nearly 2000. What required tens of µs to process on 1990s-era systems now takes hundreds of ns or fractions of 1 µs. For example, the Sparc Station 10 required 73 µs to process DRA 8; the Raspberry Pi 3 Model B requires only 3.35 µs; and a modern Intel-based PC of the order 0.2 µs. The modern single board hobby systems tested outperformed the most powerful 1994 systems.

Attempts to estimate processor time theoretically from counting the number of floating point operations that each algorithm processes has too many uncertainties due to different processor architectures. Simply counting the number of FLOPS for each algorithm ignores the differences in performance of the differing classes of floating point operation. Experimental measurement using a standard set of parameters is a more accurate means of assessing performance.

The logarithmic scales show that the relative performance of the algorithms is strikingly similar across different processors showing that dead reckoning performance increases nearly linearly with increased processing power. Forward projections for future systems can be postulated based on the historical performance observed.

In this study we have only considered single-threaded dead-reckoning performance. Parallelism can also be used to increase throughput of the dead reckoning algorithm. In its simplest form this involves running multiple instances of the dead reckoning algorithm across multiple threads or cores noting that several of the tested systems have multiple cores (for example, the Raspberry Pi 3 Model B has four cores). More sophisticated parallelism would require rewriting the dead reckoning implementation software to use vector-based instructions. For example, Intel AVX-512 instructions can perform the same floating operation to eight 64-bit doubles values simultaneously, and such a dead reckoning implementation would be capable of dead-reckoning multiple entities simultaneously.

6. Conclusions

Dead reckoning performance was studied for a range of systems from different technology eras using some 1994 data as the baseline. Results show that dead reckoning performance in even a simple modern processor such as a Raspberry Pi is far superior to the most advanced 1994 systems. Relative performance of the standard algorithms also shows similar trends across technology eras with the algorithms that use orientation always running far slower. These results can be used to predict performance of future systems. They are also relevant to the approach that will be adopted for the next generation of DIS and other simulation protocols such as the Real Time Platform Reference Object Model by providing benchmarks for DRA performance.

7. References

1. IEEE (2012). *IEEE Standard for Distributed Interactive Simulation--Application Protocols: EEE Std 1278.1-2012*.
2. Murray, R. E. (2018). Gen3: What's Up with DIS Version 8 (2017-SIW-028). In: *2018 Winter Simulation Innovation Workshop* Orlando, Florida, US: 21 - 26 Jan 2018
3. Towers, J. and Hines, J. (1994). *Highly Dynamic Vehicles in a Real/Simulated Virtual Environemt: Equations of Motion of the DIS 2.0.3 Dead Reckoning Algorithms*. Report 94-57, Defense Advanced Research Projects Agency
4. Murray, R. E. (2018). Dead Reckoning in Geodetic Coordinates for Improved LVC Interoperability (18W-SIW-029). In: *2018 Winter Simulation Innovation Workshop*, Orlando, Florida, US: 21 - 26 Jan 2018
5. Limare, N. (2014) Integer and Floating-Point Arithmetic Speed vs Precision (http://nicolas.limare.net/pro/notes/2014/12/12_arit_speed/).

6. *Standard for Information Technology - Protocols for Distributd Interactive Simulation Applications, Version 2*. (1993). Institute for Simulation and Training, University of Central Florida
7. Moore, G. E. (1965) *Cramming more components onto integrated circuits*. In *Electronics 38 (8): 114–117*.

8. Author Biographies

WILL OLIVER holds degrees in Aerospace Engineering and Mathematics. Will joined Defence Science & Technology (DST) Group's Aerospace Division in 2006 and works in the Air Operations Simulation Centre; researching interoperability issues and analysis techniques for advanced distributed simulation. Prior to joining DST, he developed software for flight simulators and simulated maintenance trainers.

DR. PETER RYAN is an Honorary Research Fellow in Defence Science & Technology Group's Aerospace Division. He has a 30 year background in the modelling and simulation of military operations. His main research interests include Advanced Distributed Simulation, real time simulation, synthetic environments, and their potential to provide enhanced training solutions for the Australian Defence Force. He is a member of SISO's DIS Product Development Group.

PETER ROSS graduated from RMIT University in 2001 with a Bachelor of Applied Science, majoring in computer science. He joined DST's Aerospace Division in 2003, where he has undertaken a role in evaluating the use of advanced distributed simulation for collective training.