Trajectory Inference Optimization Based on Improved DR Algorithm

Li Yao-yu, Hou Fei, Ren Wei, and Ma Man-hao

Abstract Trajectory planning is often encountered in distribute tactical virtual simulation. To improve the non-smooth problem caused by the nonunified step of components in the simulation of naval battlefield, we proposed an improved DR algorithm, which is verified by simulation experiments, to provide a smooth trajectory inference method, and to achieve finer grain of trajectory inference in distributed interactive simulation.

Keywords DR algorithm · Trajectory planning · Distributed simulation

1 Introduction

The position of each intermediate time cannot be determined by interpolation for a model with continuous location information constraints provided by the simulation engine because at the beginning of each simulation moment [1], it does not know the precise location information, or refinement end position, which should be provided in the following simulation moment.

Only the current location or entity information of the prior steps can be used to predict the position information of the subsequent refining points [2]. The model is then modified to reflect the comparison results in order to minimize prediction error at the subsequent simulation moment, which compares the entity position information provided by the simulation engine with the predicted information.

L. Yao-yu (✉) · M. Man-hao
Science and Technology On Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China
e-mail: garett_1984@hotmail.com

H. Fei
Hunan Tobacco Company, Changsha, China

R. Wei
No. 31675 Troops, Zhangjiakou, China

J. P. T. Mo (ed.), *Proceedings of the 8th International Conference on Mechanical, Automotive and Materials Engineering*, Lecture Notes in Mechanical Engineering,
https://doi.org/10.1007/978-981-99-3672-4_2

A common state estimation and error correction approach in distributed interactive systems is the DR algorithm. This paper enhances the current DR algorithm in accordance with the ship's movement legislation starting from building of an improved model with limits on the location information [3]. To forecast and smooth out the movement of ships, an improved DR algorithm based on a movement model is suggested.

2 DR Algorithm

2.1 Principle of Dead Reckoning

Dead Reckoning, known as DR, is a navigational technique that involves utilizing a previously established location to calculate the present position of a moving object, as well as adding estimations of its speed, heading, and course over the length of time [4].

As a result of the growing distance between the simulation nodes, and the number of nodes in the entire simulation system has expanded along with the size of distributed interaction simulation, the amount of interactive data exchanged between each node has raised system network load and decreased the effectiveness of system simulation operation, the system's synchronization has been completely disrupted by the information transmission latency.

The DR algorithm is a key component of DIS (Distributed Interactive Simulation), and it focuses on reducing the delivery of interaction information between each simulation node reasonably while compensating for transmission delays [5]. This lowers the network's transmission load and boosts the effectiveness of the entire distributed interactive system.

Local simulation nodes must forecast the state of the simulation node that changes in the interaction connection in order to limit interactive information transmission between distributed nodes and maintain the regular functioning of the full distributed simulation system at the same time. Each simulation node in the DR method has a low-level DR model for the state recursive in addition to a high-level exact model defining its own imitation of the state.

The node with the DR model used to register the entity to the node locally is also preserved by other nodes that interact with this simulation node [6]. As illustrated in Fig. 1, when the accurate state surpasses the threshold, the node communicates the precise state information of the local entity to the simulation node participating in the interaction and modifies every attribute of this node's DR model.

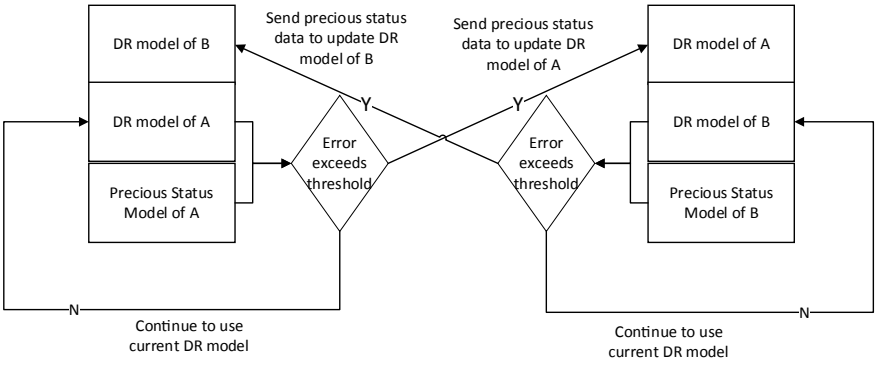

Fig. 1 DR algorithm principle

2.2 Commonly Used DR Algorithm

The DR model is mainly used for the calculation of space state, and the DR model cannot be too complicated. Commonly space status DR algorithm include as follows.

(1) First order algorithm

The formula of the space location is:

$$\begin{cases} x_i = x_0 + v_{x0} i \Delta T \\ y_i = y_0 + v_{y0} i \Delta T \\ z_i = z_0 + v_{z0} i \Delta T \end{cases} \tag{1}$$

In addition to calculating the position, the inclination angle of the entity is often calculated in the simulation, and the inclination angle is calculated based on the rotation speed:

$$\begin{bmatrix} \psi \\ \theta \\ \varphi \end{bmatrix} = \begin{bmatrix} \cos\psi tg\theta & \sin\psi tg\theta & 1 \\ -\sin\psi & \cos\psi & 0 \\ \cos\psi \sec\theta & \sin\psi \sec\theta & 0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2}$$

The formula of the inclination angle calculation is:

$$\begin{cases} \psi = \psi_{i-1} + \psi_{i-1} \Delta T \\ \theta = \theta_{i-1} + \theta_{i-1} \Delta T \\ \varphi = \varphi_{i-1} + \varphi_{i-1} \Delta T \end{cases} \tag{3}$$

(2) Second order algorithm

During the actual usage of the DR algorithm, the space posture generally uses first order algorithm, and the calculation of the position is generally used in the second order algorithm, with expression as below:

$$\begin{cases} x_i = x_0 + v_{x0} i\Delta T + \dfrac{1}{2} a_{x0} (i\Delta T)^2 \\ y_i = y_0 + v_{y0} i\Delta T + \dfrac{1}{2} a_{y0} (i\Delta T)^2 \\ z_i = z_0 + v_{z0} i\Delta T + \dfrac{1}{2} a_{z0} (i\Delta T)^2 \end{cases} \tag{4}$$

In the above formula, V is the speed vector, A is the acceleration, which is the rotation speed, and T is the calculation step (generally the simulation step size).

3 Problems with Current DR Algorithms in Simulation

In the naval tactical simulation system, the simulation engine, as an independent component, is responsible for the entire simulation scheduling control [7]. The virtual battlefield environment, as another independent component, drives the display of the three-dimensional scene, based on the physical state interaction information received from the simulation engine through the RTI (Run-time Infrastructure), shown in Fig. 2.

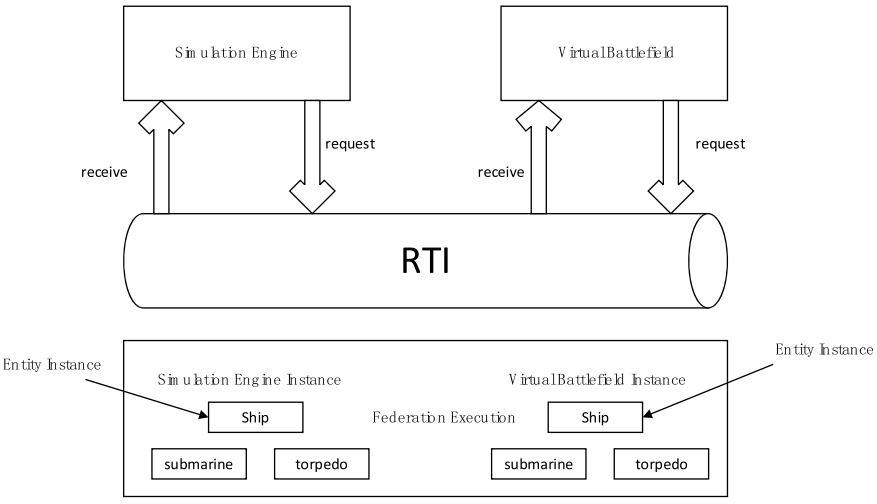

Fig. 2 HLA-based system federation structure

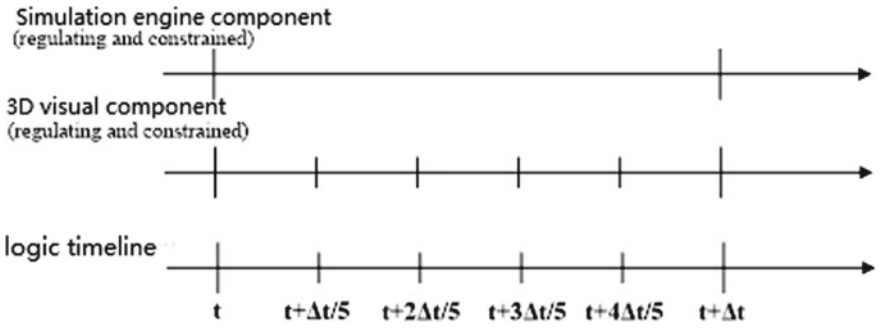

Fig. 3 Time advancement of each entity

The virtual battlefield environment must deliver data points in the right granularity in order to portray a continuous and fluid fighting process [8]. There are not enough data points available for a virtual combat setting when using tactical deduction since the simulation advancement step is typically bigger than the actual position stance required. As a result, in the virtual combat environment, it is required to use smaller steps to progress, split multiple intermediate periods inside the simulation engine, and determine the entity's position at the moment.

Because the physical location information provided by the subsequent simulation moment of the simulation engine is uncertain, the method of interpolation cannot be used for entities with a simulation engine to provide location information in the environment of the virtual battlefield at the initial moment of the advancement of the simulation engine [9]. Only previous data may be used to anticipate where each intermediate stage will be, and this prediction must be limited by the precise position data from the entity in the next simulation. Referring to Fig. 3.

Assuming ΔT is the step size of the simulation engine, and the step size of virtual battlefield environment is $\Delta t/5$, which means, it needs adding four intermediate status data between the data provided by the simulation engine, in order to ensure the consistency of the logic time of the simulation engine and the virtual battlefield environment, both components adopted the controlled and limited time management strategy in HLA (High Level Architecture). In a Δt, the interaction process of the two components is as follows:

(1) At the time T, the simulation engine requested the state information of the entity of $T + \Delta t$ from the RTI, and requests to advance to $T + \Delta t$;
(2) Three-dimensional visualization component calculates the status information of the entity at $T + \Delta t/5$ by the status information of the entity at the time of T, and requests to advance to $T + \Delta T/5$;
(3) Three-dimensional visualization State component is promoted at $\Delta t/5$. At $T + 4\Delta T/5$, after a request to advance to $T + \Delta T$, the RTI transmits the status information of the $T + \Delta T$ of the simulation engine to the three-dimensional component;

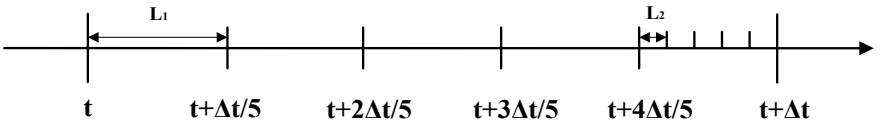

Fig. 4 Smoothing process

(4) The simulation engine and the three-dimensional visualization component are advanced to T + ΔT.

The virtual battlefield environmental element cannot be totally consistent with the entity information given by the simulation engine since it uses the anticipated approach to retrieve the entity information of intermediate instant at ΔT. There is an excessive discrepancy between the entity information given by the simulation engine and that expected by the battlefield environment (that is, exceeding the pre-set threshold). The three-dimensional scene display is prevented from changing instantly on the one hand, while the DR model of the predicted DR needs to be updated.

Let the entity state predicted by T + 4ΔT/5 is S4, the entity state predicted at the time of T + ΔT is S5, and the entity state of the T + ΔT get from the simulation engine is S5′. At the time of T + 4ΔT/5, S5′ can be obtained through the virtual battlefield environment request. The traditional smooth method is usually used to adding several intermediate points between T + 4ΔT/5 and T + Δt, transiting from S4 to S5′ by interpolation.

If the increasing entity state data point and logic time are matching, the simulation operation won't have a logical difficulty, according to the logic of simulation calculation. The adjustment of simulation time sequence has no bearing on the amount of time between each drawing in 3D performance. The sole aspect in how logic time is controlled is the quality of its own picture. As a result, as seen in Fig. 4, performance discrepancies will result from increasing the data point.

The entity state change rate in the Δt is certain. If four points are added between T + 4ΔT/5 and T + Δt, the change in the state of the entity between the data plotters before the t + 4Δt/5 time is 5 times that after the t + 4Δt/5 moment, and the interval between the data of each data has not changed significantly.

The three-dimensional performance shows the three-dimensional scene's changing condition. As a result, the conventional smooth treatment has to be enhanced to accommodate the rendering of three-dimensional real-time scenes.

4 Trajectory Inference Oriented Improved DR Algorithm

The entity that can provide location and posture information of the positioning engine for tactical simulation warfare deductions is often low-speed, such as ships and submarines [10]. The movement rule is quite straightforward for these slowly moving objects, and the transition from one state of motion to another often takes long time.

These qualities allow for the simplification of this sort of entity's movement models and the extraction of a restricted movement mode. The value utilized in the subsequent simulation step will depend on the discrepancy between the real values transmitted by the current prediction value and the actual value of the simulation engine.

In order to facilitate the description of the state of movement of the entity, the following conventions are made:

(1) The three-dimensional coordinate system is adopted, where the X–Y plane is the sea level, and Z represents the height value;
(2) Do not consider the movement of the entity with waves;
(3) Do not consider the entity rolling, only the pitch and partiality of the entity, set the pitch angle α, the angle of yaw β;
(4) The cruise speed vc, acceleration a, angular velocity $v\beta$, and vertical direction vh are constant;
(5) The physical state information sent by the simulation engine includes at least locations (x', y', z'), speeds v' and directions β'.

The movement can be classified into a uniform straight-line motion in the X–Y plane, accelerating (decelerating) straight movement, turning (changing the direction), and climbing (sinking) movement in the direction of Z, in accordance with the aforementioned agreement combined with the entity's movement law. Combining these is possible in Basic mode. It is simple to build a mathematical description of these fundamental sports modes in accordance with the law of movement.

The virtual battlefield environment states first started making predictions based on starting movement speed and motion paths for uniform speed movements. The virtual battlefield's environmental state compares based on the physical status information received and the entity information of its own forecast at each stage of simulation engine. The movement mode adjustment is carried out in accordance with the distinctions between the two, as shown in Fig. 5.

The disparities between received data and predicting data need to be handled properly in order to preserve the three-dimensional scenario display's continuity and smoothness. Traditional smooth treatment procedures cannot be used due to the demands of three-dimensional performance.

Based on the characteristics that the movement state of the entity to be predicted is low, which can only be completed within a number of simulation steps, the predicted data is still utilized at this point in time to drive the performance of the three-dimensional geometric model of the entity even if the simulation engine is much advanced.

The following actions are taken in the next simulation engine advance phase to align the anticipated data with the received data and account for the difference between the present received entity data and the new animation.

(1) According to the time of T, use predicted physical state data St and the receiving physical state data St′ to determine the movement mode in the next step;
(2) Calculate the physical state data at $T + \Delta t$ based on T and St′;

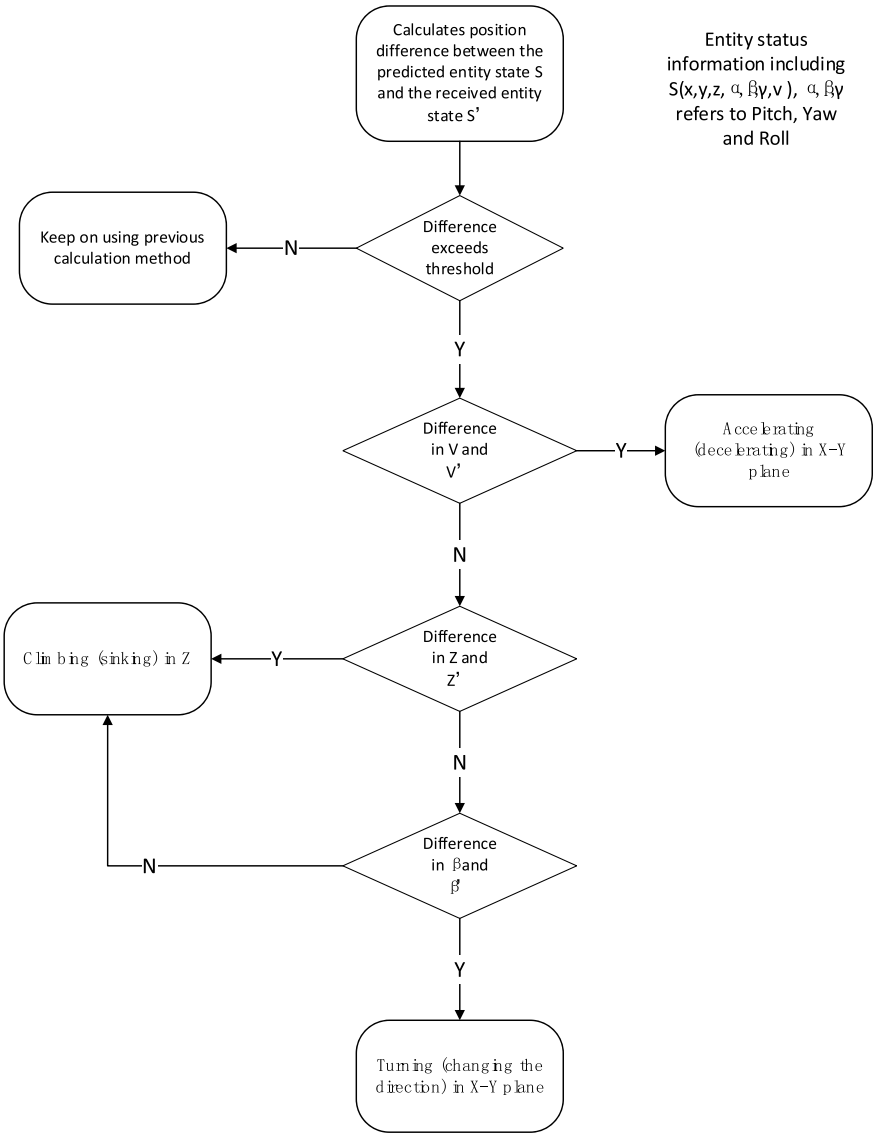

Fig. 5 Movement mode switching

(3) Interpolation calculates the entity state data for each intermediate time point within Δt by St and St + Δt.

According to the above description, the improved DR algorithm can be described as shown in Fig. 6 as follows:

```
S: Entity status in virtual battlefield, S': Entity status received from simulation engine
Do
  Compute (&S, t) for each Entity at the current time.
  Draw(S) for each Entity into the 3D scence
  If the current time is the Simulation Engine time
    GetValue(&S') from RTI
    Compare(S, S') :
     If the difference is beyond the threshold
      SelectNewMoveStyle(difference)
  TimeAdvance(NextTime)
While
```

Fig. 6 Improved DR algorithm for virtual battlefield environments

5 Experimental Verification

We employ the intricate models and their associated algorithms in this paper, using OpenGL on the VC .NET platform, to achieve the trajectory inference of all weapons, as shown in Fig. 7. This allows us to represent the ballistic and trails of missiles/torpedoes and helicopters in a way that is intuitive and understandable.

Fig. 7 Application example of trajectory refinement continuous behavior model

Explanation as follows:

(1) Virtual ballistic display of the vertical plane of anti-ship missiles;
(2) Virtual ballistic display for vertical launch of anti-submarine missile vertical plane;
(3) Virtual ballistic display for the vertical plane of an underwater submarine-launched anti-ship missile;
(4) The virtual ballistic display of the vertical plane of the anti-submarine torpedo on the water;
(5) Virtual track of the helicopter's in horizontal plane;
(6) Virtual track display for helicopter recall in horizontal plane.

The results show, this improved DR algorithm could provide a smooth trajectory inference method to achieve finer grain of trajectory inference in distributed interactive simulation.

References

1. Amit N, Kumar SA, Kumar AL (2021) Learning-based hybrid routing for scalability in software defined networks. Comput Netw 198
2. Damian SQJ, German M, Ignacio B (2021) A cloud framework for problem-based learning on grid computing. J Parallel Distrib Comput
3. Piotr B, Gianluca D, Michal R et al (2021) Unified and standalone monitoring module for NFV/SDN infrastructures. J Netw Comput Appl 175
4. Pulimeno M, Epicoco I, Cafaro M (2021) Distributed mining of time-faded heavy hitters. Inf Sci 545 (prepublish)
5. Wee J, Choi J-G, Pak W (2019) Wildcard fields-based partitioning for fast and scalable packet classification in vehicle-to-everything. Sensors 19(11)
6. Alamri A, Ansari WS, Hassan MM et al (2013) A survey on sensor-cloud: architecture, applications, and approaches. Int J Distrib Sens Netw 2013(2)
7. Jang SM, Yoo JS (2008) An efficient load balancing mechanism in distributed virtual environments. ETRI J 30(4)
8. Nativi S, Mazzetti P, Geller GN (2013) Environmental model access and interoperability: the GEO model web initiative. Environ Modell Softw 39(Jan)
9. Lu PJ, Lai MC, Chang JS (2022) A survey of high-performance interconnection networks in high-performance computer systems. Electronics 11(9)
10. Ha MJ, Kim J, Galloway Peña J et al (2020) Compositional zero-inflated network estimation for microbiome data. BMC Bioinf 21(Suppl 21)