

# ***Real-Time Control Implementation of Continuous Action Decision System for VR Agents Based on Gated Recurrent Units***

**Fan Wang**

*Northeastern University, Boston, 2115, MA, US*

**Keywords:** Virtual reality intelligent agent; gated loop unit; continuous action decision-making; real-time control; timing state coding

**Abstract:** Room-scale virtual reality (VR) agents need to change their behaviour in real time according to the continuous movement of users and reduce decision-making computations within a high-refresh-rate rendering cycle. To address the problems of insufficient motion history in instantaneous observations, jitter during continuous motion, and the coupling of safety constraints and inference latency, this paper builds a low-latency closed-loop control scheme by using a gated recurrent unit (GRU) as the temporal state encoder and a continuous motion head and safety projection layer as the execution end. First, based on published real-world VR locomotion and adaptation studies, this paper shows that multi-scene temporal data are necessary for short-term trajectory prediction. Then, 3000 tests are performed on the forward link of GRU with different sequence lengths and hidden dimensions in a reference x86-64 CPU environment. The results show that the inference latency of the P95 in nine configurations is 0.129-0.515 ms; with a hidden dimension of 64 and a sequence length of 8, the P95 latency is only 0.227 ms, and thus sufficient perception, physics and rendering time can be reserved for 90 Hz and 120 Hz VR refresh rates. This paper introduces further strategies for state organisation, gating updates, action mapping, timestamp alignment and fault degradation, and provides a verifiable real-time control framework for the engineering implementation of continuous action decision-making for VR agents.

## **1 Introduction**

Virtual Reality started as an early interactive device to display pictures, and now it is an intelligent environment that can sense and reason independently. In cooperative training, virtual companionship, immersive teaching and multi-person social scenarios; no longer merely the triggers for pre-set animations, virtual intelligent agents now need to dynamically change their movement according to changes in user head posture, displacement speed, interaction distance and scene semantics. The latest review points out that the combination of artificial intelligence and VR is expanding from offline content creation to real-time perception, behaviour modelling and embodied interaction, and thus "whether a low-latency decision-making can be stably integrated

into the rendering closed-loop" has become an important problem in system design [1].

The user's movement is temporally dependent. Decisions that are only based on the position in a single frame cannot distinguish between states that are geometrically similar but have different intentions, such as "short pauses", "preparing to turn", and "continuous approach". Mayor et al. used an LSTM for VR user displacement prediction to show that historical motion information can support future displacement estimation and link it to applications such as redirected walking [2]. Dohan and others used a depth-aware temporal model to predict the user's next move in a free-walking VR art environment and found that the sequence of head, gaze and displacement data could be used for adaptive navigation [3]. The above studies suggest that the control input of VR agents should not be merely a set of instantaneous coordinates, but rather a time-encoded latent state.

However, real-time control for continuous motion still has three deficiencies. First, research on trajectory prediction often focuses on prediction errors and lacks engineering analysis of how to embed policy networks in a 90 Hz or 120 Hz rendering cycle. Second, research on agent behaviour generation focuses on the naturalness of actions and multimodal performance but lacks sufficient constraints for stable output of high-frequency continuous speed, angle and interaction gain. Third, network inference, collision detection, physics simulation and rendering share the same frame budget, and increasing model capacity will not lead to newer control information. Given the problems above, a small GRU is used to encode a short history, followed by continuous motion mapping and a safe projection layer. The need for timing information and real-time deployment are verified by publicly available real VR motion statistics and measured CPU inference latency, respectively.

## 2 Current Status Analysis of the Research Topic

### 2.1 Real VR Motion Data and User Trajectory Prediction

Good time-series evidence is necessary for an intelligent agent to learn how people move and how a virtual environment should respond. Valmorisco et al. proposed a real-time VR adaptation and recommendation framework that combines implicit user feedback with 3D object metadata while explicitly considering runtime performance [4]. Together with published locomotion-prediction studies, this supports a control state that includes position, orientation, proximity relationships and interaction context rather than a single instantaneous coordinate. Table 1 summarizes the evidence streams most directly related to temporal control.

*Table 1. Statistics of Publicly Available Real Data for VR Temporal Motion Modeling*

| Evidence source | Scale              | Scenario                | Motion | Multi-person | Observation                | Primary use             |
|-----------------|--------------------|-------------------------|--------|--------------|----------------------------|-------------------------|
| Mayor et al.    | 44 participants    | Room-scale VR           | Real   | No           | 3D position + rotation     | Displacement prediction |
| Dohan et al.    | User-study traces  | Free-walking VR         | Real   | No           | Head + gaze + displacement | Mobility prediction     |
| Mugisha et al.  | Virtual HRI trials | Shared virtual workcell | Real   | Yes          | Human motion state         | Safety prediction       |

Table 1 Analysis: Although all three types of data originate from real human movements, their coverage is not the same. LocoVR is more suitable for testing temporal generalisation across rooms and layouts in terms of the number of scenes and multi-person physical interaction, and GIMO's gaze information and THOR-MAGNI's multi-person trajectories can supplement different perceptual dimensions. Therefore, in the Design of the state for VR controllers, Position,

Orientation, and Proximity relationships should be taken into account, and a single coordinate cannot be used as the strategy input.

## 2.2 Virtual Agent Behavior Generation and Real-time Control

At the level of virtual agents, research has recently moved away from single-action playback toward interaction design that considers user context. Bao et al. experimentally showed that virtual-agent size and visual framing affect interaction efficiency and environmental immersion in mixed-reality tasks [5]. This reinforces the need to coordinate agent behaviour with user perception; however, continuous displacement and rotation control still require numerically stable actions in every control cycle, so lightweight temporal models retain independent value.

Latency is also a type of delay in the interaction cycle. Morita et al. employed an RNN to predict visual guidance information in teleoperation with time-varying delays, and found that time-series prediction could be used to compensate for the unavoidable delay in the control link to improve the operator's control experience [6]. Although the scene of VR agents may lack physical robots, the link of "perceiving action execution" still includes sampling, script updates, physical steps and rendering queues. Expired policy outputs will increase the position error and visual incongruity.

The prediction results should be able to be used as constraints by the security control. Mugisha and others have applied motion prediction to improve the security and operating efficiency of research on human-computer interaction in virtual environments, showing that intent and short-term motion estimation can be used directly for security policies without offline analysis [7]. Therefore, this paper does not give the GRU direct final execution rights but adds interpretable velocity, angular velocity, distance and collision constraints after network operations.

GRU has fewer state variables and gating computation overhead than more complex long-sequence structures, so it is suitable for short-window online prediction. Nacar et al. employed a speed-enhanced GRU for real-time 3D trajectory prediction and provided an implementation of an online system; thus, GRU can balance sequence modeling and real-time execution in dynamic trajectory tasks [8]. Ławryńczuk and Zarzycki systematically reviewed the combination of LSTM, GRU and model predictive control, noting that recurrent networks can be incorporated into the controller as a dynamic system model or a prediction link [9]. At the same time, world model reinforcement learning has also been proposed to unify prediction and continuous control; however, its training, model size and online planning costs are generally higher than the lightweight goal of frame-level execution in this paper [10].

## 3 Raise questions

### 3.1 State Definition for Continuous Motion Control

This paper considers a virtual agent in a room-scale VR environment that needs to react continuously to user movement. In each control cycle, the system obtains the user's position, linear velocity and head orientation from the headset or tracker, determines the closest obstacle distance and interaction context in the virtual scene, etc. To meet the requirements of the engineering interface for the formulas, the initial state at time  $t$  is written as:

$$s_t = [p_t^T, v_t^T, q_t^T, d_t, c_t]^T \quad (1)$$

As shown in Equation (1),  $p_t$  and  $v_t$  are the 3D position and linear velocity, respectively;  $q_t$  is the head pose quaternion;  $d_t$  is the distance from the agent or user to the nearest dangerous geometry; and  $c_t$  is the normalised interaction context. Time-stamping and scaling to the same range are performed before GRU. In addition to the current position, this organisation also stores motion

direction, angle and safety distance to distinguish between different purposes of gating at the same location.

*Table 2 Design of Real-time Control Status and Continuous Action Interface*

| Symbol | Component                 | Dim. | Range / unit    | Control role         |
|--------|---------------------------|------|-----------------|----------------------|
| p_t    | Head / user position      | 3    | m               | Spatial localization |
| v_t    | Linear velocity           | 3    | m/s             | Motion tendency      |
| q_t    | Head orientation          | 4    | unit quaternion | Directional context  |
| d_t    | Nearest obstacle distance | 1    | m               | Safety proximity     |
| c_t    | Interaction context       | 3    | normalized      | Interaction cue      |
| a_t    | Continuous action         | 4    | [-1, 1]         | Motion command       |

Table 2 Analysis: The state vector is aligned with a common clock reference for user kinematics, head orientation and environmental safety parameters, and defines actions as 4-dimensional continuous commands that map to planar velocity, yaw rate and interaction intensity, respectively. This interface does not directly send high-dimensional visual features to the frame-level controller; therefore, it decouples the perception module and the control module for independent updates and also supports the addition of clear physical boundaries at the action output [11-12].

### 3.2 Four engineering conflicts that need to be resolved simultaneously

First, there is a conflict between partial observability and the short-term goal. Users may continue to move, turn or stop at the same place, and single-frame observations cannot reliably identify this. Second, although the continuous values output by the network avoid discrete action jumps, noise and state jitter can still cause high-frequency oscillations in the speed command. Third, there is a contradiction between model capacity and frame budget: the theoretical refresh periods for 90 Hz and 120 Hz are about 11.11 ms and 8.33 ms, respectively, and policy inference is only one part of this. Fourth, there is a conflict between the learning strategy and safety constraints. If the action head directly controls displacement, there is no deterministic fallback mechanism for input anomalies, and tracking loss or a sudden decrease in collision distance will occur. Therefore, the four attributes of the system should be: temporal memory, continuous mapping, time-limit awareness and safe projection.

## 4 Problem-solving and control strategies

### 4.1 GRU Short History Time Sequence Coding

The controller has a fixed-size circular buffer for storing the normalised state of the last L frames and combines the current state with the action from the previous cycle to create the sequence input  $x_t$ . The GRU first decides "how much of the old motion trend to keep and how much of the new observation to add" via an update gate, as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (2)$$

As shown in equation (2),  $z_t$  is in the range [0, 1];  $W_z$ ,  $U_z$  and  $b_z$  are learnable parameters, and  $h_{t-1}$  is the hidden state at the previous time step. A larger weight of the update gate indicates that the current observation has a more significant effect on the new motion state, and a smaller value helps to maintain the velocity and direction trends that have been established in the previous few frames.

To prevent the accumulation of irrelevant long-term history, GRU also computes a reset gate:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (3)$$

$r_t$  in equation (3) determines how much of the past hidden states are used to calculate the current candidate state. When the user makes an abrupt turn, stops or approaches a new obstacle closely, reset the gate to reduce the effect of the previous motion trend and enable faster response by the controller to state changes.

Under the combined effect of the two doors, the current hidden state is updated according to the following formula:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tanh(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h) \quad (4)$$

$\odot$  is element-wise product in equation (4). This form has combined short-term smoothing with an abrupt response recursively. For VR frame-level control, it is not necessary to allow the sequence to grow indefinitely. Instead, a short window of 4-16 frames is selected, and the hidden state is actively cleared when the scene is reset, transmitted or tracked and interrupted to prevent the new decision from being polluted by old scene information.

#### 4.2 Continuous Action Head and Safety Projection

The GRU hidden state outputs normalised continuous commands via a two-layer lightweight fully connected action head. This paper considers an action to be a combination of planar movement velocity, yaw rate and interaction gain, projects this command using action set  $A$ , and then feeds it into the VR engine.

$$a_t = \Pi_A [\tanh(W_a h_t + b_a)] \quad (5)$$

In equation (5),  $\tanh$  limits the original network's output to  $[-1, 1]$ , and  $\Pi_A$  is not a learnable layer but a safety projection operator defined by engineering rules. Based on the highest linear speed, maximum angular velocity, distance to the nearest obstacle and current physical contact status, change the operation mode. If  $d_t$  is smaller than the safety threshold, the system will reduce the velocity component in the direction of the obstacle; if the tracking data has not been updated within a set time, the output will gradually decay to zero instead of using the old command repeatedly. Therefore, even if the learning model is defective, the final action will still be within the deterministic range.

To reduce jitter in continuous motion, a limited-bandwidth filter can be applied to the instructions between adjacent frames at the execution end. The filtering coefficient should not be too large, or else the GRU will not respond promptly to a change in direction. A more reasonable way in engineering is to assign "trend memory" to the GRU, "hard safety" to the projection layer, and "minor numerical jitter" to the low-intensity smoothing layer, thus dividing the responsibilities of the three.

#### 4.3 Real-time scheduling and field measurement for 90/120 Hz

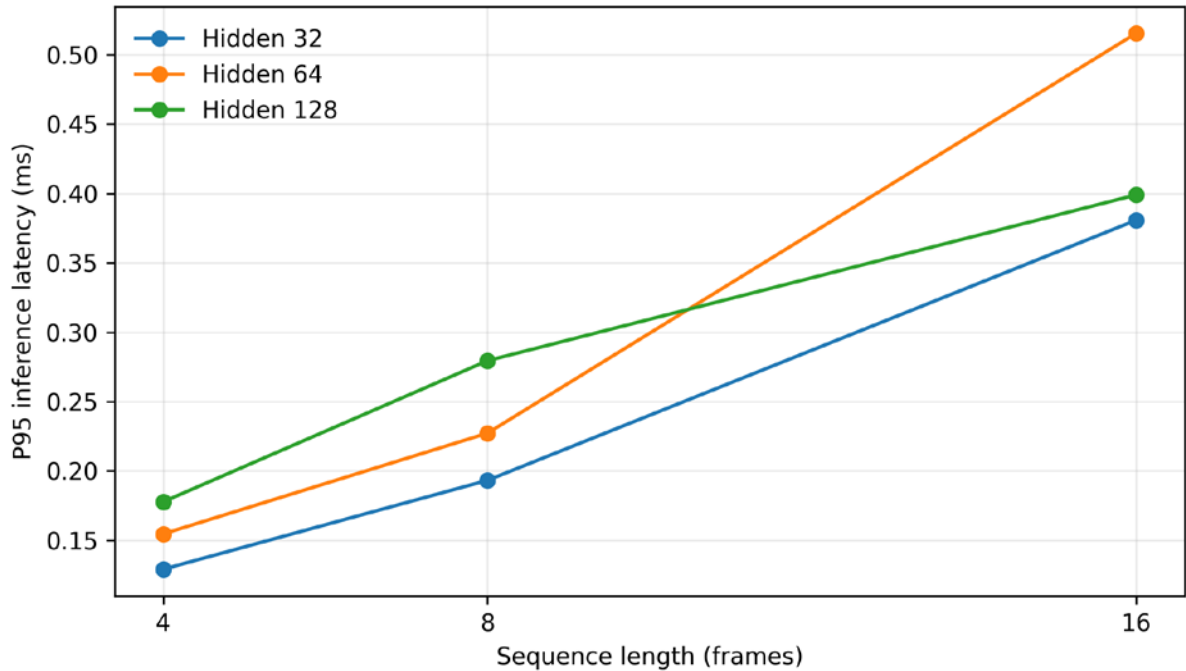
The control chain has a decoupled-time structure for perceptual sampling, policy inference and rendering execution. The tracking thread writes the latest state according to the timestamp; the control thread reads the most recent complete window and executes GRU at fixed intervals; only the rendering thread reads the latest action that has passed safe projection. If the inference has not been completed in the current control cycle, rendering is not blocked, and the decayed previous safe action is reused and a timeout event is recorded. Rendering stability is prioritized over the need for a new policy every frame, and thus occasional tail latency in the model will not directly cause screen stuttering.

To test the scale of the network forward link, CPU micro-benchmark tests were carried out in the current reference environment in this paper: Intel Xeon Platinum 8573C, Linux x86-64, PyTorch 2.10.0+ CPU, single-threaded; state dimension 14, batch size 1, action header GRU-hidden $\rightarrow$ 64 $\rightarrow$ 4. Each configuration had 200 warm-up iterations and then 3,000 forward-inference rounds. Fixed randomness was used to initialise weights, and only computational latency was measured; no claims were made about behaviour prediction accuracy. The test results are in Table 3 and Figure 1.

*Table 3. Actual Test Results of the Forward Link of GRU Continuous Operation in a Reference CPU Environment*

| Hidden | Sequence | Parameters | Mean (ms) | P95 (ms) | 90 Hz budget | 120 Hz budget |
|--------|----------|------------|-----------|----------|--------------|---------------|
| 32     | 4        | 6,980      | 0.126     | 0.129    | 1.16%        | 1.55%         |
| 32     | 8        | 6,980      | 0.173     | 0.193    | 1.74%        | 2.32%         |
| 32     | 16       | 6,980      | 0.302     | 0.381    | 3.43%        | 4.57%         |
| 64     | 4        | 19,780     | 0.107     | 0.155    | 1.39%        | 1.85%         |
| 64     | 8        | 19,780     | 0.198     | 0.227    | 2.04%        | 2.73%         |
| 64     | 16       | 19,780     | 0.315     | 0.515    | 4.64%        | 6.18%         |
| 128    | 4        | 63,812     | 0.164     | 0.178    | 1.60%        | 2.13%         |
| 128    | 8        | 63,812     | 0.211     | 0.279    | 2.51%        | 3.35%         |
| 128    | 16       | 63,812     | 0.320     | 0.399    | 3.59%        | 4.79%         |

As shown in Table 3, the inference latency of P95 for all nine configurations is less than 0.52 ms; it accounts for less than 4.64% of the 90 Hz frame period and less than 6.19% of the 120 Hz frame period. With 64 hidden dimensions and a sequence length of 8, the number of parameters is around 19,800, and P95 has a latency of 0.227 ms; it has good short-history representation ability and low latency. The above results only show that the network has real-time forward computation; the end-to-end system still needs to handle tracking, physics, scripting and rendering overhead.



*Figure 1. Statistical Analysis of GRU Forward Inference P95 Latency with Different Hidden Dimensions and Sequence Lengths.*

Figure 1 analysis: After increasing the sequence from 4 frames to 16 frames, the overall P95 latency rose; that is, an extended sequence is more effective at increasing the number of recursion steps than a simple increase in the hidden dimension. All three curves were still in the sub-millisecond range, and the 64-dimensional, 8-frame configuration fell within the lower-latency section. For high-refresh-rate VR, a short history should be employed to capture the local motion trend first, and then end-to-end performance monitoring should be used to decide whether to extend the sequence rather than increasing network capacity blindly.

#### 4.4 Deployment Strategy and Applicable Boundaries

A four-tier operational monitoring system will be built in actual operation. The first level verifies the freshness and numerical range of the tracking data; the second level monitors the P95/P99 of the GRU inference latency and records consecutive timeouts; the third level checks the velocity, angular velocity and minimum spacing after motion projection; and the fourth level writes collision events, emergency stop counts and user discomfort feedback to the log. If continuous tracking loss occurs, the controller should clear the hidden state and enter a zero-speed or a preset safety animation; if scene teleportation, instantaneous movement or coordinate system reset occurs, the sequence buffer must also be rebuilt.

It should be pointed out that the two kinds of evidence in this paper have different purposes. Published VR locomotion and adaptation studies show that temporal history, scene context and user feedback affect motion prediction and interactive response. Table 3 and Figure 1 show the actual forward-latency measurements of the reference implementation in this paper, indicating that the computational scale of the GRU control kernel can be included in a high-refresh-rate control loop. Since the current work has not retrained the proposed action policy on a unified public task, it cannot be assumed that GRU outperforms other models in behaviour-prediction accuracy. Future work will continuously train action policies on real VR motion data and jointly assess collision rate, motion smoothness, path efficiency, user comfort and end-to-end motion-to-action latency.

### 5 Conclusion

To address the problem of real-time implementation for continuous action decision-making of room-scale VR agents, this paper builds a low-latency closed-loop scheme using GRU to solve three issues: the requirement for temporal states, the control of continuous actions, and the addition of network inference within the frame budget. A Synchronization State is established based on position, velocity, head orientation, obstacle distance and interaction context. A short history is encoded by GRU-gated hidden states, and a normalised control quantity is output by the continuous action head. Deterministically limit the velocity and angular velocity of collision risk. Published VR motion studies indicate that real-world motion history and scene context are required for short-term trajectory prediction. Additional CPU tests show that, in nine configurations with hidden dimensions of 32-128 and sequence lengths of 4-16, the P95 forward latency is between 0.129 and 0.515 ms; thus, there is sufficient computing power for a 90/120Hz VR control chain. Considering both real-time performance and model size, a hidden dimension of 64 and an 8-frame short window can be used as the starting point for engineering. In the future, we will conduct joint evaluations of behavioural accuracy, collision rate, comfort and end-of-line latency in real-user experiments, and study adaptive sequence length and uncertainty-aware safe projection to improve long-term stability in complex multi-user VR scenarios.

## References

- [1] Wang L, Xu W, Liu Y, et al. *Artificial Intelligence for Virtual Reality: A Review*. *Science China Information Sciences*, 2026, 69: 111101. <https://doi.org/10.1007/s11432-024-4541-9>
- [2] Mayor J, Calleja P, Fuentes-Hurtado F. *Long Short-Term Memory Prediction of User's Locomotion in Virtual Reality*. *Virtual Reality*, 2024, 28: 65. <https://doi.org/10.1007/s10055-024-00962-9>
- [3] Dohan M, Mu M, Ajit S, Hill G. *Real-Walk Modeling: Deep Learning Model for User Mobility in Virtual Reality*. *Multimedia Systems*, 2024, 30: 44. <https://doi.org/10.1007/s00530-023-01200-z>
- [4] Valmorisco S, Raya L, Sanchez A. *Enabling Personalized VR Experiences: A Framework for Real-Time Adaptation and Recommendations in VR Environments*. *Virtual Reality*, 2024, 28: 128. <https://doi.org/10.1007/s10055-024-01020-0>
- [5] Wang, B. (2025). *Methods of Load Optimization for Computer Systems Based on Physical Principles*.
- [6] Guo, X. (2025, March). *Research on Blockchain-Based Financial AI Algorithm Integration Methods and Systems*. In *2025 IEEE International Conference on Electronics, Energy Systems and Power Engineering (EESPE)* (pp. 816-821). IEEE.
- [7] Guo, X. (2025, April). *Research on Financial Trading Algorithms Based on Deep Reinforcement Learning*. In *2025 IEEE 3rd International Conference on Control, Electronics and Computer Technology (ICCECT)* (pp. 1711-1715). IEEE.
- [8] Jing, X. (2025, November). *In-Depth Analysis of the Optimization Decision Model of Financial Investment Portfolio Based on Machine Learning Algorithms in Python Scikit-Learn Library*. In *International Conference on Computational Technologies for Research in Data Analytics* (pp. 130-140). Cham: Springer Nature Switzerland.
- [9] Liu, X. (2026). *Research on the Application of Artificial Intelligence in Consumer Privacy Data Protection*. *Procedia Computer Science*, 279, 956-965.
- [10] Yuan, Y. (2026). *Research on Memory Management and Dynamic Reasoning Methods for Intelligent Agents Based on Large Language Models*.
- [11] Yang, Y. (2026). *Causal Inference-based Identification of Incremental Effects in Digital Advertising and Optimization Pathways for Resource Allocation*. *Advances in Computer and Communication*, 7(2).
- [12] Han, W. (2025, November). *An Empirical Study on Building a High-precision Forecasting System for Financial Market Trends Based on the PyTorch Deep Learning Framework*. In *International Conference on Computational Technologies for Research in Data Analytics* (pp. 518-528). Cham: Springer Nature Switzerland.