ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20250212002363
Original scientific paper

Intelligent Simulation System of Human-Computer Interaction Motion Based on Bullet Physics Engine

Weiqiong ZHAO*, Fuchuan YE

Abstracts: With the rapid advancement in computer technology, human motion simulation has gained substantial attention across various domains; however, existing simulation models frequently encounter challenges such as poor controllability, inadequate stability, and limited anti-interference capability. To address these limitations, this study introduces an intelligent human-computer interaction (HCI) motion simulation system based on the Bullet physics engine and reinforcement learning. The proposed system leverages the proximal policy optimization (PPO) algorithm, enhanced with a multi-objective reward function designed to precisely constrain policy updates through the comparative analysis of new and old strategies. Experimental evaluations reveal significant performance improvements, with the PPO algorithm achieving maximum reward values notably higher (by 43.5 and 307.2) than comparative algorithms, and demonstrating a minimum loss error reduction of up to 0.059. The model's stability and anti-interference capabilities were further validated under challenging scenarios involving external forces and obstacles, showing quicker recovery and superior resilience. Additionally, the model demonstrated a high correlation with real human joint movements, with fitting accuracies of 91.4% for knee joints and 93.2% for hip joints, highlighting its effectiveness and practical applicability. As a result, the study effectively improves the stability and anti-interference of the human simulation model and realizes the accurate control of the model.

Keywords: bullet engine; human-computer interaction; motion control; reinforcement learning; simulation robotics

1 INTRODUCTION

Human-computer interaction (HCI) motion simulation is a research field arising from the intersection of physics, biomechanics, and computer science, in which relevant computational models are built to simulate the real physical motion of the human body under constraints [1]. Compared with traditional human animation, human motion simulation requires the synthesized human motion model to satisfy Newton's laws. Therefore, human motion simulation can be applied in entertainment industries such as movie special effects production, games, and animation to enhance the smoothness and visual effects of character movements [2]. It is also able to be applied in the operational training of physical tasks, such as military and sports training, which can enhance the training effect and guarantee the safety of personnel [3]. Existing human simulation models are mainly constructed through two modeling approaches: forward dynamics and inverse dynamics. Forward dynamics is mainly used for modeling and predictive analysis of the system to predict the behavior and performance of the system through causal reasoning [4]. Inverse dynamics is mainly used for the control analysis of the system to solve the related control problems through input-output relationships [5]. However, the models constructed by these two modeling methods have poor adaptive ability in the face of new environments and are unable to cope with unexpected situations, whereas machine learning is able to gradually optimize the adaptive ability of the human simulation model by working in the ever-changing environment [6]. Therefore, the study proposes an intelligent human-computer interaction motion simulation system based on Bullet physics engine and reinforcement learning. The system innovatively adopts reinforcement learning to improve the control algorithm, constructs a multi-objective reward function for model optimization, uses the Proximal Policy Optimization (PPO) algorithm to train the simulation model, uses the difference between the old and the new policies to constrain the policy update amplitude, performs a quaternion conversion on the Euler angle data, constructs a simplified simulation model for human-machine interaction and builds the related simulation system.

2 LITERATURE REVIEW

Human motion control and simulation has achieved a large number of results with the research of many scholars. Alvarado et al. proposed a flexible character upper body interaction model for the sake of flexibility problem in virtual character motion simulation. The model employed an adversarial controller to handle character obstacle avoidance, combining keyframe sequences with motion constraints and lightweight physics to adjust the stiffness and reaction time of the character. Experiments demonstrated that the model provided accurate character movement control and was able to seamlessly adapt to changes in the surrounding environment [7]. Yang et al. proposed a new somatosensory simulation control algorithm in an attempt to further enhance the realism and experience of virtual reality (VR) somatosensory games. The algorithm simulation control integrated vestibular perception technology, accelerated the feedback speed of the output error, continuously corrected and predicted the input angular velocity and acceleration, and realized the prediction of terrain fluctuation in animation. Experiments demonstrated that the algorithm could support phase delay, thus effectively improving the action consistency of motion simulation [8]. To increase the accuracy of motion control for a two-joint planar robot, Halilu et al. suggested a new non-negative parameter t in the conjugate gradient algorithm. The general nonlinear system of equations and the constrained monotone system of equations were solved using the conjugate gradient algorithm. Experiments indicated that the method could effectively improve the motion control accuracy of a two-joint planar robot [9].

Chen et al. proposed a new bionic mantis shrimp robot in order to realize the exploration of rugged undersea environment. The robot employed 10 rigid and flexible feet for swimming. The motion gait planning of the robot was accomplished by the motion trajectory of the mantis shrimp, and a generative controller applicable to the coupled motion of the multipod was proposed. Experiments indicated that the robot was able to swim at a speed of 0.46 body length/s with a minimum turning radius of 0.36 m [10]. A novel two-stage control technique for motion planning and robotic system control in various situations was put forth by Prasad et al. The method addressed the obstacle avoidance problem of the robot by using minimum distance computation so that the robot avoids the obstacle at the nearest point, and using the Lyapunov direct method to ensure the convergence of the two-body and the stability of the mechanical system. Experiments demonstrated that the method had a high degree of nonlinearity and a constant control law [11].

Aiming at the HCI simulation problem, Guo et al. proposed a product design model based on VR technology in order to enhance the HCI experience of product design with VR technology. The discrete mass points in this model adopted the triangular indication products, and the dynamics was analyzed in the mass point system and solved by Euler's method. Experiments indicated that the model had a high degree of operational convenience, and at the same time could effectively enhance the human-computer motion interaction [12]. Gao et al. proposed a joint vector-based plain Bayesian gesture recognition algorithm in an attempt to enhance the experience of immersive somatic interactive games. The algorithm employed a decision support system to represent skeletal and motor features as well as joint vectors, and introduced an adaptive strategy to optimize the plain Bayesian algorithm. Experiments demonstrated that the algorithm required fewer training samples and significantly improved the user's HCI experience [13]. Murray-Smith et al. proposed a HCI method based on optimal feedback control in an effort to improve the recognition accuracy of HCI motion. The state of the simulation system in this method was controlled by the user through muscle signals, and the control noise observable noise was optimized. Experiments demonstrated that the method could effectively understand and simulate the motion state of the human body at all moments [14].

To summarize, existing studies have explored the research on physics-based motion control from various aspects such as virtual character control, robot motion control, HCI, etc., and a large number of research results have been achieved. However, the existing methods also suffer from the processing capability of high latency inputs, low computational efficiency, poor stability and immunity to interference in complex terrain, and insufficient real-time performance. Therefore, a HCI motion simulation system based on Bullet physics engine is proposed. It innovatively adopts reinforcement learning to improve the control algorithm, constructs a multi-objective reward function, uses the PPO algorithm to train the simulation model, utilizes the difference between the old and the new strategies to constrain the strategy update amplitude, performs quaternion conversion on the Euler angle data, and establishes a relevant simulation system. Bullet physics engine has strong real-time physics simulation capability, supports real-time interaction in complex scenes, and provides rich physics simulation functions to flexibly adapt to different research needs. PPO algorithm is able to adapt to dynamic changes in complex environments by restricting the magnitude of the policy update, and its multi-objective reward function design can balance different optimization objectives. The research aims to further improve the stability and anti-interference of the human simulation motion model, and improve the experience of HCI.

$$R_1 = \theta \exp\left(-\frac{\left\| \bar{m} - m \right\|}{\omega} \right) \tag{1}$$

In Eq. (1), R_1 denotes the reward function of the model and the reference position. θ and ω both denote a random constant. $\bar{m}$ denotes the reference position data. m denotes the simulation model data. Reference Position Reward Function By optimizing the reference position reward, the model is able to better learn the details of the reference action and reduce the deviation of the action, thus improving the overall action fidelity. The reward function for the model linear velocity to reach the target is calculated as shown in Eq. (2).

$$R_2 = \theta \exp\left(-\frac{\left\| \bar{v} - v \right\|}{\omega} \right) \tag{2}$$

In Eq. (2), R_2 denotes the linear velocity attainment bonus value. $\bar{v}$ denotes the simulation model joint linear velocity. v denotes the simulation model linear velocity. The linear velocity reward function ensures that the model maintains a stable velocity during motion, avoiding unstable or unnatural movements caused by too fast or too slow speeds. The reward function for model angular velocity to reach the target is shown in Eq. (3).

$$R_3 = \theta \exp\left(-\frac{\left\| \bar{a} - a \right\|}{\omega} \right) \tag{3}$$

In Eq. (3), R_3 denotes the angular velocity attainment reward value (RV). $\bar{a}$ denotes the simulation model joint angular velocity. a denotes the simulation model angular velocity. Angular Velocity Reward Function: By maximizing the angular velocity reward, the model adjusts its own rotational action so that the angular velocity is as close as possible to the target value, ensuring that the model remains stable during the rotational action and avoids imbalance in the action due to a mismatch of rotational velocities. The reward function of the model

base node to the support surface is calculated as shown in Eq. (4).

$$R_4 = \theta \exp\left(-\frac{\left\|\bar{k}-k\right\|}{\omega}\right) \tag{4}$$

In Eq. (4), R_4 denotes the RV from the base node of the model to the support surface. $\bar{k}$ denotes the distance between the root node of the simulation model and the ground. k denotes the distance between the base node and the support surface at the reference location. The reward function from the base node to the support surface passes through the reward value and the model adjusts its pose so that the distance between the root node and the ground is as close as possible to the reference value. The total reward function of the simulation model is calculated as shown in Eq. (5) [19].

$$R = w_1 R_1 + w_2 R_2 + w_3 R_3 + w_4 R_4 \tag{5}$$

In Eq. (5), R denotes the total reward function of the simulation model. w_1 denotes the weight of the reference position. w_2 denotes the weight of linear velocity. w_3 denotes the weight of angular velocity. w_4 denotes the weight of the base node to the support surface. In reinforcement learning, the convergence and stability of the algorithm is closely related to the choice of the learning rate, which expresses the degree of modulation of the model parameters by affecting the update step size of the parameters at each step. A larger model learning rate allows the model to converge quickly, but may also lead to excessive fluctuations or even divergence. Smaller learning rates are stable but may lead to slow convergence [20]. Therefore, the learning rate needs to be a trade-off between convergence and overshooting. The study uses the PPO algorithm to train the model, which is used to determine the learning rate update of the model. Fig. 1 depicts the PPO algorithm's structure.

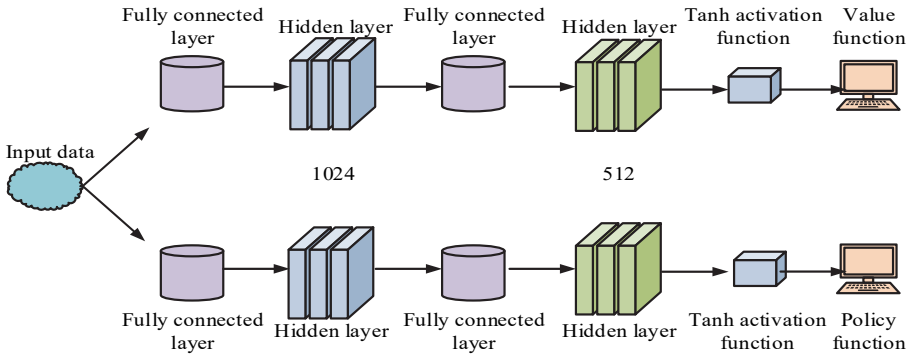

Figure 1 Specific structure of PPO algorithm

In Fig. 1, the structure of the PPO algorithm is mainly composed of two parts, in which the output of the value function includes one input layer, three fully connected layers, two hidden layers (HLs), and one activation function (AF) layer. The structure of the output of the strategy function is the same as that of the value function, the AF of both parts is Tanh function, and the number of neurons in the HLs are 1024 and 512. To guarantee training stability, the algorithm measures the difference between the old and new strategies, so constraining the amount of the strategy update. *KL* scatter is used to quantify the differences between the old and new techniques. The *KL* scatter value is calculated as shown in Eq. (6).

$$d = \hat{E}_t\left[KL\left(Q_o\left(\cdot|s_t\right), Q_n\left(\cdot|s_t\right)\right)\right] \tag{6}$$

In Eq. (6), d denotes the *KL* scatter value. $\hat{E}_t$ denotes the expected value estimate for time step t. Q_o denotes the old strategy. s_t denotes the state. Q_n denotes the new strategy. The study adjusts the penalty coefficient by calculating the magnitude of the scatter value d in relation to the preset scatter value η. When d < η/1.5, the penalty coefficient is 1/2 of the initial. When $d > 1.5\eta$, the penalty coefficient is 2 times of the initial, otherwise the penalty coefficient is unchanged. The optimization flow of PPO algorithm is shown in Fig. 2.

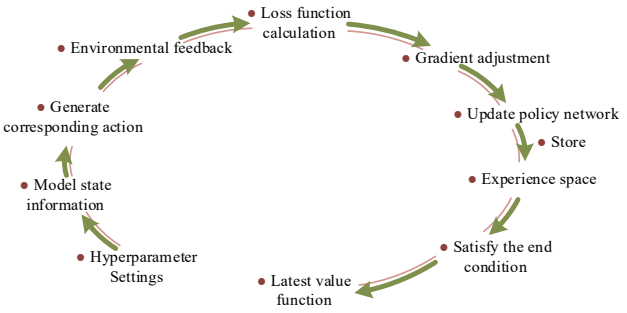

Figure 2 Specific optimization process of PPO algorithm

In Fig. 2, relevant hyperparameters such as learning rate and discount factor are set first. The input of the algorithm is the state information of the model. At each time step the corresponding action is generated according to the current strategy, and the loss function is calculated based on the environmental feedback. The gradient is adjusted according to the ratio of the new strategy to the old one, and then the strategy network is updated. The updated relevant data is stored into the experience space. The strategy network is updated using the most recent

value function until the algorithm converges or reaches the maximum number of iterations.

3 SIMULATION MOTION MODEL CONSTRUCTION AND CONTROL SIMULATION SYSTEM DESIGN

Existing physics engines perform poorly in terms of scalability and real-time performance, while the Bullet physics engine is highly optimized to achieve real-time simulation and supports cross-platform construction and testing of models. Therefore, the study uses Bullet physics engine to build a simulation experiment system. The human body consists of more than 200 bones and 78 freely rotating joints, which can accomplish most of the more complex movements. Therefore, to realize human motion simulation, it is necessary to collect human motion data and apply it to the simulation model [21]. Human data is acquired using the Kinect depth camera, which captures objects in 3D space through infrared radiation and calculates data for each major joint. Human motion data captured by the Kinect device is usually stored using Euler angles. Euler angles consist of the chapter motion angle, the rotation angle, and the spin angle, which represent the angle at which the object rotates around the *z*-axis, *x*-axis, and *y*-axis of the reference line, respectively. The orientation of a rigid body can be uniquely determined from these three angles [22]. The flow of human motion data acquisition by Kinect device is shown in Fig. 3.

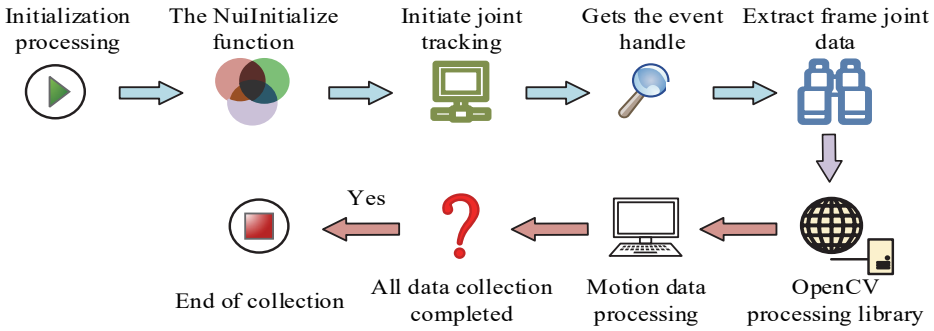

Figure 3 Specific process of collecting human motion data by kinect device

In Fig. 3, the NuiInitialize function is used to initialize the Kinect device, start the joint tracking function, and obtain the joint tracking event handles, which is the key step to collect human motion data. After the relevant settings are completed, the function is called to obtain the joint data of each frame, including the shoulder joint, elbow joint, wrist joint, hip joint, knee joint, and other important joint parts. The OpenCV image processing library is used to process the motion data after the data acquisition is completed, to judge whether the data acquisition of all joints is completed or not, and to end the acquisition when it is completed. The data in the form of Euler angles has some limitations. When the rotation of the chapter motion angle is 180 degrees, it is impossible to distinguish the rotation angle and the rotation angle, this phenomenon is called the deadlock of the universal joint. In order to avoid this situation, the study transforms the data and utilizes quaternions to represent the orientation of the rigid body. The quaternions are also more efficient than the rotation matrix consisting of Euler angles and provide smooth interpolation. The expression of quaternion is shown in Eq. (7), [23].

$$q = (a, b, c, w) \tag{7}$$

In Eq. (7), q denotes the quaternion. a denotes the coordinates of the vector (COV) in the *x*-axis. b denotes the COV in the *y*-axis. c denotes the COV in the *z*-axis. w denotes the angle of rotation. When an object is rotated in space according to a certain angle, the corresponding quaternion expression is shown in Eq. (8) [24].

$$\begin{cases} a = \cos\left(\theta_1/2\right)\cos\left(\theta_2/2\right)\sin\left(\theta_3/2\right) + \sin\left(\theta_1/2\right)\sin\left(\theta_2/2\right)\cos\left(\theta_3/2\right) \\ b = \cos\left(\theta_1/2\right)\sin\left(\theta_2/2\right)\sin\left(\theta_3/2\right) + \sin\left(\theta_1/2\right)\cos\left(\theta_2/2\right)\cos\left(\theta_3/2\right) \\ c = -\sin\left(\theta_1/2\right)\cos\left(\theta_2/2\right)\sin\left(\theta_3/2\right) + \cos\left(\theta_1/2\right)\sin\left(\theta_2/2\right)\cos\left(\theta_3/2\right) \\ w = -\sin\left(\theta_1/2\right)\sin\left(\theta_2/2\right)\sin\left(\theta_3/2\right) + \cos\left(\theta_1/2\right)\cos\left(\theta_2/2\right)\cos\left(\theta_3/2\right) \end{cases} \tag{8}$$

In Eq. (8), θ_1 denotes the angle of rotation of the object (ROO) around the *x*-axis. θ_2 denotes the angle of ROO around the *y*-axis. θ_3 denotes the angle of ROO around the *z*-axis. Substituting the above equation into Eq. (7), the quaternion expression of the object as it rotates can be obtained. The study converts all the major joint Euler angle data collected into quaternion form and saves the data according to different skeletal layers. After the data collection is completed, the study uses the Bullet physics engine to construct a simulated human model. To reduce the modeling difficulty and subsequent computational complexity, the study ignores the bones that are less

relevant to the experiment. Moreover, connecting rods are used to represent bones, and points are used to represent joints. The simplified human body simulation model is shown in Fig. 4.

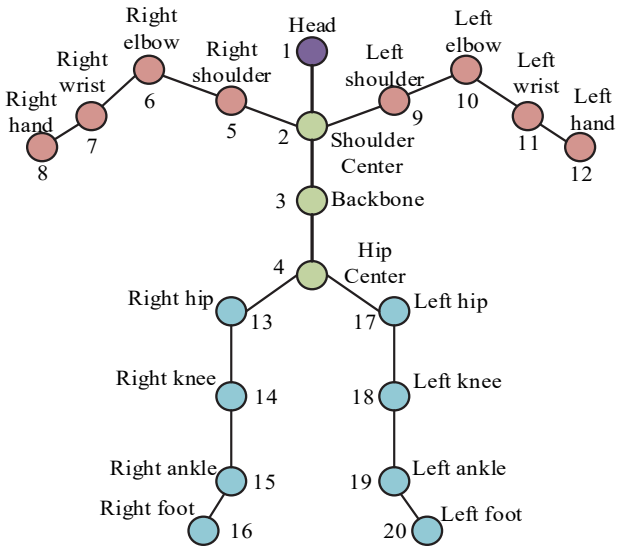

Figure 4 Simplified human simulation model

In Fig. 4, the head of the simulation model is not involved in the relevant tests, so it is simplified to one point. The simplified model has a total of 20 nodes. Among them, 3 nodes in the torso area include the shoulder center, spine, and hip center. The hand has a total of 8 nodes and the leg has the same 8 nodes. The simplified model is able to fulfill most of the motion requirements. Simplifying the human simulation model can reduce the workload and the complexity of the skeletal system; without simplification, the excessive number of bones and joints in the human body will generate a great amount of computation. The simplified model is able to perform most of the motion simulation without affecting the experimental results. The Bullet physics engine contains a number of important components consisting of linear math libraries, memory and containers, collision detection, rigid-body dynamics, soft-body dynamics, constraint definitions, and more. It is also capable of multi-threaded computation to enhance parallel processing capability [25]. The development environment of the simulation system is Inter Core i7-12600K 3.7 GHz, which is capable of reaching 4.9 GHz at high loads, with 16 GB of RAM, GTX 3060 GPU, and Visual Studio 2019 for the development software. The architecture of the simulation system is shown in Fig. 5.

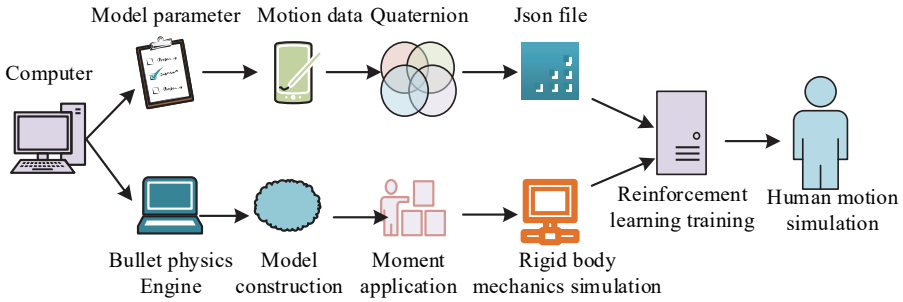

Figure 5 Simulation system architecture

In Fig. 5, the inputs include data such as the initial parameters of the simulation model, the specific state of each operation, and the motion trajectory. The relevant data are processed and generated into a Json file in quaternion expression format and transferred to the reinforcement learning module. On the other side, the Buller physics engine is used to synchronize the simulation model construction, and forces and moments are applied to the model for rigid body dynamics simulation. The simulation model is also trained with reinforcement learning algorithm to optimize its motion control. Tab. 1 displays the precise parameters of the simulation motion model that the study created.

In Tab. 1, all parts of the simulation model are rigid bodies and the model is divided into nine parts. Among them, the head and hands are spheres, the torso, hips, and feet are rectangles, and the remaining parts are ellipsoids. The torso of the model is the heaviest, which helps to ensure the stability of the model when walking. The connecting joints between each rigid body are connected by ball and socket joints. The complete operation flow of the HCI motion simulation system is shown in Fig. 6.

Table 1 Specific parameters of the simulation motion model

Position	Shape	Length / cm	Width / cm	Height / cm	Weight / kg
Head	Sphere	15	15	15	2.2
Torso	Cuboid	40	10	28	15.0
Big arm	Ellipsoid	30	10	10	1.6
Forearm	Ellipsoid	25	8	8	1.0
Hand	Sphere	8	8	8	0.8
Hip	Cuboid	15	12	26	5.8
Thigh	Ellipsoid	40	10	10	5.2
Calf	Ellipsoid	38	8	8	2.8
Foot	Cuboid	16	7	5	1.2

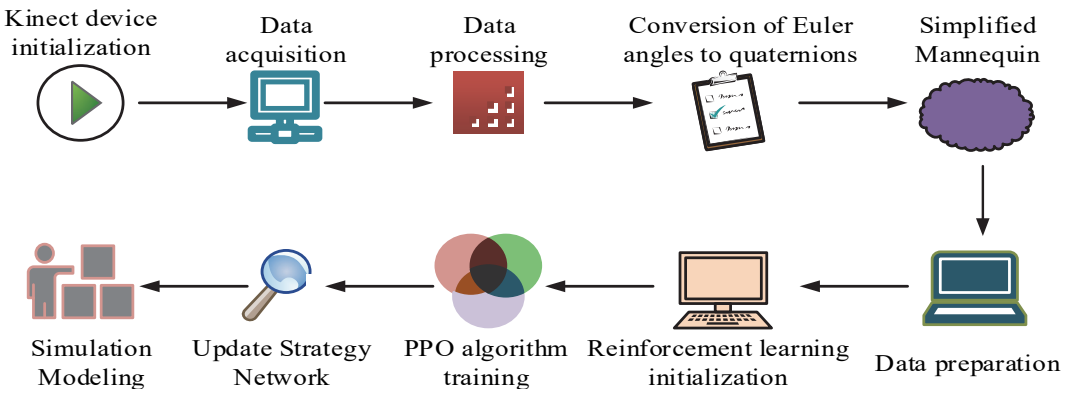

Figure 6 Complete operation flow of the human-computer interaction motion simulation system

In Fig. 6, the Kinect device is initialized using the NuiInitialize function, the human body data is collected, and the motion data is processed using the OpenCV image processing library. The data is converted from Euler angle data to quaternion form, and the simulated human body model is constructed using the Bullet physics engine. Transfer the processed data to the reinforcement learning module, initialize the reinforcement learning, update the policy network after completing the training, and finally construct the simulation model using the Bullet physics engine and perform real-time interactive simulation.

4 RESULTS AND DISCUSSION

4.1 Performance Analysis of Motion Control Algorithms

The learning rate of the PPO algorithm policy network and value function network is 3×10^{-4} and 1×10^{-3}, respectively, the discount factor is 0.99, the generalized dominance estimation parameter is set to 0.95, the batch size is 4096, the number of training iterations per round is 10, the cropping threshold is set to 0.2, the entropy reward function is set to an initial value of 0.01, and the decay of each round is 5%. The comparison algorithms used for the simulation experiments include the deep deterministic policy gradient (DDPG) algorithm and the actor-critic (AC) algorithm. Each algorithm uses the same simulation model and only changes the subsequent training method. A comparison of the algorithm performance of the simulation models under different motion postures is shown in Fig. 7.

In Fig. 7a, the higher reward value indicates the superior performance of the algorithm. The maximum reward value of PPO algorithm is significantly higher than the other algorithms with a maximum value of 463, which is 43.5 and 307.2 higher than the DDPG and AC algorithms, respectively, and it tends to converge when the sample size is 8 × 105. Although the ACA converges faster and reaches convergence when the sample size is 5×10^5, it is easy to fall into a local optimum. The DDPG algorithm, on the other hand, converges significantly more slowly, reaching convergence at a sample size of 10×10^5.

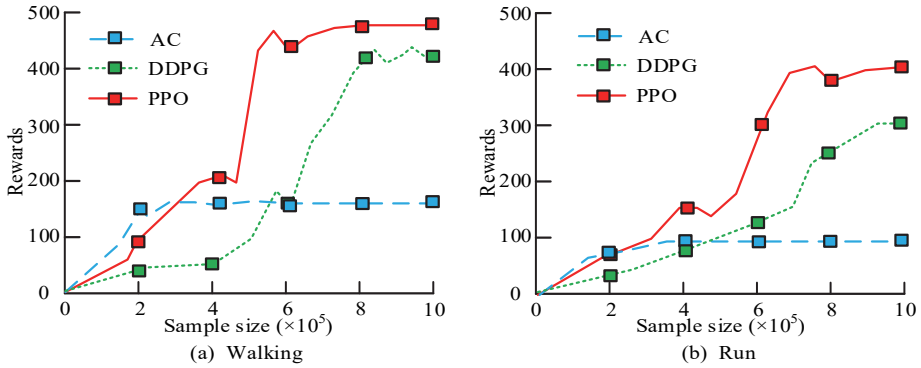

Figure 7 Comparison of algorithm performance of simulation model under different motion posture

In Fig. 7b, the model is in the running pose where the algorithm converges further back, requiring more samples for training and a decrease in the maximum reward value. Both the PPO and DDPG algorithms reach convergence at 10×10^5. Compared to the other two methods, the PPO algorithm's maximum RV is 312.5 and 95.8 higher, respectively. The variation of the loss function values of the different algorithms during training is shown in Fig. 8.

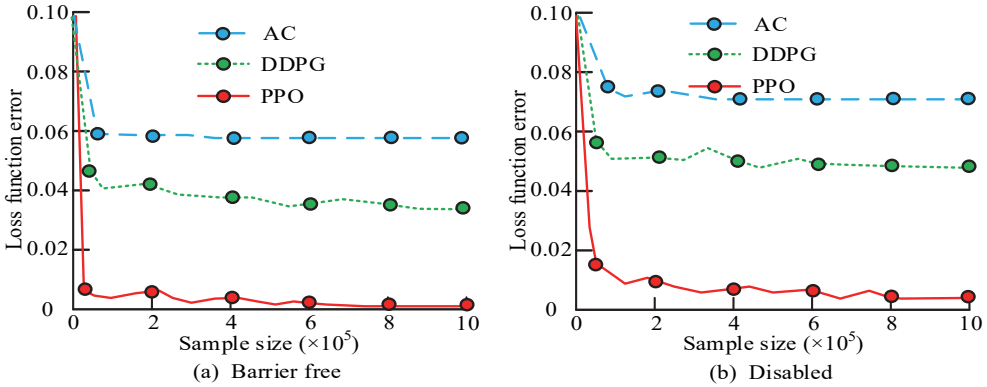

Figure 8 Changes of loss function values of different algorithms during training

In Fig. 8a, in the environment without obstacles, the loss function values of all algorithms decrease with the increase of the sample size, and the decrease is faster in the early stage, and the PPO algorithm reaches the convergence at the sample size of 8 × 105, and the minimum loss value is lower than the DDPG and AC algorithms by 0.038 and 0.059, respectively, and it fluctuates up and down a lot in the time when it does not reach the convergence. In Fig. 8b, when the model is trained in an environment with obstacles, the minimum loss error of all algorithms increases, and the minimum loss error of the PPO algorithm is 0.007, which is lower than that of the other two algorithms by 0.045 and 0.062, respectively, and the convergence position is basically unchanged.

4.2 Evaluation of HCI Simulation Systems and Analysis of their Practical Use

After constructing the corresponding human simulation motion model by the simulation system based on Bullet Physics Engine, the study evaluates and analyzes its motion control ability, and measures the stability and interference resistance of the model when it performs different actions. In this experiment, the motion evaluation of the simulation model is mainly measured when different directional forces and obstacles are applied to the model in the same experimental environment, and the performance of the simulation system is verified. A comparison of the reward curves of the model when a forward or backward force is applied to the torso of the model is shown in Fig. 9.

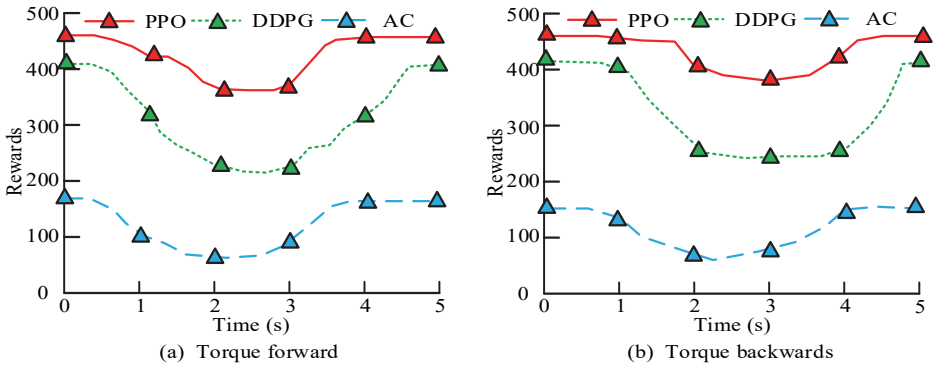

Figure 9 Comparison of the reward curve of the model

In Fig. 9a, applying a gradually increasing forward pull force at the initial moment, the reward curve of the three algorithms has basically the same trend of change, with the reward value decreasing first and then increasing. The model loses balance under the external force, resulting in a deviation between the simulated action and the actual data, and the reward value decreases. The reward curve of the PPO algorithm decreases more slowly, and at the same time recovers faster, which indicates that it is more effective in controlling the model, and is able to respond quickly to complete the model control. The RV decreased by a maximum of 62.4, which is 123.5 and 31.2 less than the DDPG and ACAs, respectively. In Fig. 9b, the model is subjected to a backward force causing the RV to first decrease and then increase. Compared to the other two methods, the PPO algorithm's maximum decreasing RV of 53.1 is 112.7 and 30.6 lower, respectively. A comparison of the model's reward curves when a larger force is applied to the model's legs and arms is shown in Fig. 10.

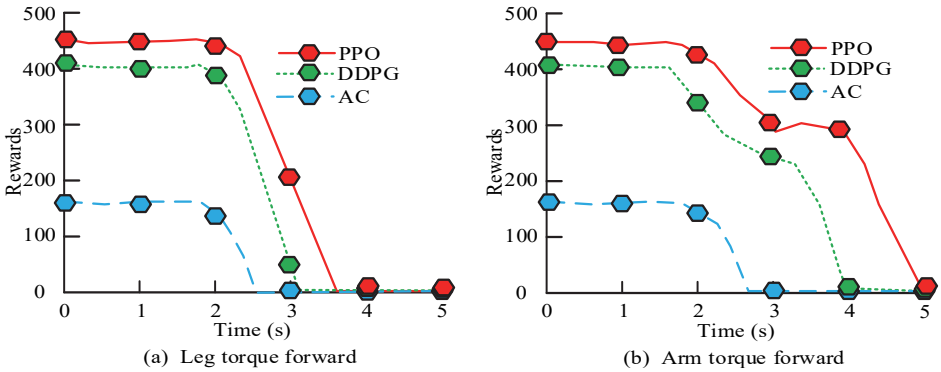

Figure 10 Comparison of the reward curve of the model when large torque is applied

In Fig. 10a, when the force applied to the model's leg is large, the model's leg changes beyond the algorithm's control ability, resulting in the RV dropping to 0 and the model falling down. However, the RV of the model controlled by the PPO algorithm falls more slowly, and the time for the RV to fall to 0 is 0.7 s and 1.3 s higher than that of the DDPG and ACAs, respectively. It shows that the PPO algorithm is able to control the balance of the model to a certain extent and slow down the fall of the model when the force on the model exceeds a limited value. In Fig. 10b, when a large unilateral moment is applied to the model's arm, the model tipped in the direction of the applied force. In this case, the model controlled by the PPO algorithm attempted to take control and the RV of the model recovered, but it eventually fell over. The time for the RV of the PPO algorithm to fall to 0 is 1.1 s and 1.8 s higher than that of the DDPG and ACAs, respectively. Comparison of the reward curves of the simulated models when the obstacle is at different positions is shown in Fig. 11.

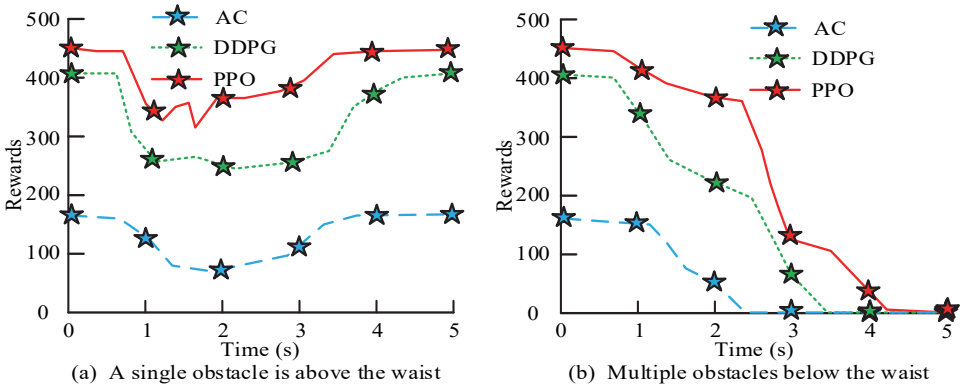

Figure 11 Comparison of reward curves of simulation models when obstacles are at different positions

In Fig. 11a, the RV of the model decreases at the time of collision in the presence of a single obstacle above the hip. Because of the control algorithm, the model's pose is constantly adjusted during the collision and the reward curves go up and down, but they all return to normal eventually. The reward change values of the PPO algorithm are 47.5 and 6.7 lower than those of the DDPG and ACAs, respectively, indicating that the model controlled by the PPO algorithm is more stable. Bonus value recovery was faster than the other two algorithms by 0.9 s as well as 0.6 s, respectively. In Fig. 11b, the model's RV decreases to 0 after hitting multiple obstacles and the motion control fails resulting in a fall. The time for the RV of the PPO algorithm to decrease to 0 is higher than that of the DDPG and ACAs by 0.8 s and 2.1 s, respectively. The study is conducted to analyze the HCI application in an indoor venue. To increase the realism and complexity of the scenario, obstacles are randomly placed in front of the experimenter. The obstacles are vases, tables, and chairs, etc., with heights capable of reaching the knees and hips of the model. The testers walk randomly in the field to test the relevant data during normal walking. A 3D motion capture system is used for motion and joint data acquisition. The researcher conducts multiple walking trials with different routes and obstacles for each trial to ensure the diversity of data collected and eliminate errors. The simulated motion model is compared with the real human joint data during HCI as shown in Fig. 12.

In Fig. 12a, the study selects the knee and hip joint angle changes during normal walking as a comparison. The number of sampling frames is chosen as 500 frames, and the actual data of human body is inputted into the model to let it be trained. After the training is completed, the same movement is performed during HCI. The angle of the knee joint of the model during the motion is a periodic change, which is the same as the real situation of the human body. The simulation data of the model is approximately the same as the actual motion data, and there is only a slight error in the joint angle turning point, and the fitting degree of the knee joint angle is 91.4%. In Fig. 12b, the trend of hip joint angle change is the same as the actual situation. The fit between the simulation model and the actual

situation is 93.2%, and the largest gap occurs at frame 410, with a difference of 2.3°. Comparison of computational complexity, real-time applicability and scalability of different methods in different computing resources are shown in Tab. 2.

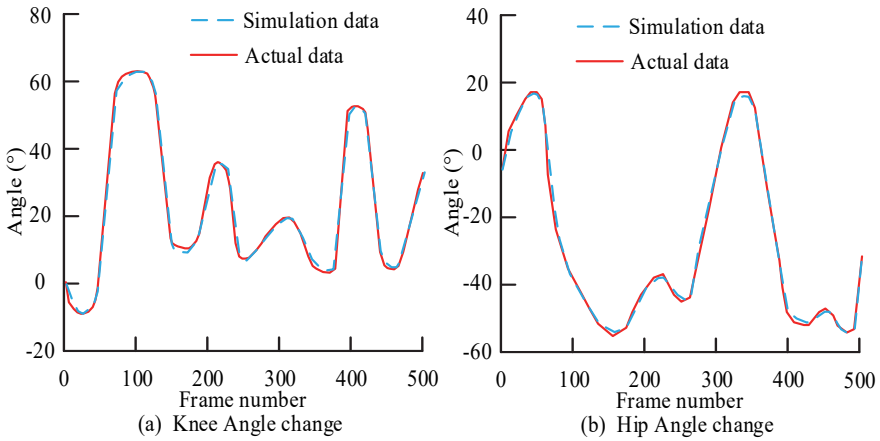

Figure 12 Comparison of angle changes of different joints during movement of the simulation model

Table 2 Comparison of computational complexity and real-time performance of different methods

Sample size	Model type	Average processing time / ms	Standard deviation / ms	Time complexity O(n2) evaluation
1000	PPO	12.5	1.7	Lower
	DDPG	29.4	2.8	Normal
2000	PPO	26.2	3.5	Lower
	DDPG	79.5	7.2	Normal
3000	PPO	40.2	5.8	Lower
	DDPG	178.6	9.4	Higher

In Tab. 2, the average processing time of the PPO algorithm is 12.5 ms, 26.2 ms, and 40.2 ms, which is 16.9 ms, 53.3 ms, and 138.4 ms lower than the DDPG algorithm at sample sizes of 1000, 2000, and 3000, respectively. The standard deviations of the PPO algorithm are lower than those of the DDPG algorithm, and the computational complexity is Lower.

5 CONCLUSION

This research developed and validated an advanced HCI motion control system leveraging the Bullet physics engine combined with reinforcement learning, specifically using the PPO algorithm. The outcomes of this study demonstrated considerable improvements over existing methods in terms of stability, precision, and resilience under disturbances and dynamic conditions. Notably, the PPO algorithm outperformed comparison models by achieving higher maximum rewards and lower loss errors in various motion scenarios, including walking and running postures. Furthermore, under external forces and obstacle interactions, the model exhibited robust adaptability and rapid recovery capabilities, reflecting enhanced stability and anti-interference performance. High fidelity between the simulation outcomes and actual human joint data underscores the model's accuracy and potential for realistic applications. However, there is room for further improvement, particularly by expanding the control methodology to include more complex skeletal structures and diverse human motions such as those found in sports and complex physical tasks, or in combination with other deep learning techniques, and expanding the potential applications of human simulation motion modeling in the industries of healthcare, entertainment, and robotics, for example, to help patients with rehabilitation training, surgical simulation, virtual reality gaming, and humanoid robots, etc.

6 REFERENCES

[1] Fischer, F., Fleig, A., Klar, M. et al. (2022). Optimal feedback control for modeling human-computer interaction. *ACM Transactions on Computer-Human Interaction*, *29*(6), 1-70. https://doi.org/10.1145/3524122

[2] Li, K. & Li, X. (2022). AI driven human-computer interaction design framework of virtual environment based on comprehensive semantic data analysis with feature extraction. *International Journal of Speech Technology*, *25*(4), 863-877. https://doi.org/10.1007/s10772-021-09954-5

[3] Sadeghi Milani, A., Cecil-Xavier, A., Gupta, A. et al. (2024). A systematic review of human-computer interaction (HCI) research in medical and other engineering fields. *International Journal of Human-Computer Interaction*, *40*(3), 515-536. https://doi.org/10.1080/10447318.2022.2116530

[4] Yi, J., Liu, J., Zhang, C. et al. (2022). Magnetic motion tracking for natural human-computer interaction: A review. *IEEE Sensors Journal*, *22*(23), 22356-22367. https://doi.org/10.1109/JSEN.2022.3215285

[5] Yusoff, Z. M., Nordin, S. A., Markom, A. M. et al. (2023). Wireless hand motion controlled robotic arm using flex sensors. *Indonesian Journal of Electrical Engineering and Computer Science*, *29*(1), 133-140. https://doi.org/10.11591/ijeecs.v29.i1.pp133-140

[6] Dong, L., He, Z., Song, C. et al. (2023). A review of mobile robot motion planning methods: From classical motion planning workflows to reinforcement learning-based

architectures. *Journal of Systems Engineering and Electronics*, *34*(2), 439-459. https://doi.org/10.23919/JSEE.2023.000051

[7] Alvarado, E., Rohmer, D., & Cani, M. P. (2022). Generating upper-body motion for real-time characters making their way through dynamic environments. *Computer Graphics Forum*, *41*(8), 169-181. https://doi.org/10.1111/cgf.14633

[8] Yang, Z. (2024). Animation VR motion simulation evaluation based on somatosensory simulation control algorithm. *Informatica*, *48*(11), 172-194. https://doi.org/10.31449/inf.v48i11.6041

[9] Halilu, A. S., Majumder, A., Waziri, M. Y. et al. (2022). Motion control of the two joint planar robotic manipulators through accelerated Dai-Liao method for solving system of nonlinear equations. *Engineering Computations*, *39*(5), 1802-1840. https://doi.org/10.1108/EC-06-2021-0317

[10] Chen, G., Xu, Y., Yang, C. et al. (2023). Design and control of a novel bionic mantis shrimp robot. *IEEE/ASME Transactions on Mechatronics*, *28*(6), 3376-3385. https://doi.org/10.1109/TMECH.2023.3266778

[11] Prasad, A., Sharma, B., Vanualailai, J. et al. (2022). Motion control of an articulated mobile manipulator in 3D using the Lyapunov-based control scheme. *International Journal of Control*, *95*(9), 2581-2595. https://doi.org/10.1080/00207179.2021.1919927

[12] Guo, Q. & Ma, G. (2022). Exploration of human-computer interaction system for product design in virtual reality environment based on computer-aided technology. *Computer-Aided Design & Applications*, *19*(S5), 87-98. https://doi.org/10.14733/cadaps.2022.S5.87-98

[13] Gao, P. (2022). Key technologies of human-computer interaction for immersive somatosensory interactive games using VR technology. *Soft Computing*, *26*(20), 10947-10956. https://doi.org/10.1007/s00500-022-07240-3

[14] Murray-Smith, R., Oulasvirta, A., Howes, A. et al. (2022). What simulation can do for HCI research. *Interactions*, *29*(6), 48-53. https://doi.org/10.1145/3564038

[15] Qiao, H., Wu, Y. X., Zhong, S. L. et al. (2023). Brain-inspired intelligent robotics: Theoretical analysis and systematic application. *Machine Intelligence Research*, *20*(1), 1-18. https://doi.org/10.1007/s11633-022-1390-8

[16] Verawati, N. N. S. P., Handriani, L. S., & Prahani, B. K. (2022). The experimental experience of motion kinematics in biology class using PhET virtual simulation and its impact on learning outcomes. *International Journal of Essential Competencies in Education*, *1*(1), 11-17. https://doi.org/10.36312/ijece.v1i1.729

[17] Lyu, Z. (2024). State-of-the-art human-computer interaction in metaverse. *International Journal of Human-Computer Interaction*, *40*(21), 6690-6708. https://doi.org/10.1080/10447318.2023.2248833

[18] Zhong, B., Zheng, J., & Zhan, Z. (2023). An exploration of combining virtual and physical robots in robotics education. *Interactive Learning Environments*, *31*(1), 370-382. https://doi.org/10.1080/10494820.2020.1786409

[19] Nenna, F., Orso, V., Zanardi, D. et al. (2023). The virtualization of human-robot interactions: A user-centric workload assessment. *Virtual Reality*, *27*(2), 553-571. https://doi.org/10.1007/s10055-022-00667-x

[20] Zhou, C., Huang, B., & Fränti, P. (2022). A review of motion planning algorithms for intelligent robots. *Journal of Intelligent Manufacturing*, *33*(2), 387-424. https://doi.org/10.1007/s10845-021-01867-z

[21] Yevsieiev, V. & Starodubcev, N. (2023). Development of a control algorithm for a small-sized mobile manipulation robot. *Scientific Collection «InterConf»*, *140*, 648-651.

[22] Qazani, M. R. C., Asadi, H., Zhang, L. et al. (2022). A new prepositioning technique of a motion simulator platform using nonlinear model predictive control and recurrent neural network. *IEEE Transactions on Intelligent Transportation Systems*, *23*(12), 23268-23277. https://doi.org/10.1109/TITS.2022.3195964

[23] Nie, Z., Yu, Y., & Bao, Y. (2023). Application of human-computer interaction system based on machine learning algorithm in artistic visual communication. *Soft Computing*, *27*(14), 10199-10211. https://doi.org/10.1007/s00500-023-08267-w

[24] Demirel, H. O., Ahmed, S., & Duffy, V. G. (2022). Digital human modeling: A review and reappraisal of origins, present, and expected future methods for representing humans computationally. *International Journal of Human-Computer Interaction*, *38*(10), 897-937. https://doi.org/10.1080/10447318.2021.1976507

[25] Barricelli, B. R. & Fogli, D. (2024). Digital twins in human-computer interaction: A systematic review. *International Journal of Human-Computer Interaction*, *40*(2), 79-97. https://doi.org/10.1080/10447318.2022.2118189

Contact information:

Weiqiong ZHAO, Master Lecturer
(Corresponding author)
School of Intelligence Technology,
Geely University of China,
Chengdu Sichuan,610000, P. R. China,
No. 123, SEC. 2, Chengjian Avenue, Eastern New District, Chengdu City, Sichuan Province
E-mail: zhaoweiqiong@guc.edu.cn

Fuchuan YE, Experimental Officer
Information and Educational Technology Center,
Southwest Minzu University, Chengdu Sichuan,610041, P. R. China,
No. 16, South 4th Section of 1st Ring Road, Wuhou District, Chengdu City, Sichuan Province
E-mail: 30300025@swun.edu.cn