frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 15 January 2021
doi: 10.3389/fnbot.2020.600885

Gait Optimization Method for Humanoid Robots Based on Parallel Comprehensive Learning Particle Swarm Optimizer Algorithm

Chongben Tao[1,2], *Jie Xue*[1], *Zufeng Zhang*[1,3,4]*, *Feng Cao*[5], *Chunguang Li*[6] and *Hanwen Gao*[1]

[1] School of Electronic and Information Engineering, Suzhou University of Science and Technology, Suzhou, China, [2] Suzhou Automobile Research Institute, Tsinghua University, Suzhou, China, [3] Department of Automation, Tsinghua University, Beijing, China, [4] Wuhan Electronic Information Institute, Hubei, China, [5] School of Computer and Information Technology, Shanxi University, Taiyuan, China, [6] School of Computer Information and Engineering, Changzhou Institute of Technology, Changzhou, China

To improve the fast and stable walking ability of a humanoid robot, this paper proposes a gait optimization method based on a parallel comprehensive learning particle swarm optimizer (PCLPSO). Firstly, the key parameters affecting the walking gait of the humanoid robot are selected based on the natural zero-moment point trajectory planning method. Secondly, by changing the slave group structure of the PCLPSO algorithm, the gait training task is decomposed, and a parallel distributed multi-robot gait training environment based on RoboCup3D is built to automatically optimize the speed and stability of bipedal robot walking. Finally, a layered learning approach is used to optimize the turning ability of the humanoid robot. The experimental results show that the PCLPSO algorithm achieves a quickly optimal solution, and the humanoid robot optimized possesses a fast and steady gait and flexible steering ability.

Keywords: RoboCup3D, humanoid robot, PCLPSO, parallel distributed, layered learning

OPEN ACCESS

Edited by:
Keke Tang,
Guangzhou University, China

Reviewed by:
Yaguang Zhu,
Chang'an University, China
Lan Anh Trinh,
Mälardalen University College, Sweden
Eiji Uchibe,
Advanced Telecommunications Research Institute International (ATR), Japan

***Correspondence:**
Zufeng Zhang
zhangzufeng@tsari.tsinghua.edu.cn

Received: *31 August 2020*
Accepted: *07 December 2020*
Published: *15 January 2021*

Citation:
Tao C, Xue J, Zhang Z, Cao F, Li C and Gao H (2021) Gait Optimization Method for Humanoid Robots Based on Parallel Comprehensive Learning Particle Swarm Optimizer Algorithm. Front. Neurorobot. 14:600885. doi: 10.3389/fnbot.2020.600885

INTRODUCTION

Gait planning is a research hotspot for humanoid robots, and it provides some technical support for humanoid robots walking like humans. The methods of gait planning can be broadly divided into three categories: human walking parameter-based methods (Baoping et al., 2015; Hereid et al., 2018), humanoid walking model-based methods (Sato et al., 2010; Winkler et al., 2018), and intelligent algorithm-based methods (Huan and Anh, 2015; Elhosseini et al., 2019). The method based on human walking parameters makes the gait of humanoid robots more similar to the way humans walk, but it costs a lot of time to find suitable gait parameters from human walking data and apply them to humanoid robots. Paparisabet et al. (2019) proposed a similar function for human-like motion, formulated kinematic constraints for humanoid robots in contact with the ground, and finally proposed humanoid walking with very high similarity to human motion. In Weon and Lee (2018), Weon et al. proposed a method for generating humanoid robot motion based on motion capture data, which corrects extracted joint trajectories based on a reprogrammed zero-moment point (ZMP) trajectory. Researchers have extensively investigated bipedal walking model-based approaches. In Graf and Röfer (2011), the three-dimensional linear inverted pendulum model (3D-LIPM) proposed is one of the most widely used simplified dynamics models for humanoid robots. The 3D-LIPM approximates the humanoid robot in the three-dimensional space to an inverted pendulum model composed of mass points and massless legs connecting the points to the support

points and constrains the center of mass to move on the constrained plane. Jadidi and Hashemi (2016) proposed a closed-loop 3D-LIPM gait for RoboCup standard platform and implemented a full range of walking on the NAO robot. The 25 degrees of freedom of the NAO robot make it have excellent omnidirectional walking and full-body motion performance. The RoboCup 3D simulation team uses the NAO robot as a reference model. At the same time, this model is also widely used in robot simulation competitions at all levels at home and abroad. Astudillo et al. (2018) used the 3D-LIPM as a model for robot and ZMP to map joint angles to achieve a humanoid robot walking on a slippery platform.

As the degrees of freedom of humanoid robots increase, the complexity of systems will increase as well. Bipedal walking model-based methods will not be sufficient for the development of humanoid robot control (Fayong et al., 2014). Besides, a variety of intelligent control methods are developed, which do not require accurate modeling (MacAlpine et al., 2015; Hong and Lee, 2016; Bonyadi and Michalewicz, 2017). However, the process of adjusting motion parameters and posture based on various models is very tedious and time-consuming. When the given parameters are not reasonable, walking instability and robots moving at a low speed may occur. This shortcoming can have a significant impact on the coordination between the needs of speed, stability, and flexibility. Therefore, various intelligent algorithms are used for robot gait planning. Many researchers have applied the central pattern generator (CPG) to generate gait trajectories, but the parameter optimization of this method is a challenge (Bai et al., 2019). Zhong et al. (2017) transformed phase signal from CPG output into a trajectory signal for the legs of a six-legged robot by adjusting it. Wang et al. (2019) proposed a gait planning method based on a reactive neuromuscular controller and CPG to achieve a power-saving human-like large walk, and the controller parameters were optimized based on an optimization algorithm in the paper. Common optimization algorithms such as central force optimization (CFO) and genetic algorithm (GA) have also been successfully used for the gait planning of humanoid robots. Kumar et al. (2018) applied GA to optimize parameters for a triple-linked humanoid robot to achieve numerical simulation of energy-controlled stable walking. Huan et al. (2018a) used CFO to optimize the foot lift amplitude of a humanoid robot, which caused an efficient and stable gait. PSO is a common intelligent optimization algorithm that solves global optimization problems simply and efficiently (Kennedy and Eberhart, 1995). Huan et al. (2018b) applied PSO to optimize joint angles to achieve stable walking for a humanoid robot with 10 degrees of freedom. Mandava et al. proposed a multi-objective particle swarm optimization algorithm method for the gait optimization of humanoid robots for the trolley table model. This method uses a sliding mode controller to optimize robust tracking control and realizes the 3D simulation walking of a humanoid robot (Mandava and Vundavilli, 2018). Gülcü and Kodaz (2015) improved the performance of comprehensive learning particle swarm optimizer (CLPSO) through parallel computation and proposed PCLPSO. Most of the studies mentioned earlier are devoted to single movements such as straight, rotating, and going up and down stairs (Faraji et al., 2019). There are relatively few studies that comprehensively consider bipedal robot forward and rotation and their arbitrary motion connection transitions. Also, during the simulation process, the individual simulation platforms always add noise in the same way, resulting in similar movements of the bipedal robots. Parallel optimization algorithms can be a good solution to this problem. Muniz et al. (2016) optimized the keyframe movements of a humanoid robot (getting up, kicking, etc.) through parallelization to improve the motion performance of the robot. However, parallel algorithms suffer from low fault tolerance when dealing with distributed tasks in RoboCup3D, and it is difficult to ensure the correctness and stability of the running process.

Based on the considerations mentioned earlier, this p*aper se*lects 13 key parameters that affect speed and stability based on the gait planning method of natural ZMP trajectories and designs two evaluation functions to address the problems of humanoid robot walking. By changing the cluster structure of the PCLPSO algorithm, a parallel distributed multi-robot gait training environment is established by using the RoboCup3D simulation platform. The training environment consists of multiple computers that can operate independently. The nodes use the computer network for information transfer to achieve a common task (gait optimization for humanoid robots). The computational efficiency is improved by running in parallel in a distributed environment. A layered learning approach is used to optimize the evaluation function layer by layer. Experimental results show that the optimized humanoid robot has a faster and more stable straight gait and excellent turning ability and has less wobble when switching between straight and turning.

GAIT PARAMETER SELECTION BASED ON NATURAL ZERO-MOMENT POINT TRAJECTORY PLANNING

In this paper, after setting a natural ZMP trajectory from heel to toe movement based on a 3D-LIPM in the single-leg support phase, a mass-centered trajectory equation is obtained (Graf and Röfer, 2011). In the double-leg support phase, a linear pendulum model was used to generate mass-centered trajectory equations. Equations for multistep planning of mass-centered trajectories in a unified coordinate system are also given. After planning the walking trajectory by natural ZMP-based mass-centered trajectory planning method, the key gait parameters are selected and optimized based on the experience of manual tuning.

Multistep Trajectory Planning in a Unified Coordinate System

During the movement of the humanoid robot, if only the front and back and up and down movements are considered and the left and right movements are ignored, it is easy to cause the robot to lose its balance and fall. Therefore, it is

necessary to extend the linear inverted pendulum to a three-dimensional environment and model the robot with a three-dimensional linear inverted pendulum. However, the process of directly analyzing and researching the multi-link structure of biped robots is often cumbersome, so this article equivalently simplified the robot to a reasonable mathematical model, which is convenient for research. Regarding the body of the robot as a mass point and the legs as massless support rods, a three-dimensional linear inverted pendulum model can be built. It does not need to know the parameters of the robot, such as the mass and the inertia of each joint, but is derived from the model for easy calculation. According to 3D-LIPM, a humanoid robot is simplified to an inverted pendulum with only a center of mass and a retractable massless pendulum, and the height of mass is assumed to remain constant, as shown in **Figure 1A**.

Assume that the height of the center of mass is fixed at z_s and the acceleration of gravity is g, the equation of motion between ZMP trajectory $p_{sx}(t)$ and the center of mass in the x-axis direction is:

$$p_{sx}(t) = x_s(t) - \frac{z_s}{g}\ddot{x}_s(t) \tag{1}$$

To make the ZMP trajectory of humanoid robot walking similar to that of human walking, this paper uses a linear equation to represent the ZMP trajectory, as shown in **Figure 1B**. Assuming that in the single-leg support phase, ZMP moves in the x-axis direction in the range of $[p_{sx\,min},\ p_{sx\,max}]$, the walking period of the single-leg support phase is T_s; the center of the foot is the origin of a coordinate system, and the trajectory of ZMP is given by the following equation:

$$p_{sx}(t) = b_1 t + b_0 \tag{2}$$

where, $t \in [0, T_s]$, $b_0 = p_{sx\,min}$, $b_1 = (p_{sx\,max} - p_{sx\,min})/T_s$.

By defining $w_s = \sqrt{\frac{z_s}{g}}$ by substituting Equation (1) into Equation (2) and solving a differential equation, one has:

$$x_s(t) = C_1 e^{t/w_s} + C_2 e^{-t/w_s} + b_1 t + b_0 \tag{3}$$

$$\dot{x}_s(t) = C_1 e^{t/w_s}/w_s - C_2 e^{-t/w_s}/w_s + b_1 \tag{4}$$

If the position and velocity $x_s(0)$ of the center of mass at the initial moment of the single-leg support phase and $\dot{x}_s(0)$ are known, then one has:

$$C_1 = \left(\left(x_s(0) - b_0\right) + \left(\dot{x}_s(0) - b_1\right) w_s\right)/2 \tag{5}$$

$$C_2 = \left(\left(x_s(0) - b_0\right) - \left(\dot{x}_s(0) - b_1\right) w_s\right)/2 \tag{6}$$

If the positions of the center of mass at the initial moment of single-leg support phase and the positions of the center of mass at the moment of termination $x_s(0)$ and $x_s(T_s)$ are known, there are:

$$C_1 = \left(\left(x_s(0) - b_0\right) e^{-T_s/w_s} - \left(x_s(T_s) - b_1 T_s - b_0\right)/ \left(e^{-T_s/w_s} - e^{T_s/w_s}\right)\right. \tag{7}$$

$$C_2 = \left(\left(x_s(0) - b_0\right) e^{T_s/w_s} - \left(x_s(T_s) - b_1 T_s - b_0\right)/ \left(e^{T_s/w_s} - e^{-T_s/w_s}\right)\right. \tag{8}$$

According to Equations (2–4), the centroid trajectory planning based on natural ZMP can be realized in the single-leg support phase.

The direct application of the single-leg support phase method for walking requires the assumption that the support leg switch is instantaneous. This will cause the center of mass acceleration to jump from the maximum to the minimum. To obtain a smooth center-of-mass velocity trajectory, the legs support phase is introduced. In this paper, a linear pendulum model is used to realize natural ZMP trajectory planning of the double-leg supporting phase.

In the double-leg support phase, according to the linear pendulum model, as shown in **Figure 1C**, the equation of the relationship between the position of robot center of mass and acceleration is given as follows:

$$x_d(t) - D = \frac{z_d}{g}\ddot{x}_d(t) \tag{9}$$

where $z_d < 0$, $t \in [0, T_d]$, T_d is double-leg support phase walk period, and D is an x-axis coordinate of the fixed end of the linear pendulum.

With $w_d = \sqrt{\frac{-z_d}{g}}$, from Equation (9) can have:

$$x_d(t) = (x_d(0) - D)\cos(t/w_d) + \dot{x}_d(0)\, w_d \sin(t/w_d) + D \tag{10}$$

$$\dot{x}_d(t) = \dot{x}_d(0)\cos(t/w_d) - \frac{x_d(0) - D}{w_d}\sin(t/w_d) \tag{11}$$

In the double-leg support phase, the starting position and speed and the ending position and the speed of the center of mass can be known, that is, $x_d(0)$, $\dot{x}_d(0)$, $x_d(T_d)$, and $\dot{x}_d(T_d)$ can be known.

$$D = \frac{\dot{x}_d(0)^2 w_d^2 - \dot{x}_d(T_d)^2 w_d^2 + x_d(0)^2 - x_d(T_d)^2}{2(x_d(0) - x_d(T_d))} \tag{12}$$

$$T_d = w_d \arccos \frac{(x_d(T_d) - D)(x_d(0) - D) + \dot{x}_d(0)\,\dot{x}_d(T_d)\, w_d^2}{(x_d(0) - D)^2 + \dot{x}_d(0)^2 w_d^2} \tag{13}$$

A smooth center-of-mass trajectory based on natural ZMP trajectory can be achieved in the double-leg support phase according to Equations (10–13).

The methods mentioned earlier have their own coordinate systems in the single-leg support phase and double-leg support phase, which do not facilitate multistep planning calculations for footprint planning. To do so, the equations mentioned earlier need to be unified in the same coordinate system. The equation for the position and velocity of the center

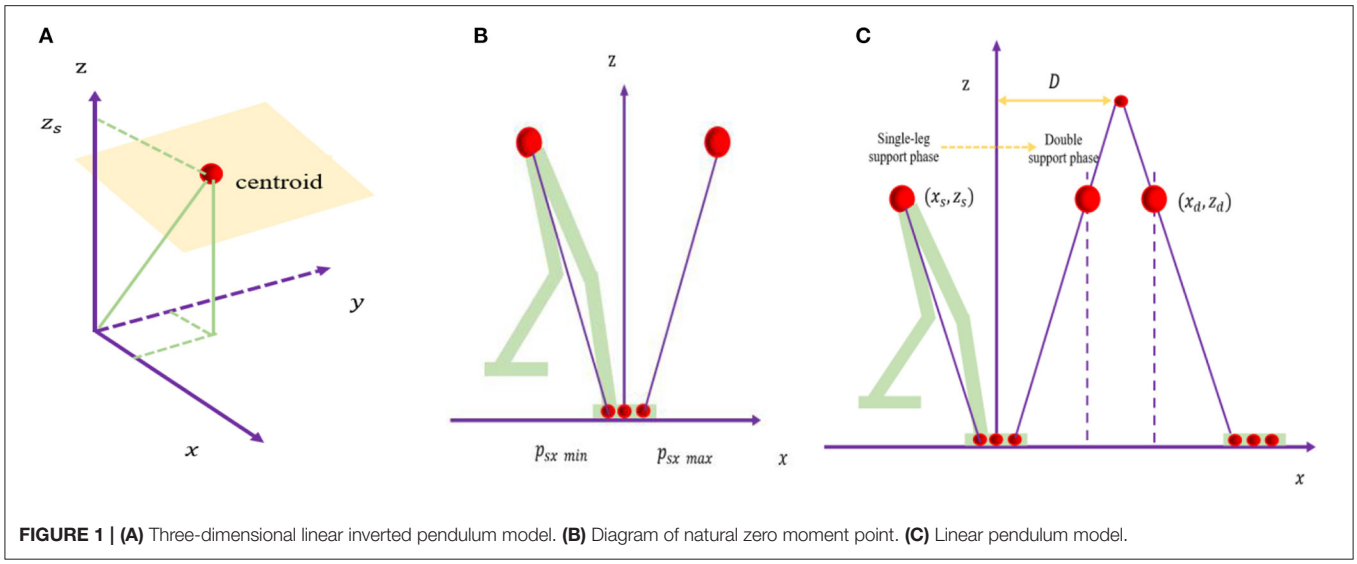

FIGURE 1 | (A) Three-dimensional linear inverted pendulum model. **(B)** Diagram of natural zero moment point. **(C)** Linear pendulum model.

of the mass in multistep planning can be expressed by the following equation:

$$x(t) = \sum_{i=0}^{n} (x_{si}(t) + x_{di}(t)) \tag{14}$$

$$\dot{x}(t) = \sum_{i=0}^{n} (\dot{x}_{si}(t) + \dot{x}_{di}(t)) \tag{15}$$

The robot gait realized by Equations (14, 15) yields a natural ZMP trajectory.

Swing Leg Trajectory Planning

Cosine functions and Bessel curves can all be used to plan swing-leg trajectories (Kajita et al., 2002; Grzelczyk et al., 2016). However, the humanoid robot is divided into two phases of single-leg support phase and double-leg support phase during walking, and only when the speed and acceleration of swing-leg start and landing are zero can the humanoid robot maintain stability when switching between single-leg support and double-leg support (Tang et al., 2003). To obtain a smoother trajectory of the oscillating leg, this paper chooses the method of simple harmonic motion synthesis:

$$z_{sw}(t) = \begin{cases} H_{sw}\left(\frac{2}{T_s}t - \frac{1}{2\pi}\sin\left(\frac{4\pi}{T_s}t\right)\right), t\epsilon[0, T_s/2] \\ H_{sw}\left(-\frac{2}{T_s}t + \frac{1}{2\pi}\sin\left(\frac{4\pi}{T_s}t\right) + 2\right), t\epsilon[T_s/2, T_s] \end{cases} \tag{16}$$

$$x_{sw}(t) = D_{sw}\left(\frac{t}{T_s} - \frac{1}{2\pi}\sin\left(\frac{2\pi}{T_s}t\right)\right) \tag{17}$$

where D_{sw} and H_{sw} are the maximum height of step length and leg lift, respectively.

Selection of Gait Parameters

There are two kinds of bipedal walking pattern generation methods. The first method uses precise knowledge of robot dynamics parameters, such as mass, centroid position, and inertia of each joint to configure walking mode. Therefore, the method mainly depends on the accuracy of the model data. In contrast, the second method uses limited knowledge of dynamics, such as the total position of the center of mass, the total angular momentum, etc. The simplified model method (3D-LIMP) used in this paper belongs to the second type. After the optimized trajectory is obtained, the robot can execute according to the corresponding trajectory, and an excellent walking gait can be obtained. By means of a gait generation method based on natural ZMP trajectory planning, humanoid robot walking is summarized in the following algorithmic steps.

Algorithm 1

1.Set the walking cycle T_s and walking parameters (D_{sw}, H_{sw}), initial foothold ($p_x^{(0)}$, $p_y^{(0)}$);
2.Initialization time $T = 0$, number of walking units $n = 0$;
3.Calculate the inverted pendulum equation from time T to $T + T_s$ and get the equation of mass center trajectory;
4. $T = T + T_s$, $n = n + 1$;
5. Calculate and determine the next foothold of the biped robot ($p_x^{(n)}$, $p_y^{(n)}$);
6. Give the next position of the bipedal robot;
7. Return to step 3.

The RoboCup3D server communicates with the robot once every 20 ms. In this article, the robot is also controlled once in 20 ms. One step is eight times, so the walk period T_s is 0.16 s. As algorithm 1 only considers the robot walking in a straight line, for the case of steering walk. If the step length is set to zero, the robot will only walk laterally, and if the step width is set to zero, the robot will only walk in a straight line. However, in a simulation match, the flexibility of the player is enhanced by changing the

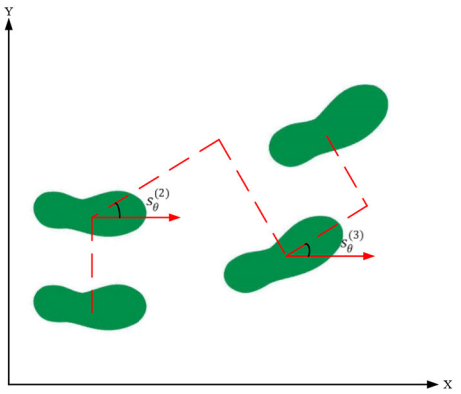

FIGURE 2 | Diagram of turning parameters.

TABLE 1 | Optimized gait parameters.

Parameter	Description
FootLength	Step size of one step (D_{sw})
FootWidth	Step width of one step
FootHeight	Maximum height of swinging leg (H_{sw})
s_θ	One step turning angle
T_s	One step time
BodyHeight	Body height in walking
Thigh	Leg length
Hip	Hip height
$Com_{off}(x,y)$	Deviation of centroid
Com_{max}	Offset of maximum center of mass
Com_{next}	Next centroid position
Arm(x,y)	Amplitude of arm swing
Zmp_{off}	Lateral offset of zero moment point

direction of travel while walking quickly. To be able to walk at any angle, additional information about the direction is required. The direction of each step is specified as s_θ, as shown in **Figure 2.**

Equation (18) represents the position of the nth step foothold of a humanoid robot $(p_x^{(n)}, p_y^{(n)})$

$$\begin{bmatrix} p_x^{(n)} \\ p_y^{(n)} \end{bmatrix} = \begin{bmatrix} p_x^{(n-1)} \\ p_y^{(n-1)} \end{bmatrix} + \begin{bmatrix} \cos s_\theta^{(n)} & -\sin s_\theta^{(n)} \\ \sin s_\theta^{(n)} & \cos s_\theta^{(n)} \end{bmatrix} \begin{bmatrix} \Delta x_x^{(n)} \\ -(-1)^n s_y^{(n)} \end{bmatrix} \quad (18)$$

Therefore, only important parameters such as step length, step width, and directional angle of humanoid robot need to be controlled to enable robot to walk. The gait parameters of the biped robot are as many as 40. If each parameter is optimized, it will inevitably affect the convergence speed because of the huge state space. And different gait parameters bias the focus of walking differently. According to natural ZMP-based mass-centered trajectory planning method and the previous experience of manual debugging, 13 key parameters affecting the gait of humanoid robot were selected, as shown in **Table 1**.

GAIT OPTIMIZATION BASED ON PARALLEL MULTIGROUP PARTICLE SWARM ALGORITHM

Aiming at the problem that a lot of manual debugging time is required when planning the robot trajectory by directly using a simplified model, this paper proposes a machine learning method based on the PCLPSO algorithm to optimize gait parameters. First, the important parameters are extracted according to the centroid trajectory planning method based on natural ZMP. Secondly, in the optimization of walking mode, different evaluation functions are set according to the requirements of game gait mode, and the PCLPSO algorithm is used to optimize the omnidirectional walking ability of a humanoid robot.

Parallel Comprehensive Learning Particle Swarm Optimizer Algorithm

PSO algorithm is a fast and efficient optimization algorithm (Kumar et al., 2018). Multiple individuals search for the target in the search area. It is a parallel and random optimization algorithm. Compared with other intelligent algorithms, it has a faster convergence speed and robustness. Each particle in PSO calculates the evaluation value based on the evaluation function. During the search of each particle, two extreme values are compared: the first is the optimal solution *pbest* of a particle; the other is the global optimal solution *gbest*. The speed and position updates of PSO are shown in Equations (19, 20):

$$v_i^{k+1} = v_i^k + c_1 * r_1 * (pbest_i^k - x_i^k) + c_2 * r_2 * (gbest^k - x_i^k) \quad (19)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (20)$$

where $i = 1, \ldots, N$ is the number of populations and $k = 0, \ldots, N_{iter}$ is the number of iterations; $pbest_i^k$ represents the local optimal solution found by the particle itself, whereas $gbest^k$ represents the global optimal solution for all current particles; c_1 and c_2 are two constants greater than zero, which are used to adjust the degree of attraction of local and global optimal to the particle; r_1 and r_2 are both uniformly distributed random numbers in the interval [0,1], which affects the random nature of the algorithm. CLPSO is a valid variant of PSO (Liang et al., 2006). The main difference between CLPSO and PSO is that the original PSO requires the use of *pbest* and *gbest*, whereas for CLPSO, updating the location only requires *pbest*. For PSO, *pbest* only needs its own *pbest* in CLPSO update, which can come from other individuals. The speed update formula is shown as follows:

$$v_i^{k+1} = v_i^k + c * r * (pbest_{f_i(d)}^k - x_i^k) \quad (21)$$

where c is the acceleration factor. r is also uniformly distributed random numbers in the interval [0,1]. $\mathbf{f_i} = [f_i(1), f_i(2), \cdots, f_i(D)]$ is determined by the probability of population P_c of randomly selected particles i. $pbest_{f_i(d)}^k$ represents the pbest value of particle, which is stored in the list $\mathbf{f_i}$ of the particle i of the dth dimension. P_c is calculated, as

shown in Equation (22); 0.05 and 0.45 are the optimal values of hyperparameters set by Liang et al. (2006) based on experience.

$$P_c = 0.05 + 0.45 * \frac{e^{\left(\frac{10(i-1)}{ps-1}\right)}}{e^{10} - 1} \tag{22}$$

where *ps* is the population size.

In PCLPSO, the particle population is divided into multiple groups, including a master group and several slave groups. All clusters run PCLPSO in the cluster environment at the same time. The *gbest*, *lbest*, and *pbest* in PCLPSO are defined as follows: *gbest* is the global optimal solution for all groups, *lbest* is the local optimal solution for populations, and *pbest* is the optimal solution for particles. Each slave group sends its *lbest* to the master group. The master group chooses the best solution from all *lbest* as *gbest* and sends *gbest* to all slaves. Each slave group randomly selects a particle to receive *gbest* to update its own *pbest*. PCLPSO algorithm updates its own *pbest* by a parallel distributed collaborative strategies. They are improving the quality of the solution and the rate of solving. Each particle in the PSO algorithm is updated by updating *pbest* and *gbest* values. The *gbest* affects the direction of population, and when it falls into a local minimum, the swarm particles tend to fall into this local minimum. PCLPSO adopts a comprehensive learning strategy, and the speed and position of the updated particles depend on all other particles. The *lbest* is selected from the *pbest* of the slave group, and the main group selects the optimal solution *gbest* from all the *lbest* of the slave group to ensure that it will not fall into a local minimum.

Construction of a Parallel Distributed Training Environment

The particles in the group in the PCLPSO algorithm are all running on a single computer, but each football robot in a RoboCup3D match is controlled by a separate client. Therefore, the PCLPSO algorithm cannot be directly applied to the RoboCup3D simulation platform. Because all clients are connected to the RoboCup3D simulated football server, the client and server can run on multiple computers in a distributed environment. Therefore, the PCLPSO algorithm is decomposed into three algorithms by changing the cluster structure to accommodate the RoboCup3D simulation platform in this paper.

In the new cluster, the structure is still based on the master–slave model of parallel computing; only one is a master cluster; the others are slaves, as shown in **Figure 3**. The gait training task of the humanoid robot is decomposed into multiple processes in the slave cluster, and a distributed training system is built to run on multiple computers. The parallel and distributed optimization framework can reduce the scale of solving gait problems. When all slaves are started and connected to the master group, and the master group sends parameters to the slave group, the communication cycle begins. Communication takes place only between the master group and slave groups; there is no communication between slave groups. In the cluster structure of this paper, the master node has no group, and its main function is to send initial parameters to the group of slave nodes (Algorithm 2, line 6), and the slave group sends its own *lbest* to the master group (Algorithm 2, line 8). The master group collects the received *lbest,* including its own *lbest,* into a pool called Elite Pool (EP). The master group finds *gbest* from the EP and eventually sends *gbest* to the slave group (Algorithm 2, lines 9, 10, and 11). The detailed algorithm of the master is shown in Algorithm 2. The slave group exchange algorithm is shown in Algorithm 3. Add an exchange program to slave group clients for exchanging data between the master client and all the clients of the slave group. In this paper, Local Pool (LP) is added to the slave group to collect all *pbest* in the group. The slave group finds *lbest* among all *pbest* and sends its own *lbest* to the master group. The master server adds them to EP, finds *gbest*, and shares it with the slave server. A robot is a member of a group during gait optimization training, a slave group has seven clients, and a client controls a humanoid robot, and eventually, multiple humanoid robots are trained in a RoboCup3D simulation court. The group client is used to calculate ZMP trajectory, COM trajectory, and swing leg trajectory when a humanoid robot is trained by the PCLPSO algorithm to walk. This part also calculates joint angle information and adaptation values for a cycle of humanoid robot walking. The detailed algorithm from the group client is shown in Algorithm 4. The gait optimization process for the client-based PCLPSO algorithm is shown in **Figure 4**.

Algorithm 2 Master algorithm

1. int *n*, *k*, *P*, *m*
2. double *w*, *c1*, *c2*
3. Array EP
4. Initialize the parameters *n*, *k*, *P*, *m*, *w*, *c1*, *c2*
5. Wait until all slave swarms have been launched and connected to this Master
6. The master sends the parameters to the slaves
7. **repeat**
8. Wait until all slave swarms have sent *lBest* to the master
9. The master stores the *lBests* into the EP
10. The master finds the *gBest* in the EP and empties the EP
11. The master sends the *gBest* to the slaves
16. **until** the stopping criterion is met
17. **return**

Algorithm 3 Switching algorithm of a slave swarm

1. int *n*, *k*, *P*, *m*
2. double *w*, *c1*, *c2*
3. Array LP
4. Connect to the Master until the parameters are received from the master
5. int $d = 0$
6. **repeat**
7. *d*++
8. Store the *pBests* into the LP until all *pBests* in this slave swarm are received
9. **if** *d* mod *P*==0 **then**
10. Find the *lBest* in the LP
11. Send the *lBest* to the master
12. Wait until the *gBest* is received from the master
13. Randomly update a *lBest* with the *gBest* in the LP
14. **end if**
15. Send the LP to the all clients in this swarm
16. **until** the stopping criterion is met
17. **return**

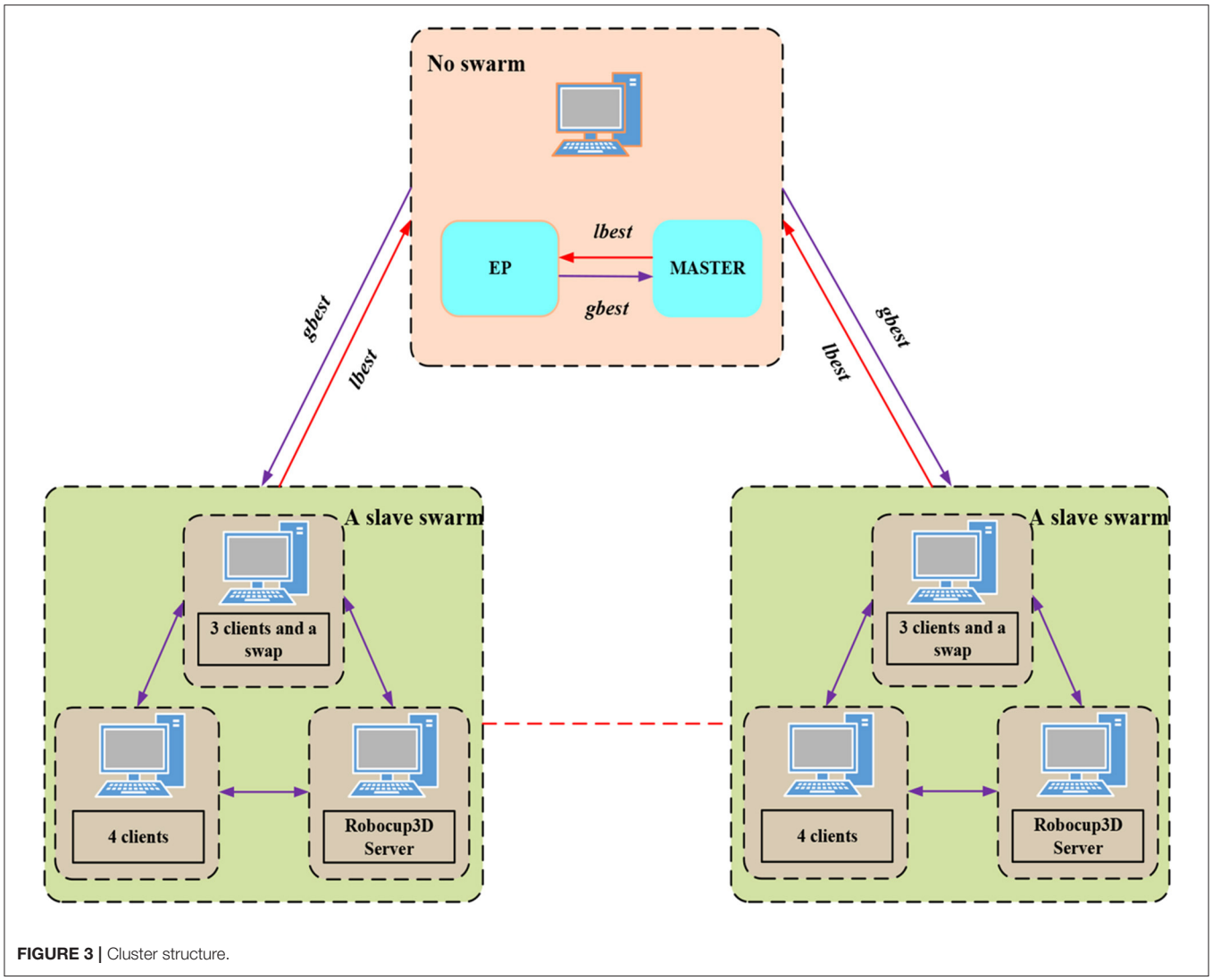

FIGURE 3 | Cluster structure.

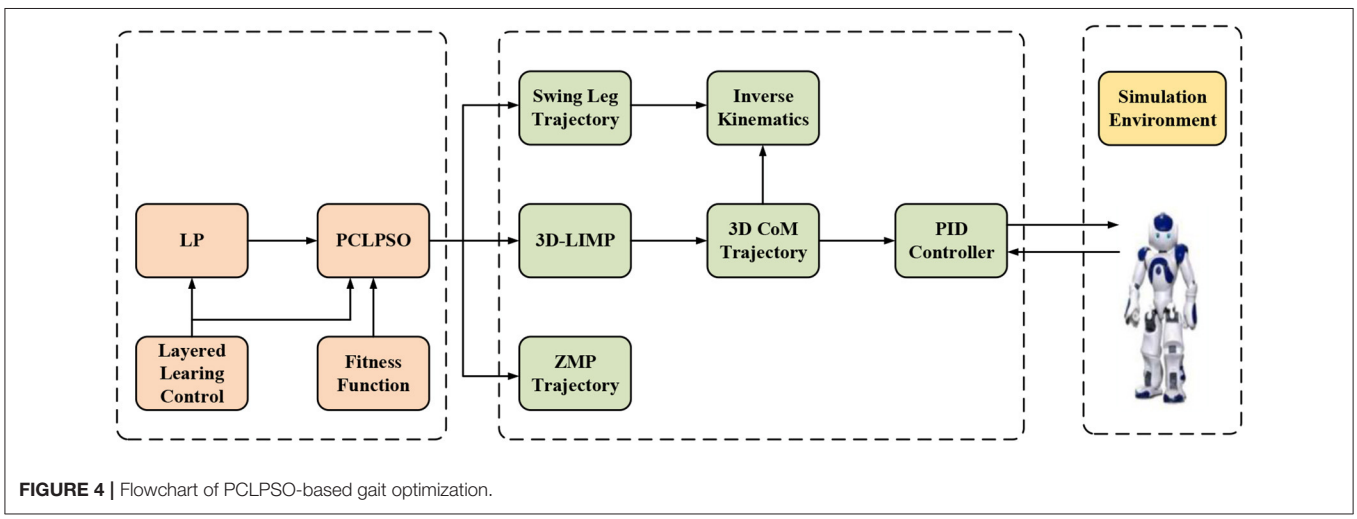

FIGURE 4 | Flowchart of PCLPSO-based gait optimization.

Algorithm 4 Algorithm in a client of a slave swarm

1. int *n*, *k*, *P*, *m*
2. double *w*, *c1*, *c2*
3. Array LP
4. Connect to the swap program in a swarm
5. Wait until the parameters are received from swap program in a swarm
6. Initialize the parameters′ and velocity
7. Calculate Swing leg Trajectory and all joint angles of Robot NAO in a walk cycle
8. Humanoid robot performs a walking training and calculates the fitness value
9. Update *pBest*
10. Send the *pBest* to the swap program until the LP is received from the swap program
11. Update the LP and find the *lBest* in the LP
12. **repeat**
13. Update the velocity and position
14. Calculate Swing leg Trajectory and all joint angles of Robot NAO in a walk cycle
15. Humanoid robot performs a walking training and calculates the fitness value
16. Update *pBest*
17. Send the *pBest* to the swap program until the LP is received from the swap program
18. Update the Local Pool and find the *lBest* in the LP
19. **until** the stopping criterion is met
20. **return**

Design of Evaluation Functions

The design of evaluation criteria is critical to achieving an excellent result (MacAlpine and Stone, 2018). In this paper, gait optimization results are judged based on the following criteria:

1. The distance the humanoid robot travels per unit time.

$$f_{dis} = a_1 * \|\varphi_{end} - \varphi_{start}\| \tag{23}$$

where φ_{start} and φ_{end} are the coordinates of start and end of the training, respectively, and a_1 is the weight.

2. Whether zero force matrix is always within the supported polygon and robot walks without falling all the time. in this paper, the zmp coordinates of the walking action sequence are chosen as the basis for calculation.

$$f_{zmp} = a_2 * \sum_{k=1}^{N} \sqrt{(P_x(k))^2 + (P_y(k))^2} \tag{24}$$

where $P_x(k)$ and $P_y(k)$ are the ZMP coordinates for each gait action sequence, and a_2 is the weight.

3. Humanoid robot in the fast walking process torso always maintains stability; body torso does not shake. In a simulation match, the body of players is in frequent contact with each other, and it is very easy to be knocked down by the opponent if the torso is not stable. At the initial time of double-support phase, the expected com coordinates are 2 ft on the center. In this paper, torso sway is detected by comparing the two coordinates of com and the center of feet during the initial stage of the dual support phase.

$$x_f = c_x - \frac{x_{foot}R + x_{foot}L}{2} \tag{25}$$

$$f_{shake} = \begin{cases} 0 & f_{abs}(x_f) < th_\theta \\ c & otherwise \end{cases} \tag{26}$$

where $x_{foot}R$ and $x_{foot}L$ are the coordinates of initial time 2 ft of the double support phase, th_θ is the set threshold, and f_{shake} and c are penalty and normal numbers, respectively.

If the robot falls during training, a constant $f_{falling}$ will be given as a penalty value. The evaluation function for speed and stability training is shown in Equation (27).

$$F_1 = f_{dis} - f_{zmp} - f_{falling} - f_{shake} \tag{27}$$

Add the lateral walking and the turning angles of the humanoid robot to the above, and the maximum distance of lateral walking for each step is limited to 0.04 m, and the maximum turning angle is 15°. In the optimization process, two different target points are set for the bipedal walking robot to train its gait. If the target point is reached, then the training of the next target point is quickly stopped. The evaluation function is as follows:

$$F_2 = f_{dis} - f_{falling} - f_{punish} + f_{reward} \tag{28}$$

where f_{punish} is the penalty for not completing the task within the specified time, and f_{reward} is the reward for completing the task.

Layered Learning

In the RoboCup3D simulation game, players can be roughly divided into two categories. The first category is to hold the ball, avoid the defense of the opponent, intercept the ball in the shortest time, and send the ball to the goal of the opponent; the second category is to run according to the situation on the field players and walk to the designated location as soon as possible. Players without the ball often complete tasks such as running and intercepting opposing players with the ball up and down. For players to complete the tasks assigned by their superiors in the shortest possible time, the walking speed of robots is very important. The player is the defensive object of an opponent after getting the ball, so frequent collisions and turns are inevitable. Improve the stability of the robot and steering ability to avoid falling and taking advantage of collisions. The omnidirectional walking of a biped robot can be decomposed into forward walking and steering motion. This paper uses a layered learning method to optimize each decomposition action of omnidirectional walking, and the final omnidirectional walking optimization is shown in **Figure 5**. Each sub-module requires the optimization algorithm to be trained through the corresponding evaluation function. Use manually adjusted parameters to drive the robot to walk, and then learn to get a fast mode with speed and stability as the goal. The final optimized parameters of the fast and stable mode are used as the initial state of the steering mode, and the learning is continued with the goal of steering stability. Through hierarchical learning of the humanoid robot, two different walking parameter sets (straight and turning) can be obtained. When different walking tasks are to be completed, the parameter sets can be switched at any time, and the flexible connection transition during switching is also obtained through learning.

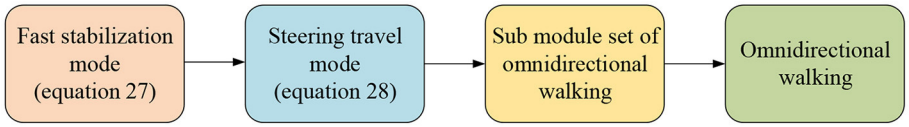

FIGURE 5 | Layered learning for omnidirectional walking process.

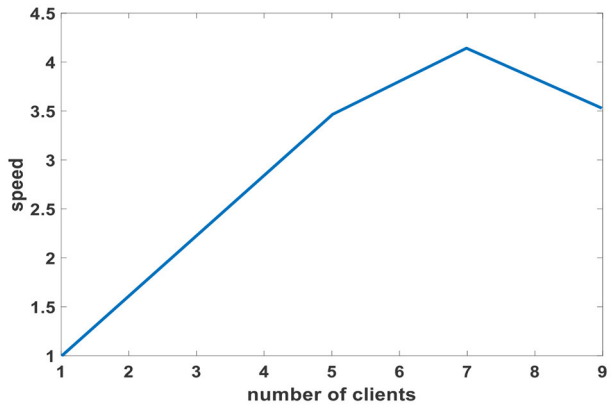

FIGURE 6 | Number of clients from the group and training speed.

FIGURE 7 | Training scenario diagram.

EXPERIMENTAL RESULTS AND ANALYSIS

This article uses a 3D-LIMP model to generate gait and opens the function of adjusting walking engine parameters to the public. OpenAI Gym acts as a bridge between optimization algorithm and environment and accepts the gait parameters computed by an optimization algorithm, performs a training session in the environment, and then provides the necessary information back to the optimization algorithm. To verify the effectiveness of the PCLPSO algorithm for gait optimization, this paper compares PCLPSO with three commonly used optimization algorithms, GA, CFO, and CMA-ES (Rongyi and Chunguang, 2018), in terms of the speed, stability, and turn capability of gait. The angle of view of players is only−120–120°. To get accurate data during training, the value of setViewCones in the server is changed from 120 to 360°. It is also necessary to change the game mode to fast mode during training so that the training is not limited by time. The mathematical properties of the four optimization algorithms of GA, CFO, CMA-ES, and PCLPSO are evolutionary algorithms. Each training uses the same population size (10) and the same number of variables (13). First, randomly produce 10 sets of parameters. The humanoid robot walks according to the 10 sets of parameters. The walking speed and stability will be different. Choose the optimal one, and then iterate the formula. Generate the next generation, and repeat the execution with 10 iterations. In this article, the coding method of the GA algorithm is standard binary coding. The number of parameters determines the length of a chromosome. When calculating the evaluation function value, the binary chromosome string should be decomposed and decoded to get the real number parameters. The specific parameters of the GA algorithm are as follows: take population size $N_p = 40$, evolutionary algebra $T = 200$, and crossover probability $P_c = 0.7$, and variation probability $P_m = 0.05$. The specific parameters of the CFO algorithm are as follows: $\alpha = 0.3, \beta = 0.3, \gamma_{start} = 0$, $\gamma_{stop} = 1$, and $\gamma = 0.1$.

Firstly, Equation (27) is chosen as an evaluation function to optimize the speed and stability of the robot. The relationship between the number of clients in a single slave group and training speed is shown in **Figure 6**. The training speed of multiple clients is compared to a single client when it takes a single client to train to an optimal value 1. The training speed increases linearly between 1 and 7 clients. The training speed reaches its maximum when the slave group contains seven clients. So, in the following comparison experiment, the slave group structures of PCLPSO are trained by seven clients controlling seven humanoid robots. The training scenario of a humanoid robot in the RoboCup3D scenario is shown in **Figure 7**.

Figure 8A shows the current optimal fitness values of four algorithms, all of which increase with the increase in the number of training. The evaluation function of the PCLPSO algorithm reaches the optimal value after 6,500 trainings, and the optimal value is 5.86. The evaluation function of the CMA-ES algorithm reaches the optimal after 14,000 trainings, and the optimal value is 4.73. GA algorithm reaches the optimal evaluation function after 17,500 trainings, and the optimal value is 3.92. The evaluation function of the CFO algorithm reaches the optimal after 16,000 trainings, and the optimal value is 4.2. The evaluation function value is averaged every 50 times of training, as shown in **Figure 8B**. As the number of iterations increases,

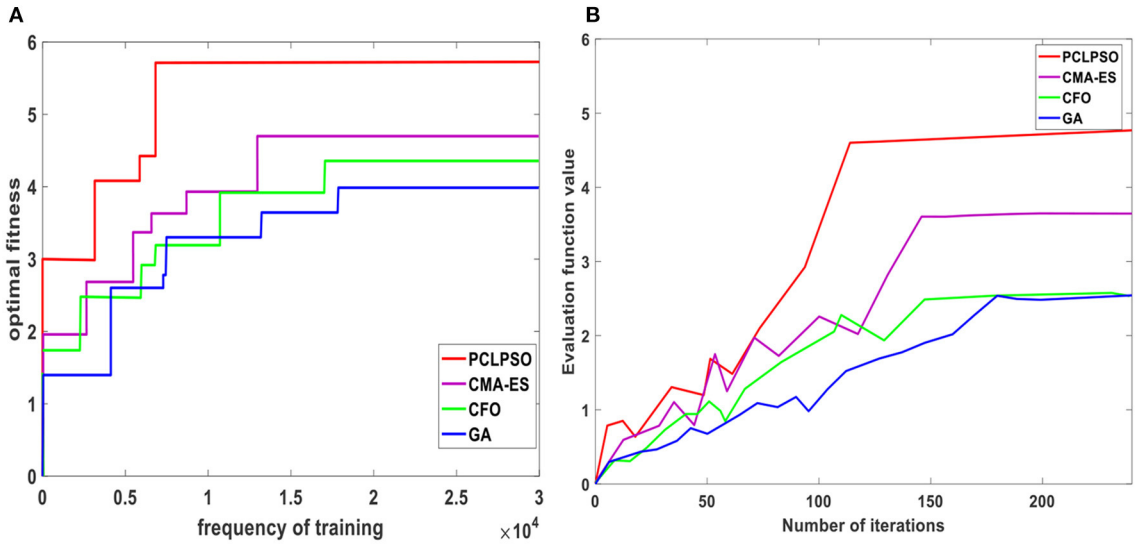

FIGURE 8 | Comparison of evaluation function values of four algorithms. **(A)** Current optimal evaluation function value. **(B)** Average function value.

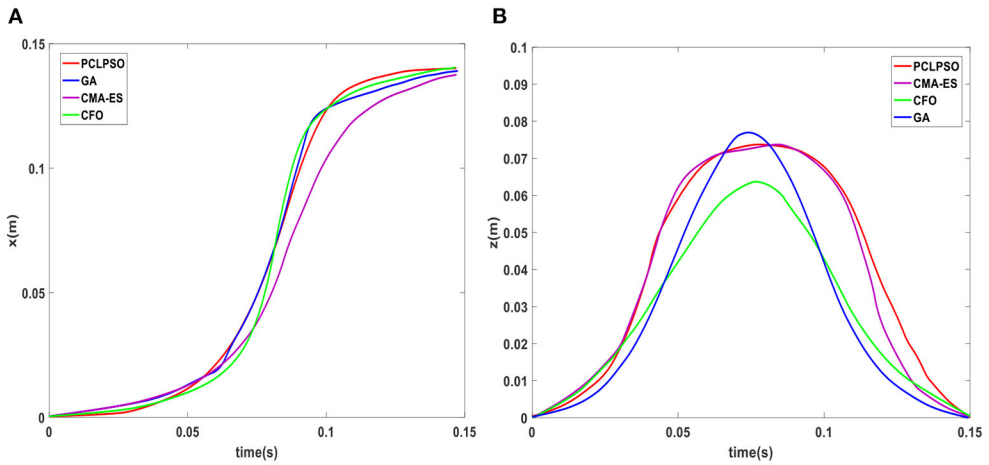

FIGURE 9 | (A) x-axis trajectory. **(B)** z-axis trajectory.

the evaluation function of all four algorithms increases steadily. One hundred twelve iterations for PCLPSO, 146 iterations for CMA-ES, 176 iterations for GA, and 148 iterations for CFO are stable. When the parallel distributed optimization framework interacts with RoboCup3D, many low-cost data samples can be obtained. Decomposition of gait problems can reduce the scale of problem-solving. Make sure to get a better solution faster.

The walking stability measured by Equation (25) shows that the x_f fluctuation range of the PCLPSO algorithm is−0.2 to 0.2 mm. The CMA-ES algorithm fluctuates between - 0.3 and 0.5. The x_f range of GA is−0.7 to 1.4 mm, and that of the CFO algorithm is−1.3 to 0.8 mm. The PCLPSO algorithm has the smallest x_f fluctuation range, and the humanoid robot is more stable. The optimized trajectories of swing leg x-axis and z-axis of four algorithms are shown in **Figures 9A,B**. At the moment of takeoff and landing, the optimized trajectory of the swing leg of the PCLPSO algorithm is parallel to the ground, which ensures stability when switching between the single-support phase and double-support phase.

The real-time changes of hip deflection pitch joint angle, hip transverse roll joint angle, and hip pitch joint angle for four algorithms are shown in **Figure 10**. The hip angle changes of the PCLPSO algorithm are very stable and better than those of the CMA-ES, CFO, and GA algorithms.

A comparison of changes in the center of mass landing points for four methods is shown in **Figure 11A**. The landing point of the center of mass in the double-leg support phase of the robot using the parameters optimized by the PCLPSO algorithm is always on the center of two-legged linkage and remains stable, whereas the landing point of the robot center of mass using

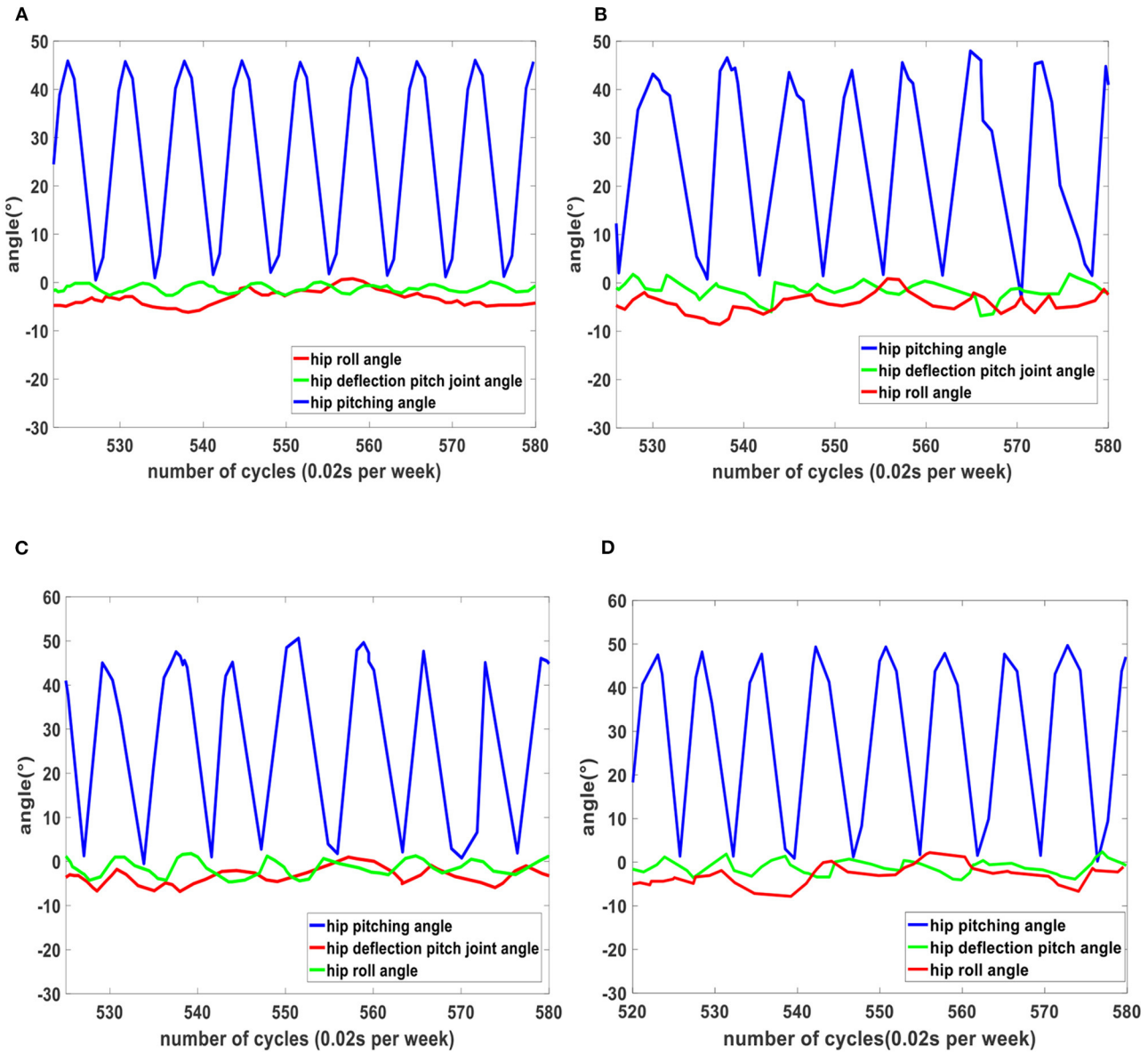

FIGURE 10 | Changes in hip angle for four algorithms. **(A)** Hip angle variation of PCLPSO algorithm. **(B)** Hip angle variation of CFO algorithm. **(C)** Hip angle variation of GA algorithm. **(D)** Hip angle variation of CMA-ES algorithm.

the parameters optimized by CFO and GA is unstable. CMA-ES algorithm optimized humanoid robot mass center fallout is more stable than the CFO and GA algorithms but inferior to the PCLPSO algorithm. ZMP trajectory after the optimization of three algorithms is shown in **Figure 11B**. The ZMP trajectory of the CMA-ES, GA, and CFO algorithms is close to the edge of the support polygon, whereas the whole trajectory curve of the PCLPSO algorithm moves toward the middle of the support polygon, in which case the stability margin of humanoid robot ZMP point is larger.

After the training mentioned earlier, it can be seen in humanoid robot gait optimization, and PCLPSO algorithm optimization of humanoid robot already has a fast and stable gait, but in simulation competition in a humanoid robot, the target point is constantly changing. When changing from a linear to a rotating state, the average speed decreases again when the rotation angle is small, and the humanoid robot is extremely unstable during rapid stops. In this paper, to better adapt to dynamic walking, layer learning is used to further optimize gait. The robot not only has a fast and stable gait but also has excellent steering ability. Next, the steering ability of the humanoid robot is optimized using Equation (28) as an evaluation function. Test experiments on the turning ability of humanoid robots were conducted under the SimSpark platform of RoboCup3D, and the

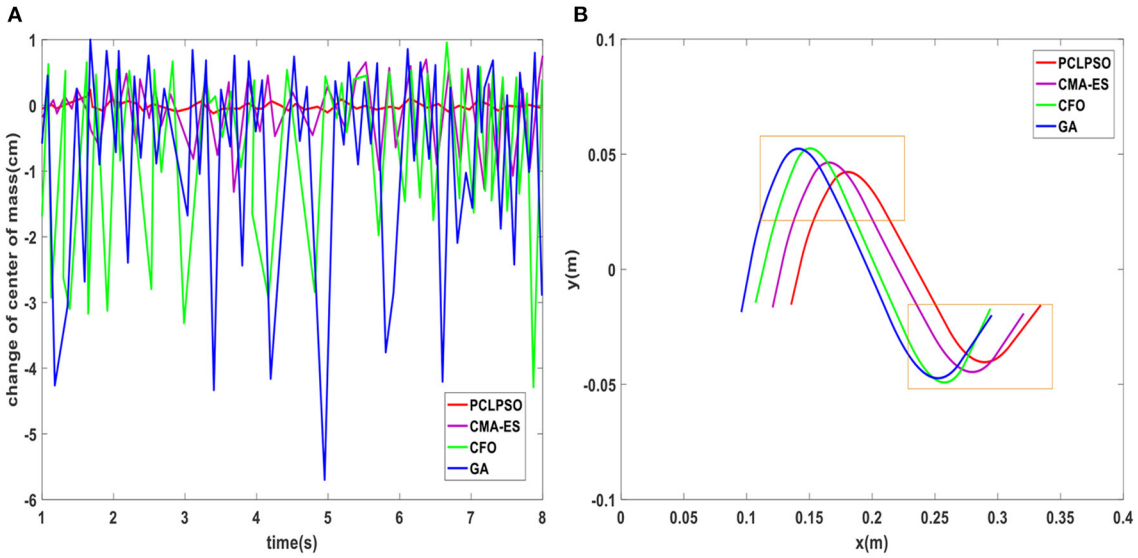

FIGURE 11 | (A) Humanoid robot center of mass landing point p_{shake} variation. **(B)** ZMP trajectory after optimization of four algorithms.

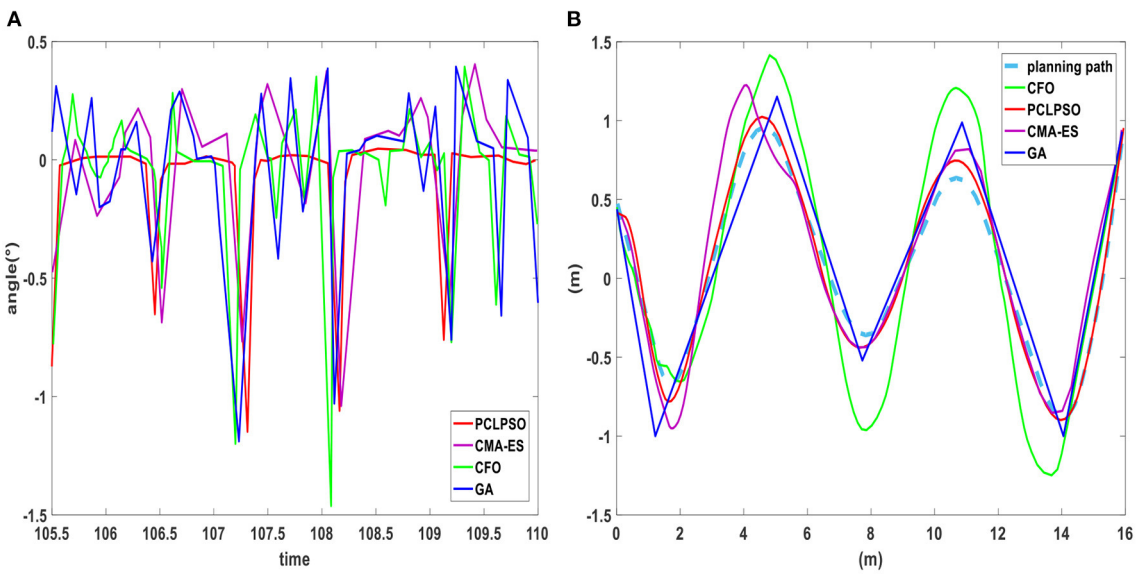

FIGURE 12 | (A) Body rotation angle. **(B)** Turning performance test.

walking path was recorded by Matlab using RoboViz observation. As shown in **Figure 12A**, the body rotation of four algorithms when the robot is optimized to make it turn continuously. The body rotation angle of robots using the PCLPSO algorithm with optimized parameters reaches a maximum of−1.19° during support leg switching, which shows that the robot is very stable during the switching process of walking and turning. Preplanning the walking path of the robot, the trajectory after layer learning using three algorithms is shown in **Figure 12B**. The GA algorithm walks a trajectory that cannot be flexible enough to maintain stability when turning. The humanoid robot optimized with the CMA-ES and CFO algorithms can turn flexibly but requires some adjustment time when turning, and the PCLPSO algorithm walks on a smooth trajectory that is almost identical to the planned path.

Let humanoid robot go straight for 15 m, test each algorithm 100 times, and take the average value. The results are shown in **Table 2**. The PCLPSO algorithm is used to walk with less time and faster speed under the same walking distance. The steering angle represents the angle between the body orientation of the initial position of the humanoid robot and the target point. The four algorithms are tested 100 times when the steering angle is

TABLE 2 | Straight speed comparison.

Algorithm	Walking distance (m)	Average time (s)	Average speed (m/s)
PCLPSO	15	16.83	0.89
CMA-ES	15	21.02	0.71
CFO	15	21.05	0.71
GA	15	25.03	0.60

TABLE 3 | Steering performance test.

Algorithm / time (s) / Angle(°)	PCLPSO	CMA-ES	CFO	GA
30°	21.07	21.32	21.37	22.47
45°	21.45	22.01	22.38	22.58
60°	21.75	22.45	22.67	23.07
90°	22.40	22.73	22.85	23.39

30, 45, 60, and 90°. The average time to reach the target point of each algorithm is shown in **Table 3**. The PCLPSO algorithm has the shortest average time to reach the target point at four angles.

CONCLUSION

This paper has built a RoboCup3D parallel distributed multi-robot gait training environment based on PCLPSO. A layer learning approach was used to optimize the existing problems in layers. It effectively reduces the influence of similar noise of a single simulation platform and has a faster optimization speed than common optimization algorithms. The final experimental results show that the PCLPSO algorithm optimizes faster and walks more quickly and steadily. During turning motion, the PCLPSO algorithm walks with a smoother trajectory and a smaller body rotation angle when switching between straight motion and turning motion, enabling stable, and flexible turning of a humanoid robot. This algorithm can also be extended to other aspects of RoboCup3D, such as the optimization of basic movements of humanoid robots such as goal shooting.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary materials, further inquiries can be directed to the corresponding author/s.

AUTHOR CONTRIBUTIONS

CT: methodology. JX: software. ZZ: funding acquisition, project administration, and supervision. FC and HG: investigation. CL: validation. All authors contributed to the article and approved the submitted version.

FUNDING

This work was supported in part by the National Natural Science Foundation of China (Grants No. 61801323), by the Science and Technology Projects Fund of Suzhou (Grant No. SYG201708, Grant No. SS2019029), by the Construction System Science and Technology Fund of Jiangsu Province (Grant No. 2017ZD066), by the Jiangsu Collaborative Innovation Center for Cultural Creativity (Grant No. XYN1703) and by the Natural Science Foundation of Changzhou Institute of Technology (Grant No. YN1603).

REFERENCES

Astudillo, D., Minchala, L. I., Astudillo-Salinas, F., Vazquez-Rodas, A., and Gonzalez, L. (2018). A simple mapping methodology of gait biomechanics for walking control of a biped robot. In: *2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)* (Lima: IEEE), 1-4. doi: 10.1109/INTERCON.2018.8526395

Bai, L., Hu, H., Chen, X., Sun, Y., Ma, C., and Zhong, Y. (2019). Cpg-based gait generation of the curved-leg hexapod robot with smooth gait transition. *Sensors* 19:3705. doi: 10.3390/s19173705

Baoping, W., Yuqian, D., Xiaoying, S., and Jiayi, L. (2015). Study of humanoid robot gait based on human walking captured data. In: *2015 34th Chinese Control Conference (CCC)* (Hangzhou: IEEE), 4480–4485. doi: 10.1109/ChiCC.2015.7260332

Bonyadi, M. R., and Michalewicz, Z. (2017). Particle swarm optimization for single objective continuous space problems: a review. *Evol. Comp.* 25, 1–54. doi: 10.1162/EVCO_r_00180

Elhosseini, M. A., Haikal, A. Y., Badawy, M., and Khashan, N. (2019). Biped robot stability based on an a-c parametric whale optimization algorithm. *J. Comp. Sci.* 31, 17–32. doi: 10.1016/j.jocs.2018.12.005

Faraji, S., Razavi, H., and Ijspeert, A. J. (2019). Bipedal walking and push recovery with a stepping strategy based on time-projection control. *Int. J. Rob. Res.* 38, 587–611. doi: 10.1177/0278364919835606

Fayong, G., Jianghai, Z., Tao, M., Minzhou, L., and Xiaobo, S. (2014). A modified gait generator for humanoid robots based on height compensation of center of mass. In: *2014 IEEE InternationalConference on Robotics and Biomimetics (ROBIO 2014)* (Bali: IEEE), 1278–1283. doi: 10.1109/ROBIO.2014.7090509

Graf, C., and Röfer, T. (2011). A center of mass observing 3d-lipm gait for the robocup standard platform league humanoid. In: *Robot Soccer World Cup* (Berlin; Heidelberg: Springer), 102–113. doi: 10.1007/978-3-642-32060-6_9

Grzelczyk, D., Stańczyk, B., and Awrejcewicz, J. (2016). Prototype, control system architecture and controlling of the hexapod legs with nonlinear stick-slip vibrations. *Mechatronics* 37, 63–78. doi: 10.1016/j.mechatronics.2016.01.003

Gülcü, S., and Kodaz, H. (2015). A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization. *Eng. Appl. Art. Intell.* 45, 33–45. doi: 10.1016/j.engappai.2015.06.013

Hereid, A., Hubicki, C. M., Cousineau, E. A., and Ames, A. D. (2018). Dynamic humanoid locomotion: a scalable formulation for hzd gait optimization. *IEEE Trans Rob.* 34, 370–387. doi: 10.1109/TRO.2017.2783371

Hong, Y.-D., and Lee, K.-B. (2016). Stable walking of humanoid robots using vertical center of mass and foot motions by an evolutionary optimized central pattern generator. *Int. J. Adv. Rob. Syst.* 13:27. doi: 10.5772/62039

Huan, T. T., and Anh, H. P. H. (2015). Novel stable walking for humanoid robot using particle swarm optimization algorithm. In: *2015 International Conference on Artificial Intelligence and Industrial Engineering* (Phuket: Atlantis Press). doi: 10.2991/aiie-15.2015.90

Huan, T. T., Thy, K. B., Trung, N. H. H., and Anh, H. P. H. (2018a). Stable gait optimization for small-sized humanoid robot using cfo. In: *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. doi: 10.1109/ICARCV.2018.8581188

Huan, T. T., Van Kien, C., Anh, H. P. H., and Nam, N. T. (2018b). Adaptive gait generation forhumanoid robot using evolutionary neural model optimized with modified differential evolution technique. *Neurocomputing* 320, 112–120. doi: 10.1016/j.neucom.2018.08.074

Jadidi, M. G., and Hashemi, E. (2016). Optimal preview control of the nao biped robot using a ukf-based state observer. In: *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)* (Banff, AB: IEEE), 52-57. doi: 10.1109/AIM.2016.7576742

Kajita, S., Nagasaki, T., Yokoi, K., Kaneko, K., and Tanie, K. (2002). Running pattern generation for a humanoid robot. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)* (Washington, DC: IEEE), 2755–2761.

Kennedy, J., and Eberhart, R. (1995). Particle swarm optimization. In: *Proceedings of ICNN'95-International Conference on Neural Networks (IEEE)*, 1942–1948. doi: 10.1109/ICNN.1995.488968

Kumar, P., Mukherjee, S., Saini, R., Kaushik, P., Roy, P. P., and Dogra, D. P. (2018). Multimodal n gait recognition with inertial sensor data and video using evolutionary algorithm. *IEEE Trans. Fuzzy Syst.* 27, 956–965. doi: 10.1109/TFUZZ.2018.2870590

Liang, J. J., Qin, A. K., and Suganthan, P. N. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comp*. 2005:281–295. doi: 10.1109/TEVC.2005.857610

MacAlpine, P., Depinet,M., and Stone, P. (2015). "Ut austin villa 2014: Robocup 3d simulation league champion via overlapping layered learning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*. doi: 10.1007/978-3-642-39250-4_8

MacAlpine, P., and Stone, P. (2018). Overlapping layered learning. *Art. Intell.* 254, 21–43. doi: 10.1016/j.artint.2017.09.001

Mandava, R. K., and Vundavilli, P. R. (2018). Near optimal pid controllers for the biped robot while walking on uneven terrains. *Int. J. Autom. Comp.* 15, 689–706. doi: 10.1007/s11633-018-1121-3

Muniz, F., Maximo, M. R., and Ribeiro, C. H. (2016). Keyframe movement optimization for simulated humanoid robot using a parallel optimization framework. In: *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)* (Recife: IEEE), 79-84. doi: 10.1109/LARS-SBR.2016.20

Paparisabet, M., Dehghani, R., and Ahmadi, A. (2019). Knee and torso kinematics in generation of optimum gait pattern based on human-like motion for a seven-link biped robot. *Multi. Syst. Dynam.* 47, 117–136. doi: 10.1007/s11044-019-09679-z

Rongyi, H., and Chunguang, L. (2018). Gait optimization of robocup3d simulation robot. *Comp. Modern.* 3, 20–25 doi: 10.3969/j.issn.1006-2475.2018.03.004

Sato, T., Sakaino, S., and Ohnishi, K. (2010). Real-time walking trajectory generation method with three-mass models at constant body height for three-dimensional biped robots. *IEEE Trans. Indust. Electr.* 58, 376–383. doi: 10.1109/TIE.2010.2052535

Tang, Z., Zhou, C., and Sun, Z. (2003). Trajectory planning for smooth transition of a biped robot. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)* (Taipei: IEEE), 2455–2460.

Wang, M., Dong, H., Li, X., Zhang, Y., and Yu, J. (2019). Control and optimization of a bionic robotic fish through a combination of cpg model and pso. *Neurocomputing* 337, 144–152. doi: 10.1016/j.neucom.2019.01.062

Weon, I.-S., and Lee, S.-G. (2018). Intelligent robotic walker with actively controlled human interaction. *ETRI J.* 40, 522–530. doi: 10.4218/etrij.2017-0329

Winkler, A. W., Bellicoso, C. D., Hutter, M., and Buchli, J. (2018). Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Rob. Autom. Lett.* 3, 1560–1567. doi: 10.1109/LRA.2018.2798285

Zhong, G., Chen, L., Jiao, Z., Li, J., and Deng, H. (2017). Locomotion control and gait planning of a novel hexapod robot using biomimetic neurons. *IEEE Trans. Control Syst. Technol.* 26, 624–636. doi: 10.1109/TCST.2017.2692727

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.