A survey of research on several problems in the RoboCup3D simulation environment

Zhongye Gao
State Key Laboratory for Novel Software Technology
Mengjun Yi
State Key Laboratory for Novel Software Technology
Ying Jin
jinying@nju.edu.cn

State Key Laboratory for Novel Software Technology
Hanwen Zhang
State Key Laboratory for Novel Software Technology
Yun Hao
State Key Laboratory for Novel Software Technology
Ming Yin
State Key Laboratory for Novel Software Technology
Ziwen Cai
State Key Laboratory for Novel Software Technology
Furao Shen
State Key Laboratory for Novel Software Technology

update

Keywords: RoboCup3D, skill generation and optimization, multi-humanoid robot cooperation, Multi-agent

Posted Date: May 16th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2925677/v1

Additional Declarations: No competing interests reported.

A survey of research on several problems in the RoboCup3D simulation environment

Zhongye Gao[1,2], Mengjun Yi[1,3], Ying Jin[1,2*], Hanwen Zhang[3], Yun Hao[3], Ming Yin[3], Ziwen Cai[3], Furao Shen[1,3]

[1*]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210000, China.

[2]Department of Computer Science and Technology, Nanjing University, Nanjing 210000, China.

[3]School of Artificial Intelligence, Nanjing University, Nanjing 210000, China.

*Corresponding author(s). E-mail(s): jinying@nju.edu.cn;
Contributing authors: gaozhongye@163.com;
mengjunyi@smail.nju.edu.cn; leoarthurzhw@hotmail.com;
191300015@smail.nju.edu.cn; 191300075@smail.nju.edu.cn;
191300001@smail.nju.edu.cn; frshen@nju.edu.cn;

Abstract

In the process of robot research and development, due to the vulnerability of hardware, simulation environment is often used to verify and test algorithms first. RoboCup3D simulation environment is developed based on open dynamic engine, and the humanoid robot NAO is modeled as the main robot, which provides a simulation platform for humanoid robot researchers to study robot movements. At the same time, it is also the official platform of RoboCup 3D events. Under the rules of soccer robot competition, it is helpful for the research of multi-robots, especially multi-humanoid robots' cooperation strategy. This paper summarizes the related research in RoboCup3D simulation environment, and first introduces the basic problems existing in this simulation environment. Secondly, the research of robot motion generation and optimization based on model and non-model in simulation environment is introduced respectively. Then, it introduces the related research of cooperation strategy design of multi-humanoid robots under RoboCup3D rules, including positioning, dynamic role assignment, etc. And sort out a typical practical solution to the above problems; Finally, the future development trend of related research in RoboCup3D simulation environment is analyzed.

Keywords: RoboCup3D, skill generation and optimization, multi-humanoid robot cooperation, Multi-agent

1 Introduction

The RoboCup 3D simulation environment is developed based on a general physics multi-agent simulation system called Simspark[1]. It uses the Open Dynamics Engine (ODE) to detect collisions and simulate rigid body dynamics, allowing for accurate simulation of physical properties such as velocity, inertia, and friction of objects. It also provides modeling support for hinge joints in humanoid robots in the simulation environment. In 2004, the simulation environment began to be used for the simulated 3D events in the Robot World Cup (RoboCup), designed and maintained with reference to actual human football rules. The physical simulation of humanoid robots was developed to enable realistic simulation of football matches, and eventually developed into a full-field football simulation environment using the 11 vs 11 format, carried by the humanoid robot Nao (produced by the French company Aldebaran Robotics, widely used as a platform for humanoid robot research), as shown in Fig. 1.

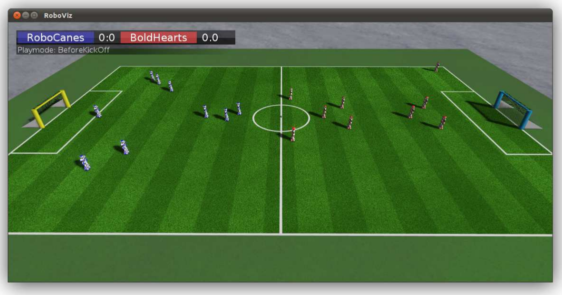


Fig. 1 RoboCup3D Simulation

RoboCup has developed into a professional robot competition with a very high level of comprehensive technology and a very wide range of participation, and has a great influence in the field of robotics worldwide. It is organized by the RoboCup International Federation, proposed by university professors such as Hiroaki Kitano, Manuela Veloso, and Minoru Asada, established in 1996, headquartered in Tokyo, Japan, and officially registered in Bern, Switzerland, as an international research and education organization. RoboCup is held annually with the aim of promoting research in robots and artificial intelligence by providing an attractive but challenging challenge to the public. The goal of RoboCup is to have a fully autonomous team of humanoid robot

[1] http://simspark.sourceforge.net/

football players capable of winning against the latest FIFA World Cup champions under official FIFA rules by the mid-21st century. To date, RoboCup includes events in multiple categories, including simulated, middle-sized, small-sized, humanoids, and four-legged robots. Each category is derived from tasks aimed at achieving the ultimate goal of RoboCup. The difficulty of the tasks varies by category but complements each other and progressively advances towards achieving the ultimate goal of RoboCup. The emergence of the simulated category was to promote more solutions to be tested and implemented in humanoid robots, on the one hand, avoiding the wear and tear of real robots, and on the other hand, allowing algorithms such as reinforcement learning that cannot be trained in a real environment to be tested and developed first in a simulated environment before gradually transitioning to real robots.

The RoboCup simulation league competition system adopts a client/server (C/S) architecture platform. The Soccer Server is provided and maintained by the RoboCup Technical Committee, and participants only need to develop client programs on the underlying code provided to control Nao robots for soccer simulation matches through programs. The difference between the RoboCup simulation league and other leagues is that it is entirely controlled by software programs and does not require physical robots. Therefore, researchers only need to consider software development and do not need to consider hardware-related issues, so they can focus more on the development of basic algorithms. The RoboCup simulation system has become an excellent platform for testing learning scenarios. The RoboCup Soccer Simulation 3D League based on this platform has also become one of the largest events in RoboCup. In the RoboCup 3D simulation environment, each simulated robot must be an independent process, as shown in Fig. 2, and it communicates with the soccer server through port 3100 using TCP or UDP protocols by default.

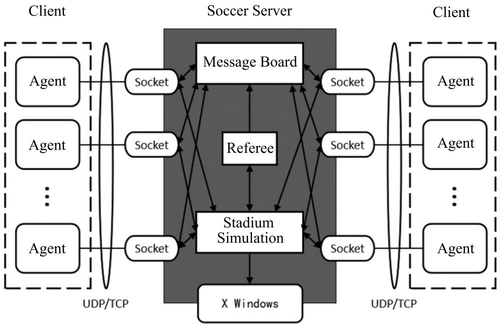


Fig. 2 C-S Structure diagram of RoboCup3D Simulation[1]

The RoboCup 3D simulation environment and its competition requirements have posed many questions and challenges for researchers. To achieve the goal of completing a match and winning, researchers mainly need to develop and design two aspects: action generation and optimization for simulated robots and multi-robot cooperative strategies, including but not limited to specific issues such as action generation and optimization methods for simulated robots, localization, obstacle avoidance, dynamic role allocation, soccer strategy design, and multi-agent cooperative design. The RoboCup 3D simulation environment, based on the Nao simulated robot and implementing the scene of multi-robot soccer matches, can help researchers develop and verify action planning and multi-agent cooperative algorithms freely in the simulation environment, free from the limitations of expensive robot hardware, based on physics engines and real soccer game rules. In addition, it is worth noting that research results in RoboCup 3D can be extended to other simulation environments [2] and real environments [3, 4], which makes research in the RoboCup 3D simulation environment more important, proving the importance and rationality of developing simulation environments and verifying algorithms. The RoboCup 3D simulation environment has attracted more and more researchers, and has gone through a period of development, with sufficient research results for key issues. This article summarizes these research results, and, based on known information, is the first article to review related research in the RoboCup 3D simulation environment.

Structure of the rest of this article is shown in Fig. 3. The first section introduces the basic issues related to research in the RoboCup 3D simulation environment. The second section introduces methods for action generation and optimization for simulated robots in the RoboCup 3D simulation environment, divided into two parts: model-based methods and non-model-based methods. The third section reviews research on strategy design aimed at winning the RoboCup 3D simulation competition. The fourth section provides a typical practical case to give a reference process for solving related problems in RoboCup 3D. The fifth section discusses future research trends, and the final section provides a summary.

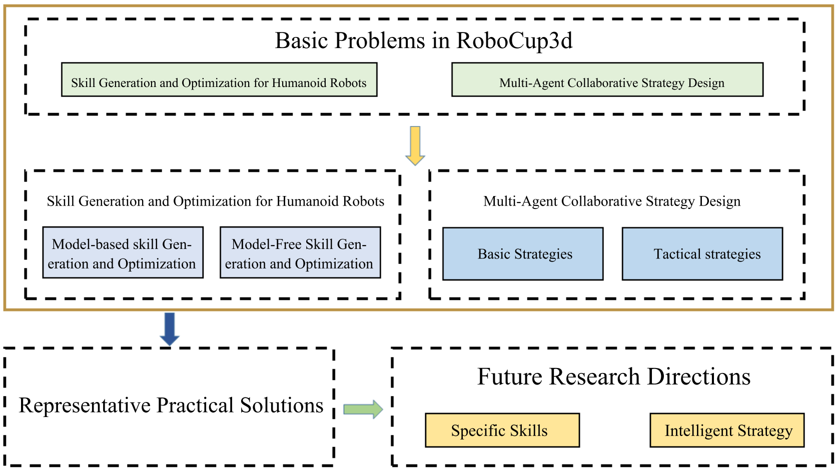

Basic Problems in RoboCup3d
Skill Generation and Optimization for Humanoid Robots
Multi-Agent Collaborative Strategy Design
Skill Generation and Optimization for Humanoid Robots
Model-based skill Generation and Optimization
Model-Free Skill Generation and Optimization
Multi-Agent Collaborative Strategy Design
Basic Strategies
Tactical strategies
Representative Practical Solutions
Future Research Directions
Specific Skills
Intelligent Strategy

Fig. 3 The structure and main content

2 Basic Problems in RoboCup3d

This section discusses some fundamental issues in research based on the RoboCup3D simulation environment, including the generation and optimization of skills for humanoid robots in the RoboCup3D simulation environment, as well as the design of strategies for achieving effective cooperation among multiple intelligent agents.

2.1 Skill Generation and Optimization for Humanoid Robots

In the RoboCup3D simulation environment, initially a sphere was used to represent a robot. By controlling the movement of the sphere through fixed actions in the simulation environment, a simple humanoid robot called Soccerbot with 20 degrees of freedom (DOF) was later modeled to better simulate real matches. Soccerbot has multiple sensors that fully utilize the physical characteristics of the RoboCup3D simulation environment. It can communicate with the simulation environment's server through a server/client architecture as shown in Fig. 2. It uses an omnidirectional visual sensor to obtain information about surrounding objects, force sensors to detect the force produced when the feet touch the ground, a gyroscope sensor to detect changes in the orientation of the body, joint sensors to obtain the angles of each joint, and a game-state sensor to obtain the environmental time. Using this sensory information, Soccerbot can determine its own state and the state of the external environment, and thus make decisions about how each joint should change its angle at each moment. It then sends the required joint angles back to the simulation environment's server, which calculates and displays them on the screen.

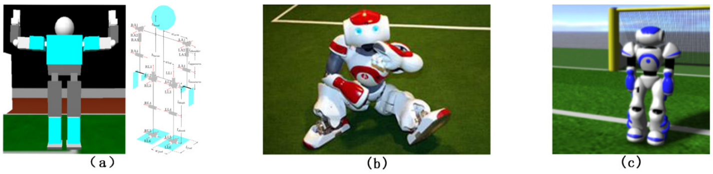


Fig. 4 (a)The Soccerbot in RoboCup3D (b) The real Nao (c) The Nao in RoboCup3D

However, Soccerbot's size is very unrealistic for a true humanoid robot, and the range of motion of the joints is not limited in this model, which leads to unrealistic movements and thus lack of realism. Due to these limitations, the Nao robot model, manufactured by Aldebaran Robotics of France, was first used in the RoboCup international league held in Suzhou in 2008. Nao is a humanoid robot with a height of about 57 centimeters and a weight of about 4.5 kilograms, with 22 degrees of freedom, making it highly maneuverable. Because it is more widely applicable, has a higher degree of similarity to humans, and is more suitable as a research carrier for RoboCup3D competitions, the Nao robot has been modeled and used in the RoboCup3D simulation environment to this day.

The Nao robot in RoboCup3D has rich sensors and effectors to perceive external information and execute corresponding joint transformations. The simulated Nao in the environment can track its radial and axial motion in three-dimensional space through a gyroscope and an accelerometer. Each foot has a force-sensitive resistor to detect contact force with the ground or other objects. There is a restricted visual sensor in the center of the head to obtain visual information of different objects in the environment. It is equipped with speech effectors and corresponding hearing sensors to achieve communication between robots. The current position of each joint is obtained by hinge joint sensors and transformed by corresponding hinge joint actuators.

The main goal of related research in the RoboCup3D environment is to complete a soccer robot match and win. To achieve this goal, stable and efficient soccer skills need to be developed, including walking, running, standing up, kicking, etc. The quality of these skills largely determines a team's performance in the RoboCup3D competition. Robots achieve different postures through changes in joint angles, and a complete action is a continuous combination of multiple postures. To generate a skill, the values of joint angles for each posture in the skill need to be designed. How to design these joint angle values is the skill generation problem of the robot. This article mainly focuses on the skill generation method for the Nao robot validated in the RoboCup3D simulation environment, using different methods to generate different skills. In addition, the stability, effectiveness, and whether closed-loop control can be achieved are all measures of the effectiveness of skill generation.

In this article, we will summarize the methods for generating robot skill based on dynamic analysis as model-based methods, including simplified models such as inverted pendulums [5–8], interpolation methods used in trajectory planning [9, 10], truncated fourier series [11–14], etc. This is because these methods all perform dynamic analysis on the robot and use the robot's dynamic stability as an important reference for generating motion. Other methods that involve little or no dynamic analysis, including reinforcement learning methods [15–17], CMAC neural network methods [18], and central pattern generator methods [19], are classified as model-free methods.

Although the initially generated skill is stable, its effectiveness is average. Optimizing the initially generated skill can make the robot walk faster and kick the ball farther, thereby obtaining better skill. Using these improved skills often makes the robot team perform better in strategy execution and actual matches, so how to optimize the skill of humanoid robots has become a key issue in RoboCup3D simulation environment research. The skill optimization of robots depends on the generation method. If the initial skill of the robot is generated based on a model-based method, the basic optimization idea is to extract some key parameters that directly determine the quality of the skill and then select optimization methods such as genetic algorithms [20] and evolutionary strategies [21] to optimize these parameters and improve the skill. However, in model-free skill generation methods, generation and optimization are often completed together or there is no clear process of skill generation. For example, reinforcement learning-based methods and central pattern generator methods use machine learning to learn from scratch to obtain good skill.

Optimizing the skills of humanoid robots involves not only the optimization of individual skills, but also the optimization of the connection between multiple skills due to the complexity of the RoboCup3D simulation environment. The instability caused by collisions between robots, balls, landmarks on the soccer field, and other objects also needs to be taken into account. Fig. 5 shows five different height models of Nao robots in the RoboCup3D simulation environment, and how to optimize skills for robots of different heights is also a problem to be solved.

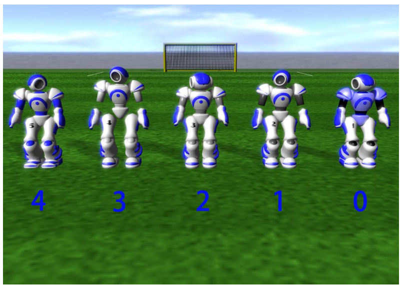


Fig. 5 Five Nao robots in RoboCup3D

2.2 Multi-Agent Collaborative Strategy Design

In the RoboCup3D simulation environment, each team can control up to 11 Nao robots. In the Robot World Cup competition, the organizing committee also stipulates that each team must participate with 11 Nao robots, and at least three different types of Nao robots(Fig. 5) must be included. These robots communicate individually with the server and can encrypt information through the server. Researchers need to design a cooperative strategy for each robot to control when to perform specific actions in order to complete the game and win.

This article divides strategy design in RoboCup3D into basic strategy design and tactical strategy design. In basic strategy, the positioning problem [22] must first be solved, including the positioning of the robot itself, other robots, the ball, and the landmarks on the football field. The Nao robots in the RoboCup3D simulation environment have a visually limited visual sensor. The server periodically communicates object information seen by the robot through the visual sensor, including its own torso, the ball, other robots, and landmarks on the football field. However, this information includes a defined Gaussian noise added on the server side, so solving the problem of accumulated errors from multiple positionings and noise interference on the server side is important.As in human soccer, formation design and role allocation are prerequisites for good decision-making design. The former needs to consider both global and local design, while the latter involves the assignment and strategic planning of the roles that the robots will take on during the game.

We categorize specific strategy design into tactical strategy design, which considers details such as dribbling and obstacle avoidance, formation maintenance, ball interception, and goalkeeper defense strategy [23]. More sophisticated global and local strategies are designed to improve offensive and defensive capabilities. Section 4.2 will provide a more detailed introduction to this part of the research.

3 Skill Generation and Optimization in RoboCup3D Simulation Environment

3.1 Model-based skill Generation and Optimization in RoboCup3D Simulation Environment

The stability of the robot during skill execution needs to be ensured to prevent continuous falling. Generally, the zero moment point (ZMP) based dynamic analysis method is chosen to determine whether the robot is stable during skill execution [8]. However, due to the complexity of the multi-link structure of humanoid robots, it is challenging to analyze their dynamics directly. Therefore, many studies use simplified models such as the gravity compensation inverted pendulum model [5], the two-centre-of-mass model [6], the multi-centre-of-mass model [6], and the three-dimensional linear inverted pendulum model [7, 8] to generate skills. After simplifying the humanoid robot into a model, its dynamic characteristics can be analyzed more easily and quickly. Keyframes for skills can be selected using the model, and basic skills can be designed. Interpolation methods can be used to design execution curves between keyframes to make skills smoother. The process of model-based skill generation and optimization is shown in Fig. 6. Firstly, a stable but slow-moving skill, such as walking, is generated for the robot using simplified models or direct trajectory planning. Then, optimization methods such as genetic algorithms and evolutionary strategies are used to improve the skill's performance. This paper mainly summarizes the model-based action generation and optimization methods for considering the robot's dynamic characteristics and corresponding optimization methods.

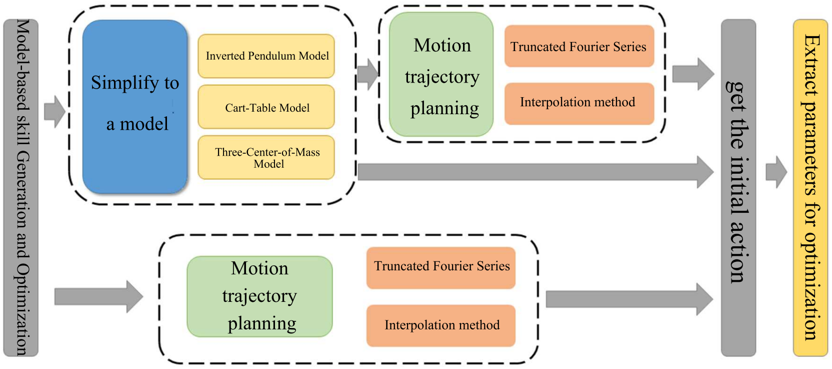


Fig. 6 Model-based skill generation and optimization

3.1.1 Inverted Pendulum Model

The inverted pendulum system is a multi-variable, fast, severely non-linear, and unstable system. The inverted pendulum model based on this system integrates knowledge from many disciplines such as mathematics, control theory, and electrodynamics and is a model of multiple cross-disciplines. In the late 1970s, Hemami et al. [24] proposed a humanoid robot control model, the inverted pendulum model, by imitating a first-order inverted pendulum model in three-dimensional space, matching the robot's legs with the pendulum's pole and the ankle joint torque drive with the base joint drive of the inverted pendulum model, thus linking the robot's walking and the inverted pendulum's upright swing. Researchers found that simulating the walking of humanoid robots using the inverted pendulum model greatly simplifies the problem, which has extraordinary significance in the field of gait research.

According to the number of pendulum rods, the inverted pendulum system can be divided into one-dimensional inverted pendulum, two-dimensional inverted pendulum, three-dimensional inverted pendulum, etc., and the pendulum rods of multi-level pendulum systems belong to free connections. To ensure that the inverted pendulum model moves linearly, the center of mass (COM) of the inverted pendulum model must maintain a constant height. This pendulum, which maintains the center of mass height constant by stretching the leg length, is called a linear inverted pendulum (LIPM). Seekircher et al. [25] implemented closed-loop control of robot walking based on the two-dimensional linear inverted pendulum model in the RoboCup3D environment. By evaluating the robot's state (velocity, acceleration, etc.) and using pre-defined control rules to update the parameters involved in the model, such as step length, they reduced the error between the model and the real robot and achieved closed-loop control. Shafii et al. [26] improved the two-dimensional linear inverted pendulum model by considering the influence of hip height on robot walking stability and implemented robot walking.

However, the two-dimensional linear inverted pendulum model only considers the dynamics of the robot in the front-back and up-down dimensions. In walking, robots often have movements in the left-right dimension as well. Therefore, the three-dimensional linear inverted pendulum model is a more comprehensive representation of the robot's motion. The three-dimensional linear inverted pendulum model approximates humanoid robots as a system composed of a mass point that concentrates all the mass and legs with no mass that connect the mass point to the support point [27].

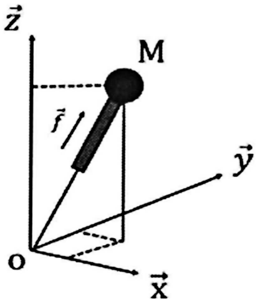


Fig. 7 Three-dimensional linear inverted pendulum[1]

Fig. 7 shows the mechanical analysis of the three-dimensional linear inverted pendulum.

Because the moment of force τ at the support point is equal to 0, the linkage can freely rotate at the support point, and the stretching force f allows the linkage to be scaled arbitrarily. The force f decomposition in the three-dimensional coordinate axis direction for the stretching force is as follows:

$$f_x = (\frac{x}{r})f \quad f_y = (\frac{y}{r})f \quad f_z = (\frac{z}{r})f \tag{1}$$

In equation (1), r represents the distance between the center of mass and the support point. Because only gravity and stretching forces act on the center of mass, equation (2) can be derived:

$$M\ddot{x} = (\frac{x}{r})f \quad M\ddot{y} = (\frac{y}{r})f \quad M\ddot{z} = (\frac{z}{r})f - Mg \tag{2}$$

In order to constrain the center of mass of the three-dimensional inverted pendulum, the equation for the constraint plane is defined as follows:

$$z = k_x x + k_y y + z_c \tag{3}$$

In equation (3), k_x and k_y control the slope of the constraint plane and z_c control the height of the constraint plane, respectively. In order for the center of mass to always be within the constraint plane, its acceleration must be perpendicular to the normal vector of the constraint plane, that is:

$$\begin{bmatrix} (\frac{x}{r})f & (\frac{y}{r})f & (\frac{z}{r})f - Mg \end{bmatrix} \begin{bmatrix} -k_x \\ -k_y \\ 1 \end{bmatrix} = 0 \tag{4}$$

According to equation (4), f can be solved, and according to equation (3), it can be obtained that:

$$f = \frac{Mg}{z_c r} \tag{5}$$

According to equation (5), the stretching force f is proportional to the length of the linkage r. Therefore, the motion of the center of mass within the constraint plane can be controlled by the stretching force f. The horizontal motion of the center of mass can be obtained as follows:

$$\begin{aligned} \ddot{x} &= \frac{g}{z_c}x \\ \ddot{y} &= \frac{g}{z_c}y \end{aligned} \tag{6}$$

Equations (5) and (6) are both linear equations that do not contain the slope parameters k_x, k_y of the constraint plane, but only contain the intercept parameter z_c of the constraint plane. Therefore, the inclination of the constraint plane is independent of the horizontal motion of the center of mass. This type of inverted pendulum model is called a three-dimensional linear inverted pendulum.

The humanoid robot can be simplified as a three-dimensional linear inverted pendulum model, and simple robot skills can be generated through dynamic analysis based on ZMP stability. The equation for the motion of the center of mass of the bipedal humanoid robot in the horizontal direction is given above, but in the actual motion process of the humanoid robot, the height of the center of mass on the Z-axis is often set to a fixed value [28]. Therefore, when the value of the intercept of the constraint plane along the Z-axis is constant, the equation for the motion of the center of mass along the horizontal direction of the humanoid robot can be obtained from equation (7) and (8):

$$\begin{cases} x(t) = x(0)\cosh(t/T_c) + T_c\dot{x}(0)\sinh(t/T_c) \\ \dot{x}(t) = x(0)/T_c\sinh(t/T_c) + \dot{x}(0)\cosh(t/T_c) \\ T_c = \sqrt{z_c/g} \end{cases} \tag{7}$$

$$\begin{cases} y(t) = y(0)\cosh(t/T_c) + T_c\dot{y}(0)\sinh(t/T_c) \\ \dot{y}(t) = y(0)/T_c\sinh(t/T_c) + \dot{y}(0)\cosh(t/T_c) \\ T_c = \sqrt{z_c/g} \end{cases} \tag{8}$$

Here, T_c is a constant determined by the height of the center of mass and the acceleration of gravity, and $x(0)$,$\dot{x}(0)$ are the initial position and velocity of the center of mass along the axis $\mathbf{X}$ at time 0 respectively, $y(0)$,$\dot{y}(0)$ are the initial position and velocity of the center of mass along the axis $\mathbf{Y}$ at time 0 respectively. In order to generate skills, it is also necessary to calculate the time required for the center of mass to move between adjacent positions. Given the initial conditions $(x_0, \dot{x}_0)$ and a certain target state $(x_1, \dot{x}_1)$, the equation representing the relationship between the two states can be obtained through equation (7).

$$\begin{aligned} x_1 &= \frac{x_0 + T_c\dot{x}_0}{2}e^{\frac{\varphi}{T_c}} + \frac{x_0 - T_c\dot{x}_0}{2}e^{\frac{-\varphi}{T_c}} \\ \dot{x}_1 &= \frac{x_0 + T_c\dot{x}_0}{2T_c}e^{\frac{\varphi}{T_c}} - \frac{x_0 - T_c\dot{x}_0}{2T_c}e^{\frac{-\varphi}{T_c}} \end{aligned} \tag{9}$$

The time φ it takes for the center of mass to go from the initial state $x(0)$,$\dot{x}(0)$ to the final state $y(0)$,$\dot{y}(0)$ is represented by equation (9). Two equations are derived from equation (9) and are shown below as equation (10).

$$x_1 + T_c\dot{x}_1 = (x_0 + T_c\dot{x}_0)e^{\varphi/T_c} \tag{10}$$

φ can be obtained by equation (11):

$$\varphi = T_c \ln \frac{x_1 + T_c\dot{x}_1}{x_0 + T_c\dot{x}_0} \tag{11}$$

or:

$$\varphi = T_c \ln \frac{x_0 - T_c\dot{x}_0}{x_1 - T_c\dot{x}_1} \tag{12}$$

In general, equation (11) and equation (12) represent the same result, but they are different in cases where the numerator and denominator approach zero in each equation.

In the RoboCup3D environment, the three-dimensional linear inverted pendulum model is widely used to generate walking motions for robots [1, 18, 29–32]. Chun-Guang Li [33] used the 3D linear inverted pendulum model based on natural ZMP trajectory in the single-leg support phase and the two-dimensional linear inverted pendulum model in the double-leg support phase to obtain the center of mass trajectory. After initial manual parameter tuning, a good gait was obtained.

3.1.2 Cart-Table Model

The cart-table model was used by Snafii et al. [34] to propose an implementation of a ZMP-based NAO humanoid robot omnidirectional walking engine. The cart-table model was used to model human-like walking skills of the robot, and a new analytical method based on Fourier ZMP approximation was proposed to generate the COM reference trajectory from a predetermined ZMP trajectory. A new time segmentation method was proposed to parameterize different double support cycles, and an active balance method was used to keep the robot upright when subjected to environmental disturbances. The cart-table model makes assumptions and simplifications, such as assuming that all weight is concentrated on the cart and that the supporting legs have no weight. Although this assumption does not hold in reality, modern robots typically have heavy backpacks containing components and batteries, so the weight of the legs relative to the upper body is small and consistent with the assumptions of the cart-table model. Fig. 8(a) shows the cart-table model of the NAO robot.

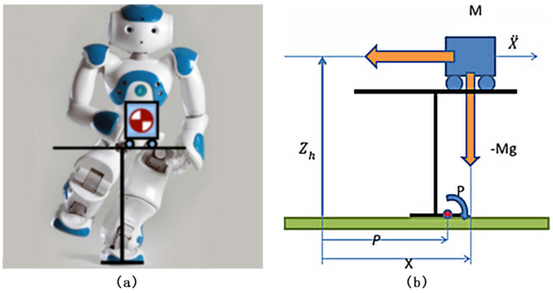


Fig. 8 (a) The cart-table model used on an NAO robot[34] (b) Schematic view of a cart-table model[34]

By using two sets of cart-table models, the three-dimensional motion of the robot can be simulated, with one set for front-back motion and the other set for coronal plane motion. Fig. 8(b) shows a diagram of the cart-table model.

The position of the center of mass M, denoted as x. Z_h is defined in the coordinate system O. The gravity g and acceleration of the cart produce a moment T_p around the pressure center P_x. Equation (13) calculates the moment T_p around the point P.

$$T_p = Mg(x - P_x) - M\ddot{x}Z_h \tag{13}$$

When the robot is in dynamic balance, ZMP and CoP are the same [35]. Therefore, the moment around CoP is 0, i.e. $T_p = 0$. By setting the left side of equation (13) to 0, equation (14) can be used to calculate the position of ZMP. Using the same assumptions and reasoning, the cart-table model can also be used in the coronal plane (y-direction), and the position of the COM in the coronal plane can be calculated using equation (14).

$$P_x = x - \frac{Z_h}{g}\ddot{x} \quad P_y = y - \frac{Z_h}{g}\ddot{y} \tag{14}$$

To apply the cart-table model to the walk generation problem of humanoid robots, the first step is to plan and define the position of the feet during walking, then derive the ZMP trajectory based on the ZMP position and the constraints of the supporting polygon. Next, use equation (14) to calculate the position of the COM. Finally, based on the planned position of the feet and the computed COM position, use inverse kinematics to obtain the angle trajectory of each joint.

3.1.3 Three-Center-of-Mass Model

Sato et al. [36] proposed a three-center-of-mass model that simplifies humanoid robots as a trunk center of mass, a support leg center of mass, and a swing leg center of mass, under the condition of constant body height. They generated walking skills for robots in the RoboCup3D simulation environment. In the single-leg support phase, given the ZMP and swing leg trajectory, the body trajectory of the humanoid robot can be solved using the ZMP equation to plan the walking motion.

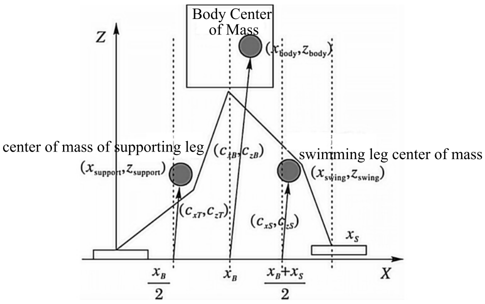


Fig. 9 Three COG model of single support phase in sagittal plane[37]

In the three-center-of-mass model, under the condition of constant body height, the equations for the ZMP trajectory, body trajectory, and swing leg trajectory are shown in equation (15):

$$
\begin{aligned}
p_x(t) =& \frac{m_B + m_L}{M} x_B(t) - \frac{E_z}{g} \ddot{x}_B + E_x + \\
& \frac{m_L}{2M} (x_S(t) - \frac{c_{zS}}{g} \ddot{x}_S(t)) \\
p_y(t) =& \frac{m_B + m_L}{M} y_B(t) - \frac{E_z}{g} \ddot{y}_B + E_y + \\
& \frac{m_L}{2M} (y_S(t) - \frac{c_{zS}}{g} \ddot{y}_S(t))
\end{aligned}
\tag{15}
$$

$$
\begin{aligned}
E_x &= \frac{m_B c_{xB} + m_L c_{xT} + m_L c_{xS}}{M} \\
E_y &= \frac{m_B c_{yB} + m_L c_{yT} + m_L c_{yS}}{M} \\
E_z &= \frac{m_B c_{zB} + m_L c_{zT}/2 + m_L c_{zS}/2}{M} \\
M &= m_B + 2m_L
\end{aligned}
\tag{16}
$$

$(x_B(t), y_B(t))$ is the geometric center trajectory of the robot body, $(x_S(t), y_S(t))$ is the heel trajectory of the robot swing leg, $(p_x(t), p_y(t))$ is the ZMP trajectory of the robot, (c_{xB}, c_{yB}) is the offset of the trunk center of mass relative to the geometric center of the trunk, (c_{xT}, c_{yT}) is the offset of the support leg center of mass relative to the geometric center of the support leg, (c_{xS}, c_{yS}) is the offset of the swing leg center of mass relative to the geometric center of the swing leg, c_{zB}, c_{zT}, c_{zS} are the heights of the trunk center of mass, support leg center of mass, and swing leg center of mass respectively, m_B, m_L are the masses of the trunk and a single leg respectively. After planning the ZMP trajectory and swing leg trajectory based on equation (15), the robot body trajectory can be obtained by solving the differential equations directly.

Li et al. [37] applied the method for generating walking skills using the three-center-of-mass model to generate kicking skills. They used the three-center-of-mass model to obtain the ZMP trajectory considering the swing leg and trunk, and then used a cubic Bezier curve to plan the swing leg and trunk trajectory. In ZMP trajectory planning, they considered three types of trajectories: fixed points, straight lines, and cubic Bezier curves. They demonstrated through experiments that using a cubic Bezier curve to plan the ZMP trajectory could result in the maximum trunk velocity and the best kicking effect. Finally, they combined the three-dimensional linear inverted pendulum model with the two-legged support phase to calculate the center of mass trajectory and adjust the posture.

3.1.4 Truncated Fourier Series

In the design of walking skills for humanoid robots, truncated fourier series can be used to synthesize trajectories of the hip and knee joints [38]. This is because the natural gait of humans is periodic, and the typical trajectory shapes of the hip and knee joints within one cycle of skill generated based on the model are shown in Fig. 10.

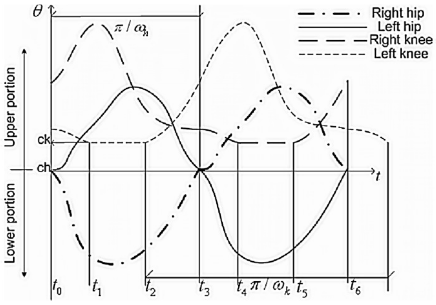


Fig. 10 Typical shapes of hip and knee trajectories[38]

Just like human gait, the trajectories of the two legs are similar in shape, but offset from each other in time by half a walking cycle. From the perspective of joint angles, the upper and lower parts of the trajectory are both similar to a part of a sine curve. Therefore, the fourier series representation of these curves can be simplified by choosing not to involve too many high-order terms.

The general fourier series can be represented as:

$$f(t) = \frac{1}{2}a_0 + \sum_{i=1}^{\infty} a_i \sin(\frac{2\pi i}{T}t) + \sum_{i=1}^{\infty} b_i \cos(\frac{2\pi i}{T}t) \tag{17}$$

All joint trajectories in one gait cycle can be divided into two parts, each of which can be seen as an odd function output based on the intersection with the angle axis. Therefore, the sine series in the fourier series function is simplified, and a truncated Fourier series (TFS) is used to simulate each part, as shown in equation (18):

$$f(t) = \sum_{i=1}^{n} a_i \sin(i\omega t) + c_f \qquad \omega = \frac{\pi}{T_s} \tag{18}$$

a_i,n,cf are variables that need to be determined. ω represents the step frequency, which is defined as half T_s the period T. The joint angle trajectory can be defined through the truncated fourier series, and different gaits can be generated by changing the variables [11–14].

3.1.5 Interpolation method

Interpolation refers to the method of constructing a function passing through a specified series of points. It can be proved that there is one and only one function passing through N different points no higher than $N-1$ times. In order to obtain a good approximation effect, the distance between interpolation nodes should be small, so there are generally many nodes. If a single interpolation polynomial is used for the entire interpolation interval, the degree of the interpolation polynomial is generally high, called high-degree polynomial interpolation. However, high-degree polynomial interpolation is prone to produce oscillation phenomenon, so a cubic spline interpolation function with first and second derivative continuity is used to fit the data.

Generally, interpolation methods have consistent convergence, but they only guarantee the overall continuity of the interpolation function. Although the left and right derivatives exist at the connection points of each small section, they are not necessarily equal, which means they are not smooth at the connection points. Early engineers used a flexible slender wooden strip (or metal strip), called a spline, to draw a curve passing through a given point. They forced it to bend through the known points. Elastic mechanics shows that the deflection curve of the spline has a second-order continuous derivative function, and is a cubic polynomial between adjacent given points, which is the mathematical cubic spline interpolation curve.

To make the planned trajectory smooth, while maintaining the continuity of displacement and velocity, it is also necessary to ensure the continuity of acceleration.

Huang [39] used the cubic spline interpolation method for walking motion trajectory planning. First, a multi-link model is established, and then the trajectory planning of the foot inclination angle, ankle joint, hip joint, and knee joint is carried out in the radial and lateral directions of walking using the cubic spline interpolation method. By selecting key points in the walking process of the humanoid robot, as shown in Fig. 11, and determining their displacement, velocity, acceleration, and other state parameters at these key points, the entire time period of the robot is planned using the cubic spline interpolation method to obtain the planned trajectory. The ZMP trajectory can be calculated based on the established multi-link model and joint trajectories, and finally, the ZMP trajectory is ensured to always be within the supporting polygon.

Using cubic spline interpolation to generate motion trajectories, and considering stability based on ZMP, it is possible to simplify the model of an inverted pendulum, and generate walking using interpolation between interpolated trajectories or keyframes [9, 10]. By designing the endpoint of the trajectory in advance [40–42], cubic spline interpolation can also be used to plan kicking action trajectories. Other interpolation methods such as Hermite interpolation have also been used in action development in RoboCup3D [43, 44].

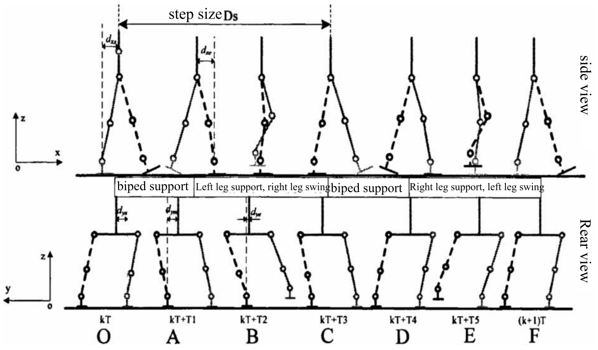


Fig. 11 The diagram of walking robot's process[39]

3.1.6 Optimization methods for generating skills based on models

Model-based methods consider robot dynamics stability, so the generated skills are stable and executable. However, initial implementations of these skills are often mediocre, such as walking skills being stable but slow, or kicking skills not resulting in sufficient distance. From the perspective of competition goals and research, these initial skills need to be optimized to improve the performance of robot skills.

Model-based skill generation involves some model-related parameters, such as the parameters directly affecting the stability and speed of walking generated based on a three-dimensional linear inverted pendulum model, as shown in Table 1.

The process of optimization can be equivalent to finding the optimal parameters , x represent the parameters that need to be optimized, and the corresponding skill performance value is y. In the case of walking, the parameter can be set as the distance

Table 1 Gait paremeter[1]

symbol	Parameter Description	Parameter Classification
$maxStep_i$	maximum step size of x, y, θ	speed
COM_{shift}	offset of the center of mass	walking dimension
H_{torso}	torso height	
H_{step}	the maximum height of the swing foot from the ground	
f_g	ground hold phase before lifting the foot	pre-swing phase
f_a	swinging foot in swing phase	
f_s	the resting phase before the swing foot moves in the x-y plane	The time of the movement of the swinging foot
f_m	the stage where the swing foot moves in the x-y plane	
X_{offset}	fixed offset between torso and feet	center of gravity control
X_{factor}	the step factor is applied to the forward position of the torso	
err_{norm}	maximum error COM before gait slows down	Center of mass control
err_{max}	maximum error COM before speed drops to 0	
D_{pos}	expected differences between theoretical COM and perceived COM	
φ_{pid}	proportional controller value for torso angle measured by IMU	proportional controller
COM_{pid}	proportional controller value that controls COM	
Arm_{pid}	move the arm to control the proportional controller value of COM	
δ	how fast the unit cycle step changes (proportional controller value)	
S_{length}	single step duration	other
$Swing_{ankle}$	angle of foot landing	

walked within a specified time. Therefore, the problem of model-based skill optimization is a black-box optimization problem. We do not need to consider how it works, we just need to find the parameter that leads to the global minimum or maximum.

$$y = f(x) \tag{19}$$

For skills such as kicking and getting up, Depinet et al. [45] extract the entire skill through keyframe sampling, where each keyframe represents the joint angles of the robot for the current cycle. Multiple joint angles directly affecting the action are selected from multiple keyframes, and these angles are taken as the parameters to be optimized. In short, the problem of model-based skill optimization can be transformed into selecting some important parameters that determine the skill effect and finding the optimal solution for these parameters.

Kasaei et al. [46] generated initial walking skills based on a two-mass point model and improved the LIPM model by considering the upper body mass, by adding trunk movements and changing the height of the center of mass to improve walking performance. The LQG controller is used to fit the trajectory during the control phase, and genetic algorithms are used to optimize the parameters involved in the model and controller. The final walking speed reaches 0.805m/s. Nezami et al. [11] used a learning automata to improve the genetic algorithm and optimize the walking skills by optimizing the parameters in the truncated fourier series.

In addition to genetic algorithms, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm is currently widely used to optimize skills in the RoboCup3D simulation environment, including walking skills [18, 22, 25, 26, 29, 32, 47–50], kicking skills [22, 40, 41, 45, 51, 52], and getting up actions after falling [53]. Based on most reports and actual competition results, CMA-ES currently achieves the best optimization results. CMA-ES [54–56] is an improved evolution strategy algorithm that has been proven to perform better than other evolution strategy algorithms in many works [57, 58] on numerical optimization.

Apart from the standard CMA-ES algorithm, several research studies have improved the CMA-ES algorithm for optimizing goals in the RoboCup3D simulation environment, achieving better results in terms of optimization effectiveness and speed. Abdolmaleki et al. [59, 60] proposed a Contextual Relative Entropy Policy Search with Covariance Matrix Adaptation (CREPS-CMA) algorithm for learning distance-controllable kicking skills, which outputs parameters selected through keyframe selection based on the expected kicking distance. Lu et al. [61] used PCA to reduce the number of parameters before applying the CMA-ES algorithm to optimize the reduced parameters, while Jouandeau et al. [62] used the CLOP algorithm to improve the training speed of the CMA-ES optimization. Uchitane et al. [63, 64] proposed a mask-CMA-ES algorithm that ignores unimportant parameters to speed up the optimization process and ensure optimization effectiveness.

Patrick et al. [65] designed a random obstacle walking trajectory to re-optimize the walking skills that had already been optimized, as the actual performance in real situations may differ from the predefined walking trajectory in the optimization process. Muniz et al. [10] also decomposed existing model-based walking skills into keyframes, extracted parameters, and optimized them using the CMA-ES algorithm. Urieli et al. [66] used a distributed computing cluster to automatically evaluate skill data and compared the effectiveness of hill climbing, cross-entropy, genetic algorithms, and CMA-ES algorithms, reporting that the CMA-ES algorithm achieved the best optimization results for both walking and kicking skills. Melo et al. [67] used the CMA-ES algorithm to develop kicking skills of different distances, selecting the midpoint of the parameters of kicking skills with a lower distance and a higher distance as the initial seed for optimization, and then optimizing the kicking skill parameters for a middle distance.

Optimizing individual skills alone may not be beneficial for the execution of continuous skills [66], as multiple skills need to be executed continuously when robots execute strategies in RoboCup3D, such as transitioning from walking left to walking straight, walking straight to stopping, walking straight to stopping and then walking straight again. This means that individual skills cannot be optimized alone and different skills' transitions need to be considered. Hierarchical optimization methods [1, 40, 68–70, 80] have been used to optimize multiple skills, fixing lower-level parameters to ensure continuous optimization between different skills and maintaining the effectiveness of individual skills during continuous execution.

Table 2 Comparison of skill optimization methods for humanoid robots in RoboCup3D simulation environment

skill	author	generation and optimization	uniqueness
W	Shafii[26](2015)	LIPM+CMA-ES	Consider hip height
	Seekircher[25](2016)	LIPM + CMA-ES	reduce error close-loop control with real robots
	Xu[32](2014)	3D LIPM + CMA-ES	Hierarchical Optimization
	Shen[29](2015)	3D LIPM + CMA-ES	Based on ZMP
	Seekircher[47](2015)	3D LIPM + CMA-ES	\
	Patrick[65](2016)	3D LIPM + CMA-ES	Design walking trajectories with random obstacles
	Zixuan[68](2016)	3D LIPM + CMA-ES	Hierarchical Optimization
	Li[1](2019)	3D LIPM + CMA-ES	Hierarchical Optimization
	Lu[61](2019)	3D LIPM + CMA-ES	PCA algorithm
	Jouandeau[62](2013)	3D LIPM + CLOP	Train Faster
	Cai[75](2013)	3D LIPM + DE	Compared Multiple Methods
	Tao[72](2021)	3D LIPM + PSO	Parallel PSO Algorithm
	Picado[77](2009)	PFS + GA	\
	Shafii[13](2009)	TFS + GA	\
	Shafii[14](2009)	TFS + PSO	\
	Nezami[11](2012)	TFS + GA	Combining learning automata to improve GA
	Haider[12](2012)	TFS + ES	\
	Huang[39](2011)	CSI + PSO	Design Stability Margin Evaluation Function
	Uchitane[63, 64](2011,2010)	CPGs + mask-ES	Train Faster
	Lattarulo[79](2011)	CPGs + AA	\
	Halataei[73](2015)	ZTP + ABC	\
	Muniz[10](2016)	Keyframe + CMA-ES	Walking skill is also disassembled for keyframes
	Simoes[48](2017)	CTM + CMA-ES	\
	Kasaei[46](2019)	TMM + GA	Add the LQG controller and optimize its parameters
K	Depinet[45](2014)	Keyframe + CMA-ES	First proposed the key frame method
	MacAlpine[69](2014)	Keyframe + CMA-ES	Hierarchical Optimization
	Abdolmaleki[59, 60] (2016,2019)	Keyframe + CREPS-CMA	Kicking distance is controllable
	Melo[67](2019)	Keyframe + CMA-ES	Kicking skills at different distances
	He[80](2019)	Keyframe + CMA-ES	Hierarchical Optimization
	Liu[18](2015)	FTP + CMA-ES	Three-tier incremental accumulation optimization
	Jouandeau[74](2014)	CBC + CLOP	Train Faster
	Li[51](2015)	CBC + CMA-ES	shorter execution time
	Hecheng[41](2015)	FTI + CMA-ES	control trunk to maintain balance
	Dorer[52](2017)	IM + CMA-ES	Mainly optimize TYPE4 robot
	Baur[71](2018)	RS + CMA-ES	Without keyframe method
WK	Urieli[66](2010)	3D LIPM,Keyframe + CMA-ES	Compare effects of multiple algorithms and prove that CMA-ES works best
	MacAlpine[22](2011)	3D LIPM + CMA-ES	\
	Urieli[70](2011)	3D LIPM + CMA-ES	Hierarchical Optimization
	Li[33](2015)	3D LIPM, TCM + PSO	Hierarchical Optimization
	Feng[40](2017)	IM + CMA-ES	Hierarchical Optimization
SWK	Rei[76](2010)	TFS + PSO	Compare Multiple Methods
SK	MacAlpine[53](2012)	Keyframe + CMA-ES	\
SIK	Cruz[78](2012)	SI + GSSOA	\
KW	MacAlpine[49](2017)	Keyframe + CMA-ES	Model-Based Optimization

Table 2 in this paper summarizes the comparison of model-based robot skill generation and optimization methods validated in the RoboCup3D simulation environment.

To adapt to the width of the article page, we have defined some abbreviations to shorten the length of each line. It should be noted that some of these abbreviations are not universal. In the column of skills, W represents walking skills; K represents kicking skills; WK represents walking and kicking skills; SWK represents standing up, walking, and kicking skills; SK represents standing up and kicking skills; SIK represents side kicking skills; KW represents kicking while walking skills. In the column of generation and optimization, each row represents the method for generating and optimizing skills in the study, and the methods for generation and optimization are separated by a plus sign. In this column, LIPM represents two-dimensional linear inverted pendulum; 3D LIPM represents three-dimensional linear inverted pendulum; TFS represents Truncated Fourier series; GSSOA represents global step-by-step optimization algorithm;TCM represents Three-Center-of-Mass Model; PFS represents Partial Fourier series; AA represents Alliance Algorithm; CSI represents Cubic spline interpolation; ZTP represents ZMP trajectory planning; CTM represents Cart-Table Model; TMM represents Two-Mass Model; FTP represents Foot Trajectory Planning; CBC represents Cubic Bezier curve; FTI represents Five point trajectory interpolation; IM represents Interpolation method; SI represents Sine interpolation; RS represents Random Setting. Although many studies used similar optimization methods, they differ in the initial skill generation methods and possess unique optimization and generation techniques.

3.2 Model-Free Skill Generation and Optimization in RoboCup3D Simulation Environment

This section provides an overview of the model-free skill generation and optimization methods in the RoboCup3D simulation environment, which completely disregard or mainly ignore the kinematic analysis of humanoid robots. In contrast to the model-based methods summarized in Section 3.1, model-free methods skip the complex kinematic analysis and can generate skills from scratch without any prior knowledge, continually optimizing them during the generation process. The model-free skill generation and optimization methods validated in the RoboCup3D simulation environment mainly include reinforcement learning methods, cerebellar model neural network methods, and central pattern generator methods, which transform the robot skill generation and optimization process into robot learning processes. Not only do these methods avoid the complex kinematic analysis process, but they can also generate some skills that cannot be obtained through model design (such as running, walking, and kicking), thus becoming the main research direction in humanoid robot skill generation and optimization.

3.2.1 Reinforcement Learning Methods

Reinforcement learning (RL) algorithms belong to a subfield of machine learning that determines how to change the environment by executing certain actions under given environmental states to maximize the rewards [81]. In the RL process, one or more intelligent agents (learners and decision-makers) interact with the environment continuously, obtain environmental information outside themselves, adopt learning algorithms, and respond appropriately to environmental changes to maximize cumulative rewards [82].

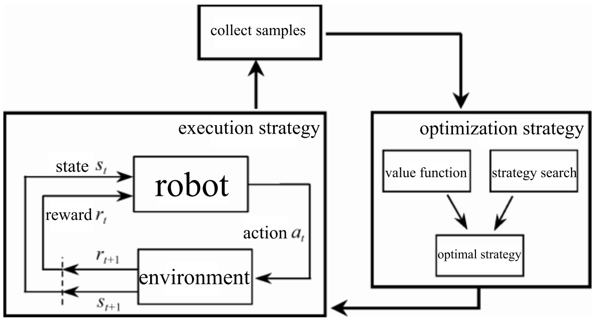


Fig. 12 Illustration of manipulation skills learning method based on reinforcement learning[83]

As shown in Fig. 12, during the policy execution stage, the intelligent agent executes an action a_t according to the current policy π based on the state s_t, obtains a reward value r_{t+1}, and reaches a new state s_{t+1} according to the state transition probability $p(s_{t+1}/s_t, a_t)$. This process is repeated until the intelligent agent reaches the termination state. During the sample collection stage, a trajectory sequence $\tau : s_0, a_0, s_1, a_1, \cdots, s_H$ is obtained, where H is the length of the trajectory sequence. The cumulative reward $R(\tau)$ obtained by the intelligent agent executing the policy π in the environment is given by:

$$R(\tau) = \sum_{t=0}^{H} \gamma^t r_t, \qquad 0 < \gamma \leq 1 \tag{20}$$

where γ is the discount factor. The value function $V^\pi(s)$ corresponding to the state s represents the cumulative reward value obtained by the intelligent agent executing the policy π in the state s.

$$V^\pi(s) = \mathbf{E}[\sum_{k=0}^{H-t} \gamma^k r_{t+k} | s_t = s; \pi] \tag{21}$$

The state-action value function $Q_\pi(s, a)$ obtained by executing the action a in the state s is defined as:

$$Q^\pi(s, a) = \mathbf{E}[\sum_{k=0}^{H-t} \gamma^k r_{t+k} | s_t = s, a_t = a; \pi] \tag{22}$$

The iterative relationship for the action-state value function can be obtained from the Bellman equation [84]:

$$Q^{\pi}(s_t, a_t) = \mathbf{E}_{s+1}[r_{t+1} + \gamma Q_{\pi}(s_{t+1}, \pi(s_{t+1}))] \tag{23}$$

The optimal action a_t^* that the intelligent agent in the state s should execute is:

$$a_t^* = \arg\max_{a_t} Q^{\pi}(s_t, a_t) \tag{24}$$

In the policy optimization stage, depending on whether the value function or the state-action value function is required to obtain the optimal action, the reinforcement learning method can be divided into value function reinforcement learning and policy search reinforcement learning. With the development of deep learning, various deep reinforcement learning methods have been formed by combining deep learning and reinforcement learning, achieving remarkable results in games, robots, natural language processing, etc. [85–92]. Especially in 2016, AlphaGo [93], which was trained by deep reinforcement learning algorithm, defeated the world champion in the game of Go and proved that even without prior human knowledge, a deep reinforcement learning algorithm can train a Go agent beyond human capabilities from scratch [94].

The execution of robot skills is essentially the change of joint angles. We can consider robot skills as a continuous sequence of multiple frames. Assuming the robot has n joints, a certain skill can be decomposed into m frames, and each frame is a combination of n joint angles $K : k_1, k_2, \cdots, k_n$, where k_n represents the angle of the nth joint. Thus, the skill can be represented as a sequence of frames $K_1, K_2, \cdots, K_m$. In order to achieve better performance, the robot needs to execute the optimal combination of frames $K_1, K_2, \cdots, K_m$ for the skill, just like the interaction between an intelligent agent and the environment in reinforcement learning. The agent needs to learn a complete strategy to maximize the reward value, and the complete strategy is the combination of the agent's actions during the interaction process. Analogously, in the generation and optimization of robot skills, the strategy that the robot needs to learn is the combination of frames $K_1, K_2, \cdots, K_m$ at each moment of the skill, and each frame K_m is the action a that the robot interacts with the environment.

In model-based methods, it is generally necessary to first build a model for the humanoid robot, and then generate various skills through dynamic analysis, and extract parameters for optimization. Even if the humanoid robot is simplified to a model, complex dynamic analysis for planning of robot joint trajectories and ensuring stability are still required. However, deep reinforcement learning methods can enable humanoid robots to learn walking skills [15, 16] and kicking skills [17] from scratch. Depending on the different skills to be generated, different state spaces, action spaces, and reward functions are set to achieve learning of different skills. In this process, dynamic analysis is not required. The robot interacts with the environment, and continuously learns during the process of generating skills, thus eliminating the optimization process. Spitznagel et al. [95] used the PPO algorithm to enable the type 4 robot with toes to learn kicking skills at different directions and distances. Since the state space in the algorithm includes information such as the robot's gyroscope, the kicking skill can be adjusted based on the current state when the robot executes the skill, thus achieving closed-loop control. Different from the fixed kicking skills obtained by the CMA-ES method optimization at several directions and distances, this method can take direction and distance as input, which can achieve more effective kicking skills. In addition to basic walking and kicking skills, Abreu et al. [96] used the PPO algorithm to learn running and dribbling skills from scratch to achieve faster movement. Although the learned skills are different from the human running posture, the speed exceeds the known walking skills at that time. Abreu et al. [97] also used the PPO algorithm to learn running skills, and improved the parameters of the state space by using the Akaike information criterion in the selection of the state space, resulting in running posture that is more symmetric than that learned by Abreu et al. [96]. Melo et al. [98, 99] made improvements based on Abreu et al.'s [97] research, not only reducing the training time, but also achieving faster robot speed in the end, and without changing the framework of the underlying server. By adding mirrored data during the data collection stage in the algorithm, the robot's final learned running skill is more similar to human skill. The highest reported speed in this study was 3.5 m/s, which is currently the fastest known robot running speed.

In most research and competition teams of RoboCup3D, walking and kicking skills are separated. Robots first walk to the kicking point and then adjust the kicking point with small steps based on the position of the ball, in order to smoothly execute the pre-designed kicking skill. This process takes a lot of time, especially when adjusting the kicking point with small steps, collisions and interceptions of the ball may occur, resulting in the failure of the kicking skill. In contrast, human kicking skills are performed continuously while running. Abreu et al. [100, 101] used the PPO algorithm to enable robots to learn the kicking skill while moving from scratch, just like humans, without the need to adjust the kicking point and waste time. In this study, the robot first uses its existing walking skills to walk to a kicking point close to the ball. This kicking point does not require an accurate coordinate within the priority range. Then, the situation is divided into six categories based on the relative position of the robot's foot and the ball(Fig. 13), and the kicking skill is learned for each category. In addition, the state space includes not only the global coordinates of the ball but also the relative position of the ball to the robot's foot. Therefore, this method requires higher accuracy in self-positioning and ball positioning. The study reported that in all six categories, kicking skills during walking can be achieved within an average distance of 6m in 0.33s, although the kicking distance is not very far, it greatly increases the attacking ability of the robot. The study also open-sourced the reinforcement learning tool FCP Gym based on the Gym development of RoboCup3D simulation environment, providing a convenient reinforcement learning interface for other researchers.

In addition to learning skills from scratch, reinforcement learning algorithms are also used to optimize existing skills. Kasaei et al. [102] used a combination of reinforcement learning and robot dynamics models to generate and optimize the omnidirectional walking skill of robots. This study used the ZMP as the main criterion for robot stability and designed a linear inverted pendulum model considering the motion of the center of mass. The robot's motion trajectory was planned through a dynamic planner, and the LQG controller was used to control the robot's joint to track the

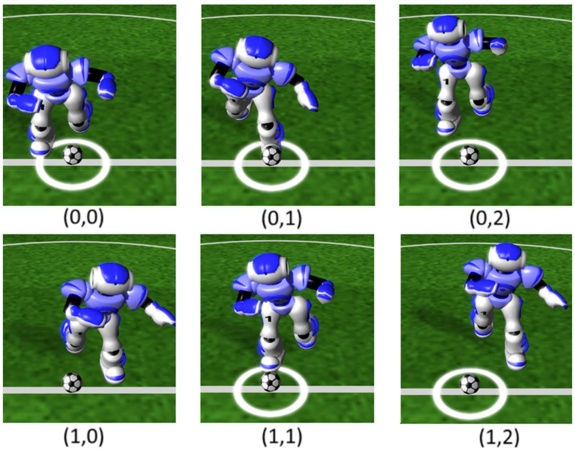


Fig. 13 Initial condition cases fora kick in motion[101]

trajectory, thereby achieving the initial walking skill of the robot. In the optimization phase, genetic algorithms were first used to optimize the parameters of the planner to achieve stable straight-line walking skills. However, considering that the dynamics model did not consider the state of the robot's upper body, and the researchers believed that arm movements are also important for turning during walking, PPO algorithm was used to learn the arm movements of the robot during walking, especially during turning. By using the gyroscope data, robot acceleration, joint angle positions and velocities, and LQG controller adjustment data as the state space, the angle variables of the robot's arm joints and the adjustment values of the center of mass were obtained, which were combined with the results of the LQG controller to control the robot's walking, thus forming closed-loop control. The study results showed that the combination of the dynamics model and the PPO algorithm can enable robots to achieve faster walking speed and better stability. However, the study proposed that because the dynamics model ignored the weight of the swing leg, the controller still has a large error, which is a consistent problem with the dynamics model.

Muzio et al. [103, 104] used deep reinforcement learning algorithms to optimize dribbling skill while walking. Firstly, they implemented a walking model based on a model-based method. The model's input is the walking direction and rotational speed, and then the deep reinforcement learning algorithm learns what the input data is like when dribbling, indirectly optimizing the dribbling walking skill. The study compared three deep reinforcement learning algorithms: PPO, DDPG, and TRPO, and proved that the PPO algorithm optimized the longest duration of dribbling skills, i.e., the robot's dribbling ability was the strongest.

Rezaeipanah et al. [105] optimized kicking skill during walking using reinforcement learning algorithms. Firstly, they designed a kicking model based on inverse dynamics, which requires the robot to walk to a fixed kicking point. The more accurately the kicking point is reached, the better. To kick the ball while walking, the robot needs to be controlled to walk to this point. The study believes that the posture in the second time period of the entire walking skill (alternating between the supporting and swinging legs) is the most suitable for kicking. The ultimate goal is to control the robot to accurately walk to this point while being in the second state of the walking skill. This requires precise control of the robot's walking speed and angle, such as slowing down the speed as it approaches the kicking point. Then, to kick the ball better, a curved walking route was calculated for the robot, which allows the robot to face the direction of the ball as much as possible when reaching the kicking point. Finally, Q-Learning algorithm was used to learn how to control the speed and angle of walking. Here, the speed is replaced by step length, because Q-Learning deals with discrete action state spaces. The study also designed discrete state and action spaces based on the distance to the ball. The final report showed an average distance of about 6 meters for kicking during walking, but the kicking time was very short. Compared with the methods in references [100, 101], this method did not optimize the kicking skill, i.e., the original kicking skill was not the best. If the kicking skill is optimized again, the kicking effect proposed by this study is believed to be better.

Wang et al. [107] combined reinforcement learning with a dynamic model to optimize multi-directional stationary kicking skills. Firstly, they designed a cubic spline interpolation kicking model to obtain the kicking trajectory, thereby realizing basic kicking. The highest position of the swinging leg and the x-directional offset of the kicking point were used as the action space for reinforcement learning. Then, Q-Learning algorithm was used, where the action space was the coordinates of the highest point of the swinging leg and the x-offset during kicking, and the state space was the coordinates of the ball landing point. These continuous variables were discretized to meet the requirements of Q-Learning algorithm. Finally, they learned how to perform stationary kicking within a range of 5-12 meters and at an angle between -45 degrees to 45 degrees, and the actual kicking point was relatively close to the target point.

Melo et al. [108, 109] used the PPO algorithm to optimize existing walking skill. By adding the output of reinforcement learning to the joint angle control values of the gait model, the robot learns how to maintain balance when subjected to external disturbances during walking. The state space includes various dynamic physical quantities and joint angles of the robot, and the action space is the hip and ankle joint angles, with rewards defined as the time spent maintaining balance.

The literature [110–112] also uses the Q-learning algorithm to optimize walking skill, but it does not implement closed-loop control, only optimizing the four important parameters involved in the double inverted pendulum. When used in practice, the optimized parameters are directly used.

Pavse et al. [113] combined imitation learning with reinforcement learning. Imitation learning only considers whether it matches the trajectory to be imitated, but if the expert's trajectory is locally optimal, the imitated trajectory will not be the best. Reinforcement learning can be used to consider global rewards during imitation, i.e., to increase the reward rather than blindly sticking to the expert trajectory. Based on this method, expert trajectories can be obtained through a single demonstration example, and the skill can be optimized using reinforcement learning methods, which is higher than the demonstration examples and the baseline.

3.2.2 Cerebellar Model Articulation Controller

The CMAC (Cerebellar Model Articulation Controller) neural network model is a neural network model based on the principles of the cerebellum controlling limb movements. It is a locally approximated, simple and fast neural network that can learn any number of nonlinear fits. The most important feature of CMAC is that it provides an understanding of the human brain's computational processes, which leads to a unique insight into system integration [114]. The CMAC neural network model imitates the unique function of the cerebellum and can be widely used in the field of robot motion control. As shown in Fig. 14, the neural network mainly consists of input (decoding), information storage, and output.

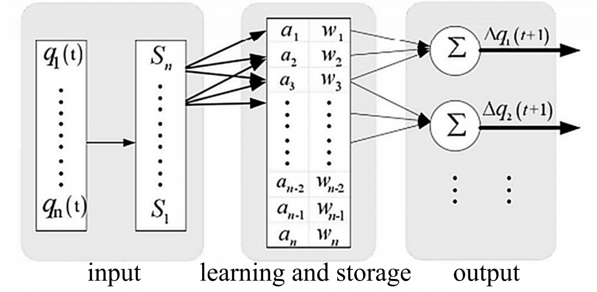


Fig. 14 Cerebellar Model Articulation Controller[18]

The general learning process based on CMAC requires few unresolved objects in the system. In the process of interacting with the system environment, different degrees of reward and punishment are obtained to further understand the unresolved objects and turn them into a known part of the system. The main control process is divided into the following three steps:

Step 1: Input Parameter Quantization Process:

$$q_i = \frac{s_{q_i} - s_{q_{min}}}{s_{q_{max}} - s_{q_{min}}} \times Q_i \tag{25}$$

Where Q_i is the number of controller quantization levels, where $i = 1, 2, \cdots n$.

Step 2: Concept and Physical Mapping Process:

1) Obtain X logical addresses for each activation intensity from the input section, and take the remainder after mod X. If there is no remainder after mod, then let the address remainder be X.

2) Arrange the calculated results in order from small to large according to the remainder.

3) Arrange each of the X logical addresses that have been sorted vertically, and the sorting criteria are also based on the above requirements. This creates a coding table that contains X virtual addresses.

4) The rolling combination forms a virtual storage address, and through hash mapping, the hashed address code is stored, thereby achieving the mapping of physical storage address space.

Step 3: Calculate the sum of X values stored in the actual physical space and obtain the final output result.

Liu [18] introduced CMAC neural network learning control into the inverse kinematics model of humanoid robots, thus achieving the mapping from the robot joint position space to the joint angle space. At the current time, the joint angle values and the predetermined joint positions for the next moment are used as the inputs of the controller, and the output value is the joint angle increment for the next moment. CMAC is used as the optimization process of inverse kinematics. Since there may be non-unique solutions in the inverse kinematics solution process of robot legs, it is necessary to consider using certain joint angle optimization algorithms to achieve the optimal selection of multiple output angle values. By introducing the CMAC inverse kinematics control method into the basic inverse kinematics calculation process, the accuracy of the robot trunk position and joint angle values is continuously corrected and optimized to form closed-loop control. The closed-loop control process based on CMAC is shown in Fig. 15.

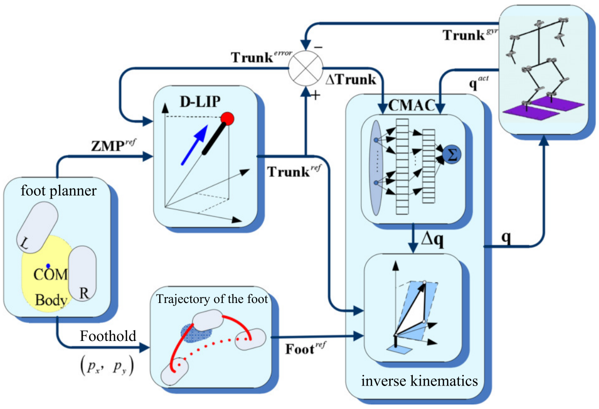


Fig. 15 Omnidirectional walking based on CMAC closed-loop control[18]

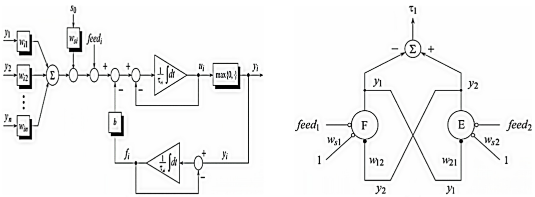
3.2.3 Central Pattern Generators

The Central Pattern Generators (CPGs) is a type of neural circuit that exists in both invertebrates and vertebrates. It can generate rhythmic patterns of neural activity without receiving rhythmic inputs. The CPG model is used to control the movement of robots, and it is a new control method that can replace methods based on finite-state machines, sine generators, and preset reference trajectories. When robots perform skills such as walking, the joints change periodically and rhythmically. Therefore, the CPG model can be used as a reference to set the kinematic model of the robot's skill. The gait generation method based on CPG does not consider ZMP, and because the CPG model carries multiple parameters, it is generally optimized using genetic algorithms after designing the CPG model to make the rhythmic signal generated by the CPG model result in stable walking for the robot.

Bavani et al. [19] used the Matsuoka oscillators to form the CPG model, which is based on the mutual inhibition of two artificial neurons to produce a periodic signal as output. The model of each neuron is represented by two equations of two state variables, as shown in Fig. 16(a):

$$\begin{aligned}
\tau_{ri}\frac{du_i}{dt} &= -u_i + \sum_{j=1}^{n} w_{ij}y_j + w_{s_0}s_0 - bf_i + feed_i \\
\tau_{ai}\frac{df_i}{dt} &= -f_i + y_i \\
y_i(u_i) &= max\{0, u_i\}
\end{aligned} \tag{26}$$

The first state variable u_i describes the membrane potential of the neuron, and the second state variable f_i represents the degree of adaptation or self-inhibition of the neuron, which y_i is the output of the neuron. The output frequency is roughly determined by $1/\tau_{ri}$, and the specified rise time τ_{ri} when the step time is given. In addition, τ_{ai} is the time constant that specifies the delay of the adaptation effect. w_{ij} describes the inhibitory synaptic connection weight from the j-th neuron to the i-th neuron. $\sum_{j=1}^{n} w_{ij}y_j$ represents the total input from the internal neurons of the neural network, s_0 represents the driving input, and w_{s_0} represents the connection weight of the driving input. Self-inhibition is achieved through bf_i, and mutual inhibition is achieved through $\sum_{j=1}^{n} w_{ij}y_j$. In the closed-loop CPG model, $feed_i$ is used to represent the sensor signal that is fed back to the neuron as input, and represents the interaction between the robot and the environment. Each Matsuoka oscillator contains two neurons, the flexor neuron and the extensor neuron, which are interconnected, and also inhibit and excite each other to generate oscillatory outputs, as shown in Fig. 16(b).



(a) (b)

Fig. 16 (a) General neuron based on Matsuoka model[19] (b) Matsuoka neuron oscillator with extensor and flexor neurons[19]

In RoboCup3D, Bavani et al. [19] used genetic algorithms to optimize the parameters of the Matsuoka oscillator model to generate stable walking and used the Center of Pressure (COP) as real-time feedback control input to form closed-loop control.

4 Multi-agent Collaboration Strategy Design

The goal of research in the RoboCup3D simulation environment is to complete a soccer robot game and win it. Therefore, the design of multi-robot collaboration strategies is crucial to achieving this goal. Robots have skills such as walking on the field, dribbling, kicking the ball, and getting up after falling, which are the basic guarantees for strategy execution. With these skills, efficient completion of a series of specified tasks can be achieved through coordination with other robots, thus winning the game.

As shown in Fig. 17, we divide strategies into two categories: basic strategies and tactical strategies. Basic strategies include three types: field positioning, formation design, and role allocation. Tactical strategies are divided into three types: offensive tactics, defensive tactics, and tactical training.

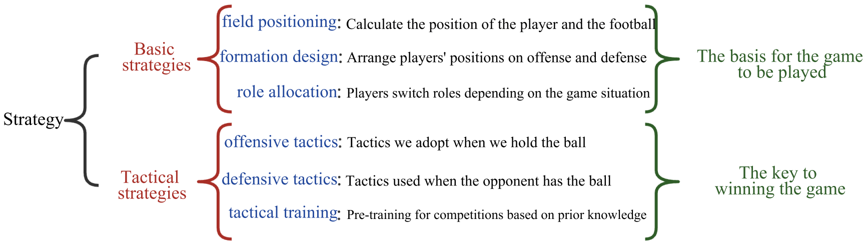

Strategy
Basic strategies
Tactical strategies
field positioning: Calculate the position of the player and the football
formation design: Arrange players' positions on offense and defense
role allocation: Players switch roles depending on the game situation
offensive tactics: Tactics we adopt when we hold the ball
defensive tactics: Tactics used when the opponent has the ball
tactical training: Pre-training for competitions based on prior knowledge
The basis for the game to be played
The key to winning the game

Fig. 17 Policy classification

4.1 Basic Strategies

The basic strategies of robots are the fundamental guarantees for the successful execution of other strategies. According to their different functions, they can be divided into three categories: field positioning, formation design, and role assignment.

4.1.1 Field Positioning

Accurate positioning of our players, enemy players, and the ball on the field is the prerequisite for good cooperation between robots in the simulation environment. Positioning techniques can be divided into relative positioning and absolute positioning. Relative positioning refers to the robot estimating its position for the next moment based on its position at the previous moment, as shown in Fig. 18, where the dashed line is the actual trajectory and the arrow is the position change vector estimated by the robot relative to the previous moment. The relative positioning method accumulates errors over time and is not suitable for long-term accurate positioning, while the absolute positioning method, which relies on visual information and fixed markers on the field, is not completely accurate due to limited visual information and noise.

As shown in Fig. 19, there are 8 fixed markers on the soccer field in the RoboCup 3D simulation environment. Prior to 2009, robots had perfect vision [115], meaning that they could see all the markers on the field regardless of their location. To simulate a more realistic environment, since 2009 robots have had limited vision [116], with

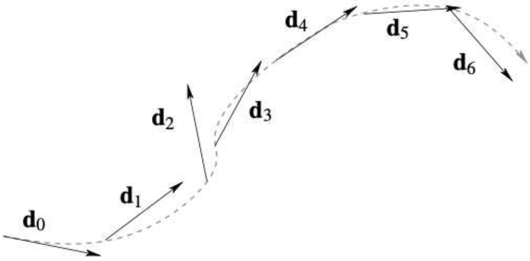


Fig. 18 Relative Positioning

a viewing angle of only ±120 degrees. When a marker is outside the field of view, its position information cannot be obtained. In addition, the visual information has added noise in the form of a normal distribution centered at zero $\mu = 0.0$, with errors in distance, horizontal angle, and vertical angle, all satisfying a normal distribution with mean of zero. These errors can cause the position information calculated by the robot based on visual information to be less accurate.

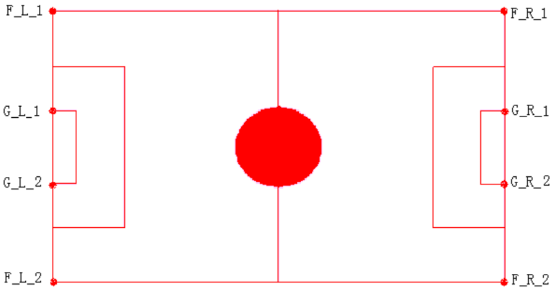


Fig. 19 Fixed Marker Post

The most basic method of robot localization is the landmark localization method [117–119]. The robot can obtain the position of the landmarks through visual information, and its own localization can be obtained through algorithmic processing. Because the robot's vision is limited by angles, it is difficult to see all the landmarks at every moment. Therefore, different numbers of landmarks within the visual range are selected for localization according to different situations. In the common case where there are three or more landmarks in the field of view, a system of three linear equations can be solved using the coordinates of the three landmarks $(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)$ and the distances between them and the robot d_1, d_2, d_3 obtained from the visual information. The global coordinates of the robot

(x, y, z) can be obtained by solving the three equations simultaneously.

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = d_1^2, \\ (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = d_2^2, \\ (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = d_3^2, \end{cases} \tag{27}$$

When the markers cannot be seen, the dead reckoning method [120] is used, which is a relative positioning method. The basic idea is to use gyroscopic information to estimate the robot's posture, and then calculate the current coordinates based on the last known position and the change in posture. The estimation of posture change mainly uses the rotation matrix R, assuming that the robot's posture rotates α, θ, β around the X-axis, Y-axis, and Z-axis, and the original coordinates transform from $P(x, y, z)$ to the new coordinates $P(x', y', z')$ using the formula:

$$P(x', y', z') = R \cdot P(x, y, z) \tag{28}$$

The R means:

$$\begin{bmatrix} C_\alpha C_\theta - S_\alpha C_\theta + C_\alpha S_\theta S_\beta & S_\alpha S_\beta + C_\alpha S_\theta C_\beta \\ S_\alpha C_\theta C_\alpha C_\theta + S_\alpha S_\theta S_\beta & -C_\alpha S_\beta + S_\alpha S_\theta C_\beta \\ -S_\theta C_\theta S_\beta & C_\theta C_\beta \end{bmatrix} \tag{29}$$

$cos\theta$ is denoted as C_θ here for simplicity. The other steps are similar.

If the head pose of the robot is obtained based on this, the poses of other parts of the robot can be obtained using the chain multiplication rule of homogeneous transformations [8].

In addition, when there is only one or two landmarks, the robot's global coordinates are usually obtained by combining the position and pose changes and the relative position between the robot and the landmarks [121].

However, since visual information contains noise, there are errors in any landmark localization method. In RoboCup3D, the Kalman filter method [122, 123] is commonly used for denoising. The Kalman filter has two stages, prediction and correction. The prediction stage uses the optimal result from the previous iteration to predict the current value, and the correction stage uses the observation value to correct the current value, thereby obtaining the current optimal result.

(1) Prediction stage:

$$x_k = Ax_{k-1} + Bu_{k-1}, \tag{30}$$

$$P_k = AP_{k-1}A^T + Q, \tag{31}$$

(2) Correction stage:

$$K_k = P_k H^T (H P_k H^T + R)^{-1}, \tag{32}$$

$$x_k = x_k + K_k(z_k - Hx_k), \tag{33}$$

$$P_k = (I - K_k H)P_k, \tag{34}$$

where x_k is the state at time k, A is the state transition matrix, u_k is the effect of external factors on the system at time k, B is the input control matrix, P is the error matrix, Q is the covariance matrix of prediction noise, R is the covariance matrix of measurement noise, H is the observation matrix at time k, K_k is the Kalman gain at time k, and z_k is the observation value at time k.

The Kalman filter algorithm calculates the predicted value of the robot's current position based on the previous position, and uses information obtained from the visual field as measurement values, combines the predicted and measured values to obtain the optimal value of the robot's current position. Since the relationship between the estimated variables and the process is usually nonlinear in actual positioning, Sun et al. [122] and Seekircher et al. [124] considered using the extended Kalman filter for positioning in nonlinear situations. This method has a similar basic idea to the Kalman filter, but it linearizes the nonlinear function by Taylor expansion, omitting high-order terms and retaining the first-order terms of the expansion to achieve linearization.

Particle filtering is another commonly used positioning method, also known as Monte Carlo localization [125], which is widely used in RoboCup3D [22, 126, 127]. The basic steps of particle filtering are to initialize particles, and each particle returns a prediction of a robot's location in the form of (x, y, θ) and the likelihood of the prediction result. Then, the weights of the particles are updated based on the measurement values extracted from the walking engine and the observation values from the markers, and then resampled according to the weights, and the weights are updated again, and the resampling is repeated, continuously optimizing the self-position information. The accuracy of the particle filtering method is generally stronger than that of the Kalman filter, but the positioning efficiency of the Kalman filter is usually higher, so the specific filtering method used depends on the situation.

In RoboCup3D, when a robot is removed from the field due to a foul or other reasons, resulting in a sudden change in position, it is referred to as the ”Kidnapping problem”. In such cases, the resampling of particles may fail to locate the target successfully. To address this problem, a common approach is to add random particles during each round of resampling. Bustamante et al. [126] enhance the model's robustness by completely resampling a portion of particles from the field area each time. Shen et al. [29] add two parameters to track the long-term and short-term changes in particle weight average, and release random particles when the particle weight decreases.

In addition to the commonly used localization methods mentioned above, Wang et al. [128] and Fu et al. [129] assist in localization by using the endpoints formed by the marker lines and the marker poles. Simoes et al. [130] and Fernandes et al. [131] use communication to assist in localization. Lu et al. [132] use a Long Short-Term Memory (LSTM) network for localization, with visual information, gyroscope information, accelerometer information, joint motion information, etc. as network inputs, and robot position information as output. Abreu et al. [101] further consider the z-axis direction and use a three-dimensional coordinate system for localization.

4.1.2 Formation Design

The scale of robots in the Robocup3D simulation environment is constantly expanding, from the initial 3v3, to 6v6, and then to 11v11, fully simulating the real scale of human soccer matches. With the increase in the number of players on both sides, the arrangement of these players, namely formation design, plays a crucial role in the development of the entire game.

At first, most teams used manual formation plans. Chen [133] manually divided the robots into two groups, offense and defense. The offense group positions itself by adding a certain offset to the position of the ball, while the defense group positions itself by adding a certain offset to the line connecting the ball and the center of the goal. During the game, the offset is continuously adjusted according to the situation on the field, following predetermined rules to avoid special situations such as robots moving out of the field. This type of method positions the players of the entire formation based on adding a certain offset to the position of the ball, which is singular and fixed, unable to respond to multiple situations.

Feng [40] and He et al. [134] used Delaunay triangulation for formation design, considering a more comprehensive situation. The main idea is to use the method of Delaunay triangulation [135], as shown in Fig. 20, to divide the field into a triangular mesh model, where each vertex of the triangle represents a key position of the ball. Formation positions for all key positions are designed in advance. When the ball is at any point, the formation is calculated using linear interpolation based on the pre-set formation for the key positions. This method increases the diversity of formations.

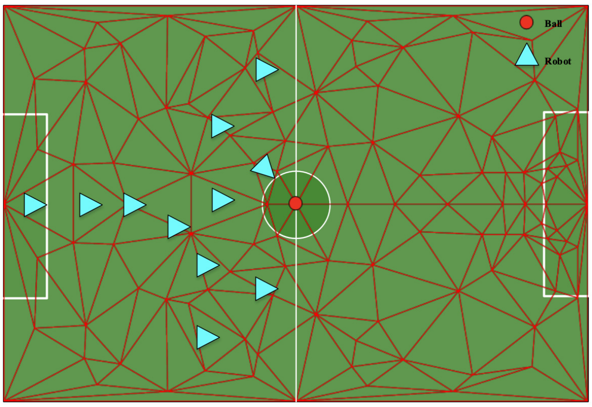


Fig. 20 Delaunay[40]

With the development of artificial intelligence technology, Larik et al. [136] no longer limit themselves to hand-crafted approaches but instead use evolutionary algorithms for formation design. It mainly considers defensive scenarios and simplifies the defensive player formation into nine distance parameters as input, which are passed into the evolutionary algorithm for optimization. The fitness function maximizes the difference between the number of goals scored in this match and the previous one, the time of ball possession, and the time the ball is not in a dangerous area. After multiple iterations, the algorithm terminates when the set evolution generations are reached, and returns the optimized parameters. Compared with hand-crafted methods, this approach has improved the number of goals scored against the same opponent.

4.1.3 Role Allocation

In the RoboCup3D simulation environment, role allocation is an important method of multi-robot collaboration, which assigns different positions to several robots on the field according to the current situation, so that they can take on different role tasks. If each robot's role remains the same throughout the game, it is easy to lead to defensive gaps or weak attacks in certain positions. Therefore, dynamic role allocation based on changes in the situation on the field is needed at all times. As shown in Fig. 21, A_1, A_2, A_3 represents three robot players, P_1, P_2, P_3 represents the target position, and the arrow represents the current robot's mapping scheme with the target position. The most important problem in role allocation is to find a mapping scheme that allows multiple robots to move to their respective target positions in the shortest possible time.

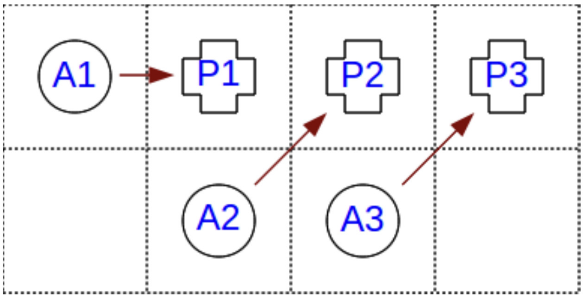


Fig. 21 Role Mapping[137]

Assuming that the target position is fixed, the shorter the time it takes from the start of the robot movement to the last robot reaching the target, the better the result of role allocation. The most primitive role allocation scheme uses brute force search to calculate the total distance required for each robot to reach the target role position under all mapping scenarios, and the mapping method with the smallest distance is the optimal solution after sorting.

Chen et al. [138] proposed a role assignment algorithm based on path cost, which assigns roles to each robot by comparing path costs. The calculation of path cost is obtained by weighting the distance from the robot to the target position, the angle required for the robot to turn to the target position, and the number of obstacles between the robot and the target position. The roles in this strategy are selected in a fixed order. The robot with the minimum path cost is assigned the corresponding role each time, and it will no longer participate in the subsequent calculations. To avoid frequent reassignments, a cost function is introduced, that is, the change will only be made when the benefit after changing the current role assignment strategy is greater than the change cost.

Ulusoy et al. [139] use a case-based reasoning (CBR) approach to assign player roles. The main idea of this method is to dynamically allocate the number of robots playing center and back roles, based on different scenarios in the case library, except for the goalkeeper and one forward player. For example, in a disadvantageous scenario such as being one goal behind, more center forwards will be assigned to attack in the hope of achieving a draw.

Considering that the original brute force search method requires comparing the cost of all role mappings, when the number of roles involved in allocation is n, there will be $n!$ mappings, which results in a huge calculation workload. Furthermore, there is a real-time requirement in the game. Therefore, MacAlpine et al. [42, 137] proposed using dynamic programming to find the mapping with the minimum cost. The original problem is decomposed into several sub-problems, which are then solved to obtain the solution to the final problem. Specifically, in the game, the method can be used to find the mapping with the minimum cost for k players to the role positions based on the minimum cost mapping of $k - 1$ players to the role positions. The number of evaluations in this iterative method is $n2^{n-1}$ times. In an 11v11 game, n is 10 after excluding the goalkeeper. Brute force search requires 3628800 evaluations, while dynamic programming requires only 5120 evaluations. In addition, since the players cannot obtain complete and accurate information, there may be differences in the best mapping calculated by each robot. Therefore, they rely on player communication and establish a voting system among players to determine the final role allocation plan based on the mapping with the most votes.

Li et al. [140] further improved the dynamic programming method by using the idea of matrix minimum adjustment for role allocation. The main idea is to use a distance matrix to record the cost of each robot moving to the target position, and then solve the optimal allocation plan step by step using the upper limit minimum incremental matrix adjustment algorithm. The time complexity of this algorithm is further reduced compared to the previous dynamic programming method.

MacAlpine et al. [141] further extended the dynamic programming method by proposing a priority role assignment of target positions. The difference from previous methods is that in the calculation process, the robots are assigned to higher priority positions first, which are the ones closest to the ball or the opponents. Although this method increases the completion time and distance for all robots to reach all targets, it shortens the time and distance for all high-priority targets, which is more practically meaningful in real game scenarios.

Chen et al. [142] divided the robots into three categories: one forward, one goalkeeper, and nine other players. With the goalkeeper position fixed, they first constructed a value function based on various factors such as distance, to select the robot with the highest value among the remaining ten players as the forward. Then, they used the KM (Kuhn-Munkres) algorithm with distance as edge weight to make the optimal match between other players and target positions, while also adopting a priority assignment scheme similar to that in reference [141]. This method set higher priority for defending the vacant areas, thereby improving the team's defensive ability against long-range passes from the opponents.

Abeyruwan et al. [143] introduced reinforcement learning strategies for role assignment. This method first represents the robot soccer scene with knowledge [144], and then uses two methods, Greedy-GQ(λ) and OP-GTD, to learn the dynamic role assignment function. That is, the robots learn which role they should assume to maximize the overall team reward. However, this method only performed better than manually programmed methods in 3v3 games, and its performance decreased as the number of players on both sides increased.

4.2 Tactical Strategy

Tactical strategy can be divided into three types according to different tactical purposes: attacking tactics, defensive tactics, and tactical training.

4.2.1 Attacking Tactics

Attacking tactics are usually divided into ball possession attacks and passing cooperation. Path planning is the key to ball possession attacks, and its main task in RoboCup3D is to plan the optimal or suboptimal path from the starting point to the target point while avoiding obstacles. Specifically, it is to find a path for the ball holder to reach the specified position that can avoid obstacles such as opposing defensive players and our fallen players. When there are no obstacles between the robot and the target point, moving straight ahead is the optimal path. If there are obstacles, obstacle avoidance measures need to be taken. The commonly used path planning algorithms for obstacle avoidance include the shortest tangent method and the grid method.

The shortest tangent obstacle avoidance algorithm [145] is a simple and commonly used obstacle avoidance method. As shown in Fig. 22, first set a collision radius R for the obstacle P, and the circular area within this radius is set as a possible collision area. To bypass such a circular area, you can walk along the tangent direction of the circle. If the robot O is closer to point A, it chooses the tangent of point A; if it is closer to point B, it chooses the tangent of point B; if the distance is the same, it randomly chooses one, and then moves towards the target point after avoiding the obstacle, thereby achieving local obstacle avoidance. To avoid frequent changes in direction, when the obstacle is still moving within the rolling window, the current path is not easily changed.

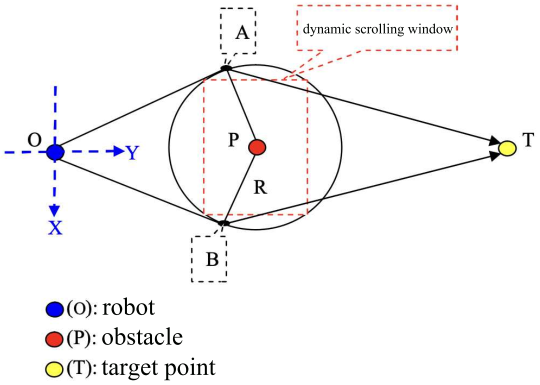


Fig. 22 Shortest tangent method[137]

Zheng et al. [146] and Su et al. [147] considered that the method of walking along the tangent line can lead to a less smooth obstacle avoidance path, which can cause the robot to easily fall when adjusting direction. Therefore, they proposed a smooth Nearness Diagram (ND) obstacle avoidance algorithm. This algorithm takes into account the movement of obstacles and regards them as active circular areas for path planning. The actual turning angle is obtained by adding the expected direction angle and the avoidance turning angle, which makes the turning angle smoother and maintains the stability of the robot's movement.

The grid method is another commonly used method, which divides the field into multiple regions called grids. The grid with no obstacles is marked as passable, and the grid with obstacles is marked as impassable. These passable grids can form a connected graph, and the shortest path from the starting grid to the ending grid can be obtained by searching on the graph. In RoboCup3D, the commonly used shortest path solving algorithms are the global planning A* search [148] and the local planning artificial potential field method [149]. The A* algorithm directly searches for the shortest path from the starting point to the ending point using a greedy strategy on the gridded field with obstacles. The basic idea of the artificial potential field method is to simulate the robot's movement in the field as a movement in an abstract artificial gravitational field, where the target position exerts "attraction" on the robot and obstacles exert "repulsion" on the robot. Finally, the motion of the robot is controlled by the resultant force.

In addition, Li et al. [33, 150] proposed the swarming control algorithm, which is based on quantized information swarming control for multi-robot formation and obstacle avoidance. It achieves automatic maintenance of hand-crafted formations through the swarming control algorithm and achieves the effect of group obstacle avoidance. Muzio et al. [103] proposed a ball-carrying strategy using a deep reinforcement learning method. By establishing a ball-carrying and obstacle avoidance environment when facing a single opponent, it learns the appropriate walking actions that should be taken in this environment.

The outcome of a game depends on the score, and scoring requires shooting. When the ball holder avoids the defending players and comes to the front of the goal, it is necessary to judge the appropriate shooting point based on the situation. Yao et al. [151] proposed a shooting position judgment method. Firstly, the weighted sum of the current goal distance, our team's adjustment time, and the opponent's interception situation is used as the benefit value to measure whether direct shooting is possible. When the benefit value is greater than the set value, direct shooting is performed. Otherwise, the A* search algorithm is used to find the appropriate shooting path. Shen et al. [152] used the Q-learning algorithm in reinforcement learning to train the forward shooting. The action space is the power and angle parameters of the shot, and the state space is the position of the ball in the eight areas around the goal, and the corresponding reward is set according to the position of the goal. The experimental results show that compared with the hand-coding method, the reinforcement learning scheme significantly improves the goal efficiency.

Ball possession and attack improve the single combat capability of our robots, but football is a team cooperation game. Good passing strategies can sometimes play a very critical role.

Rettinger et al. [153] proposed the ExpBoost algorithm based on AdaBoost, which enhanced the utilization of the knowledge learned in the past and applied this method to the RoboCup3D passing task, which had more advantages when encountering new opponents outside the training set. Shi Guoqiang et al. [118] proposed a means of two-person cooperation attack, in which one person holds the ball, and the other person receives it in a parallel position. When the ball holder is intercepted, consider passing it to the receiver. At this time, based on role allocation, the receiver becomes the ball holder and executes the relevant attack strategy.

Due to the limitations of robot movements, early robots rarely used passing tactics to cooperate in attacks. Generally, the player closest to the ball would hold it, and other teammates would be responsible for receiving passes. However, with the continuous improvement of movements, especially the emergence of long-distance kicking actions, more passing methods gradually appeared in the game. The traditional passing tactics training was often used in RoboCup2D, with a classic task called Keepaway [154]. Keepaway refers to a training where two teams compete for control of the ball in a certain-sized field. The ball control team is responsible for holding the ball as long as possible in the field area, while the ball-snatching team is responsible for taking the ball away from the ball control team as quickly as possible. The scale of the Keepaway task can be arbitrarily set in terms of the size of the field, the number of ball control and ball-snatching players, etc. Fig. 23 shows a schematic diagram of a 3v2 Keepaway task. The team [29, 155–157] introduced the Keepaway task from RoboCup2D to the RoboCup3D platform based on its characteristics, and established a reinforcement learning training model under the 3D platform, combined with the Sarsa(λ) algorithm to train local passing tactics to obtain longer ball control time. Gupta et al. [158] divided the continuous training area in Keepaway into several discrete points and used 8-person-defined input features to enter the neural network to evaluate the value of each point, in order to select the best passing point.

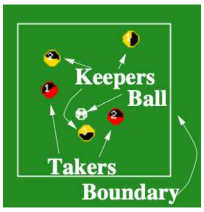


Fig. 23 Keepaway:3V2

4.2.2 Defensive Tactics

Defensive tactics involve both the goalkeeper and the defenders. The goalkeeper is the last line of defense and their main focus is to decide when to dive for the ball. One approach is to set a diving line in front of the goal, and when the ball crosses this line, the goalkeeper dives for the ball. However, this approach can be problematic as the goalkeeper may continue to dive even when the ball has stopped inside the diving line, which could give the opposing team an opportunity to score. Another approach is to set a threshold speed along with the diving line so that the goalkeeper only dives when the ball is above a certain speed within the diving line. However, selecting an appropriate threshold speed can be difficult, as a high threshold speed may cause the goalkeeper to miss a threatening ball, while a low threshold speed may cause the goalkeeper to dive unnecessarily.

Huang et al. [159] proposed a ball prediction-based diving decision-making method. They first established a free motion model for the ball and predicted its position after a certain amount of time to determine whether the goalkeeper should dive and in which direction. Masterjohn et al. [160] used two methods to improve the goalkeeper's efficiency. The first method was similar to Huang et al.[159]'s approach and used linear regression and Kalman filtering to estimate the ball's position. The second method used a psychological simulation framework called Orpheus [161], where the goalkeeper used mental simulation to evaluate which action was most advantageous before executing it. In addition, they also considered the goalkeeper's positioning, forming an isosceles triangle to defend both sides of the goal. Chen et al. [133] mainly considered the goalkeeper's positioning, attempting to make the distance between the goalkeeper and both posts equal and ensuring that the goalkeeper faced the ball as soon as they reached their target position to make an immediate decision. Wei et al. [162] first measured parameters such as movement speed, shot speed, and player movement direction and established a motion model using these parameters to calculate the optimal position for the goalkeeper to defend.

Apart from the goalkeeper, the defenders also play an important role in defense. When the ball is in the opposing team's possession, our players need to actively fight for possession. Yao et al. [151, 163] studied robot interception techniques and proposed using neural network technology to select long-distance interceptions when close-range interceptions are not suitable. Input to the three-layer BP neural network included the ball's speed, direction, and size, the distance between the ball and the robot, and the robot's speed and direction. The output was the robot's action, and the training set consisted of 500 interception sequences. The experimental results showed that the neural network method was faster and more successful than the original method, but acquiring a suitable training set may be difficult. Zhu et al. [164] proposed a real-time interception algorithm, which first calculated the relative velocity between two robots and then estimated the required time to approach based on their relative distance at the current time. The algorithm then estimated the opponent's future position and determined the interception point. Predicting the ball's movement trend can also facilitate interception, and Mirmohammad et al. [165] proposed a method to predict the ball's path in the next time period by combining KNN regression and autoregressive methods. The KNN method predicted the path based on previously observed path data, while the autoregressive method predicted based on the current path. The two methods were combined to form the final prediction model. In addition, they also considered the possibility of multiple robots intercepting the ball simultaneously, and used a voting method to select the final interception point.

4.2.3 Tactical Training

Teams can improve their tactical level through tactical training based on data from past games and experts' experience. First, robots can be trained to assess game situations. Abadi et al. [166] proposed an evolutionary neural network to predict future game states, with world state parameters as input and opponents' next positions as output. They used a multi-layer feedforward neural network trained with backpropagation and optimized with a genetic algorithm. Yang et al. [167] proposed a deep neural network-based method for football robot situation assessment, designing scenario factors based on experts' experience to construct training data. They then proposed a deep neural network with multiple hidden layers. Finally, during training, scenario factors were used as input and corresponding evaluation results as output.

Secondly, teams can learn from excellent teams. Larik et al. [168, 172, 173] extracted data from game log files, labeled data on different behaviors by experts, and used the WEKA software [174] and PART algorithm [175] to obtain a rule library for recognizing robot behaviors. Finally, they predicted opponents' behaviors based on the rule library. Raza et al. [169] did not use manually defined rule libraries, but learned from observing others' behaviors. This approach can avoid complex problems such as role allocation and path planning encountered in manual definition. The main idea is to select two robots from excellent teams as teachers, extract corresponding data from the teacher robots' log files, and train a multi-layer perceptron to predict the player's next position.

At the same time, tactics can be planned in advance. For example, Simoes et al. [170, 176–179] and their team have been committed to the research of set play technology for many years. The main idea is to make the team perform a series of predefined action sequences during the game. The specific method is to determine the termination time and termination condition, the player numbers and their corresponding positions, and establish a deterministic finite automaton to execute related behaviors and transition to the next state, continuing to execute related behaviors until termination. In addition, to exercise the team's temporary teamwork ability, the team [171, 180] proposed a challenge called Drop-In Player, in which robots from different teams are selected for cooperation to improve tactical level. Table 3 summarizes the different research angles and solutions for RoboCup3D tactical strategies.

5 Practical Solutions to RoboCup3D Related Issues

To complete a full game in the RoboCup3D simulation environment, many issues need to be addressed in research related to this environment. Researchers have proposed many solutions to these problems, as summarized in the previous two sections. However, an effective and universal process for solving the many problems in the RoboCup3D simulation environment still needs to be established. Teams participating in RoboCup3D simulation games not only research these related issues but also combine them with actual gameplay, and have produced a large number of reference materials. Therefore, We have summarized a set of solutions, many of which refer to the research of University of Texas at Austin(UT). The team of University of Texas at

Table 3 Tactical Strategies

Strategie	Author	Research	Problem	Solution
Attacking Tactics	Cheng[145](2005)	R1	P1	Shortest tangent obstacle avoidance algorithm
	Rayermann[148](2016)	R1	P1	Grid-based method and A* search
	Muzio[149](2016)	R1	P1	Grid-based method and artificial potential field
	Muzio[103](2020)	R1	P1	Deep reinforcement learning
	Li[33](2015)	R1	p2	Swarm control algorithm
	Yao[151](2012)	R1	P3	Value-based judgment and A* search
	Shen[152](2011)	R1	P3	Q-learning
	Rettinger[153](2006)	R2	P4	ExpBoost
	Shi[118](2010)	R2	P5	Passing and receiving
	Shen[29](2015)	R2	P6	Sarsa(λ) algorithm
	Gupta[158](-)	R2	P6	Neural Network
Defensive Tactics	Huang[159](2008)	R3	P7	Ball motion model
	Masterjohn[160](2015)	R3	P7	Psychological simulation
	Chen[133](2014)	R3	P8	Position assignment
	Wei[162](2022)	R3	P9	Motion model
	Yao[151](2012)	R4	P10	Neural Network
	Zhu[164](2012)	R4	P11	Ball motion model
	Mirmohammad[165](2021)	R4	P11	Regression model
Tactical Training	Abadi[166](2006)	R5	P12	Evolutionary neural networks
	Yang[167](2019)	R5	P13	Deep Neural Network
	Larik[168](2012)	R6	P14	PART algorithm
	Raza[169](2012)	R6	P14	Multilayer perceptron(MLP)
	Simoes[170](2018)	R7	P15	setplay algorithm
	MacAlpine[171](2014)	R8	P16	Drop-In Player algorithm

Here are the meanings of the respective abbreviations. R1: attack with the ball; R2: pass the ball; R3: goalkeeper defense; R4: defender defense; R5: situation assessment; R6: team simulation; R7: tactical planning; R8: Temporary cooperation; P1: Single-body obstacle avoidance problem in dribbling path planning; P2: Group obstacle avoidance problem in dribbling path planning; P3: Obstacle Avoidance Problem in Shooting Path Planning; P4: Passing problems when facing a new opponent; P5: Two-player cooperative passing problem; P6: Keepaway; P7: The problem of goalkeeper skill selection; P8: The issue of goalkeeper's regular positioning; P9: The issue of goalkeeper's movement to defensive positioning; P10: The selection of interception methods; P11: The selection of interception points; P12: Predicting the match state; P13: Determining the match scenario; P14: Predicting opponent behavior; P15: Predefining action sequences; P16: Collaboration between teammates from different teams.

Austin has won the World Cup championship 9 times between 2011 to 2022, making their research methods of great reference value.

5.1 skill Generation and Optimization

Based on considerations of football robots, it is necessary to generate optimized skills, including necessary walking, kicking, getting up, and some unique skills.

The generation and optimization of walking skill is often the top priority for a team. The research[70] discussed the generation and optimization of walking skill at the earliest stage, and transferred the walking skill of the real Nao robot to the simulated Nao environment at the beginning. They designed keyframes for the simulated

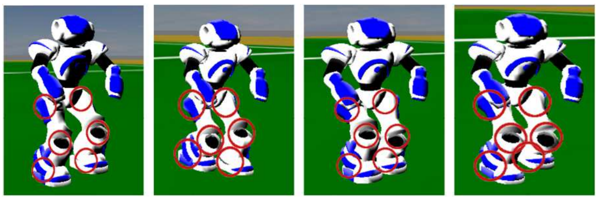
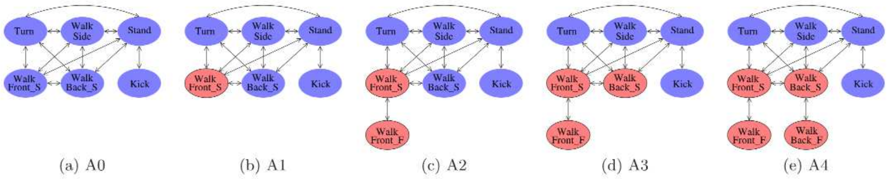


Fig. 24 Nao robot walk frames with the joints[70]

Nao environment to achieve walking skills in multiple directions (Fig. 24), including forward, backward, sideways, and turning, and then used the CMA-ES algorithm to optimize the parameters extracted from the keyframes. Finally, each type of walking skill was improved. They also considered the connection effect between different walking skills, such as optimizing combination skills like straight walking-turning-straight walking, and proposed a hierarchical optimization method to optimize combination skills such as carrying the ball to the target point (this skill requires a combination of straight, sideways, and turning walking). The specific method is to optimize each skill one by one, and fix the parameters of the optimized skill to optimize the next skill (Fig. 25).



(a) A0 (b) A1 (c) A2 (d) A3 (e) A4

Fig. 25 Constraints on transitions between skills represented as state diagrams. For Agent A0 neither the WalkFront S nor the WalkBack S skills is optimized;the former is optimized (shown with thick border) under A1. Further skills are added and optimized subsequently under agents A2, A3, and A4. Agent A5 is identical to A4, except for retuning thresholds and the logic for selecting and invoking our new learned skills.[70]

However, the skill generated by the keyframe method is too fixed and cannot achieve omnidirectional walking. In order to develop omnidirectional walking, The research[22] used a three-dimensional linear inverted pendulum model to build a walking engine for the Nao robot in the simulated environment, and achieved omnidirectional walking. The parameters optimized by CMA-ES were also transformed into parameters in the three-dimensional linear inverted pendulum, and the walking was divided into three types: sprinting for speed, going to the target for stability, and positioning for quick position adjustment. They optimized the parameters of each walking type through hierarchical optimization and dynamic reward setting(Fig. 26), and finally achieved omnidirectional walking that can handle all situations, with significant improvements in both speed and stability. This method of generating and optimizing walking skill by the team [22] is still widely used by many teams today.

Stable walking skill is crucial, while a powerful kicking skill can provide a team with a winning opportunity, as kicking the ball into the goal is often more convenient

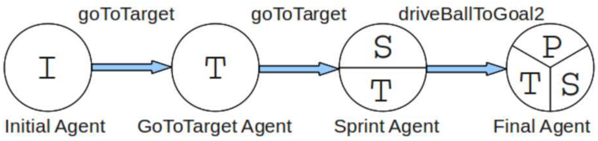


Fig. 26 Hierarchical optimization of different walking movements[22]

and unstoppable than dribbling ball into the goal. The research[70] also transferred the kicking skill developed on the real robot to the simulation environment and used CMA-ES to optimize the keyframes, achieving a kicking distance of about 5.09m. The research [182] designed a kicking trajectory for the robot using kinematics and implemented a complete robot skill using cubic spline interpolation (Fig. 27), and optimized it using CMA-ES. The kicking skill designed based on dynamics is more robust, but the final kicking distance is not as good as that obtained by the keyframe method. In research[53], it was reported that the keyframe-based kicking can be optimized to about 12m, while the kinematics-based kicking is about 6m.

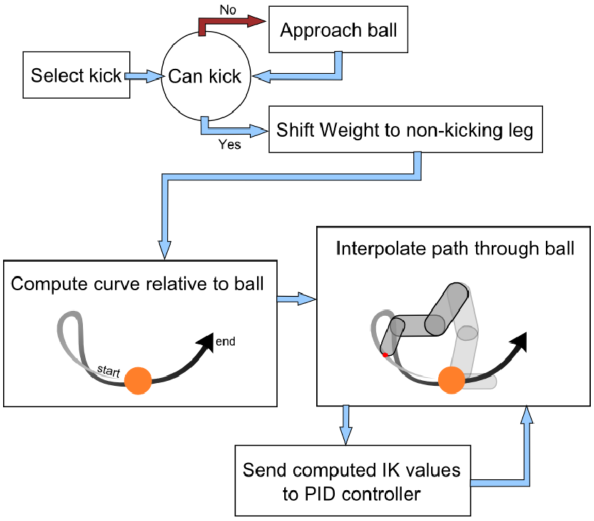

No
Approach ball
Select kick
Can kick
Yes
Shift Weight to non-kicking leg
Compute curve relative to ball
start
end
Interpolate path through ball
Send computed IK values to PID controller

Fig. 27 Kinematically based kicking[22]

The reasch[45] aimed to further improve the distance of kicking the ball to score a goal from a far distance away from the opponent's goal post. The upper limit of the CMA-ES optimization depends on the quality of the initial seed, which in the case of optimizing kicking behavior, is the quality of the initial keyframe kicking skill. The research [45] used the idea of imitation learning to obtain keyframes of good kicking skills by observing the kicking skills of other teams, and used these keyframes as the initial seed for CMA-ES optimization. This process is called KSOBI (Keyframe Sampling, Optimization, and Behavior Integration). Ultimately, the research [45] reported that the farthest kicking distance achieved was around 20 meters, which is almost two-thirds the length of the football field and could allow scoring a goal from one's own half of the field. It is because of such kicking skills, UT dominated the competition for many years, winning almost every competition with dozens of goals scored and none conceded. The KSOBI method of generating and optimizing kicking skills has been widely adopted by most teams to optimize their own kicking skills. The KSOBI process is shown in Fig. 28.

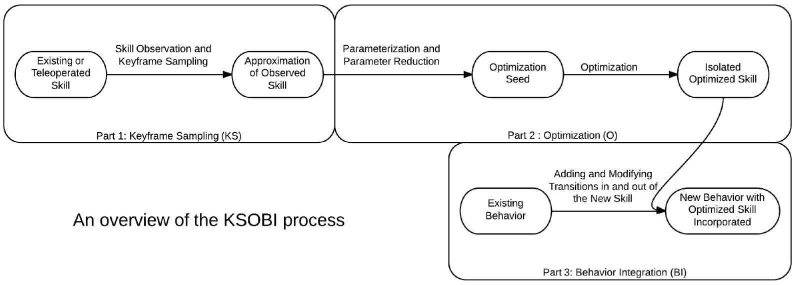

Existing or Teleoperated Skill
Skill Observation and Keyframe Sampling
Approximation of Observed Skill
Part 1: Keyframe Sampling (KS)
Parameterization and Parameter Reduction
Optimization Seed
Optimization
Isolated Optimized Skill
Part 2 : Optimization (O)
Adding and Modifying Transitions in and out of the New Skill
Existing Behavior
New Behavior with Optimized Skill Incorporated
Part 3: Behavior Integration (BI)
An overview of the KSOBI process

Fig. 28 An outline of KSOBI[45]

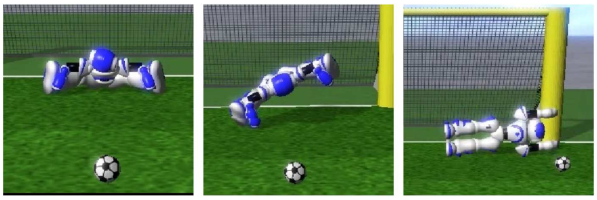


Fig. 29 Blocking action by the goalie[22]

Collisions causing robots to fall down is a common occurrence in RoboCup3D matches, usually happening during ball fights. Whichever team's robot can get up first will have a faster chance of gaining ball possession. The research[70] initially used the transfer of real machine skills to simulate a getting-up skill, and then optimized it using the CMA-ES algorithm to make it even faster. Additionally, the research[22] implemented special skills such as diving and leg blocking for goalkeepers through keyframe methods (Fig. 29). In research[49], walking and kicking skills were directly connected to achieve kicking while walking, saving time on adjusting before kicking. This was achieved by optimizing parameters using CMA-ES, resulting in a walking and kicking skill that could reach 20 meters.

As mentioned in section 3.1.6, optimizing a single skill does not necessarily guarantee that different skills performed consecutively will achieve the same effect. Therefore, most teams choose a layered optimization approach to optimize the combination of different skills. The research[183] proposed an overlapping layered optimization method (Fig. 30), which not only fixes the parameters of optimized skills but also optimizes different skill combinations by adopting different fixed parameter methods based on the relationships between different skills. Through a more granular layered optimization method, robots can perform more stably in real matches.

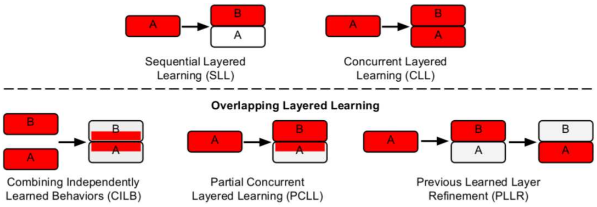
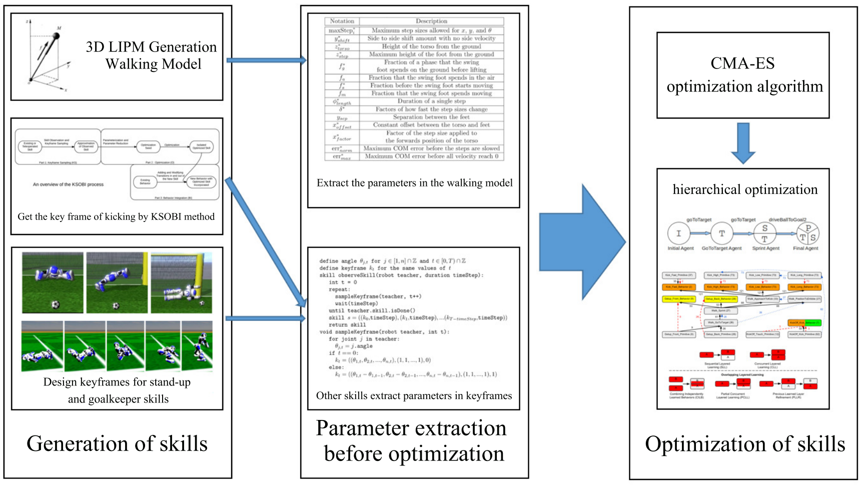

A
B
A
Sequential Layered Learning (SLL)
A
B
A
Concurrent Layered Learning (CLL)
Overlapping Layered Learning
B
A
B
A
Combining Independently Learned Behaviors (CILB)
A
B
A
Partial Concurrent Layered Learning (PCLL)
A
B
A
B
A
Previous Learned Layer Refinement (PLLR)

Fig. 30 Overlapping Layered Learning[183]

We summarize skill generation and optimization methods in Fig 31:


3D LIPM Generation Walking Model
Get the key frame of kicking by KSOBI method
Design keyframes for stand-up and goalkeeper skills
Generation of skills
Extract the parameters in the walking model
Other skills extract parameters in keyframes
Parameter extraction before optimization
CMA-ES optimization algorithm
hierarchical optimization
Optimization of skills

Notation	Description
maxStep_i^*	Maximum step sizes allowed for x, y, and θ
y_{shift}^*	Side to side shift amount with no side velocity
z_{torso}^*	Height of the torso from the ground
z_{step}^*	Maximum height of the foot from the ground
f_g^*	Fraction of a phase that the swing foot spends on the ground before lifting
f_a	Fraction that the swing foot spends in the air
f_s^*	Fraction before the swing foot starts moving
f_m	Fraction that the swing foot spends moving
ϕ_{length}^*	Duration of a single step
δ^*	Factors of how fast the step sizes change
y_{sep}	Separation between the feet
x_{offset}^*	Constant offset between the torso and feet
x_{factor}^*	Factor of the step size applied to the forwards position of the torso
err_{norm}^*	Maximum COM error before the steps are slowed
err_{max}^*	Maximum COM error before all velocity reach 0

```
define angle $\theta_{j,t}$ for $j \in [1,n] \cap \mathbb{Z}$ and $t \in [0,T) \cap \mathbb{Z}$
define keyframe $k_t$ for the same values of $t$
skill observeSkill(robot teacher, duration timeStep):
   int t = 0
   repeat:
     sampleKeyframe(teacher, t++)
     wait(timeStep)
   until teacher.skill.isDone()
   skill $s = ((k_0, \text{timeStep}), (k_1, \text{timeStep}), ...(k_{T-timeStep}, \text{timeStep}))$
   return skill
void sampleKeyframe(robot teacher, int t):
   for joint $j$ in teacher:
     $\theta_{j,t} = j$.angle
   if $t == 0$:
     $k_t = ((\theta_{1,t}, \theta_{2,t}, ..., \theta_{n,t}), (1, 1, ..., 1), 0)$
   else:
     $k_t = ((\theta_{1,t} - \theta_{1,t-1}, \theta_{2,t} - \theta_{2,t-1}, ..., \theta_{n,t} - \theta_{n,t-1}), (1, 1, ..., 1), 1)$
```

Fig. 31 Skill generation and optimization of the overall process

5.2 Strategy Design

In Section 4 of this article, the strategy design for RoboCup3D is divided into basic strategies and tactical strategies. A complete strategy should include the localization and role assignment in the basic strategies and also explores special strategies such as kick-off design and passing design in the tactical strategies.

Accurate localization is necessary to ensure the correct execution of basic actions and subsequent strategies. Localization mainly involves locating the robots (both own and opponent) and the ball. Due to the communication data's inherent error in the server, the research [22] uses particle filters and Kalman filters to locate the robots and ball's position. Moreover, the research [69] adds the field line information into the particle filter and solves the issue of robots losing themselves by communication between players when the robots walk out of the field and cannot obtain server information.

The role assignment in RoboCup3D differs from the role assignment in real human matches. In RoboCup3D, the same role may be assigned to different robots. In real matches, one role usually corresponds to one player because robots may be ejected from the field in case of violations, and the role must be reassigned, especially when the role carries significant tasks. Therefore, the role assignment in RoboCup3D is a real-time process, and it needs to continuously assign roles to eleven robots. The research [22] uses dynamic programming to minimize the total distance of all robots' movements and corrects the assignment by player voting to achieve the initial role assignment. Although this method minimizes the movement distance, it does not consider the collision problem when the robots move after role assignment. The research [184] optimizes the role assignment plan by using the Hungarian algorithm and avoids robot collisions. This process is called SCRAM (Scalable Collision-avoiding Role Assignment with Minimal-makespan). The research [141] adds the consideration of role priority based on SCRAM, giving priority to higher priority roles and assigning robots first, depending on the relationship between the role, the ball, and the opponent robot.

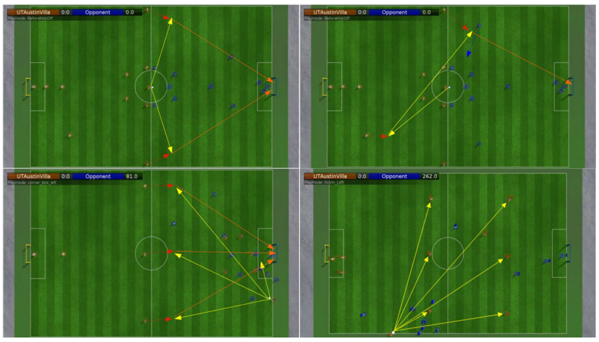

UTAustinVilla 0:0 Opponent 0.0
Playmode: BeforeKickOff
UTAustinVilla 0:0 Opponent 0.0
Playmode: BeforeKickOff
UTAustinVilla 0:0 Opponent 81.0
Playmode: corner_kick_left
UTAustinVilla 0:0 Opponent 262.0
Playmode: KickIn_Left

Fig. 32 The design of tactics and strategies for different match states[187]

In addition to the basic strategies, UT also designs many tactical strategies. The research [185] specifically designs a kick-off strategy by using different kicking skills with various distances to achieve different landing points and using different kick-off strategies. Moreover, considering that interference from the opponent can cause a failure of the kick, the logistic regression is used to classify whether to kick, further improving the success rate of the kick. The research [186] uses reinforcement learning to learn how to pass, using the data collected from real matches to train a network to evaluate the passing quality and determine how to pass based on the network output. In RoboCup3D, in addition to the overall game state, there are also various game states such as corner kicks, free kicks, and sideline kicks. To ensure effective cooperation of the robot team in each game state, The research [185, 187] designs tactical strategies for each game state, including each player's position, kick-off strategy, and passing strategy, etc. (see Fig. 32).

5.3 RoboCup3D Competition Results in Recent Years

Based on available data, this article compiles the results of RoboCup3D World Cup from 2016 to 2021 in Table 4 . The World Cup was not held in 2020 due to the pandemic.

Table 4 The results of the RoboCup3D World Cup in recent years

Year	1st Place	2nd Place	3rd Place
2011	UT Austin Villa	CIT3D	Apollo 3D
2012	UT Austin Villa	RoboCanes	BoldHearts
2013	Apollo 3D	UT Austin Villa	FC Portugal
2014	UT Austin Villa	RoboCanes	MagmaOffenburg
2015	UT Austin Villa	FUT-K	FC Portugal
2016	UT Austin Villa	FUT-K	FC Portugal
2017	UT Austin Villa	MagmaOffenburg	FUT-K
2018	UT Austin Villa	MagmaOffenburg	FC Portugal
2019	UT Austin Villa	MagmaOffenburg	Wright Ocean
2021	UT Austin Villa	MagmaOffenburg	Apollo3D
2022	FC Portugal	MagmaOffenburg	UT Austin Villa

By analyzing the results, it can be observed that the top spots in the RoboCup3D competition are consistently taken by a few fixed teams. On the one hand, this is because these teams have participated for many years and have accumulated rich experience. On the other hand, it is also due to the fact that related research on RoboCup3D is relatively scattered and the use of the simulation environment is difficult, which poses certain difficulties for new researchers. Therefore, this article summarizes the relevant research on RoboCup3D and provides an overview of this research field.

6 Future Research Directions

This section discusses and looks forward to the future development trends of research related to the RoboCup3D simulation environment, including the generation and optimization of more skills that reference specific human football movements, as well as the design of intelligent strategies.

6.1 Generation and Optimization of Specific Skills

The development potential of humanoid robot skills in the RoboCup3D simulation environment goes far beyond basic skills such as walking, kicking, and getting up. The combination of skills and the development of more new skills is the future trend of research on robot skill generation and optimization in the RoboCup3D simulation environment. Although basic walking, kicking, and getting up skills are already relatively mature and have mature generation and optimization methods, as mentioned in section 3.1.6, optimizing a single skill alone does not necessarily facilitate the execution of continuous skills. Therefore, it is necessary to explore how to optimize the continuous behavior of different skills to ensure that different skills can still achieve the effects of individual skilsl when executed continuously. Currently, besides hierarchical optimization [69], reinforcement learning methods have also been tried on continuous skills such as walking and kicking [100, 101], but they still make too many assumptions and hypotheses. In the future, there is still a lot of exploration space for optimizing or directly generating the continuity of different skills. In addition, referring to human football games, football players require more diverse skills to be generated in the RoboCup3D simulation environment, such as tackling, side kicking, lobbing, and goalkeeper diving, etc. If these skills can be implemented in the RoboCup3D simulation environment, it will make the simulation football game more similar to human games and further advance towards the goal of 2050. Therefore, in future development trends, it is necessary to break the current fixed skill generation method and explore more imaginative robot football skills.

6.2 Intelligent Strategy Design

Due to the complexity of the RoboCup3D simulation environment, designing strategies for multiple robots is a very difficult task. It not only requires processing information such as the positions of robots on both sides, ball position, and game status in the environment but also requires real-time strategy design for 11 robots while considering external factors such as collisions and out-of-bounds penalties. Section 4 of this paper summarizes the current methods for overall strategy design in relevant research. These methods all have too much prior knowledge or artificially established rules, and the strategies of robots are not intelligent. The overall strategy can only be limited to pre-planned scenarios. In the face of new situations, robots cannot make better adjustments to their strategies. There is no learning process for the overall strategy design of robots. It is more like an artificially defined process. Therefore, developing intelligent strategies is an important trend for future research.

In recent years, multi-agent deep reinforcement learning algorithms have achieved good results in multi-agent strategy design research, especially in the gaming field. Intelligent agents trained by large-scale deep reinforcement learning algorithms have surpassed human players. Multi-agent deep reinforcement learning algorithms train multiple intelligent agents without prior knowledge. They only need to input the current field state to output the agent's strategy (i.e., the skill to be executed). Through the training of thousands of games, the intelligent agents have learned real strategies that can adapt to different situations. This method has achieved good results in cooperative and competitive games, and it has the potential to be applied to the RoboCup3D simulation environment in the future.

7 Conclusion

Research on several issues in the RoboCup3D simulation environment has always been a popular research area, and the Robocup World Cup 3D simulation project based on this has also attracted teams from various countries to participate. From the information currently available to the author, this article is the first overview article based on this research area. The article first introduces the relevant information about RoboCup3D, and then based on whether there is a model, it provides an overview and analysis of robot motion generation and optimization, and also classifies and summarizes the existing strategies based on their respective implementation goals.

Although a lot of research has been conducted on RoboCup3D, various methods still have some shortcomings, especially in dealing with sudden problems during matches, and there is still a lot of room for development. There are often cases where one side surpasses the other in terms of motion optimization and strategy writing, but loses due to inadequate adaptability to sudden changes. Traditional motion models are difficult to handle such situations, but with the development of machine learning technologies, further breakthroughs can be sought in areas such as neural networks and reinforcement learning. From the literature summarized in this article, related articles have also grown rapidly in recent years. These methods can bring further development to RoboCup3D research, and at the same time, they may also obtain new breakthroughs and be applied to other fields from this simulation platform.

Declarations

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Li X. Research and Implementation of RoboCup3D Soccer Humanoid Robot Gait Optimization[D]. Hefei University Of Technology, 2019.

[2] C. Hong, I. Jeong, L. F. Vecchietti, D. Har and J. -H. Kim, "AI World Cup: Robot-Soccer-Based Competitions," in IEEE Transactions on Games, vol. 13, no. 4, pp. 330-341, Dec. 2021, doi: 10.1109/TG.2021.3065410.

[3] Depinet M, MacAlpine P, Stone P. Keyframe sampling, optimization, and behavior integration: Towards long-distance kicking in the robocup 3d simulation league[C]//Robot Soccer World Cup. Springer, Cham, 2014: 571-582.

[4] Hanna J P, Desai S, Karnan H, et al. Grounded action transformation for sim-to-real reinforcement learning[J]. Machine Learning, 2021, 110(9): 2469-2499.

[5] Kajita S, Kanehiro F, Kaneko K, et al. Biped walking pattern generation by using preview control of zero-moment point[C]//2003 IEEE international conference on robotics and automation (Cat. No. 03CH37422). IEEE, 2003, 2: 1620-1626.

[6] Vukobratovic, Miomir & Borovac, Branislav. (2004). Zero-Moment Point - Thirty Five Years of its Life.. I. J. Humanoid Robotics. 1. 157-173. 10.1142/S0219843604000083.

[7] Hirai K, Hirose M, Haikawa Y, et al. The development of Honda humanoid robot[C]//Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146). IEEE, 1998, 2: 1321-1326.

[8] Shuji Kajita, Guan Y. Humanoid Robots[M]. Tsinghua University publishing house co., ltd, 2007.

[9] S. Wang, M. Hu, H. Shi, S. Zhang, X. Li and W. Li, "Humanoid robot's omni-directional walking," 2015 IEEE International Conference on Information and Automation, 2015, pp. 381-385, doi: 10.1109/ICInfA.2015.7279317.

[10] F. Muniz, M. R. O. A. Maximo and C. H. C. Ribeiro, ”Keyframe Movement Optimization for Simulated Humanoid Robot Using a Parallel Optimization Framework,” 2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), 2016, pp. 79-84, doi: 10.1109/LARS-SBR.2016.20.

[11] Nezami O M, Meybodi M R. Biped Robot Walking using a Combination of Truncated Fourier Series and GALA (Genetic Algorithm parameters adaption using Learning Automata)[J]. International Journal of Machine Learning and Computing, 2012, 2(5): 598.

[12] S. Haider, S. R. Abidi and M. Williams, ”On evolving a dynamic bipedal walk using Partial Fourier Series,” 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2012, pp. 8-13, doi: 10.1109/ROBIO.2012.6490935.

[13] Shafii N, Javadi M H S, Kimiaghalam B. A truncated fourier series with genetic algorithm for the control of biped locomotion[C]//2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. IEEE, 2009: 1781-1785.

[14] Shafii N, Aslani S, Nezami O M, et al. Evolution of biped walking using truncated fourier series and particle swarm optimization[C]//Robot Soccer World Cup. Springer, Berlin, Heidelberg, 2009: 344-354.

[15] Braun H. Analyse domnenseitiger Optimierungen für Deep Reinforcement Learning in der RoboCup Umgebung[D]. Hochschule Offenburg, 2021.

[16] Tao C, Xue J, Zhang Z, et al. Parallel Deep Reinforcement Learning Method for Gait Control of Biped Robot[J]. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, 2022, 69(6).

[17] Spitznagel M. Analyse des Deep Reinforcement Learning Algorithmus PPO2 in der RoboCup Umgebung[J]. 2020.

[18] Liu J. The Motion Planning and Cooperation Mechanism of Biped Robots in RoboCup3D Simulation Environment[D]. Nanjing University Of Posts And Telecommunications, 2015.

[19] A. M. Bavani, H. Ahmadi and H. R. Nasrinpour, ”A closed-loop Central Pattern Generator approach to control NAO humanoid robots’ walking,” The 2nd International Conference on Control, Instrumentation and Automation, 2011,pp.1036-1041,doi:10.1109/ICCIAutom.2011.6356804.

[20] Holland, John H. “Genetic Algorithms.” Scientific American, vol. 267, no. 1, 1992, pp. 66–73.

[21] Hansen N. The CMA evolution strategy: a comparing review[J]. Towards a new evolutionary computation, 2006: 75-102.

[22] MacAlpine P, Urieli D, Barrett S, et al. UT Austin Villa 2011: 3D Simulation Team Report[R]. University of Texas at Austin Austin United States, 2011.

[23] MacAlpine P M. Multilayered skill learning and movement coordination for autonomous robotic agents[D], 2017.

[24] Hemami H. Reduced order models for biped locomotion[C]. IEEE Trans.Systems Man Cybernetics,1978,8(4): 321-351.

[25] Seekircher, A., Visser, U. An Adaptive LIPM-based Dynamic Walk using Model Parameter Optimization on Humanoid Robots. Künstl Intell 30, 233-244 (2016).

[26] Shafii N, Lau N, Reis L P. Learning to walk fast: Optimized hip height movement for simulated and real humanoid robots[J]. Journal of Intelligent & Robotic Systems, 2015, 80(3): 555-571.

[27] Sui Z, Yu W, Tian Y, Xu M. Gait Planning of Biped Robot Based on Reference Trajectory and COM Balance[J]. Journal of Jilin University(Information Science Edition),2017,35(2):175-182.

[28] Urieli D,MacAlpine P,Kalyanakrishnan S,Bentor Y,Stone P.On optimizing interdependent skills:A case study in simulated 3D humanoid robot soccer[C] International Conference on Autonomous Agents and Multiagent Systems, 20 11:769.776.

[29] Shen P. Omni-directional Walking Skill and Cooperation Mechanism of Biped Robots in RoboCup3D Simulation Environment[J]. 2015.

[30] Liang Z , Zhao H , Yue H . An omnidirectional walk for a biped robot based on gyroscope-accelerometer measurement[C]. IEEE International Conference on Mechatronics & Automation. IEEE, 2014.

[31] V. Hugel and N. Jouandeau, ”Walking patterns for real time path planning simulation of humanoids,” 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, 2012, pp. 424-430, doi: 10.1109/ROMAN.2012.6343789.

[32] Xu J. Gait Research and Implementation of Soccer Humanoid Robot Based on CMA-ES Algorithm[D]. Hefei University Of Technology, 2014.

[33] LI C. Research on Motion Planning and Flocking Control for Humanoid Robot[D]. Jiangnan University,2015.

[34] Snafii, N., Abdolmaleki, A., Lau, N., & Reis, L. P. (2015). Development of an Omnidirectional Walk Engine for Soccer Humanoid Robots. International Journal of Advanced Robotic Systems.

[35] Marko B Popovic, Ambarish Goswami, and Hugh Herr. Ground reference points in legged locomotion: Definitions, biological trajectories and control implications.

The International Journal of Robotics Research, 24(12):1013–1032, 2005.

[36] Sato T , Sakaino S , Ohnishi K . Real-Time Walking Trajectory Generation Method With Three-Mass Models at Constant Body Height for Three-Dimensional Biped Robots[J]. IEEE Transactions on Industrial Electronics, 2011, 58(2):376-383.

[37] LI C. Shooting method for humanoid robot based on three-mass model[J]. Journal of Computer Applications, 2014, 34(6): 1657.

[38] L. Yang, C. M. Chew and A. N. Poo, "Adjustable bipedal gait generation using Genetic algorithm optimized Fourier Series formulation," in Proc. IEEE/RSJ. International Conference on Intelligent Robots and Systems, 2006, PP. 4435-4440.

[39] Huang C L. Research of Gait Planning for Biped Robot[D]. Guangdong University Of Technology,2011. DOI:10.7666/d.y1941788.

[40] Feng H. The Optimization of Skills and Cooperation with Machine Learning in RoboCup3D[D]. Nanjing University Of Posts And Telecommunications, 2017.

[41] Hecheng Z, Zhiwei L, Qingyuan W. Long range kick for RoboCup3D—A practical approach[C]//2015 34th Chinese Control Conference (CCC). IEEE, 2015

[42] MacAlpine P, Urieli D, Barrett S, et al. UT Austin Villa 2011: a champion agent in the RoboCup 3D soccer simulation competition[C]//AAMAS. 2012: 129-136.

[43] Shi, Haobin Li, Xuesi Chen, Huahui Wang, Shixiong. (2016). Adaptive Omnidirectional Walking Method with Fuzzy Interpolation for Biped Robots. International Journal of Networked and Distributed Computing.

[44] H. Shi, X. Li, W. Liang, M. Dang, H. Chen and S. Wang, "A novel fuzzy omni-directional gait planning algorithm for biped robot," 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2016, pp. 71-76, doi: 10.1109/SNPD.2016.7515880.

[45] Depinet, M., MacAlpine, P., Stone, P. (2015). Keyframe Sampling, Optimization, and Behavior Integration: Towards Long-Distance Kicking in the RoboCup 3D Simulation League. In: Bianchi, R., Akin, H., Ramamoorthy, S., Sugiura, K. (eds) RoboCup 2014: Robot World Cup XVIII. RoboCup 2014. Lecture Notes in Computer Science(), vol 8992. Springer, Cham.

[46] Kasaei, M., Lau, N., Pereira, A. (2019). A Fast and Stable Omnidirectional Walking Engine for the Nao Humanoid Robot. In: Chalup, S., Niemueller, T., Suthakorn, J., Williams, MA. (eds) RoboCup 2019: Robot World Cup XXIII. RoboCup 2019. Lecture Notes in Computer Science(),vol.11531.Springer,Cham.

[47] Seekircher, A. (2015). Adaptive dynamic walking and motion optimization for humanoid robots (Order No. 3720027). Available from ProQuest Dissertations & Theses Global; ProQuest Dissertations & Theses Global A&I: The Sciences and Engineering Collection. (1718489352).

[48] Simoes M A C, Ramos C E, Argollo E, et al. Bahiart 2018: Team description paper for robocup 3d soccer simulation league[J]. RoboCup, 2017.

[49] MacAlpine, P., Stone, P. (2018). UT Austin Villa: RoboCup 2017 3D Simulation League Competition and Technical Challenges Champions. In: Akiyama, H., Obst, O., Sammut, C., Tonidandel, F. (eds) RoboCup 2017: Robot World Cup XXI. RoboCup 2017. Lecture Notes in Computer Science(), vol 11175. Springer, Cham.

[50] Farchy A. Learning in simulation for real robots[D], 2012.

[51] Li X, Liang Z, Feng H. Kicking motion planning of Nao robots based on CMA-ES[C]//The 27th Chinese Control and Decision Conference (2015 CCDC). IEEE, 2015: 6158-6161.

[52] Dorer, K. (2018). Learning to Use Toes in a Humanoid Robot. In: Akiyama, H., Obst, O., Sammut, C., Tonidandel, F. (eds) RoboCup 2017: Robot World Cup XXI. RoboCup 2017. Lecture Notes in Computer Science(), vol 11175. Springer,Cham.

[53] MacAlpine, P., Collins, N., Lopez-Mobilia, A., Stone, P. (2013). UT Austin Villa: RoboCup 2012 3D Simulation League Champion. In: Chen, X., Stone, P., Sucar, L.E., van der Zant, T. (eds) RoboCup 2012: Robot Soccer World Cup XVI. RoboCup 2012. Lecture Notes in Computer Science(), vol 7500. Springer, Berlin, Heidelberg.

[54] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. Evolutionary computation, 9(2):159–195, 2001.

[55] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cmaes). Evolutionary computation, 11(1):1–18, 2003.

[56] Nikolaus Hansen and Stefan Kern. Evaluating the cma evolution strategy on multimodal test functions. In International Conference on Parallel Problem Solving from Nature, pages 282–291. Springer, 2004.

[57] Stefan Kern, Sibylle D Müller, Nikolaus Hansen, Dirk Büche, Jiri Ocenasek, and Petros Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms–a comparative review. Natural Computing, 3(1):77–112, 2004.

[58] Nikolaus Hansen. The cma evolution strategy: a comparing review. In Towards a new evolutionary computation, pages 75–102. Springer, 2006.

[59] Abdolmaleki A, Simoes D, Lau N, et al. Learning a humanoid kick with controlled distance[C]//Robot World Cup. Springer, Cham, 2016: 45-57.

[60] Abdolmaleki A, Simoes D, Lau N, et al. Contextual direct policy search[J]. Journal of Intelligent & Robotic Systems, 2019, 96(2): 141-157.

[61] Y. Lu, Z. Liang, H. He, C. Xu, B. Yang and F. Fang, "3D Humanoid Robot Multi-gait Switching and Optimization," 2019 Chinese Control And Decision Conference (CCDC), 2019, pp. 4196-4201, doi: 10.1109/CCDC.2019.8832817.

[62] Jouandeau N, Hugel V. Simultaneous evolution of leg morphology and walking skills to build the best humanoid walker[C]//IEEE-RAS International Conference on Humanoid Robots 2013, 8th Workshop on Humanoid Soccer Robots. 2013.

[63] T. Uchitane and T. Hatanaka, "Applying evolution strategies for biped locomotion learning in RoboCup 3D Soccer Simulation," 2011 IEEE Congress of Evolutionary Computation (CEC), 2011, pp. 179-185, doi: 10.1109/CEC.2011.5949616.

[64] Uchitane T, Hatanaka T, Uosaki K. Evolution strategies for biped locomotion learning using nonlinear oscillators[C]//Proceedings of SICE Annual Conference 2010. IEEE, 2010: 1458-1461.

[65] Patrick MacAlpine, Elad Liebman, and Peter Stone. 2016. Adaptation of Surrogate Tasks for Bipedal Walk Optimization. In Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion (GECCO '16 Companion). Association for Computing Machinery, New York, NY, USA, 1275–1276.

[66] Urieli D, MacAlpine P, Kalyanakrishnan S, et al. Optimizing Interdependent Skills for Simulated 3D Humanoid Robot Soccer[C]//The Fifth Workshop on Humanoid Soccer Robots at Humanoids. 2010.

[67] Melo L C, Maximo M R O A, da Cunha A M. Bottom-up meta-policy search[J]. arXiv preprint arXiv:1910.10232, 2019.

[68] Zixuan Z, Yu Z, Jiawen W. Omnidirectional walk design of humanoid robots using layered learning method based on CMA-ES[C]//2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). IEEE, 2016: 464-468.

[69] MacAlpine P, Depinet M, Liang J, et al. UT Austin Villa: RoboCup 2014 3D simulation league competition and technical challenge champions[C]//Robot Soccer World Cup. Springer, Cham, 2014: 33-46.

[70] Urieli D, MacAlpine P, Kalyanakrishnan S, et al. On optimizing interdependent skills: a case study in simulated 3D humanoid robot soccer[C]//AAMAS. 2011, 11: 769.

[71] Baur M, Christmann K, Dorer K, et al. The magmaOffenburg 2018 RoboCup 3D simulation team[C]//RoboCup 2018 Symposium and Competitions: Team Description Papers. 2018.

[72] Tao C, Xue J, Zhang Z, et al. Gait Optimization Method for Humanoid Robots Based on Parallel Comprehensive Learning Particle Swarm Optimizer Algorithm[J]. Frontiers in Neurorobotics, 2021, 14: 600885

[73] F. Halataei and A. K. Kayhani, "Optimizing NAO humanoid walking using ABC algorithm," 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), 2015, pp. 1142-1144, doi: 10.1109/KBEI.2015.7436208.

[74] Jouandeau N , Hugel V . Optimization of parametrised kicking motion for humanoid soccer player[C]. IEEE International Conference on Autonomous Robot Systems & Competitions. IEEE, 2014.

[75] C. Cai and H. Jiang, "Performance Comparisons of Evolutionary Algorithms for Walking Gait Optimization," 2013 International Conference on Information Science and Cloud Computing Companion, 2013, pp. 129-134, doi: 10.1109/ISCC-C.2013.100.

[76] Rei J L M. Optimizing simulated humanoid robot skills[J]. 2010.

[77] Picado H, Gestal M, Lau N, et al. Automatic generation of biped walk behavior using genetic algorithms[C]//International Work-Conference on Artificial Neural Networks. Springer, Berlin, Heidelberg, 2009: 805-812.

[78] Cruz, L., Reis, L.P., Lau, N., Sousa, A. (2012). Optimization Approach for the Development of Humanoid Robots' Behaviors. In: Pavón, J., Duque-Méndez, N.D., Fuentes-Fernández, R. (eds) Advances in Artificial Intelligence – IBERAMIA 2012. IBERAMIA 2012. Lecture Notes in Computer Science(), vol 7637. Springer, Berlin,Heidelberg.

[79] Lattarulo V, Dijk S G. Application of the "alliance algorithm" to energy constrained gait optimization[C]//Robot Soccer World Cup. Springer, Berlin, Heidelberg, 2011: 472-483.

[80] H. He, Z. Liang, Y. Lu, C. Xu, B. Yang and F. Fang, "Dynamic Kick Optimization Of Humanoid Robot Based on Options Framework," 2019 Chinese Control And Decision Conference (CCDC), 2019, pp. 5176-5181, doi: 10.1109/CCDC.2019.8833269.

[81] Sutton R S, Barto A G. Reinforcement Learning: An Introduction (2nd edition). MIT Press, 2018.

[82] Liang Xing-Xing, Feng Yang-He, Ma Yang, Cheng Guang-Quan, Huang Jin-Cai, Wang Qi, Zhou Yu-Zhen, Liu Zhong. Deep multi-agent reinforcement learning: a

survey. Acta Automatica Sinica, 2020, 46(12): 2537-2557

[83] LIU Nai-Jun, LU Tao, CAI Ying-Hao, WANG Shuo. A Review of Robot Manipulation Skills Learning Methods. ACTA AUTOMATICA SINICA, 2019, 45(3): 458-470

[84] Bellman R. On the theory of dynamic programming. Proceedings of the National Academy of Sciences of the United States of America, 1952, 38(8): 716-719

[85] Wang Q, Zhao X, Huang J C, Feng Y H, Liu Z, Su Z H, et al. Addressing complexities of machine learning in big data: Principles, trends and challenges from systematical perspectives. Preprints, 2017, DOI: 10.20944/preprints201710.0076.v1

[86] Mnih V, Kavukcuoglu K, Silver D, Rusu A A, Veness J,Bellemare M G, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529-533

[87] Zhao Dong-Bin, Shao Kun, Zhu Yuan-Heng, Li Dong, Chen Ya-Ran, Wang Hai-Tao, et al. Review of deep reinforcement learning and discussions on the development of computer Go. Control Theory and Applications, 2016, 33(6): 701-717

[88] Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, et al. Mastering the game of Go without human knowledge. Nature, 2017, 550(7676) : 354-359

[89] Graves A, Wayne G, Reynolds M, et al. Hybrid computing using a neural network with dynamic external memory[J]. Nature, 2016, 538(7626): 471-476.

[90] Zhang T Y, Huang M L, Zhao L. Learning structured representation for text classification via reinforcement learning. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence. New Orleans, LA, USA: AAAI Press, 2018. 6053-6060

[91] Su P H, Gasic M, Mrksic N, Rojas-Barahona L M, Ultes S, Vandyke D, et al. On-line active reward learning for policy optimisation in spoken dialogue systems. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Berlin, Germany: Association for Computational Linguistics, 2016.

[92] Zhou Zhi-Hua. AlphaGo special session: An introduction. Acta Automatica Sinica, 2016, 42(5): 670

[93] Silver D , Huang A , Maddison C J , et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature.

[94] Silver, D., Schrittwieser, J., Simonyan, K. et al. Mastering the game of Go without human knowledge. Nature 550, 354–359 (2017). https://doi.org/10.1038/nature24270

[95] M. Spitznagel, D. Weiler and K. Dorer, "Deep Reinforcement Multi-Directional Kick-Learning of a Simulated Robot with Toes," 2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2021, pp. 104-110, doi: 10.1109/ICARSC52212.2021.9429811.

[96] M. Abreu, N. Lau, A. Sousa and L. P. Reis, "Learning low level skills from scratch for humanoid robot soccer using deep reinforcement learning," 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2019, pp. 1-8, doi: 10.1109/ICARSC.2019.8733632.

[97] Abreu M, Reis L P, Lau N. Learning to run faster in a humanoid robot soccer environment through reinforcement learning[C]//Robot World Cup. Springer, Cham, 2019: 3-15.

[98] Melo L C, Melo D C, Maximo M R O A. Learning Humanoid Robot Running Motions with Symmetry Incentive through Proximal Policy Optimization[J]. Journal of Intelligent & Robotic Systems, 2021, 102(3): 1-15.

[99] L. Carvalho Melo and M. R. Omena Albuquerque Máximo, "Learning Humanoid Robot Running Skills through Proximal Policy Optimization," 2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education(WRE),2019,pp.37-42,doi:10.1109/LARS-SBR-WRE48964.2019.00015.

[100] H. Teixeira, T. Silva, M. Abreu and L. P. Reis, "Humanoid Robot Kick in Motion Ability for Playing Robotic Soccer," 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2020, pp. 34-39, doi: 10.1109/ICARSC49921.2020.9096073.

[101] Abreu M, Silva T, Teixeira H, et al. 6D localization and kicking for humanoid robotic soccer[J]. Journal of Intelligent & Robotic Systems, 2021, 102(2): 1-25.

[102] Kasaei M, Abreu M, Lau N, et al. Robust Biped Locomotion Using Deep Reinforcement Learning on Top of an Analytical Control Approach[J]. arXiv preprint arXiv:2104.10592, 2021.

[103] A. F. V. Muzio, M. R. O. A. Maximo and T. Yoneyama, "Deep Reinforcement Learning for Humanoid Robot Dribbling," 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), 2020,pp.1-6,doi:10.1109/LARS/SBR/WRE51543.2020.9307084.

[104] Muzio A F V, Maximo M R O A, Yoneyama T. Deep Reinforcement Learning for Humanoid Robot Behaviors[J]. Journal of Intelligent & Robotic Systems, 2022, 105(1): 1-16.

[105] Rezaeipanah A, Amiri P, Jafari S. Performing the Kick During Walking for RoboCup 3D Soccer Simulation League Using Reinforcement Learning Algorithm[J]. International Journal of Social Robotics, 2020: 1-18.

[106] J. Wang, Z. Liang, Z. Zhou and Y. Zhang, ”Kicking motion design of humanoid robots using gradual accumulation learning method based on Q-learning,” 2016 Chinese Control and Decision Conference (CCDC), 2016, pp. 5274-5279, doi: 10.1109/CCDC.2016.7531941.

[107] Ma J, CAMERON S A. Learning robust and energy-efficient biped walking patterns using QWalking[M]//Mobile Robotics: Solutions and Challenges. 2010: 599-606.

[108] D. C. Melo, M. R. O. A. Máximo and A. M. da Cunha, ”Push Recovery Strategies through Deep Reinforcement Learning,” 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), 2020,pp.1-6,doi:10.1109/LARS/SBR/WRE51543.2020.9306967.

[109] Melo D C, Maximo M R O A, da Cunha A M. Learning Push Recovery Behaviors for Humanoid Walking Using Deep Reinforcement Learning[J]. Journal of Intelligent & Robotic Systems, 2022, 106(1): 1-18.

[110] Liang Z, Shen P, Li X. Walking motion design of humanoid robots in RoboCup3D simulation platform[J]. International Journal of Modelling and Simulation, 2015, 35(1): 35-42.

[111] Liang Z, Zhu S. Walking Parameters Training Algorithm of Humanoid Robot Based on Reinforcement Learning[J]. Computer Engineering, 2012, 38(8): 13-15.

[112] Liang Z, Zhu S, Jin X. Walking parameters design of biped robots based on reinforcement learning[C]//Proceedings of the 30th Chinese Control Conference. IEEE, 2011: 4017-4022.

[113] B. S. Pavse, F. Torabi, J. Hanna, G. Warnell and P. Stone, ”RIDM: Reinforced Inverse Dynamics Modeling for Learning from a Single Observed Demonstration,” in IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 6262-6269, Oct. 2020, doi: 10.1109/LRA.2020.3010750.

[114] Li X. Study on the CMAC Based Coordinated Control of Multidrive Systems[D]. Hefei University Of Technology, 2010.

[115] Xu Y. Agent localization in RoboCup3D[J]. Proceedings of 2005 China Robot Competition, 2005.

[116] Chen S, Yi Y, Tan Y. RoboCup3D football localization method based on α-β-γ filtering[C]//Proceedings of the 2009 China Conference on Intelligent Automation

(Volume 2). 2009

[117] Shen L, Ye P, Ding X, et al. Research on key technologies of RoboCup 3D simulation[J]. Robot Technique and Application, 2010 (4): 54-56.

[118] Shi G. Implementation of Omni-directional Walking and High-Level Decision for Humanoid Robots in RoboCup3D Simulation System[D]. Hefei University Of Technology, 2010.

[119] Yin Z, Chen W H. Fantasia 2009 Team Description[J]. coordinates, 11(12): 13.

[120] Vorst P. Readylog agents for the robocup 3d soccer simulation league[J]. RWTH Aachen University Thesis, 2006.

[121] Zhao Y. Self localization under restricted vision in RoboCup3D[J]. Electronic Test, 2010 (4): 81-85.

[122] Sun Z, Huang H, Ren X. Application of an Enhanced Location System with the Virtual Multi-Sensor[C]//2009 WRI Global Congress on Intelligent Systems. IEEE, 2009, 1: 36-41.

[123] Wang X, Yan X, Zhang Y, et al. Kalman filter in the robocup 3d positioning[C]//2012 International Conference on Computer Science and Electronics Engineering. IEEE, 2012, 3: 47-52.

[124] Seekircher A, Abeyruwan S, Visser U. Accurate ball tracking with extended kalman filters as a prerequisite for a high-level behavior with reinforcement learning[C]//The 6th Workshop on Humanoid Soccer Robots at Humanoid Conference, Bled (Slovenia). 2011.

[125] Fox D, Burgard W, Dellaert F, et al. Monte carlo localization: Efficient position estimation for mobile robots[J]. AAAI/IAAI, 1999, 1999(343-349): 2-2.

[126] Bustamante Horta C F. Probabilistic Agent Localization and Fuzzy-Bayesian Pass Evaluation for the RoboCup Simulation 3D League-Edicion Unica[J].

[127] Hao Y, Liang Z, Liu J, et al. The framework design of humanoid robots in the robocup 3D soccer simulation competition[C]//2013 10Th IEEE International conference on control and automation (ICCA). IEEE, 2013: 1423-1428.

[128] Wang J R, Yu Y, Wei J G, et al. A Robot Simulation System For Self-location by using field visual information[C]//Advanced Materials Research. Trans Tech Publications Ltd, 2013, 748: 690-694.

[129] Fu H, Cao F. A Survey of Robot Self-localization and the Research in RoboCup3D[J]. Computer Knowledge and Technology: Academic Edition, 2016 (4): 172-174.

[130] Simoes M A C, Ramos C E, Argollo E, et al. Bahiart 2017: Team description paper for robocup 3d soccer simulation league[J]. RoboCup, 2017.

[131] Fernandes G C G, Dias S S, Maximo M R O A, et al. Cooperative Localization for Multiple Soccer Agents Using Factor Graphs and Sequential Monte Carlo[J]. IEEE Access, 2020, 8: 213168-213184.

[132] Lu W, Zhang J, Zhao X, et al. Multimodal sensory fusion for soccer robot self-localization based on long short-term memory recurrent neural network[J]. Journal of Ambient Intelligence and Humanized Computing, 2017, 8(6): 885-893.

[133] Chen T. Research and Implement of RoboCup 3D Simulation Robot[D]. Anhui University Of Technology, 2014.

[134] He K, Liang Z, Cui T, et al. Formation optimization of RoboCup3D soccer robots using delaunay triangulation network[C]//2018 Chinese Control And Decision Conference (CCDC). IEEE, 2018: 224-229.

[135] Akiyama H, Noda I. Multi-agent positioning mechanism in the dynamic environment[C]//Robot Soccer World Cup. Springer, Berlin, Heidelberg, 2007: 377-384.

[136] Larik A S, Haider S. On using evolutionary computation approach for strategy optimization in robot soccer[C]//2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI). IEEE, 2016: 11-16.

[137] MacAlpine P, Barrera F, Stone P. Positioning to win: A dynamic role assignment and formation positioning system[C]//Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence. 2012

[138] Chen W, Chen T. Multi-robot dynamic role assignment based on path cost[C]//2011 Chinese Control and Decision Conference (CCDC). IEEE, 2011: 3721-3724.

[139] Ulusoy O, Talay S S. Distributed Team Formation for Humanoid Robot Soccer[C]//ICAART (1). 2012: 605-613.

[140] Li L, Fang Y. Research on decision system model of Robocup3D robotics team[J]. Computer Engineering and Applications, 2015 (6): 37-41.

[141] MacAlpine P, Stone P. Prioritized role assignment for marking[C]//Robot World Cup. Springer, Cham, 2016: 306-318.

[142] Chen L, Qin S, Chen K, et al. Efficient Role Assignment with Priority in Robocup3D[C]//2020 Chinese Control And Decision Conference (CCDC). IEEE, 2020: 2697-2702.

[143] Abeyruwan S, Seekircher A, Visser U. Off-policy general value functions to represent dynamic role assignments in RoboCup 3D soccer simulation[J]. arXiv preprint arXiv:1402.4525, 2014.

[144] Abeyruwan S, Visser U. A New Real-Time Algorithm to Extend DL Assertional Formalism to Represent and Deduce Entities in Robotic Soccer[C]//Robot Soccer World Cup. Springer, Cham, 2014: 270-282.

[145] Chang E C, Choi S W, Kwon D Y, Park H and Yap C K. Shortest path amidst disc obstacles is computable.Proceedings of the Annual Symposium on Computational Geometry, 2005, pp.116-125.

[146] Zhen C. The Motion Planning and Intelligent Decision of Biped Robots in RoboCup3D Simulation Environment[D]. Nanjing University Of Posts And Telecommunications, 2013.

[147] Su L, Liang Z. Collision-avoidance planning of soccer robot in RoboCup3D simulation environment[C]//Proceedings of the 32nd Chinese Control Conference. IEEE, 2013: 5739-5743.

[148] Rayermann M, Cake T M. Cooperative Pathfinding in 3D Robot Soccer[J].

[149] Muzio A, Melo D, Henrique E, et al. Itandroids soccer3d team description paper 2016[J]. 2016.

[150] Li C, Tao C, Liu G, et al. Quantized flocking control for second-order multiple agents with obstacle avoidance[J]. Advances in Mechanical Engineering, 2016, 8(1): 1687814015624646.

[151] Yao Q. The Research of RoboCup 3D Simulation Robot[D]. Guangdong University Of Technology, 2012.

[152] Shen X, Liu G. Robocup vanguard's goal-scoring ability based on Q-learning[J]. Jisuanji Gongcheng yu Yingyong(Computer Engineering and Applications), 2011, 47(18): 53-55.

[153] Rettinger A, Zinkevich M, Bowling M. Boosting expert ensembles for rapid concept recall[C]//AAAI. 2006: 464-469.

[154] Stone P, Sutton R S. Scaling reinforcement learning toward RoboCup soccer[C]//Icml. 2001, 1: 537-544.

[155] Li X. Soccer Robots Local Passing and Tactics Cooperation[D]. Nanjing University Of Posts And Telecommunications, 2016.

[156] Xuanyu C, Zhiwei L, Yongyi Y, et al. Multi-robot collaboration based on Markov decision process in Robocup3D soccer simulation game[C]//The 27th Chinese Control and Decision Conference (2015 CCDC). IEEE, 2015: 4345-4349.

[157] Zhao Q, Liang Z, Fang F, et al. Local passing-ball tactics based on a Keepaway algorithm[C]//2017 29th Chinese Control And Decision Conference (CCDC). IEEE, 2017: 4884-4889.

[158] Gupta N, Kalyanakrishnan S. Learning complex behaviours and Keepaway in 3D Robocup Environment[J].

[159] Huang R, Xu Y, Tan Y. Research and Implementation of RoboCup Goalkeeper Action and Strategy[J]. China Science and Technology Information, 2008 (21): 36-37.

[160] Masterjohn J G, Polceanu M, Jarrett J, et al. Regression and mental models for decision making on robotic biped goalkeepers[C]//Robot Soccer World Cup. Springer, Cham, 2015: 177-189.

[161] Polceanu M. ORPHEUS: Reasoning and Prediction with Heterogeneous rEpresentations Using Simulation[D]. Université de Bretagne Occidentale (UBO), 2015.

[162] Wei S, Qin H. Application of RoboCup 3D and Intelligent Technology in Football Simulation League[J]. Computational intelligence and neuroscience, 2022, 2022.

[163] Yao Q. Research on Robocup3D Simulation Robot Interception Technology[J]. Shandong Industrial Technology, 2015 (11): 52-53.

[164] Zhu J. Prediction of the real-time interceptor algorithm for robot football match simulation[J]. Electronic Test, 2012 (2): 31-34.

[165] Mirmohammad Y, Khorsandi S, Shahsavari M N, et al. Ball Path Prediction for Humanoid Robots: Combination of k-NN Regression and Autoregression Methods[C]//Robot World Cup. Springer, Cham, 2021: 3-14.

[166] Abadi M M N, Lucas C. Evolving Artificial Neural Networks for Prediction in Robocup Soccer[J].

[167] Yang C, Chang X, Chen J, et al. Situation Assessment for Soccer Robots using Deep Neural Network[C]//2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC). IEEE, 2019: 1-4.

[168] Larik A S, Haider S. Rule-based behavior prediction of opponent agents using robocup 3D soccer simulation league logfiles[C]//IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, Berlin, Heidelberg, 2012: 285-295.

[169] Raza A, Sharif U, Haider S. On learning coordination among soccer agents[C]//2012 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, 2012: 699-703.

[170] Simoes M, Nogueira T. Towards setplays learning in a multiagent robotic soccer team[C]//2018 Latin American robotic symposium, 2018 Brazilian symposium on robotics (SBR) and 2018 Workshop on Robotics in Education (WRE). IEEE, 2018: 277-282.

[171] MacAlpine P, Genter K, Barrett S, et al. The RoboCup 2013 drop-in player challenges: Experiments in ad hoc teamwork[C]//2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014: 382-387.

[172] Larik A S. Opponent Modeling in RoboCup Soccer Simulation 3D[C]//German Conference on Multiagent System Technologies. Springer, Berlin, Heidelberg, 2013: 416-419.

[173] Larik A S, Haider S. Opponent classification in robot soccer[C]//International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer, Cham, 2015: 478-487.

[174] Holmes, G., Donkin, A., Witten, I.H.: WEKA: A Machine Learning Workbench. In: Proceedings of Second Australia and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia (1994)

[175] Eibe, F., Witten, I.H.: Generating Accurate Rule Sets without Global Optimization. In: Proceedings of the 15th International Conference on Machine Learning, San Francisco, USA (1998)

[176] Simoes M A C, Nobre J, Sousa G, et al. Strategy planner: enhancements to support better defense and pass strategies within an LfD approach[C]//2020 IEEE international conference on autonomous robot systems and competitions (ICARSC). IEEE, 2020: 46-52.

[177] de Sousa Pereira V. FCPortugal-Multi-Robot Action Learning[J]. 2020.

[178] da Silva R M, de Souza J R, Simoes M A C, et al. Framework for modeling autonomous multi-robots systems[C]//2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE). IEEE, 2018: 13-18.

[179] Simōes M A C, Mascarenhas G, Fonseca R, et al. BahiaRT Setplays Collecting Toolkit and BahiaRT Gym[J]. Software Impacts, 2022, 14: 100401.

[180] Miikkulainen R, Grauman K. Making Friends on the Fly: Advances in Ad Hoc Teamwork[J].

[181] MacAlpine P, Stone P. Using dynamic rewards to learn a fully holonomic bipedal walk[C]//Adaptive Learning Agents Workshop. 2012.

[182] Lopez-Mobilia A. Inverse Kinematics Kicking in the Humanoid RoboCup Simulation League[D]. Master's Thesis, University of Texas at Austin, 2012.

[183] MacAlpine P, Depinet M, Stone P. UT Austin Villa 2014: RoboCup 3D simulation league champion via overlapping layered learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2015, 29(1).

[184] MacAlpine P, Price E, Stone P. SCRAM: Scalable collision-avoiding role assignment with minimal-makespan for formational positioning[C]//Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

[185] MacAlpine P, Hanna J, Liang J, et al. UT Austin Villa: RoboCup 2015 3D simulation league competition and technical challenges champions[C]//Robot Soccer World Cup. Springer, Cham, 2015: 118-131.

[186] MacAlpine P, Torabi F, Pavse B, et al. UT Austin Villa: RoboCup 2018 3D simulation league champions[C]//Robot World Cup. Springer, Cham, 2018: 462-475.

[187] MacAlpine P, Stone P. UT Austin Villa: RoboCup 2016 3D simulation league competition and technical challenges champions[C]//Robot World Cup. Springer, Cham, 2016: 515-528.