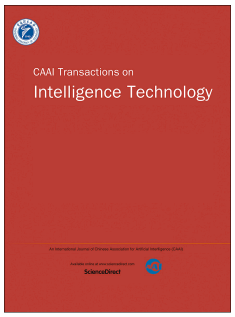


Trajectory planning for biped robot walking on uneven terrain—taking stepping as an example

Qiu-bo Zhong, Fei Chen

PII: S2468-2322(16)30066-X
DOI: 10.1016/j.trit.2016.10.009
Reference: TRIT 26

To appear in: *CAAI Transactions on Intelligence Technology*

Trajectory planning for biped robot walking on uneven terrain—taking stepping as an example

Qiu-bo ZHONG [1*], Fei CHEN[2]
*Corresponding Author mail: zhongqiubo@nbut.edu.cn

[1] School of Electronic and Information Engineering, Ningbo University of Technology, 315016, China;
[2] Advanced Robotics Department, Istituto Italiano di Tecnologia, Genova, Italy;

Abstract: According to the features of movements of humanoid robot, a control system for humanoid robot walking on uneven terrain is present. Constraints of stepping over stairs are analyzed and the trajectories of feet are calculated by intelligent computing methods. To overcome the shortcomings resulted from directly controlling the robot by neural network (NN) and fuzzy logic controller (FLC), a revised particle swarm optimization (PSO) algorithm is proposed to train the weights of NN and rules of FLC. Simulations and experiments on different control methods are achieved for a detailed comparison. The results show that using the proposed methods can obtain better control effect.

Key words: Humanoid robot, PSO, NN, FLC, motion planning

1. Introduction

In order to make humanoid robot able to “live” in the human being environment, it must be able to perform some complex motions. There are two methods to plan these complex motions: The first method is to first establish the appropriate model according to the type of movement, and then plan for the trajectory of motions. Usually, the trajectories of ankle and hip of the robot are assumed by parameter interpolation. Secondly, trajectories of other joints are deduced by geometric relationship. Finally, the optimal trajectories are confirmed by optimal algorithms according to the MAX of stability or MIN of energy consumed or similar strategies[1][2]. The second method is firstly to design an ideal trajectory of motion according to the stability or optimal targets. Then the movement of each joint will be calculated to match the ideal trajectory through inverse kinematics. Finally, during process of the real movement, the error between real trajectory and ideal trajectory will be adjusted[3].

It is worth to study of humanoid robot of integrating into society and serving in the home, and it is a very challenging subject. In face of all kinds of uncertain environment, for walking, robot might need to complete the task in various of complex environment regularly, as shown in Fig.1.

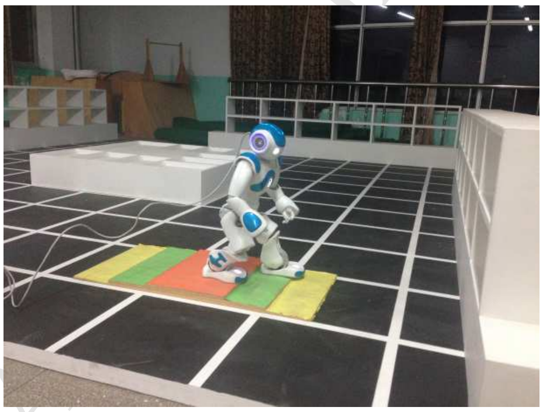


Fig. 1 Robot walks in complex environment

Walking on uneven terrain for humanoid robot is one of the complex motions. It involves in planning and controlling more than ten degrees of freedom of the robot, and this system is a high-dimensional nonlinear complex system. If traditional methods are used in this system, the optimized control effect is not very satisfactory. And especially when there are many more uncertain initial variables, it is easy to fall into local optimization, slow optimization and poor adaptive ability of the algorithm. Therefore, because almost all soft computing algorithms are highly versatile, and analyticity of objective function is not required, evolutionary algorithms and other soft computing techniques can be considered in order to control this system. Rahul used GA with fuzzy logic method to control biped robot walking, and obtain good results[4]. Based on soft computing Pandu designed a gait planner that keep the robot stable when it walked on slop[2].

PSO algorithm has proven to be very effective in solving complex optimization problems. However, as with other evolutionary algorithms, with PSO algorithm it is also easy to fall into local optimal solution. Maintaining the diversity of population and avoiding prematurely falling into local optimal solution is a direct way to improve the PSO. One way to improve the evolutionary algorithm is an idea based on multi-population[5], and it has achieved very good results solving practical problems[6][7]. In order to control the multi-robots formation, Seung-Mok used a coevolving particle swarm optimization(CCPSO) algorithm to optimize the model predictive control, and obtained good results[8]. Micael S.

Couceiro introduced an extension of PSO optimal method called RDPSO to control with 15 robots. RDPSO can benefit from the dynamical partitioning of the whole population of robots into multiple groups[9]. To avoid multi-robots obstacles, Ezequiel introduced a noise-resistant PSO algorithm to optimize the resource constraints of multi-robots[10]. Vojtech Vonasek utilized PSO technique to find the CPG-based motion primitives, and then planed the global motion for robots using RRT algorithm [11].

Artificial neural network (ANN) has a strong approach to solve the problem of nonlinear mapping. It can be introduced to control the robot arm, motion optimization and many problems of robot control. Tao Li used a new recurrent neural network to control the online robot arm to learn the behavior. Because the training time is too long, only weights of output were trained[12]. Pedro introduced NN to recognize the gestures and to control the moves of the industrial robot. He gathered the data from data gloves and used two ANN to recognize different kinds of gestures. The rate of recognition is up to 95%[13]. However, how to reduce the training time for NN is a good question to discuss.

FLC is one of the soft computing techniques. It uses the controlling experience from human experts to make up the adverse effects brought from nonlinear and uncertain factors in dynamic characteristics of robot. However, it also has shortcomings. Its control rules and membership functions cannot be modified once established. Park[14] designed a trajectory generator based on ZMP FLC. The trajectories of legs were the input, and the effectiveness of the algorithm was proved by simulation. The ZMP trajectory produced by this algorithm can increase the stability of robot. However, the main defect of this algorithm is that the trajectory cannot be the optimum because of the lack of optimizer. Zhou[15] presented a fuzzy reinforcement learning structure, which can maintain the dynamic stability of biped robot while moving. However, this algorithm used a gradient descent learning method, with which is easy to fall into local optimum. One way to jump the local optimum is to use the intelligence optimal algorithm. PSO algorithm is less dependent on the objective function and it can be introduced to optimize the rules of FLC.

In this paper, constraints of movement for humanoid robot are analyzed at first. Then, an example of stepping over stairs is taken to feasibility analysis. And the kinematics formulas of stepping over stairs are achieved. Based on the PSO algorithm present in [5], an idea of crossover from GA is introduced to make the algorithm jump the local optimum, which is called MPSO. In order to test the effects of different control methods, three methods of soft computing consisting of MPSO, MPSO optimizing NN(MPSONN), MPSO optimizing FLC(MPSOFLC) of are used to control the motion of walking on uneven terrain for humanoid robot. Finally, simulations of different methods are achieved.

2. External non-collision constraint for humanoid robot

External non-collision constraint for humanoid robot can be divided into two cases, one is collision between legs (mainly considering the lower body of humanoid robot) and external part, and the other is between feet and floor.

Constraint for the first case can be described by 2D geometric constraints in sagittal plane[16]. According to a fundamental theorem in computational geometry, given three points, p_1, p_2, p_3, the point p_3 is on the left(right) of the directed line segment $\overrightarrow{p_1 p_2}$ if and only if the area composed of these three points $A_{p_1 p_2 p_3} > 0 (< 0)$. The point p_3 is on the segment $\overrightarrow{p_1 p_2}$ if and only if $A_{p_1 p_2 p_3} = 0$. From the figure 2, $A_{p_1 p_2 p_3}$ and be easily computed by the coordinate values of p_1, p_2, p_3. The formula of computing is shown as bellow.

$$A_{p_1 p_2 p_3} = \frac{1}{2}\left[x_1 (y_2 - y_3) + x_2 (y_3 - y_1) + x_3 (y_1 - y_2) \right]$$

(1)

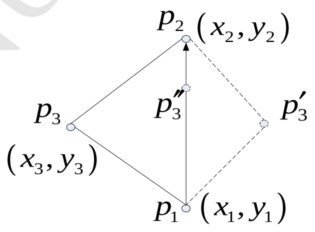


Fig. 2 Distance of lines

When humanoid robot walks on uneven terrain, if its foot trajectory is not handled properly, external obstacles collision happen very easily, and so the robot will lose stability. We will take robot stepping across obstacles within single supporting period as example to analyze the constraints. As shown in figure 3, the supposed the starting point of foot is on the point $s(s_x, 0)$, the final location is on the point $e(e_x, 0)$, the highest point of obstacle is on $c(c_x, h)$, and the height is h. f is defined as a fourth-order polynomial function. Let f contain those three points above, and $f'(s) = f'(c) = f'(e) = 0$. After a simple mathematical iteration, the expression of $f(s_x, e_x, c_z) = 0$ can be achieved. During the movement of robot, if the heel (h_x, h_z) and toe (t_x, t_z) are always over the movement trajectory, avoidance will succeed, which $h_z \geq f(h_x)$ and $t_z \geq f(t_x)$.

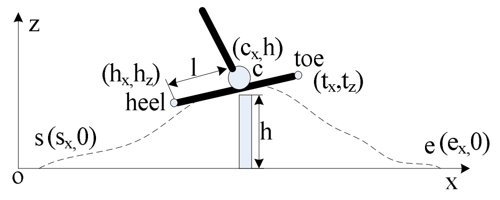


Fig.3 Model of foot for stepping over obstacle

3. Motion planning on uneven terrain

Uneven terrain can be assumed to be convex part and concave part in 2D. For humanoid robot, crossing convex part and concave part can be analyzed as stepping on and off stairs respectively. The model of movement is shown in figure 4.

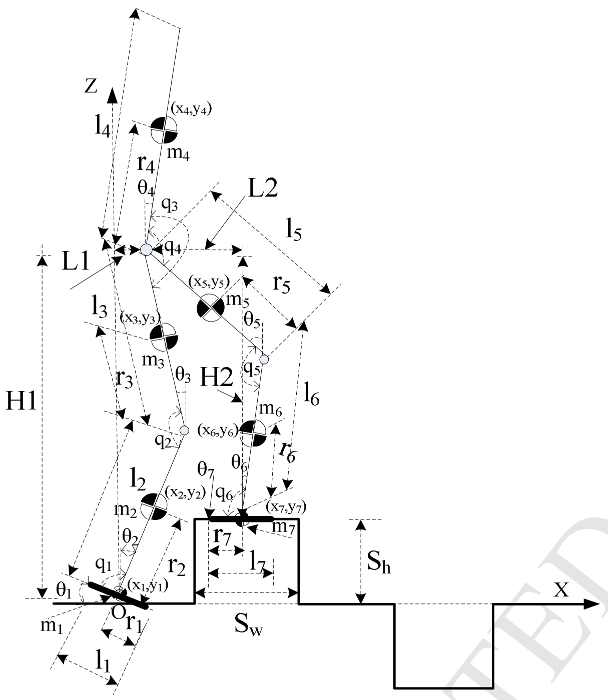


Fig.4 Parameters of model of going uneven terrain for humanoid robot

3.1 Geometric constraints

According to the stability constraints in [17], constraints of stepping off stairs for humanoid robot are shown as in figure 5.

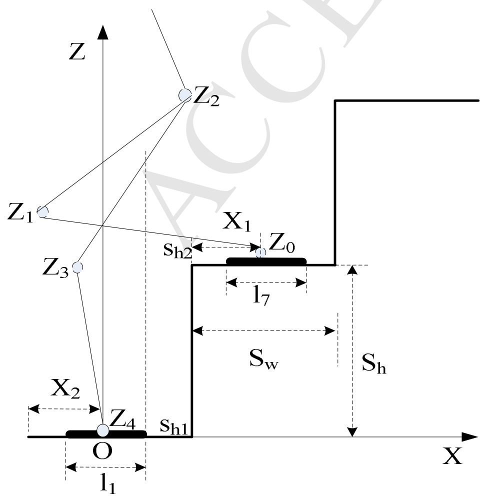


Fig.5 Feasible model of going downstairs for humanoid robot

From formula (1), combined with geometric model of humanoid robot in stepping over stairs, constraints during the double supporting period can be obtained. The details of constraints are shown in formula (2)

$$\begin{cases} \max\left(A_{z_0 z_3 s_{h_1}} A_{z_0 z_3 s_{h_2}}, A_{z_0 s_{h_2} s_{h_1}} A_{s_{h_1} z_3 s_{h_2}}\right) > 0 \\ \max\left(A_{z_3 z_4 s_{h_1}} A_{z_3 z_4 s_{h_2}}, A_{z_3 s_{h_2} s_{h_1}} A_{s_{h_1} z_4 s_{h_2}}\right) > 0 \\ A_{z_0 z_3 z_4} > 0 \\ -\frac{l_1}{2} \le x_{com} \le S_w - X_1 + X_2 + \frac{l_7}{2} \end{cases} \tag{2}$$

Where, S_w and S_h are the width and height of the stairs respectively. $z_i\left(i=0,1,2,3,4\right)$ represent the joints in legs of robot. l_1, l_7 are the width of left and right legs respectively. X_1, X_2 is distance between the center of foot and the edge of stairs. x_{com} represents the abscissa of center of gravity. The processing of stepping on stairs is similar to this motion, the details can be found in[17].

3.2 Stability constraints

Regarding the stability constraints for a humanoid robot, we supposed the friction between the robot foot and stairs is large enough, and ensures that the robot does not slide during the movement. Therefore, the stability constraints can be judged by calculating the ZMP of robot.

3.3 Kinematic analysis on uneven terrain

The period of stepping over stairs T is similar with walking on flat terrain, which can be divided into double support period T_d and single support period T_s. In a case study of sagittal plane motion it is supposed that the initial location of robot is the following: its right leg is on the stairs, and the left leg is under the right one. The distance between the two legs is one step. After one period T of walking, the robot reached the final location, where the left leg is on the right one, and the distance between the two legs is also one step. The movement is shown in figure 3. The details of parameters of the robot are present above.

The single support period of the robot can be similar to the multi-link inverted pendulum motion. Supposing the link mass of robot is even, and each COM coordinates $\left(x_i, z_i\right)\left(i=1,2...,7\right)$ leads to the formula (3)

$$\begin{aligned}
&x_1 = x\\
&z_1 = z_f\\
&x_2 = r_2 \times \sin(\theta_2) + x\\
&z_2 = r_2 \times \cos(\theta_2) + z_f\\
&x_3 = -r_3 \times \sin(\theta_3) + l_2 \times \sin(\theta_2) + x\\
&z_3 = r_3 \times \cos(\theta_3) + l_2 \times \cos(\theta_2) + z_f\\
&x_4 = r_4 \times \sin(\theta_4) - l_3 \times \sin(\theta_3) + l_2 \times \sin(\theta_2) + x\\
&z_4 = r_4 \times \cos(\theta_4) + l_3 \times \cos(\theta_3) + l_2 \times \cos(\theta_2) + z_f\\
&x_5 = (l_5 - r_5) \times \sin(\theta_5) - l_3 \times \sin(\theta_3) + l_2 \times \sin(\theta_2) + x\\
&z_5 = -(l_5 - r_5) \times \cos(\theta_5) + l_3 \times \cos(\theta_3) + l_2 \times \cos(\theta_2) + z_f\\
&x_6 = (l_6 - r_6) \times \sin(\theta_6) + l_5 \times \sin(\theta_5) - l_3 \times \sin(\theta_3) + l_2 \times \sin(\theta_2) + x\\
&z_6 = -(l_6 - r_6) \times \cos(\theta_6) + l_5 \times \cos(\theta_5) + l_3 \times \cos(\theta_3) + l_2 \times \cos(\theta_2) + z_f\\
&x_7 = S_w - X_1 + X_2 - l_7 / 2\\
&z_7 = S_h
\end{aligned} \tag{3}$$

Where, r_i represents the distance between joint and COM, l_i is the length of link, x and z_f are the trajectories of ankles[17]. The trajectory of ZMP of robot during the processing of stepping on stairs is shown in formula (4)

$$x_{zmp} = \frac{\sum_{i=1}^{7} \left(m_i (\ddot{z}_i + g) x_i - m_i \ddot{x}_i z_i - I_{ix} \ddot{\theta}_{ix} \right)}{\sum_{i=1}^{7} m_i (\ddot{z}_i + g)} \tag{4}$$

Where, m_i is the mass of each link, I_{ix} represents moment of inertia of each link, $\ddot{\theta}_{ix}$ is joint rotation angular acceleration. $(i = 1, 2, \ldots 7)$

4. Hybrid Particle Swarm Optimization (MPSO)

In normal PSO algorithm, updating the location only involves $x_{i,j}(t+1)$ and $x_{i,j}(t)$, and actually, the middle points $x_1(t+1)$ and $x_2(t+1)$ have been counted more than once. However, these two points may be better than point $x_i(t+1)$, therefore, single-step sequential addition in normal PSO is updated into distributed computing. The middle points can be obtained as shown in formula (5) and (6).

$$x_1(t+1) = x_i(t) + v_i(t+1) \tag{5}$$

$$x_2(t+1) = x_1(t+1) + c_1 \cdot r_1 \cdot \left(pbest_i(t) - x_i(t) \right) \tag{6}$$

The next step is to calculate the function values of the two middle location, and compare them with the objective function $f(x_i(t+1))$ .The smaller one is used to update $x_i(t+1)$ and the detail of algorithm is shown as bellow[5].

$$v_1(t+1) = v_i(t);\ x_1(t+1) = x_i(t) + v_1(t+1) \tag{7}$$

$$\begin{aligned}
&v_2(t+1) = v_1(t+1) + c_1 \cdot r_1 \left(pbest_i(t) - x_i(t) \right);\\
&x_2(t+1) = x_1(t+1) + v_2(t+1)\\
&v_3(t+1) = v_2(t+1) + c_2 \cdot r_2 \left(gbest_i(t) - x_i(t) \right);\\
&x_3(t+1) = x_2(t+1) + v_3(t+1)
\end{aligned} \tag{8}$$

$$x_i(t+1) = \begin{cases} x_1(t+1)\ if\ f(x_1(t+1)) \le f(x_2(t+1)) \\ and\ f(x_1(t+1)) \le f(x_3(t+1)) \\ x_2(t+1)\ if\ f(x_2(t+1)) \le f(x_1(t+1)) \\ and\ f(x_2(t+1)) \le f(x_3(t+1)) \\ x_3(t+1) \quad others \end{cases} \tag{9}$$

This algorithm divided the single-step updating formula of particle velocity in normal PSO into three steps, and selects the best one of three position vectors. It can refine the process of searching particle trajectories and increase the update rate of individual extreme.

This algorithm improves the performance of the algorithm based on normal PSO without increasing the complexity. However, this method has defect on the avoiding local optima. To solve this problem, the concept of crossover in genetic algorithm is introduced. In each iteration, a specified number of crossover particles are selected to put into the hybrid pool according to the crossover probability. The particles in hybrid pool will crossover randomly to produce the same number of child particles($child$) and update the parent particles($parent$) using child particles. The position of child can be obtained by cross computing the positions of parents. The details are shown in formula (11).

$$child(x) = p \cdot parent_1(x) + (1 - p) \cdot parent_2(x) \tag{11}$$

Where, p is a random number between 0 and 1. The velocity of child can be reached by formula (12)

$$child(v) = \frac{parent_1(v) + parent_2(v)}{\left| parent_1(v) + parent_2(v) \right|} \left| parent_1(v) \right| \tag{12}$$

Meanwhile, for balancing the global search ability and local improvement capacity in PSO algorithm, a non-linear dynamic inertia weight coefficient formula is proposed. The expression of the formula is shown as bellow.

$$w = \begin{cases} w_{\min} - \dfrac{(w_{\max} - w_{\min}) * (f - f_{\min})}{(f_{avg} - f_{\min})}, & f \le f_{avg} \\ w_{\max} & , f \ge f_{avg} \end{cases} \tag{13}$$

Where, $w_{\max}$ and $w_{\min}$ represent the MAX and MIN of w respectively. f is the current value of objective function. f_{avg} and $f_{\min}$ are the current average target and minimum target for all particles. When the target value of each of the particles tends to converge into local optimum, the inertia weight will be increased. While the target values tend to disperse, the inertia weight will be decreased. Meanwhile, for the particles whose objective functional value is better than average ones, their corresponding factor of inertia weight should be smaller, in order to be protected. The details of the algorithm are shown as bellow:

Step 1: Initial the position of particles $x_{i,j}$ and velocity

$v_{i,j}$ randomly, and initial the personal best value P_{best} and global best value G_{best} .

Step 2: Evaluate fitness of each particle, store the current position and fitness of each particle in P_{best} , and store the best position and fitness from the whole P_{best} in G_{best} .

Step 3: Update the velocity and position from formula (7) to formula (10).

Step 4: Update the weight by formula (13)

Step 5: For each particle, compare its fitness to the best position it passed, and if it is better, update the personal best position by current position and then compare the whole current P_{best} to G_{best} , and update the G_{best} .

Step 6: Select number of particles to put into the hybrid pool according to the crossover probability. Select two particles randomly to crossover the same number of child particles, calculate the position and velocity of child particle by formula (11) and (12).

Step 7: If the finishing condition is confirmed (predefined computing precision or iteration numbers), stop searching, output the result, else return step 3 to go on searching.

5. Design of neural network optimized by MPSO algorithm

5.1 Design of neural network

According to the features of humanoid robot stepping over stairs, two three-forward networks are designed[17]. The input layer of first network are parameters X_1 and X_2 , which are in the double support period of stepping over stairs for robot. The hidden layer have M nerve cells and the output layer are parameters $L2$ and $H2$. Based on different inputs X_1 and X_2 , let $F_d = \left| ZMP_x - \frac{X2}{2} - \frac{1}{4}(l_1 + l_7) \right|$ be fitness function, which means the ZMP in initial condition is in the center of double support area of robot. V^1_{ij} $(i=1,2; j=1,2,...M)$ are weights between input layer and hidden layer while W^1_{ij} $(i=1,2...M; j=1,2)$ are weights between hidden layer and output layer.

When parameter $L2$ and $H2$ are confirmed, the value of $\theta_i(i=5,6)$ in the most stable condition during the double support period can be obtained. During the single support period, let the value of $\delta\theta_i(i=5,6)$ be the input of the second neural network, and the output are $\delta\theta_i(i=1,2,3,4)$, $V^2_{ij}(i=1,2; j=1,2,...N)$ are the weights between input layer and hidden layer, W^2_{ij} $(i=1,2...N; j=1,2,3,4)$ are the weights between hidden layer and output layer.

5.2 Neural Network optimized by MPSO

The advantage of using PSO algorithm to train the Neural Network is without using the gradient information. Although the global convergence of PSO algorithm is relatively good, the speed of convergence of PSO algorithm is slower. A MPSO algorithm is introduced to train the weight of NN. Given a NN, only coding for connection weight is required, and mapping to individual with code string representation. Meanwhile, each fitness function of weight is evaluated, and fitness value is calculated. Therefore, the problem of training for NN can be converted to a problem of finding a group of best weights to make the connection weights of NN MAX. The detail of the algorithm is shown as bellow.

Step 1: Select suitable hidden cells, and initial the NN.

Step 2: Code the particles of PSO, which consists of V^1_{ij} , W^1_{ij} and b_i .

Step 3: Initial the position of particles $x_{i,j}$ and velocity $v_{i,j}$, and initial the personal best value P_{best} and global best value G_{best} .

Step 4: Each particle is mapped to the weights of NN and constitutes a NN.

Step 5: Select the samples randomly from sample space to train NN.

The next steps are similar with the algorithm MPSO described above.

Final Step: Take the best G_{best} to be used as a group of weights for NN.

6. Design of Fuzzy Logic Controller optimized by MPSO algorithm

A fuzzy logic controller is designed and the rules are optimized offline by MPSO algorithm. There are two fuzzy logic controllers. The first one is used to calculate the node angle of robot during the double support period. The second one is used to calculate the node angle changes of robot during the single support period.

6.1 design of fuzzy logic controller

The membership function distributions for input and output variables of two fuzzy logic controllers are shown in figure 6 and 7.

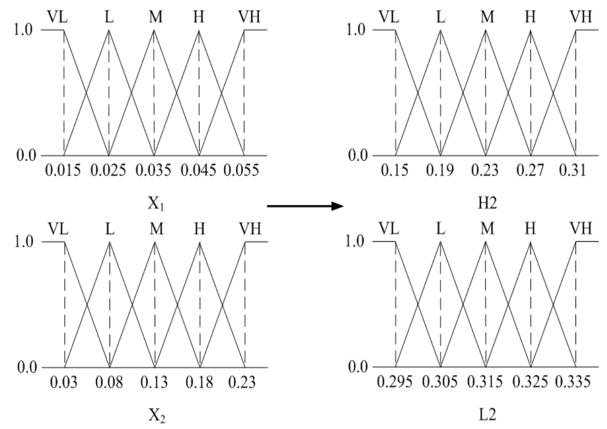


Fig.6 Membership function distributions for input and output variables of the first module of fuzzy logic controller

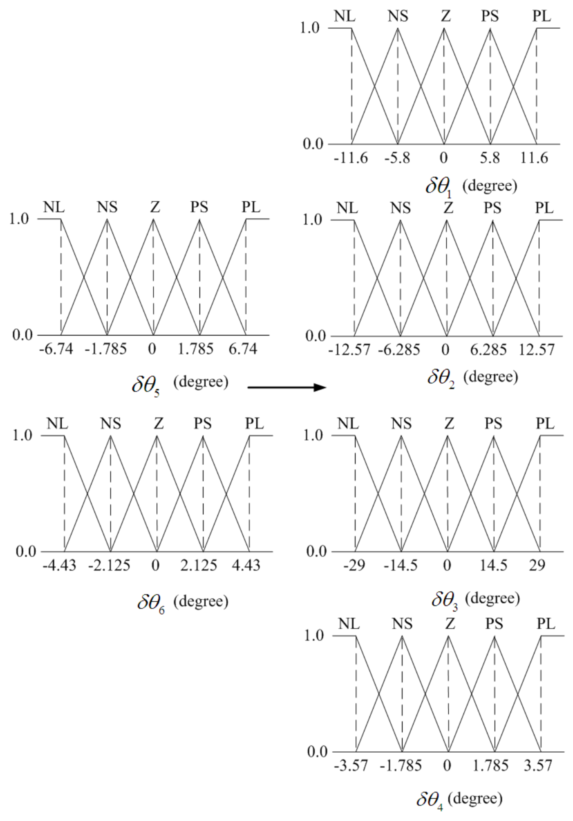


Fig.7 Membership function distributions for input and output variables of the second module of fuzzy logic controller

Where VL,L,M,H,VH represent very small, small, medium, high, very high. NL,NS,Z,PS,PL are negative, slightly negative, zero, slightly positive, positive.

6.2 Fuzzy logic system optimized by MPSO

6.2.1 System framework

In the fuzzy control system the key factors to efficient of control are rules and membership. With traditional trial and random method it is difficult to select the correct control rules and membership, resulting in incomplete control rules and impact of the effectiveness of the system control. Therefore, multiple control rules which are selected randomly are used to control the system, and according to the control effect, optimal control rules are determined through PSO optimization algorithm.

According to the first controller, each of two input parameters has five linguistic variables, and resulting in 25 rules. First, the condition parts of the 25 rules are created by hand, and then two group integers are generated randomly, each group has 25 integers varying between [-2,2], -2 corresponding to VL, -1 corresponding to L, and so on. Therefore, the two group integers correspond to the L2 and H2 respectively, which are results of X_1 and X_2 with fuzzy rules. According to the second controller, which is similar with the first one, 25 rules are created by hand and four group integers are generated randomly, each group corresponding to $\delta\theta_i\ i=1,2,3,4$, which are the results of $\delta\theta_5$ and $\delta\theta_6$. The vector dimension of each particle is set to 150($25\times2+25\times4$),where, each particle represents a rule set. The details of algorithm are shown as bellow.

Step 1: Select a proper fitness function and determine the fuzzy area and fuzzy reasoning methods.

Step 2: Code the particle of PSO, which are composed of linguistic variables such as VL, L, M, H, VH and NL, NS, Z, PS, PL

Step 3: Initiate the position of particles $x_{i,j}$ and velocity $v_{i,j}$, and initiate the personal best value P_{best} and global best value G_{best}.

Step 4: Give a group value to particle randomly. The group values are composed by the rules of fuzzy logic controller.

Step 5: Calculate the result according to the rules.

The next steps are similar with the algorithm MPSO described above.

Final Step: Take the best G_{best} to be used as a group of weights for fuzzy logic controller.

6.2.2 Sensitivity Analysis for System

For the optimization of fuzzy logic controller, both the fitness and control rules can be optimized. Considering optimization may increase the sensitivity of system and reduce the robustness of system. In this paper, only rules are optimized.

Take the first fuzzy logic controller as an example. As input values are the variables of X_1 and X_2, output values are the variables of H_2 and L_2, a control rule can be expressed as bellow:

$$\text{IF } X_1 = A_i \text{ and } X_2 = B_i \text{ then } H_2 = C_i \text{ and } L_2 = D_i \qquad (16)$$

Where, A_i, B_i, C_i, D_i, are the five linguistic variables of output, and $i \in \{\text{VL、L、M、H、VH}\}$

When optimizing fuzzy rules based on MPSO, the fitness can be transformed. The sensitivity of $f(x_i)/\sum_{i=1}^{n} f(x_i)$ can be changed along with the iteration.

$$f'(x_i) = af(x_i) + \frac{e - e^{k/K_{\max}}}{e + e^{k/K_{\max}}}(f_{\max} - f_{\min}) \qquad (17)$$

Where, $f_{\max}$ is the current MAX fitness value, $f_{\min}$ is the current MIN fitness value, k is the current iteration numbers, $K_{\max}$ is the MAX iteration number, and $a > 0$ refers to a constant. From the formula (17) we know that this control method increases the computational efficiency, but depends on the system robust performance. Therefore, this method can be used on line.

7. Experiments

Simulation for humanoid robot walking on uneven terrain is created in Matlab. The parameters of each joint are described as shown in table 1.

Tab.1 Parameters of robot

Joint	Mass(m)	L(m)	r(m)	MI(kgm^2)
1	0.4	0.05	0.02	0.0005
2	1.8	0.10	0.15	0.0922
3	4.8	0.15	0.16	0.1882
4	12	0.25	0.40	3.7004
5	4.8	0.15	0.16	0.1882
6	1.8	0.10	0.15	0.0922
7	0.4	0.05	0.02	0.0005

7.1 Simulation of neural network optimized by MPSO(MPSONN)

A BP neural network is used as a motion control network. The number of hidden cells of BP is 8, and the transfer function between the layers are purelin, tansig and logsig. There are 500 samples for training. The learning rate is 0.3, error is 10^{-5}. The number of dimensions of selected PSO is 62, the number of particles is 30, learning factors C_1, C_2 are both 2, and number of iteration is 500.

The trajectories of all joints of robot during the process of stepping over stairs are shown in figure 8.

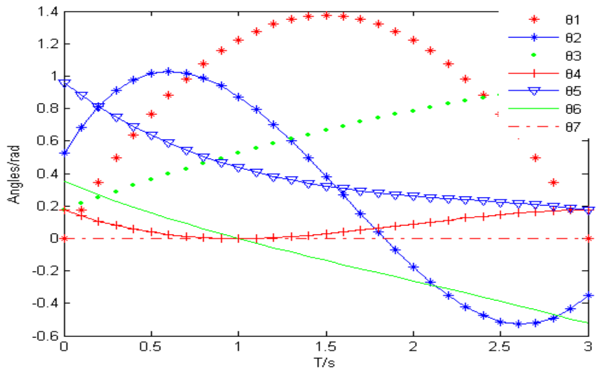


Fig.8 Trajectories of joints based on MPSONN

The weights of NN optimized by MPSO are shown in table 2.

Tab.2 Weights of NN obtained after MPSO

V_{ij}^1	$i=1$	$i=2$	W_{ij}^1	$j=1$	$j=2$				
$j=1$	0.95	1.01	$i=1$	0.97	0.73				
$j=2$	0.94	1.02	$i=2$	1.02	0.95				
$j=3$	0.98	0.99	$i=3$	0.93	0.86				
$j=4$	0.76	1.03	$i=4$	1.05	0.77				
$j=5$	0.91	0.92	$i=5$	0.99	0.94				
V_{ij}^2	$i=1$	$i=2$	W_{ij}^2	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
$j=1$	0.84	1.23	$i=1$	1.06	0.92	0.76	0.56	1.03	1.21
$j=2$	0.82	0.51	$i=2$	1.21	1.03	1.02	1.11	0.45	0.53
$j=3$	0.96	0.69	$i=3$	0.54	0.92	0.44	0.56	0.56	1.23
$j=4$	0.86	0.94	$i=4$	0.76	0.64	1.34	1.03	1.67	0.93
$j=5$	0.90	0.82	$i=5$	1.23	0.86	0.84	1.02	1.55	0.69

7.2 Simulation of fuzzy logic controller optimized by MPSO(MPSOFLC)

The number of particles used to simulate is 30, learning factors are both 2, number of iteration is 500. The most common max-min from Mamdani method is introduced to reason. A common trigonometric function is used as the membership function. The weighted average judgments method of membership function $u = \frac{\sum_{i=1}^{n} \mu(U_i) \cdot U_i}{\sum_{i=1}^{n} \mu(U_i)}$ is adopted to take defuzzification. The rules of two fuzzy logic controller optimized by MPSO are shown in table 3.

Using method MPSOFLC, the trajectories of all joints of robot are show in figure 9.

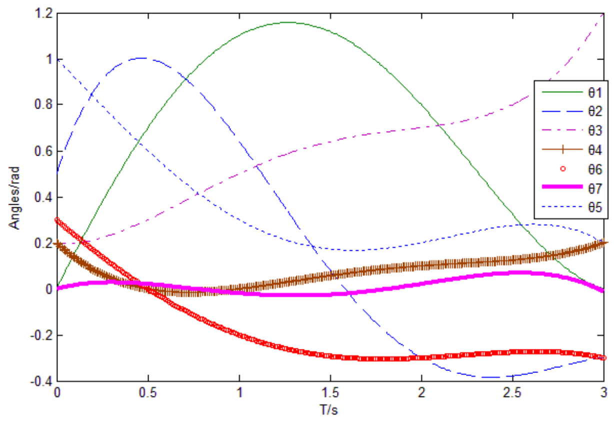


Fig.9 Trajectories of joints based on MPSOFLC

Tab.3 Optimized rule based for the module of fuzzy logic controller

X1	X2	H2	L2	$\delta\theta_5$	$\delta\theta_6$	$\delta\theta_1$	$\delta\theta_2$	$\delta\theta_3$	$\delta\theta_4$
VL	VL	-	-	NL	NL	NL	NL	NL	NL
VL	L	L	L	NL	NS	NS	NS	NS	NS
VL	M	-	-	NL	Z	Z	Z	NL	Z
VL	H	H	H	NL	PS	NL	Z	NL	NS
VL	VH	VH	VH	NL	PL	NS	NL	NS	Z
L	VL	L	VL	NS	NL	Z	NS	Z	PL
L	L	-	-	NS	NS	PS	Z	NL	NL
L	M	L	M	NS	Z	PL	PS	S	NS
L	H	L	VH	NS	PS	-	-	-	-
L	VH	VL	L	NS	PL	NL	NS	PS	PL
M	VL	VH	VL	Z	NL	NS	Z	Z	PS
M	L	L	L	Z	NS	-	-	-	-
M	M	H	H	Z	Z	NS	NL	NS	PL
M	H	-	-	Z	PS	PS	NS	Z	PL
M	VH	VH	VH	Z	PL	PS	Z	NL	PL
H	VL	VL	VL	PS	NL	NL	NL	NS	NL
H	L	L	L	PS	NS	NS	NS	Z	NS
H	M	H	M	PS	Z	Z	NL	NL	Z
H	H	H	H	PS	PS	NL	NS	NS	PL
H	VH	-	-	PS	-	-	-	-	-
VH	VL	VL	VL	PL	NL	Z	PS	PL	NS
VH	L	L	L	PL	NS	NS	NL	NL	Z
VH	M	VH	M	PL	Z	Z	NS	NS	NS
VH	H	H	H	PL	PS	NL	Z	Z	Z
VH	VH	VH	VH	PL	PL	NL	PS	PS	PS

7.3 Comparison of different methods

7.3.1 Comparison of different PSO

In the process of using the basic PSO and MPSO to optimize the weights of NN, the same number of particles, the same initial particles and the same learning factors are arrived. After 500 iterations, the error is 10^{-5}, the process of iteration is shown in figure 10. In figure 11, basic PSO and MPSO are used to optimize the rules of FLC, where the number of dimensions is 150 and number of particles is 40.

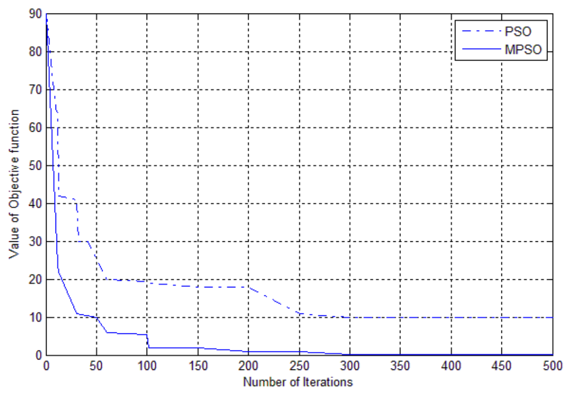


Fig.10 Iterations of NN based on different PSO

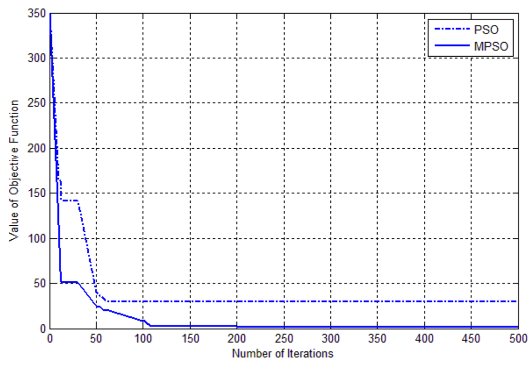


Fig.11 Iterations of optimizing rules of fuzzy logic controller based on different PSO

7.3.2 Comparison of control methods

In the process of stepping over stairs for humanoid robot, NN, PSONN, PSOFLC, MPSONN, and MPSOFLC are introduced to control the robot. The time consumed by different methods is displayed in figure 12.

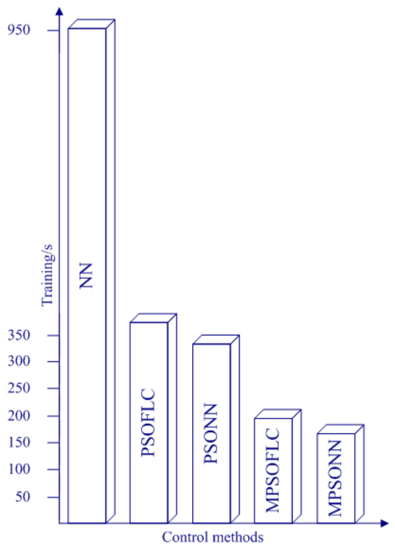


Fig.12Trainning time under different control methods

From the figure 12, we can conclude that MPSONN is the fastest method to control the robot to step over stairs compared to other methods, and the NN is the slowest one. The ZMP trajectory is used to compare the stabilization robot through different methods.

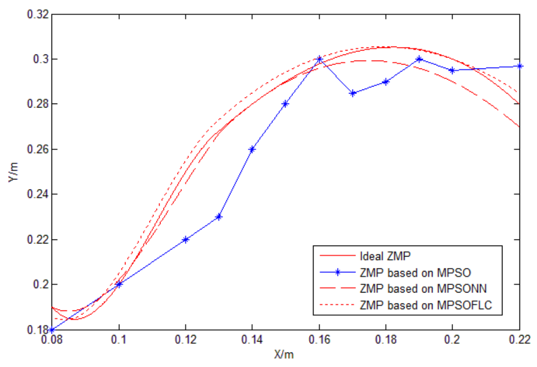


Fig.13 ZMP Trajectories under different controll methods

From figure 13, we can see, using direct PSO to control the robot is not a good method. MPSONN and MPSOFLC can control the robot to step over stairs and obtain better stability.

7.4 Simulation of process of stepping over stairs for humanoid robot

The simulations of stepping over stairs through different methods are described in figure 14, 15 and 16.

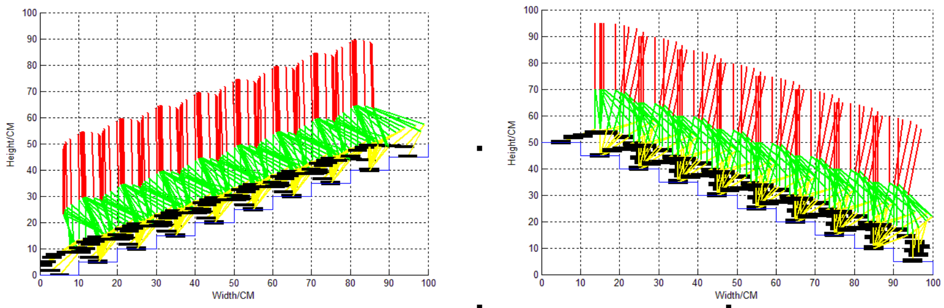


Fig.14 Simulations of going up and down stairs for humanoid robot (MPSO)

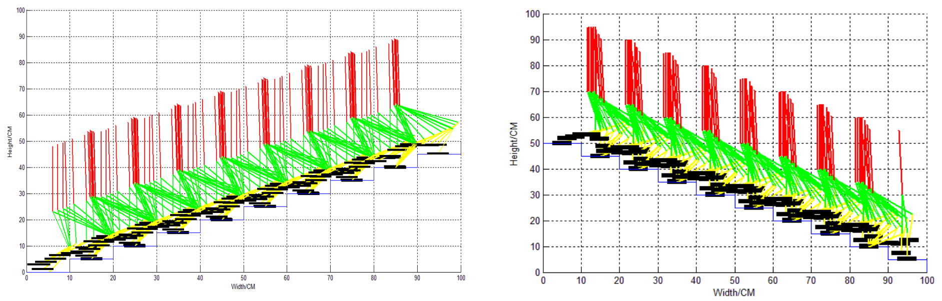


Fig.15 Simulations of going up and down stairs for humanoid robot (MPSOFLC)

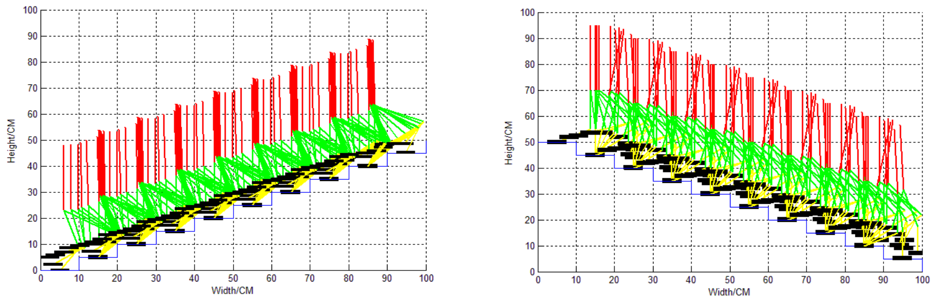


Fig.16 Simulations of going up and down stairs for humanoid robot (MPSONN)

7.4 Experiments on real humanoid robot (NAO)

Nao is a humanoid robot desigend by Aldebaran Robotics, it can run the method present in this paper effectively. The structure of robot NAO is described as in Fig.17 . Method of MPSONN is selected to achieve the stepping motion by NAO. The parameters are shown in Tab.4-Tab.6. and snapshots of stepping motion are shown in Fig.18.

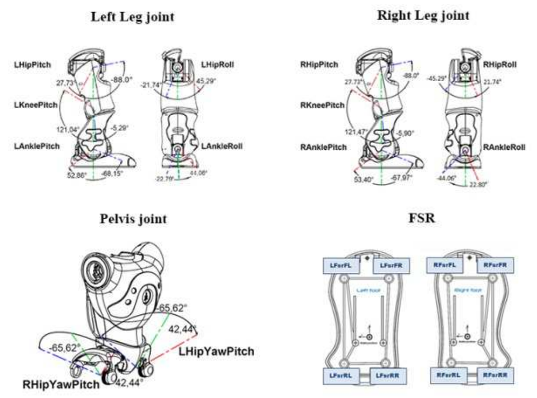


Fig.17 structure of robot NAO

To perform the rotation of the body parts, we place a frame at each joint. When the robot is at the zero pose, all joint frames have the same orientation. Then, roll rotations take place around the X axis, pitch rotations around the Y axis and yaw rotations around the Z axis. The detail of parameters of robot between distance and jonts are described as in Tab.4 and Tab.5

Tab.4 The Roll Motion

Distance(cm) Joint(rad)	0	14	28	42	56	70	84	98
LAnkleRoll	0.45	0.59	0.59	0.44	0	0	0.16	0
LHipRoll	0.78	0.48	0.38	0.21	0.75	1.04	0.20	0.37
RAnkleRoll	0.50	0.36	0.38	0	0	0.49	0.61	0.52
RHipRoll	0.64	0.06	0.52	0.29	0	0.40	0	1.03

Tab.5 The Pitch Motion

Distance(cm) Joint(rad)	0	14	28	42	56	70	84	98
LAnklePitch	0.00	0.88	0.93	0.00	0.05	0.00	1.12	0.71
LKneePitch	0.76	0.27	0.32	0.94	0.84	1.09	0.08	0.38
LHipPitch	0.00	0.27	1.06	0.25	0.20	0.08	1.15	0.03
LHipYawPitch	0.84	0.61	0.68	0.50	0.55	0.53	1.02	0.15
RHipPitch	1.04	0.96	0.29	1.05	0.94	1.20	0.78	1.70
RKneePitch	0.21	0.45	0.24	0.49	0.11	0.85	0.30	0.11
RAnklePitch	1.02	0.25	1.35	0.75	1.05	0.16	1.21	0.76
RHipYawPitch	0.84	0.61	0.68	0.50	0.55	0.53	1.02	0.15

In ladder, the sole of robot's feet can feel different pressure.Through the monitor, we can get the value of FSC(Feedback Shift Register)of four point under feet: FsrFL, FsFR, FarRL,FarRR, and the distance of offset: COOx, COOy. The value returned for each FSR is similar to Kg. If FSR are calibrated (see robot configuration keys), the value in kg is about 20% precision (depending time and real force position). Without calibration the error could be more important, and is specific to each sensor. The detail experimental datas are shown in Tab.6

Tab. 6 Datas from FSR by NAO

Distance(cm) Point(rad)	0	14	28	42	56	70	84	98
LFsrRR	1.13	0.36	0.41	1.27	1.04	0.73	0.49	0.01
LFsrRL	0.01	0.20	0.01	0.01	0.38	0.85	2.11	1.21
LFsrFR	0.18	0.00	0.01	0.01	0.00	0.00	0.07	0.02
LFsrFL	0.02	0.02	0.02	0.02	0.02	0.00	0.03	0.03
LCOPy	0.00	1.07	0.00	0.21	0.07	0.69	0.12	0.00
LCOPx	0.89	0.08	0.26	0.31	0.04	0.00	0.00	2.24
RFsrRR	0.00	0.00	0.00	0.00	0.00	0.23	0.18	0.00
RFsrRL	0.43	0.00	0.34	0.00	0.00	0.00	0.00	0.00
RFsrFR	0.02	0.02	0.02	0.00	0.01	0.02	0.02	0.01
RFsrFL	0.01	0.03	0.02	0.03	0.03	0.01	0.03	0.03

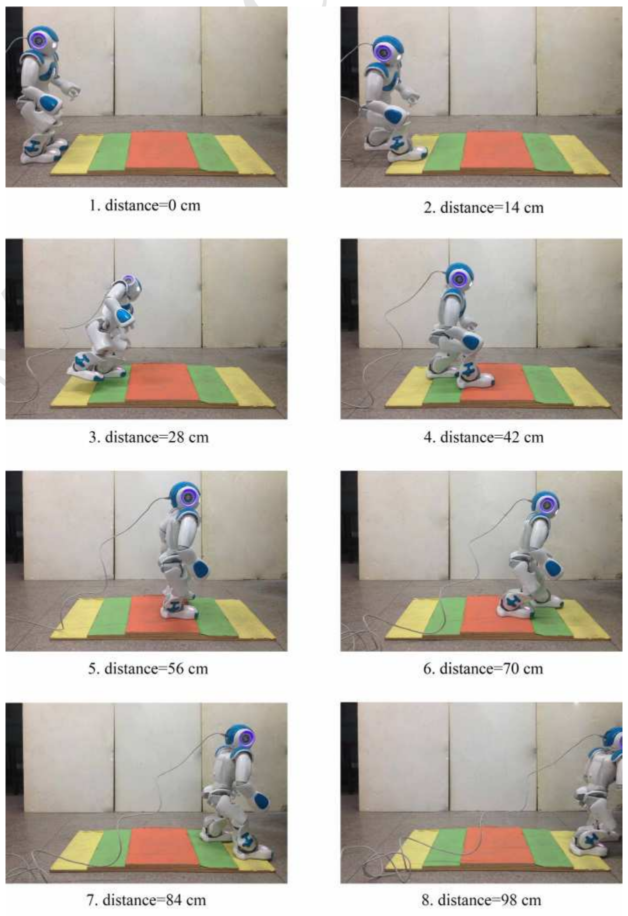


Fig. 18 Snapshots of stepping motion by NAO

8. Conclusion

Stepping over stairs can be taken as an example of walking on uneven terrain for humanoid robot. A controller based on NN and FLC are designed to control the robot walking. A mixed PSO algorithm is present to optimize the weights of NN and rules of FLC. Comparison of different control methods are achieved through the simulations and tests the effectiveness of the proposed methods.

Acknowledgement

This material is based upon work funded by State Key Laboratory of Robotics and System (HIT)Foundation of China under Grant No.SKLRS-2012-MS-06, China Postdoctoral Science Foundation under Grant No.2013M531022, Research project of laboratory work in universities of Zhejiang Province under Grant No. ZD201504, Educational technology research program of Zhejiang Province under Grant No. JA027.

Reference

[1] Shouwen F, Min S, Mingquan S. Real-time Gait Generation for Humanoid Robot Based on Fuzzy Neural Networks[C]. The Third International Conference on Natural Computation, 2007:142-150.

[2] Pandu Ranga Vundavilli, Dilip Kumar Pratihar, Soft computing-based gait planners for a dynamically balanced biped robot negotiating sloping surfaces. Applied Soft Computing, 9(2009) 191-208

[3] Zhang tong. Research on walking control and motion planning for humanoid robot[Dissertation], Guangzhou : SOUTH CHINA UNIVERSITY OF TECHNOLOGY, 2010:22-23.

[4] Rahul Kumar Jha, Balvinder Singh, Dilip Kumar Pratihar, On-line stable gait generation of a two-legged robot using a genetic-fuzzy system, Robotics and Autonomous Systems 53 (2005)15-35

[5] Gao fang. Research on intelligent particle swarm optimization algorithm.[Dissertation] Harbin: Harbin Institute of Technology, 2008:42-57

[6] Si shubing, Sun shudong, Xu yaping. Multi-Colony Diploid Immune Algorithm(MCDIA) for Better Job-Shop Scheduling. Journal of Northwestern Polytechnical University, 2007,25(1):27-31

[7] Alba E, Luna F, Nebro A J, et al. Parallel Heterogeneous Genetic Algorithms for Continuous Optimization[J]. Parallel Computing, 2004, 30：699-719.

[8] Seung-Mok Lee, Hanguen Kim, and Hyun Myung, Cooperative coevolution-based Model Predictive control for Multi-robot Formation, 2013 IEEE International conference on robotics and Automation(ICRA), Karlsruhe, Germany, May 6-10, 2013, pp:1882-1887

[9] Micael S. couceiro, Rui P. Rocha, Nuno M. Fonseca Ferreira, Fault-Tolerance Assessment of a Darwinian Swarm Exploration Algorithm under communication Constraints, 2013 IEEE International conference on robotics and Automation(ICRA), Karlsruhe, Germany, May 6-10, 2013, pp:2000-2005

[10] Ezequiel Di Mario, Inaki Navarro, Alcherio Martinoli, The Role of Environmental and Controller Complexity in the Distributed Optimization of Multi-robot Obstacle Avoidance.2014 IEEE International conference on robotics and Automation(ICRA), Hong Kong Convention and Exhibition Center, Hong Kong, China, May 31-June 7, 2014, pp:571-577

[11] Vojtech Vonasek, Martin Saska, Karel Kosnar and Libor Preucil, Global motion planning for modular robots with local motion primitives. 2013 IEEE International conference on robotics and Automation(ICRA), Karlsruhe, Germany, May 6-10, 2013, pp:2450-2455

[12] Tao Li, Kohei Nakajima and Rolf Pfeifer. Online Learning for Behavior Switching in a Soft Robotic Arm. 2013 IEEE International conference on robotics and Automation(ICRA), Karlsruhe, Germany, May 6-10, 2013, pp:1288-1294

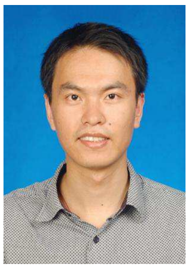
[13] Pedro Neto, Dario Pereira, J. Norberto Pires, A. Paulo Moreira. Real-time and Continuous Hand Gesture Spotting: an Approach Based on Artificial Neural Networks. 2013 IEEE International conference on robotics and Automation(ICRA), Karlsruhe, Germany, May 6-10, 2013, pp:178-183

[14] Park J H. Fuzzy-logic zero-moment-point trajectory generation for reduced trunk motions of biped robots[J]. Fuzzy Sets System, 2003：189-203

[15] Zhou C, Meng Q. Dynamic balance of a biped robot using fuzzy reinforcement learning agents[J]. Fuzzy Sets System, 2003, 134：169-187.

[16] S´ebastien L, Nacim R, Philippe F. Planning and Fast Re-Planning of Safe Motions for Humanoid Robots: Application to a Kicking Motion[C]. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009：441-447.

[17] Zhong Qiubo, Piao Songhao, Gao Chao. Motion planning for humanoid robot based on hybrid evolutionary algorithm. International Journal of Advanced Robotic Systems, 2010, 7(3):209-216



Qiubo Zhong, he was born in 1980, and received his B.S and M.S in Computer Science of Technology from Harbin University of Science and Technology, Harbin ,China, in 2003 and 2006, He received his Ph.D degrees in Computer Application from Harbin Institute of Technology, Harbin, China, in 2011. He is currently working in School of Electronic and Information Engineering, Ningbo University of Technology, Ningbo China. His interests include humanoid robot, motion planning.

Fei Chen, he was born in 1983, and received his B.S in Computer Science of Technology from Xi'an Jiaotong University, Xi'an ,China, in 2006 He received his M.S. degree in computer Application from Harbin Institute of Technology, Harbin, China, in 2008, He received his Ph.D degrees in Computer Application from Nagoya University, Japan, in 2012. He is currently working in Advanced Robotics Department, Istituto Italiano di Tecnologia, Genova, Italy. His interests include robot, motion planning, image processing.