Autonomous Evolution of High-Speed Quadruped Gaits Using Particle Swarm Optimization

Chunxia Rong, Qining Wang, Yan Huang, Guangming Xie, and Long Wang

Intelligent Control Laboratory, College of Engineering,
Peking University, Beijing 100871, China
xiegming@mech.pku.edu.cn
http://www.mech.pku.edu.cn/robot/fourleg/

Abstract. This paper presents a novel evolutionary computation approach to optimize fast forward gaits for a quadruped robot with three motor-driven joints on each limb. Our learning approach uses Particle Swarm Optimization to search for a set of parameters automatically aiming to develop the fastest gait that an actual quadruped robot can possibly achieve, based on the concept of parameterized representation for quadruped gaits. In addition, we analyze the computational cost of Particle Swarm Optimization taking the memory requirements and processing limitation into consideration. Real robot experiments show that the evolutionary approach is effective in developing quadruped gaits. Satisfactory results are obtained in less than an hour by the autonomous learning process, which starts with randomly generated parameters instead of any hand-tuned parameters.

1 Introduction

Over the past years, plenty of publications have been presented in the biomechanics literature which explained and compared the dynamics of different high-speed gaits including gallop, canter, bound, and fast trot (eg. [15], [16]). To study and implement legged locomotion, various robot systems have been created (eg. [3], [4], [5]). However, most of the high-speed machines have passive mechanisms which may be not easy to perform different gaits. To understand and apply high-speed dynamic gaits, researchers have implemented different algorithms or hand-tune methods in the simulation [6] and real robot applications [2], [7], [8]. Much published research in learning gaits for different quadruped robot platforms used genetic algorithm based methods. Different from genetic algorithms, Particle Swarm Optimization (PSO) described in [1], [13], [14] eliminated the crossover and mutation operations. Instead, the concept of velocity was incorporated in the searching procedure for each solution to follow the best solutions found so far. PSO can be implemented in a few lines of computer code and requires only primitive mathematical operators. Taking the memory and processing limitation onboard into account, PSO is more appropriate in gaits

L. Iocchi et al. (Eds.): RoboCup 2008, LNAI 5399, pp. 259–270, 2009.

learning comparing with the genetic algorithm based methods for quadruped robots, especially those commercial robots with kinds of motors.

Our research focused on the gait optimization of legged robot with motor-driven joints. The commercial available quadruped robot, namely the Sony Aibo robot, which is the standard hardware platform for RoboCup four-legged league, is the main platform that we analyze and implement algorithms. Aibo is a quadruped robot with three degrees of freedom in each of its legs. The locomotion is determined by a series of joint positions for the three joints in each of its legs. Early research in gait learning for this robot employed joint positions directly as parameters to define a gait, which was the case in the first attempt to generate learned gait for Aibo. However, being lack of consistency in representing the gaits, these parameters failed to exhibit the gait in a clear way. Most of the recent research used higher lever parameters to symbolize the gait which focus on the stance of the body and the trajectories of paw. An inverse kinematics algorithm was then implemented to convert these higher lever parameters into joint angles. The general high-lever parameters used to describe the gait for Aibo can be divided into three groups. One group is for determining the gait patter by the relative phase for each leg. [11] mentioned that there exist eight types of gait patters for quadruped animals in nature. [7] described three of the most effective gaits for quadruped robot especially for Aibo, which are the crawl, trot and pace. Another group of the parameters is associated with the stance of robot. The last group of parameters describes the locus of the gait. Most of the gaits developed for Aibo based on this high lever parameter represent method differ in the shaped of the locus of paws or the representation of the locus, that is the actual parameters used to trace out the locus, eg. [9] [10].

In this paper, we present the implementation of Particle Swarm Optimization in generating high-speed gaits for the quadruped robot, specifically the Aibo. First, an overview of the basic PSO and Adaptive PSO (APSO) are introduced. Our gait learning method is based on APSO. With the knowledge of using higher lever parameters to represent the gait which focus on the stance of the body and the trajectories of the paw, the inverse kinematics model is explained. Moreover, the control parameters and optimization problem are proposed. In addition, how to implement PSO in the quadruped gaits learning is introduced in detail. The whole learning process is running automatically by the robot.

In the following section, we introduce the basic PSO and APSO specifically. In section III, we present the optimization problem in gaits learning. In section IV, the process of implementing PSO in quadruped gaits learning is introduced in detail. At last, gaits optimization results are shown in experiments with the Sony Aibo ERS-7 robot.

2 Particle Swarm Optimization

2.1 Overview of the Basic PSO

Particle Swarm Optimization (PSO) is a stochastic optimization technique, inspired by social behavior of bird flocking or fish schooling [12]. It is created by

Dr. Eberhart and Dr. Kennedy in 1995 [1]. Similar with Genetic Algorithms, PSO method searches for optimal solutions through iterations of a population of individuals, which are called a swarm of particles in PSO. However, the crossover and mutation operation are replaced with moving inside the solution space decided by the so-called velocity of each particle. PSO has proved to be effective in solving many global optimization problems and in some areas outperform many other optimization approaches including Genetic Algorithms.

PSO theory derives from imitation of social behavior of bird flocking or fish schooling. It is discovered that each bird, when hunting for food in a bird flock, changes its flying direction based on two aspects: one is the information of food found by itself; the other is information of flying directions of other birds. When one of the birds gets food, the whole flock has food. It is similar to social behavior of human being. People's decision making is not only influenced by their own experience but also affected by other people's behavior.

For an optimization procedure, hunting food by bird flock becomes searching for an optimal solution to this problem. One solution of the problem corresponds to the position of one bird (called particle) in the searching space. Each particle remembers the best position which was found by itself so far, and this information together with its current position makes up the personal experience of that particle. Besides, every particle is informed of the best value obtained so far by particles in its neighborhood. When a particle takes the whole flock as its topological neighbors, the best value is a global one. Each particle then changes its position in according to its velocity relied on this information: the personal best position, current position and the global best position.

In the realization of the PSO algorithm, a swarm of N particles is constructed inside a D-dimensional real valued solution space, where each position can be a potential solution for the optimization problem. The position of each particle is denoted X_i $(0 < i < N)$, a D-dimensional vector. Each particle has a velocity parameter V_i $(0 < i < N)$, which is also a D-dimensional vector. It specifies that the length and the direction of X_i should be modified during iteration. A fitness value attached to each location represents how well the location suits the optimization problem. The fitness value can be calculated by the objective function of the optimization problem.

At each iteration, the personal best position $pbest_i$ $(0 < i < N)$ and the global best position $gbest$ are updated according to fitness values of the swarm. The following equation is employed to adjust the velocity of each particle:

$$v_{id}^{k+1} = v_{id}^{k} + c_1 r_{1d}{}^{k}(pbest_{id}^{k} - x_{id}^{k}) + c_2 r_{2d}^{k}(gbest_{d}^{k} - x_{id}^{k}) \tag{1}$$

Where v_{id}^{k} is one component of V_i (d donates the component number) at iteration k. Similarly, x_{id}^{k} is one component of X_i at iteration k. The velocity in equation (1) consists of three parts. One is its current velocity value, which can be thought as its momentum. The second part is the influence of the personal best. It tries to direct the particle back to the best place it has found. The last part associated with the global best attempts to move the particle toward the $gbest$. c_1 and c_2 are acceleration factors. They are used to tune the maximum length of flying in

each direction. r_1 and r_2 are random numbers uniformly distributed between 0 and 1. They contribute to the stochastic vibration of the algorithm. It should be noted that each component of the velocity has new random numbers, not that all the components share the same one. In order to prevent particles from flying outside the searching space, the amplitude of the velocity is constrained inside a spectrum $[-v_d^{max}, +v_d^{max}]$. If v_d^{max} is too big, the particle may fly beyond the optimal solution. If v_d^{max} is too small, the particle will easily step into the local optimum. Usually, v_d^{max} is decided by the following equation:

$$v_d^{max} = kx_d^{max} \tag{2}$$

where $0.1 \leq k \leq 1$. Now the current position of particle i can be updated by the following equation:

$$x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1} \tag{3}$$

PSO algorithm is considerably easy to realize in computer coding and only a few primitive mathematical operators are involved. Furthermore, it has the advantage of multiple points searching at the same time. Most importantly, the speed of converging is remarkably high in many learning processes. It is a critical virtue when it comes to learning gaits in a physical robot, because it minimizes damage to the robot.

The basic PSO is an algorithm base on stochastic searching, so it has strong ability in global searching. However, in the final stage of searching procedure, it is difficult to converge to a local optimum because the velocity still has much momentum. To improve the local searching ability in the final stage of optimization process, the influence of previous velocity on the current velocity needs to decrease. Thus, we proposed the using of adaptive PSO with changing inertia weight in this study.

2.2 Adaptive PSO with Changing Inertia Weight

In equation (1), by multiplying inertia weight to the momentum part of the velocity vibration can control the impact of previous velocity on the current velocity. The update equation for velocity with inertial weight is as follows:

$$v_{id}^{k+1} = wv_{id}^k + c_1 r_{1d}^k (pbest_{id}^k - x_{id}^k) + c_2 r_{2d}^k (gbest_d^k - x_{id}^k) \tag{4}$$

where w is the inertia weight. PSO with larger inertial weight results in better global searching ability for the reason that the search area is expanded with more momentum. Small inertial weight limits the search area thus improving local searching ability. Empirical results show that PSO has faster convergent rate when w falls in the range from 0.8 to 1.2. With the intention of realizing both fast global search at the beginning and intensive searching in the final stage of iteration, the value of w should vary gradually from high to low. It is similar to the annealing temperature of Simulated Annealing Algorithm. In this way, both global searching in a broaden area at the beginning and intensive search in a currently effective area at the end can be realized.

3 Optimization Problem

3.1 Inverse Kinematics Model

The high-lever parameters that we adopt to represent the gait need to be transferred to joint angles of legs before they can be implemented by the robot. An inverse kinematics model can be used to solve this problem. For a linked structure with several straight parts connecting with each other, the position of the end of this structure relative to the starting point can be decided by all angles of linked parts and only one position results from the same angle values. The definition of the kinematics model is the process of calculating the position of the end of a linked structure when given the angles and length of all linked parts.

In this robot Aibo case, given the angles of all the joints of the leg, the paw positions relative to the shoulder or the hip will be decided. Inverse kinematics does the reverse. Given the position of the end of the structure, inverse kinematics calculates out what angles the joints need to be in to reach that end point. In this study, the inverse kinematics is used to calculate necessary joint angles to reach the paw position determined by gait parameters. Fig.1 shows the inverse kinematics model and coordinates for Aibo. The shoulder or hip joint is the origin of the coordinate system. l_1 is the length of the upper limb, while l_2 is the length of the lower limb. Paw position is represented by point (x, y, z). The figures and equations below only give the view and algorithm to get the solution for left fore leg of robot. In according to the symmetrical characteristic of legs, all other legs can use the same equations with some signs changing.

The following equations shows the inverse kinematics model:

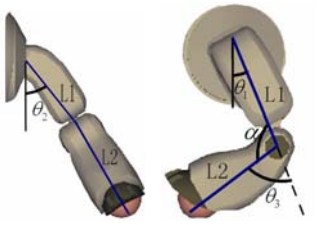
$$\begin{aligned} x &= l_2 \cos\theta_1 \sin\theta_3 + l_2 \sin\theta_1 \cos\theta_2 \cos\theta_3 + l_1 \sin\theta_1 \cos\theta_2 \\ y &= l_1 \sin\theta_2 + l_2 \sin\theta_2 \cos\theta_3 \\ z &= l_2 \sin\theta_1 \sin\theta_3 - l_2 \cos\theta_1 \cos\theta_2 \cos\theta_3 - l_1 \cos\theta_1 \cos\theta_2 \end{aligned} \tag{5}$$

The inverse kinematics equation to get θ_1, θ_2, θ_3 by the already known paw position (x, y, z) is as follows:

$$\begin{aligned} \theta_3 &= \cos^{-1} \frac{x^2 + y^2 + z^2 - l_1^2 - l_2^2}{2 l_1 l_2} \\ \theta_2 &= \sin^{-1} \frac{y}{l_2 \cos\theta_3 + l_1} \\ \theta_1 &= -\tan^{-1} \frac{a}{b} \pm \cos -1 \frac{x}{a^2 + b^2} \end{aligned} \tag{6}$$

where $a = l_2 \sin\theta_3$, $b = -l_2 \cos\theta_2 \cos\theta_3 - l_1 \cos\theta_2$.

One problem with the inverse kinematics is that it always has more than one solution for the same end point position. However, as to Aibo, only one solution is feasible due to the restriction on the joint structure. As a result, when using inverse kinematics to calculate joint angles, it is necessary to take joint structure limitation into consideration to get the right solution. Otherwise, it will possibly cause some physical damage to the robot platform.



(a) (b)

Fig. 1. The inverse kinematics model and coordinates for Aibo. (a) is the front view of left fore leg. (b) is the side view of left fore leg.

3.2 Control Parameters

Before we run the learning gait procedure, the control parameters representing a gait need to be decided. There are two rules based on which we choose our parameters: One is the sufficient representation of the gait that makes it possible to get a high-performance gait in an expanded area. The other one is the attempt to limit the number of control parameters in order to reduce the training time. These two rules are to some extent contradicted with each other. We have to find a better way to compromise these two policies manually. We have done some work on the robot's gait patters and found out that trot gait is almost always the most effective pattern in terms of both stability and speediness,thus we limit the gait pattern to mere trot gait.

For stance parameters, based on our observation and analyze of the motion for Aibo, we conclude that forward-leaning posture can speed up the walking, thus

Table 1. Control parameters in gaits evolution

fore height	vertical height from paw to chest
hind height	vertical height from paw to hip
fore width	transverse distance between paw and chest
hind width	transverse distance between paw and hip
fore length	forward distance between paw and chest
hind length	forward distance between paw and hip
step length	time for one complete step in 0.008 second units
fore step height	fore height of the locus
hind step height	hind height of the locus
fore step width	fore width of the locus
hind step width	hind width of the locus
fore ground time	fore paw fraction of time spent on ground
hind ground time	hind paw fraction of time spent on ground
fore lift time	fore paw fraction of time spent on lifting
hind lift time	hind paw fraction of time spent on lifting
fore lowing time	time spent on fore paw lowing around locus
hind lowing time	time spent on hind paw lowing around locus

we constrain the range of stance parameters to keep robot in forward-leaning posture, that is the height of hip higher than that of chest. As to locus, we choose rectangle shape because it has proved to be effective in quadruped gaits and it is simple to be represented. And because of the symmetry of right and left side when moving straight forward, we use the same locus for right legs and left legs. In all, we choose our parameters of gait as shown in Table 1.

4 Implementation of PSO

Given the parametrization of the walking defined above, we formulate the problem as an optimization problem in a continuous multi-dimensional real-value space. The goal of the optimization procedure is to find a possibly fastest forward gait for the robot, therefore the objective function of the optimization problem is simply the forward speed of the walking parameters. Particle Swarm Optimization is then employed to solve this problem with a particle corresponding to a set of parameters. A predetermined number of sets of parameters construct a particle swarm which will expose to learning by PSO, with the forward speed of each parameter being the fitness.

4.1 Initialization

Initially, a swarm of particles are generated in the solution space, which is a set of feasible gait parameters. These particles can be represented by $\{p_1, \Lambda, p_N\}$ (where $N = 10$ in this case).These sets of parameters are acquired by random generation within the parameter limits decided by the robot mechanism. A lot of previous work done on learning gaits start from a hand-tune set of parameters. Comparing with previous work, random generation of initial values has the advantage of less human intervention, and more importantly, has more possibility to lead to different optimal values among different experiments. Initial velocities for all particles are also generated randomly in the same solution space within given ranges. The width of the range is chosen to be half of that of the corresponding parameters. Velocity calculated later is also constrained inside the spectrum. The spectrum is denoted by $(-V^{max}, +V^{max})$, where $V^{max} = \frac{1}{4}(x^{max} - x^{min})$, with (x^{min}, x^{max}) as the changing range of particle P_i. The ranges we chosen turn out to be appropriate to avoid the two problems mentioned in Section II.A. To expedite the search process, c_1 and c_2 are set to 2. The initial *pbest*s are equal to the current particle locations. There is no need to keep track of *gbest* while it can be acquired from *pbest*s, that is the *pbest* with the best fitness is *gbest*.

4.2 Evaluation

The evaluation of parameters is performed using sole speed. Since the relation between gait parameters and speed is impossible to acquired, we do not know the true objective function. There is no sufficiently accurate simulator for Aibo

due to the dynamics complexity. As a result, we have to perform the learning procedure on real robots.

In order to automatically acquiring speed for each parameter set, the robot has to be able to localize itself. We use black and while bar for Aibo to localize, because given the low resolution of Aibo's camera, it is faster and more accurate to detect black-while edge than other things. We put two pieces of boards with the same black and while bars in parallel so the robot can walk between them.

During evaluation procedure, the robot walks to a fixed initial position relative to one of the boards, then load the parameter set needed to be evaluated, walk for a fixed time, 5s, stop and determine the current position. It should be noted that both before and after the walk, robot is in static posture, so the localization is better compare to localizing while running. Now the starting and ending location have been acquired from detecting the bars, speed can be calculated out. After that, robot turns around by 90 degree, and localizes according to the other board, if the position is far from the fixed position, adjust it or else go to the next step, loading another set of parameters and then begin another trial. The total time for testing one set of parameter including turning, localizing, and walking time. Because of the ease of localizing, usually it takes less than 3s to turn and get to the right position. As a result, the test time of one particle is less than 8s.

4.3 Modification

After all particles of the swarm are evaluated, *pbest*s are updated by comparing them with corresponding particles. If the performance of P_i is better than $pbest_i$, which means the fitness value of P_i is higher than that of $pbest_i$, $pbest_i$ will be replaced by the new position of P_i. In addition, the fitness value of P_i is recorded as fitness value of $pbest_i$ for future comparing. Subsequently, the new *gbest*, the best among *pbest*s can be acquired. It should be noted that the update of *gbest* is not done anytime a particle is evaluated but after the whole swarm is evaluated. The difference does not change the principle of the algorithm or empirically influence the converge rate.

As mentioned in section II, in order to realize global search in a broaden area at the beginning of the learning procedure and intensive search in a currently effective area at the end, we employ adaptive PSO with piecewise linearity declining inertial weight to perform the learning procedure. When inertial weight value w is around 1, it presents global search characteristics and results in fast converge rate. when w is a lot less than 1, intensive search is realized.

5 Experimental Results

Using the method described above, we take two separate experiences and achieve favorable results. In the first experience, since large inertial weight will extend the searching area, resulting in a long time of training, we take a conservative move and reduce inertial weight quickly from the start with initial value being 1. The inertial weight is determined by equation (7). Fig. 2(a) shows the vibration

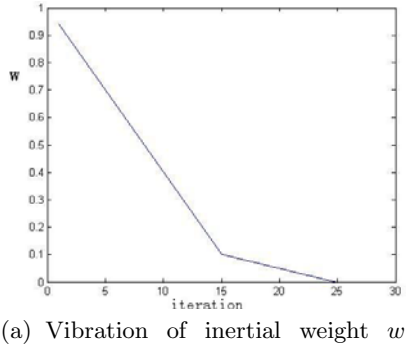
of w through iterations. By iteration 15, w has decreased to 0.1. The global search is diminished, while the intensive search is enhanced. Fig. 3(a) shows the result through iterations. We can see that the learning process is converging quite fast from 1 to 10 iteration. After that, the result improve slowly but firmly until around 25 iteration. Although we get a high-performance gait in a short time in this experiment, we think it is possible that we can have a better result when extending the search area a little by not reducing w so fast. So we tried to use another equation (8) to update w, Fig. 2(b) shows the vibration of w, and Fig. 3(b) shows the learning result.

$$\begin{aligned} w &= 1 - 0.06 \times iter(iteration \leq 15) \\ w &= 0.1 - 0.01 \times (iter - 15)(15 < iteration \leq 25) \\ w &= 0(iteration > 25) \end{aligned} \tag{7}$$

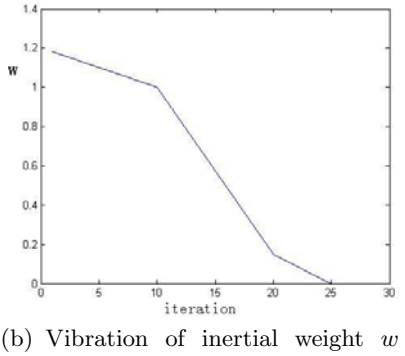
$$\begin{aligned} w &= 1.2 - 0.02 \times iter(iteration \leq 10) \\ w &= 1 - 0.085 \times (iter - 10)(10 < iteration \leq 20) \\ w &= 0.15 - 0.03 \times (iter - 20)(20 < iteration \leq 25) \\ w &= 0(iteration > 25) \end{aligned} \tag{8}$$

We can see that the second experiment achiever better result than the first one. It is interesting that they both reach their peak in the 25 iteration, that's the time when w become zero. It's possible that PSO has little local optimization when current velocity is no longer influenced by previous velocity which is contradicted to what we assumed.

We can also note that there are both advantage and disadvantage comparing these experiments with each other. For one thing, the learning curve of the first experiment is a lot smoother than that of the second one. It means that the second learning process has more undulation. In fact, during the second experiment, there are still new sets of parameters that perform very poor after the 10

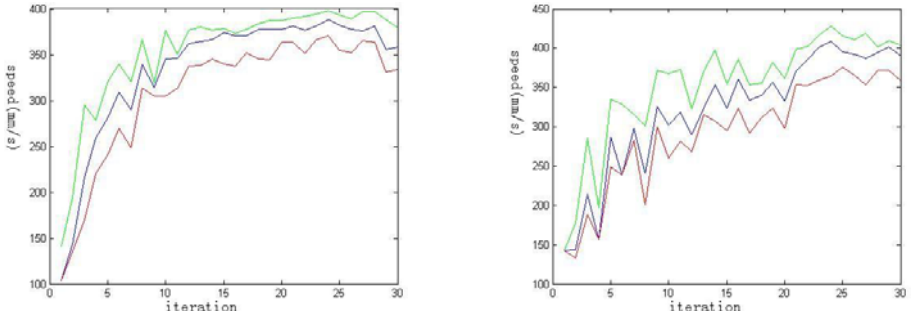


(a) Vibration of inertial weight w through iterations in the first experiment.



(b) Vibration of inertial weight w through iteration in the second experiment.

Fig. 2. Vibration of inertial weight w through iteration in real robot experiments



(a) The first experiment (b) The second experiment

Fig. 3. Optimization results. (a) is the best(in the green line), average of the whole swarm(in the red line)and average of the best half part of the swarm(in the blue line) in real robot experiments. (b) the best result of every iteration in both the two experiments. The green is the first one, and the blue is the second one.

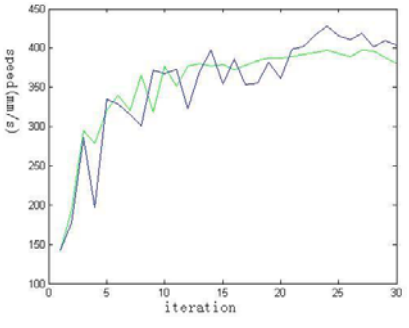


Fig. 4. The best result of every iteration in both the two experiments. The green line is the first one, while the blue line is the second one.

iteration due to the extended searching area. This problem cause more damage to physical robot. However, the second experiment acquire better parameters also because of the extended searching area. Fig. 4 shows the best result of every iteration in both the two experiments.

6 Conclusion

In this paper, we have demonstrated a novel evolutionary computation approach to optimize fast forward gaits using Particle Swarm Optimism. PSO has been proven to be remarkable effective in generating optimal gaits in the robot platform Aibo. Our method was easily coded and computationally inexpensive. Moreover, by using PSO, the evolution converged extremely fast and the training time was largely reduced. That is an essential advantage for physical robot learning, minimizing possible damage to the robot. Another contribution

of our method was its initial sets of parameters are randomly generated inside the value range instead of mutation from a hand-tune set of parameters. It reduced the human work as well as generating evolutional results varied a lot in different experiences. Through experiments which took about 40 minutes each, we achieved several high-performance sets of gait parameters which differ a lot from each other. These gait parameter sets were among the fastest forward gaits ever developed for the same robot platform.

In the future, we will compare different high-performance gait parameters and analyze the dynamics model of the robot and in an attempt to get a deeper sight into the relation between parameter and its performance. After that, we will be able to generate more effective gaits in less learning time. Through analysis, we find that the gait actually executed by robot differ significantly from the one we design. There are several reasons accounting for that. The most important one is the interaction with environment prevents the implement of some strokes of robot legs. Although with learning approach, factors that cause the difference between actual gait and planned gait do not have to be taken into consideration. However, we assume that if the planned gait and actual gait can conform with each other, Aibo will walk more stable and fast. In order to solve the problem, the analysis of dynamics between the robot and the environment is necessary. In this gait learning procedure, we only evolve fast forward gait and choose forward speed as the fitness. Later on, we will try to learn effective gaits in other directions, for example, gaits for walking backward, sideward and turning. We also consider exploring optimal omnidirectional gaits. With gaits working well at all directions, robots will be able to perform more flexibly and reliably.

Acknowledgements

The authors gratefully acknowledge the contribution of the team members of the sharPKUngfu Legged Robot Team. This work was supported in part by the National Science Foundation of China (NSFC) under Contracts 60674050, 60528007, and 60635010, by the National 973 Program (2002CB312200), by the National 863 Program (2006AA04Z258) and 11-5 Project (A2120061303).

References

1. Eberhart, R., Kennedy, J.: Particle Swarm Optimization. In: Proceedings of the IEEE Conference on Neural Network, Pelfh., Australia, pp. 1942–1948 (1995)
2. Papadopoulos, D., Buehler, M.: Stable running in a quadruped robot with compliant legs. In: Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, pp. 444–449 (2000)
3. Holmes, P., Full, R.J., Koditschek, D., Guckenheimer, J.: The dynamics of legged locomotion: Models, analyses, and challenges. SIAM Review 48(2), 207–304 (2006)
4. Raibert, M.H.: Legged robots that balance. MIT Press, Cambridge (1986)
5. Collins, S., Ruina, A., Tedrake, R., Wisse, M.: Efficient bipedal robots based on passive dynamic walkers. Science 307, 1082–1085 (2005)

6. Krasny, D.P., Orin, D.E.: Generating high-speed dynamic running gaits in a quadruped robot using an evolutionary search. IEEE Transactions on Systems, Man, and Cybernatics - Part B: Cybernetics 34(4), 1685–1696 (2004)
7. Hornby, G.S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O.: Autonomous evolution of gaits with the Sony quadruped robot. In: Banzhaf, W., et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, USA, vol. 2, pp. 1297–1304. Morgan Kaufmann, San Francisco (1999)
8. Kim, M.S., Uther, W.: Automatic gait optimisation for quadruped robots. In: Australasian Conference on Robotics and Automation (2003)
9. Röfer, T., Burkhard, H.D., Düffert, U., Hoffmann, J., Göhring, D., Jüngel, M., Lötzsch, M., Stryk, O.v., Brunn, R., Kallnik, M., Kunz, M., Petters, S., Risler, M., Stelzer, M., Dahm, I., Wachter, M., Engel, K., Osterhues, A., Schumann, C., Ziegler, J.: GermanTeam RoboCup 2004. Technical report (2004)
10. Röfer, T., Burkhard, H.D., Düffert, U., Hoffmann, J., Göhring, D., Jüngel, M., Lötzsch, M., Stryk, O.v., Brunn, R., Kallnik, M., Kunz, M., Petters, S., Risler, M., Stelzer, M., Dahm, I., Wachter, M., Engel, K., Osterhues, A., Schumann, C., Ziegler, J.: GermanTeam RoboCup 2005. Technical report (2005)
11. Ian Stewart. Nature's Numbers (1996)
12. Reynolds, C.: Flocks, herds, and schools:Adistributed behavioral model. Comp. Graph. 21(4), 25–34 (1987)
13. Angeline, P.: Using selection to improve particle swarm optimization. In: Proc. IEEE Int. Conf. Evolutionary Computation, pp. 84–89 (1998)
14. Naka, S., Genji, T., Yura, T., Fukuyama, Y.: Hybrid particle swarm optimization based distribution state estimation using constriction factor approach. In: Proc. Int. Conf. SCIS & ISIS, vol. 2, pp. 1083–1088 (2002)
15. Alexander, R.M., Jayes, A.S.: A dynamic similarity hypothesis for the gaits of quadrupedal mammals. J. Zoology Lond. 201, 135–152 (1983)
16. Alexander, R.M., Jayes, A.S., Ker, R.F.: Estimates of energy cost for quadrupedal running gaits. J. Zoolog. 190, 155–192 (1980)