Legged Robot Gait Locus Generation Based on Genetic Algorithms

Kai Xu[1], Xiao-ping Chen[1], Wei Liu[2], and Mary-Anne Williams[3]

[1] Department of Computer Science, The University of Science and Technology of China, HeFei 230027, China
[2] The University of Western Australia, Perth, Australia
[3] The University of Technology, Sydney, Australia

Abstract. Achieving an effective gait locus for legged robots is a challenging task. It is often done manually in a laborious way due to the lack of research in automatic gait locus planning. Bearing this problem in mind, this article presents a gait locus planning method using inverse kinematics while incorporating genetic algorithms. Using quadruped robots as a platform for evaluation, this method is shown to generate a good gait locus for legged robots.

1 Introduction

Legged robots have a wide range of applications, and are used for many tasks that cannot be accomplished by wheeled ones. A legged robot is a typical multivariable, tight coupling, non-linear and time-variant kinetic system, subjected to the gravitational field. The sophisticated structure necessary for flexible gaits and the dynamic balance required by stable gaits, make the design and realization of real-time robotic walking control systems a highly challenging and complex task. Currently, one of the main challenges is that legged robots are typically controlled by multiple motors and therefore have a very high degree of freedom, which increases the difficulty of gait design even further.

Theoretically, optimal gait design can be achieved through dynamics analysis if a legged robot can be described using dynamical models. However, legged robots' dynamical states are often too complex. In particular, when dealing with the whole robot not just the individual parts it becomes a high-dimensional gait locus planning problem. Such problems are exceedingly difficult to solve for most robots, except for extremely simple ones. Therefore, current practical legged robots generally resort to Inverse Kinematics [1] for gait design; converting legged robot's walking design into a gait locus plan. Normally, robotic gait design is inspired by and borrows from animal behaviors and walking patterns, including both static and dynamic gaits. However, for a specific robotic platform, not much research has been done to investigate how to select a particular gait locus, and which gait locus is more appropriate, feasible and suitable to the robot's physical structure and the specific recognition problem. In reality, expert experience is often used to compensate the gait imperfection due to incomplete information.

Furthermore, a lot of testing and adjustment are needed before some reasonable gaits can be generated. However, when walking conditions change (e.g. mass focus variation due to the change of postures, raised requirement on walking speed and flexibility), a new round of tedious manual testing needs to be carried out. The problem becomes acute when key motor information affecting the gaits for the specific robot platform is not available, such as motor operation and feedback parameters, since it is impossible to verify if the motor is following the designed gait locus and taking the appropriate moves. Kohl and Stone [2], Düffert and Hoffman [3] report strategies for such situations. However, their work does not consider how to generate the gait loci, instead it is based on fixed gaits. Our work in this paper, in contradistinction, considers how to obtain the optimal gait locus under various conditions.

In this paper, we propose a Genetic Algorithm based approach for legged gait locus generation; in particular the situations when the physical structure and motor information of the robots are missing. In such situations, robot kinematics is no longer applicable, therefore, our approach uses the geometric analysis based inverse kinematics, using genetic algorithms to adaptively generate stable gait loci. Experiments show that our approach can achieve reasonably effective results, and also reduce the testing and adjustment phase, which makes it applicable to practical legged robot motion analysis and control.

The rest of the paper is organised as follows: in Section 2, we propose the idea of legged robot gait locus generation based on inverse kinematics; in Section 3, we develop a Genetic Algorithm based gait locus generation algorithm; in Section 4, the Sony AIBO[1] ERS-7 a quadrupled legged robot, is used as the experimental platform to compare the proposed approach with the commonly used rectangular gait. Experiment results are presented in this section as well. Section 5 concludes the paper with some outlook to the future work.

2 Generating Gait Locus for Legged Robots

2.1 Robot Inverse Kinematics

Robot inverse kinematics calculate the joint parameters when given the limbs' geometric parameters and the posture of the limbs relative to the robot's coordinate system. An inverse kinematics problem may have many, unique or no solutions. To illustrate some features of the inverse kinematics, consider the simple manipulator with only two joints as shown in Figure 1(a).

According geometric analysis, we have,

$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \tag{2.1}$$

$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \tag{2.2}$$

An inverse kinematics problem is: given x and y to determine θ_1 and θ_2.The solution is illustrated in Figure 1(b), $\theta_2$in the figure can be calculated according

[1] Sony AIBO Robots. http://www.aibo.com

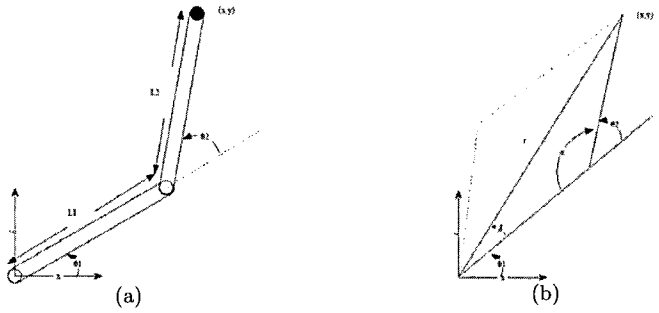

Fig. 1. Inverse kinematics of a manipulator with 2 joints

to $r = \sqrt{x^2 + y^2}$ and the Law of Cosines,

$$\theta_2 = \pi \pm \alpha \qquad \text{where} \qquad \alpha = \arccos(\frac{l_1^2 + l_2^2 - r^2}{2l_1 l_2}) \tag{2.3}$$

When $\alpha \neq 0$, θ_2 has two different values. The dotted line in Figure 1(b) shows the other value.Then we can work out θ_1 according to every possible value of θ_2. Now we have

$$\theta_1 = \alpha \tan 2(y, x) \pm \beta \qquad \beta = \arccos(\frac{r^2 + l_1^2 - l_2^2}{2l_1 r}) \tag{2.4}$$

where the sign before β is in accordance with that of α.

Typically, inverse kinematics problems can be classified into two categories, closed-form solutions and numerical solutions.Most robots have closed-form solutions.Interested readers are referred to Craig [4] for some common ways of solving such problems.

2.2 Idea of Generating Curve for Gait Locus

A legged robot's gait locus is the limb movement from one position to another with certain speed and acceleration, in a specific space within a certain time frame. According to robot inverse kinematics, we know that the joint parameters can be determined so long as the expected posture of the limbs is given,therefore, robot gait locus can be planned based on inverse kinematics in *Cartesian coordinate space*.When Cartesian coordinate space is used for gait locus planning, it is often assumed that the motors can generate enough power for joints acceleration and deceleration,simple planning can lead to great differences between the real-time response of the motor controlled limb and the designed trajectory. Stronger and Stone[5] have shown that such differences greatly affect the robot's real-time performance Figure 2.

Through experiments, we can see that the gait locus in Cartesian Space is a curve made of discrete points of the limb position. Very often the curve is of

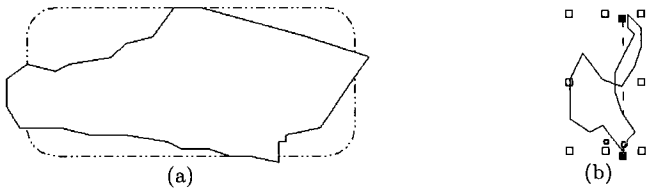

Fig. 2. The contrast between the AIBO locus planning curve (dashed) and actuator response curve (solid line)

irregular shape. Mathematically, *interpolation* and *curve fitting* are two dominant approaches in describing irregular curves. During the gait locus planning, the real concern is the robot's operating performance rather than the actual gait shape, since the sampling points may not be exactly on the optimal gait locus, there are certain errors. Interpolation that requires the curve to go through every data point will not work well, the use of a more general curve fitting method that approximates all sampling points may avoid such errors through optimization so long as the sampling points are close to the optimal gait locus.

The general principle of a curve fitting method is to obtain a group of sampling points through manual observation, assume the points $\{(x_i, y_i)\}_{i=0}^m$ are for fitting curve $y = f(x)$.Here, the analytical expression of $y = f(x)$ is not available beforehand.Meanwhile, no matter how the discrete set of data is obtained, be it through observation, testing or generation,errors are inevitable. Therefore, eventually rather than asking the curve $\phi(x)$ to exactly go through these points,we need to minimize $\sum_{i=0}^m (\phi(x_i) - y_i)^2$,the squared error between the discrete points $\{(x_i, \phi(x_i))\}_{i=0}^m$ on the curve $\phi(x)$ and the sampling points $\{(x_i, y_i)\}_{i=0}^m$ obtained through observation. The $\phi(x)$ so obtained is then called the fitting curve $y = f(x)$ on Φ.The most commonly used fitting curve is obtained on $\Phi = P_n$,where P_n is a n degree polynomial, where the n-th term's coefficient is non-zero.Curves obtained on P_n are known as polynomial fitting curves.

Given a set of discrete points $\{(x_i, \phi(x_i))\}_{i=0}^m$, take a P_n with coefficients $1, x, x^2, \ldots, x^n$, the polynomial function is then

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{2.5}$$

To minimize the squared error,

$$Q(a_0, a_1, \ldots, a_n) = \sum_{i=0}^{m} (P(x_i) - y_i)^2 = \sum_{i=0}^{m} (a_0 + a_1 x_i + \cdots + a_n x_i^n - y_i)^2$$

onP_n, $a_0, a_1, \ldots, a_n$ should meet

$$\frac{\partial Q}{\partial a_k} = 2 \sum_{i=0}^{m} x_i^k (a_0 + a_1 x_i + \cdots + a_n x_i^n - y_i) = 0 \qquad (k = 0, 1, \ldots, n)$$

After simplifying the above, we have

$$a_0\sum_{i=0}^{m}x_i^k+a_1\sum_{i=0}^{m}x_i^{k+1}+\cdots+a_n\sum_{i=0}^{m}x_i^{k+n}=\sum_{i=0}^{m}x_i^k y_i \qquad (k=0,1,\ldots,n) \quad (2.6)$$

Formula (2.6) is a group of $n+1$ degree linear equations on $a_0, a_1, \ldots, a_n$. The matrix presentation for the linear equations is

$$\begin{pmatrix} m+1 & \sum_{i=0}^{m}x_i & \sum_{i=0}^{m}x_i^2 & \cdots & \sum_{i=0}^{m}x_i^n \\ \sum_{i=0}^{m}x_i & \sum_{i=0}^{m}x_i^2 & \sum_{i=0}^{m}x_i^3 & \cdots & \sum_{i=0}^{m}x_i^{n+1} \\ \sum_{i=0}^{m}x_i^2 & \sum_{i=0}^{m}x_i^3 & \sum_{i=0}^{m}x_i^4 & \cdots & \sum_{i=0}^{m}x_i^{n+2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sum_{i=0}^{m}x_i^n & \sum_{i=0}^{m}x_i^{n+1} & \sum_{i=0}^{m}x_i^{n+2} & \cdots & \sum_{i=0}^{m}x_i^{2n} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{m}y_i \\ \sum_{i=0}^{m}x_i y_i \\ \sum_{i=0}^{m}x_i^2 y_i \\ \vdots \\ \sum_{i=0}^{m}x_i^n y_i \end{pmatrix} \quad (2.7)$$

In (2.7), m is the number of discrete data points, n is the degree of the fitting curve. Let

$$A=\begin{pmatrix} m+1 & \sum_{i=0}^{m}x_i & \sum_{i=0}^{m}x_i^2 & \cdots & \sum_{i=0}^{m}x_i^n \\ \sum_{i=0}^{m}x_i & \sum_{i=0}^{m}x_i^2 & \sum_{i=0}^{m}x_i^3 & \cdots & \sum_{i=0}^{m}x_i^{n+1} \\ \sum_{i=0}^{m}x_i^2 & \sum_{i=0}^{m}x_i^3 & \sum_{i=0}^{m}x_i^4 & \cdots & \sum_{i=0}^{m}x_i^{n+2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sum_{i=0}^{m}x_i^n & \sum_{i=0}^{m}x_i^{n+1} & \sum_{i=0}^{m}x_i^{n+2} & \cdots & \sum_{i=0}^{m}x_i^{2n} \end{pmatrix} B=\begin{pmatrix} \sum_{i=0}^{m}y_i \\ \sum_{i=0}^{m}x_i y_i \\ \sum_{i=0}^{m}x_i^2 y_i \\ \vdots \\ \sum_{i=0}^{m}x_i^n y_i \end{pmatrix} X=\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

Then 2.7 can be expressed as a n degree linear algebra equations $AX = B$. Solving this group of linear equations will give use the coefficients $a_0, a_1, \ldots, a_n$ of the fitting polynomial function $P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$. This is the fitting curve for the give set of discrete data points which belongs to the unshaped gait locus.

2.3 Gait Locus Generation

At first, to generate the gait locus of a legged robot based on its physical form and structure, the relation between joint angles and limb stance is established by Inverse Kinematics. Given a legged robots walking speed (or velocity) V and the duration of the operation cycle, T, the expected walking distance can be calculated according to this simple formula $S = V \times T$. Therefore, we can work out the possible limb movement space during a walking cycle, which is a rectangular prism, as shown in Figure 3.

Now, we can get the sample points for the gait locus from the space and classify them according to the movement phases of robot limb, and obtain the fitting curve for each points group. The generated curve is a possible gait locus. Due to the inability to conduct verification, each point in space may be on the optimization gait locus. Based on this idea, we must generate a curve aggregate Φ which could include an infinite number of elements and then select an optimized gait locus which is virtually impossible to realize. Bearing this problem in mind,

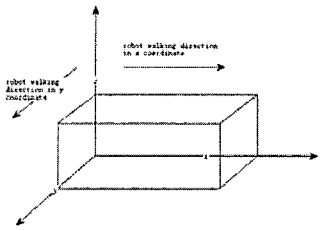

Fig. 3. The limb movement space of legged robot

traditional optimization methods can be adopted, however using traditional optimization methods(analysis optimization, enumeration optimization etc.) gives rise to many potential problems such as local optima traps, difficulties if the objective function is not continuous and differentiable, and low efficiency etc. The optimization methods of machine learning such as *Reinforcement Learning* and *Genetic Algorithms* are more efficient.

The generated fitting curve of a gait locus can not be evaluated before being applied on the robot, the fitting curve aggregate is infinite, and the generated fitting curve needs to be evaluated in time to help optimize the gait locus globally. Therefore, a global optimization method is needed to avoid local optima traps. In addition, it needs to be efficient and stable, especially in the face of incomplete information. Genetic Algorithms (GA) [6, 7] are based on natural selection and genetic mechanisms. The mechanism of biological evolution is simulated in computer algorithms for optimization. The search space (solution domain) is mapped to a genetic space (i.e. candidate solutions are represented as a vector of chromosomes). A population consists of n so *chromosomes*. Every chromosome is evaluated according to a predefined function, i.e. the *fitness function*. According to the "survival of the fittest" principle, the best candidate is selected and the less fit are abandoned in the next generation so that gradually the population converges to the optimal solution.

As an optimization method, GA exhibits the following features:

(1) It has "memory" during searching.
(2) It has fewer requirements on the objective function. The objective function only needs to be defined, but need not necessarily be continuous or differentiable.
(3) It can simultaneously search many solution domains thereby greatly reducing the chance of being trapped by local optimum.
(4) It has the potential of running in parallel, and is therefore suitable for optimizing large scale problems.

According to Suzuki (1995)'s[8] analysis using Markov chains, Simple Genetic Algorithm(SGA) can stochastically converge to the optimal solution. Therefore, GA is suitable for solving the optimization problem we described above. Because GA obtains the optimization solution stochastically, in this paper we use SGA

guided by the "survival of the fittest" strategy, that is, during selection, the SGA with the best solution is kept. Using this method, we can get a suboptimal solution surely and still obtain the optimization solution efficiently using stochastic methods.

3 Gait Locus Generating Method Based on Genetic Algorithms

3.1 Individuals of Generated Curve

The limb's movement space is 3 dimensional (3D), and therefore, the gait locus generation should be carried out in a 3D space. According to the projection theory in 3D space, we first carried out gait locus generation on the limb and the projection plane, respectively, and then synthesize the results to obtain the 3D curve. This effectively avoids a direct description of a 3D curve. In addition, it is very easy to change the locus on one plane without affecting the other. The actual gait locus of a legged robot can be very complex. Therefore, in order to ensure the generated curve is as close to the real locus as possible, the generated curve and each section must be able to adapt flexibly in and only in the limb movement space. If we directly apply genetic manipulation on parameters of the generated curves, the results may not be in the movement space. Therefore, we program genetic operations directly on the discrete data points instead. Moreover, because the 3D curve is synthesized from the projections on 2D planes, we only need to sample and process the corresponding data points Ψ on the two 2D planes $Y-Z$ and $X-Z$ separately:

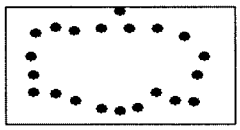

Fig. 4. Illustration of discrete data points

As shown in Figure 4, it is required that all data points $p(x, y) \in \Psi$ are in the 2D projection planes of the limb movement space. Therefore, in the genetic algorithm, an individual representing the information about a gait locus is the discrete data points for the curve generation. The length of an individual is then determined by the number of data points. The initial point set $\Psi_i = \{p_1, p_2, \ldots, p_n\}$ can be obtained randomly from a uniform distribution, where n is the number of data points, i is the number of individuals in the genetic algorithm.

3.2 Individual Encoding, Genetic Operators, Fitness Function Design and Convergence

According to Holland's modeling theory Genetic Algorithms, the shorter the encoding binary string, the more global the information maintained. However, the parameters for the gait locus generation are relatively independent, have different value ranges in the Real Number space, and vary in the requirement on the accuracy and the distribution of the data points on the 2D plane. As a result we use real numbers for the encoding of individuals.

Our crossover operator is shown as Formula 3.1

$$child\_val = father\_val \times father\_fac + mother\_val \times (1 - father\_fac) + noise \tag{3.1}$$

where $father\_val$ and $mother\_val$ are the values for the parents respectively, $fahter\_fac$ is the father's gene's ratio that will pass on to the children according to the parent's fitness calculation. The mother's ratio is $(1 - father\_fac)$, and $noise$ is a uniformly distributed error compensating the numerical difference between the parents, as shown in the formula below:

$$noise = (random - 0.5) \times 2 \times |father\_val - mother\_val|$$

$random$ is a random value between 0 and 1. The main purpose of is to reduce the effects of errors in the evaluation of the parents' fitness.
There are two types of mutation operators, one based on uniform distribution and the other based on Gaussian distribution.

The mutation operator based on uniform distribution is:

$$value = value + (random - 0.5) \times factor \times (max\_value - min\_value) \tag{3.2}$$

The mutation operator based on Gaussian distribution is:

$$\begin{aligned} value = value + factor \times (max\_value - min\_value) \times \sqrt{-2 \times log(random)} \\ \times \sin(2\pi \times random) \end{aligned} \tag{3.3}$$

$random$ is a random value between 0 and 1, $factor$ is the coefficient taking into account the effect of the different value ranges,max_value and min_value are the maximum and minimum of $value$ respectively.The uniform distribution based mutation is more applicable when all individuals are obtained randomly. Therefore, no initial values are required in this case. The Gaussian distribution based mutation, on the other hand, is more useful when there are better initial values.

In addition to walking speed, stability is also a requirement for developing an effective walk for legged robots. Therefore, the fitness function is defined as:

$$Fitness = V - \phi - \varphi - \delta \tag{3.4}$$

where $Fitness$ is the fitness value, V is the average linear velocity on the specified trajectory, ϕ is the error during walking φ is the vibration factor, δ is the factor that accounts for the influence on the robust operation of other equipments.

Convergence: According to the traditional GA convergence conditions, we set our convergence conditions as a fitness threshold, the number of iterations and the changing ratio of the current fittest individuals, when one of these conditions is met, the convergence of algorithm has been achieved.

3.3 The Program Flow of the Algorithm

The algorithm is composed of the following steps:

Step 1: Determine the range of the limb movement space;
Step 2: Given the experimental data, determine the fitness function;
Step 3: Randomly obtain the parameters for describing the key points at different walking stages in n individuals;
Step 4: Apply gait locus generation for every individual in order to obtain the gait locus function in Step 5; Use mutation on the individuals that cannot generate a gait locus to ensure that all individuals can generate a gait locus; initialize the selection pool of the fitter individuals;
Step 5: Use inverse kinematics such that the robot follows every individual's gait locus under the same initialization in the same environment. Data obtained during these experimental walking are used to find the individual's fitness according to the fitness function;
Step 6: Evaluate the convergence conditions, when one of them has been achieved, turn to Step 8, otherwise, go to Step7;
Step 7: Compare the fitness values of the individuals in this generation with the ones in the selection pool, keep the fitter ones or record the no change frequency of it; then apply cloning, crossover, and mutation to obtain the new generation, go to Step 3;
Step 8: The program terminates, the optimal gait locus under the current initialization and the specified convergence conditions is therefore obtained.

Since the stability of the gaits during gait locus generation must be taken into account, the gait locus generation problem itself is a constraint based optimization problem. Many ways of dealing with constraints are available in Genetic Algorithms. In this paper, the main issues of concern are the generation of the initial population and the selection from the children generation. In the above algorithm, the key points for describing the gait locus of each individual is obtained through random generation first, and then the individual is tested to see if it can generate a valid gait locus. The same testing on individuals is also necessary during crossover and mutation.

4 Experiments, Result Analysis and Application

4.1 The Experiment Platform and Environment

The standard robot platform of the RoboCup[9] 4-legged League - Sony AIBO ERS-7 (as shown in Figure 5) is used in our experiments to test and evaluate

the gait locus generation algorithm proposed in this paper. The experimental environment adopted is that used by Düffert and Hoffmann [3], which is also shown in Figure 5.

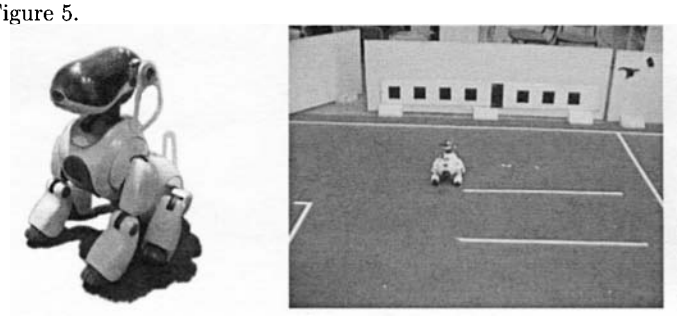

Fig. 5. Quadruped robot - Sony AIBO ERS7 and the experiment environment

4.2 Experiment Design and Results

Fixed Rectangular Gait Loci vs. Generated Gait Loci. While keeping all other parameters consistent, we compared the common rectangular gaits with the proposed adaptive gaits. According to its own sense of distance, the same robot is used on a straight line walking for distances between $100cm$ and $300cm$. Over a 3 hour period we obtained the results given in Table 1, and we also carried out the same experiments on the official carpet used in RoboCup 4 Legged League, and those results are shown in Table 2.

Common Carpet	Initial Speed	Finishing Speed
Rectangular Gait	$250mm/s$	$306mm/s$
Generated Gait	$250mm/s$	$350mm/s$

Table 1. The straight walking experiment results of AIBO on common carpe

Official Carpet	Initial Speed	Finishing Speed
Rectangular Gait	$250mm/s$	$310mm/s$
Generated Gait	$250mm/s$	$370mm/s$

Table 2. The straight walking experiment results of AIBO on official carpet

The results demonstrate that the GA generated curve gait can achieve higher finishing speeds than the rectangular gait.

Detailed Experiments for Generated Gait Locus We increase the sections of the generated curve to 8, to repeat the above experiments. The result is shown below in Table 3.

Official Carpet	Initial Speed	Finishing Speed
Generated Gait(4 Sections of Quadratic Curves)	$250mm/s$	$392mm/s$
Generated Gait(8 Sections of Quadratic Curves)	$250mm/s$	$475mm/s$

Table 3. The straight walking experiment results of AIBO on official carpet

The experiment shows that by increasing the number of sections, we can get closer fitted gait locus to ensure stable and fast walking for legged robots. In addition, we also tried to use more sections for the experiment. However due to the limitation of Sony AIBO ERS-7's computation power and mechanical parts, experiments on using more sections are not conducted.

Experiments on the adaptiveness of the Gait Locus In this section, we will apply the algorithm on side-way and multi-direction moves. The multi-direction moves followed the experimental design in [2] and the results are:

Official Carpet	Rectangular Gait	Generated Gait
Initial Speed of Side-way Move	$152mm/s$	$152mm/s$
Finishing Speed of Side-way Move	$302mm/s$	$346mm/s$
Initial Speed of Multi-direction Move	$250mm/s$	$250mm/s$
Finishing Speed of Multi-direction Move	$350mm/s$	$410mm/s$

Table 4. The straight walking experiment results of AIBO on official carpet

During side moves, although AIBO's legged movement direction changed from forward to horizontal, the proposed algorithm is still applicable. As we can see, faster walking speed is again achieved as compared to the rectangular gait.During multi-direction movement, AIBO is no longer just following one fixed direction. Instead, it combines forward, side-way moves with rotation concurrently. The experiment results indicate that the generated curve gaits again outperforms the rectangular ones.

4.3 Results Summary

From the above three sets of experiments, the proposed algorithm for gait locus generation has better performance for different movement in different environment, because it does generate the gait loci in global and the designed fitness function includes not only velocity but also other realtime information about walking which help itself to be convergence efficiently. The results also show the algorithm is adaptive and robust under various circumstances for relatively stable and fast walking of legged robots.

The researchers of UNSW also used the AIBO to work in the same field [2]. They optimize the gait locus by relative fixed gait. In 2004, they got $340mm/s$, and achieved $430mm/s$ in 2005. This suggests that our algorithm which can generate gait loci globally and find the optimal gaits stochastically is more effective

[2] RobocupThesis2005_weiming.pdf.
http://www.cse.unsw.edu.au/~robocup/2005site/reports05/

and efficient. Using this algorithm, the WrightEagle team of the University of Science and Technology of China successfully entered the quarter finals of the RoboCup 2006 competition. During the competition, it took the robots 40 mins to achieve a very good $450mm/s$ walking speed.

5 Conclusion and Future work

Bearing how to describe the walking gait locus of legged robot in mind, this article proposes the idea of using curve fitting to generate gait locus through inverse kinematics. Based on Genetic Algorithms, it can find the optimal solution in the curve aggregate that approaches an infinite number of elements. As demonstrated by experiments, the algorithm can be used by legged robots to generate gait loci autonomously, after iterated optimization, the optimal or suboptimal gait locus is obtained.

In future, we will investigate the gait distribution in the 3D movement space for searching the optimal gait locus and look for methods that will reduce the computation cost during the optimization. Moreover, we will analyze more optimization methods to determine those that will lead to even better results.

Acknowledegment

This project is supported by grant 60275024 of the National Natural Science Foundation of China and the Australia China Project CH050103 from the Australia Department of Education, Science and Training (DEST).

References

1. R. M. Murray, Z. X. Li, and S. S. Sastry. A Mathematical Introduction to Robotic Manipulation Boca Raton, FL: CRC Press, 1994.
2. N. Kohl and P. Stone. Machine learning for fast quadrupedal locomotion.In The Nineteenth National Conference on Artificial Intelligence,pages 611C616, July 2004.
3. U. Düffert and J. Hoffman. Reliable and Precise Gait Modeling For a Quadruped Robot. In RoboCup 2005: Robot Soccer World Cup IX, Lecture Notes in Artificial Intelligence. Springer, 2005.
4. John J. Craig. Introduction to Robotics: Mechanics and Control, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1989
5. Daniel Stronger and Peter Stone. A model-based approach to robot joint control. In RoboCup-2004: Robot Soccer World Cup VIII, pages 297C309. Springer Verlag, Berlin, 2005.
6. DeJong, Kenneth. Adaptive System Design: A Genetic Approach. IEEE Transactions on Systems, Man and Cybernetics 10 (9), 566-574,1980.
7. Tom M. Mitchell, Carnegie-Mellon University. Machine Learning. McGraw-Hill Inc.,1997
8. J. Suzuki. A Markov chain analysis on simple genetic algorithms. IEEE Trans. Syst., Man., Cybern., vol. 25, no. 4, pp. 655-659, 1995.
9. H. Kitano, Y. Kuniyoshi, I. Noda, M. Asada, H. Matsubara, E. Osawa. RoboCup: A challenge problem for AI. AI Magazine 18 (1) (1997) 73-85.