28

Collaborative Localization and Gait Optimization of SharPKUngfu Team

Qining Wang, Chunxia Rong, Guangming Xie and Long Wang
Intelligent Control Laboratory, College of Engineering, Peking University
China

1. Introduction

In this chapter, we introduce the recent progress of sharPKUngfu Team which participates in the RoboCup Four-Legged League since 2004. sharPKUngfu Team is a robot soccer team from Peking University, China. In July 2005, we got the third place in the RoboCup China Open. In June 2006, our sharPKUngfu Team has participated in the technical challenge of RoboCup 2006. In this event, our *Medal Awarding* challenge got the eighth place in the Open Challenge. In October 2006, we got the champion in the RoboCup China Open 2006, both in soccer competition and technical challenge. In July 2007, we participate in the RoboCup 2007 and got the fourth place in the technical challenge. Our research in robot soccer focuses on robot vision, multi-robot cooperation strategy, collaborative localization in dynamic environment, quadruped gaits optimization and intelligent behavior.

We focus this chapter on localization and gait optimization which are the fundamental parts in soccer robotics. Recently, we successfully apply self-learning image-retrieval approach and collaboration in self localization in robot soccer. This improvement eliminates the problems of image-retrieval method and collaboration mentioned in previous research. By using this approach, robots can play soccer under more natural conditions towards real human soccer environment. We organize the localization part as follows. At first, a brief overview of current self-localization approaches is presented. Secondly, we introduce the human cognition inspired localization with self-learning experience. Specific algorithms for image features collection and self-learning process are described. Then, the dynamic reference object based method for collaborative localization is demonstrated in detail. Experimental results in real robot soccer are shown in the end. We also discuss current challenges and future works of localization in soccer robotics.

How to get high-speed walking and running gaits is another problem in soccer robotics. Different to existing literature which uses Genetic Algorithms (GA) based gait optimization methods, we present the implementation of Particle Swarm Optimization (PSO) in generating high-speed gaits for a quadruped robot, specifically the Aibo, which is the commercial robot made in Sony. PSO has been proven to be effective in solving many global optimization problems and in some areas outperform many other optimization approaches including Genetic Algorithms. In this part, at first, we overview the basic PSO and Adaptive PSO (APSO) with comparison to other optimization approaches. After that, with the

Source: Robotic Soccer, Book edited by: Pedro Lima, ISBN 978-3-902613-21-9,
pp. 598, December 2007, Itech Education and Publishing, Vienna, Austria

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 with onboard processor. In robot experiments, we achieved an effective gait faster than previous hand-tuned gaits, using Aibo as the test platform.
Our progress of intelligent behaviors in real soccer competition is described briefly in the end of the chapter. All the details about real robot experiments and how to use the debugging tools can be found in (Wang, 2006b) or the official website of sharPKUngfu team.

2. Multi-Robot Collaborative Localization

In soccer robotics, for example in the RoboCup, several probabilistic methods for global self-localization have been implemented in various teams from different leagues (eg. Fox et al., 2000; Röfer et al., 2003; Schmitt et al., 2002). However, most current localization approaches used in robot soccer depend on standard landmarks and static environment. On the move to real human soccer conditions, current localization approaches in robot soccer seem not enough. In the human soccer, there are two aspects which may inspire the self localization of mobile robot systems. On the one hand, the features surrounding the soccer field may be exploited as the sensory information in probabilistic approaches. Inspired by the features, some robot systems have applied image-retrieval approach in localization (Wolf et al., 2005; Wang et al., 2006a, 2006b). There are several limitations by using such image-retrieval method. First, the computational cost of this approach is expensive. Besides, the requirement of building a huge database is not so practical, especially in the complex environment. On the other hand, collaboration among the robot team, which is only used in the strategy modules of current robot soccer teams, may be considered as another part of the sensory information. Previous research in localization has proven that the cooperation in self-localization among multiple robots has impressive performance in real robot systems (see Arkin & Balch, 1998 for overview). The limitation of such robot systems is that the robot needs to identify other one precisely. It is quite difficult to perform collaborative localization for robots dealing with situations where they can detect but not identify other robots. In addition, taking the uncertainty of sensors into account, the result of detecting individual robot is not so reliable. Those limitations of the approach make it not so applicable for real robots localizing in complex environments.
To apply image-retrieval approach and collaboration in self localization in robot soccer, we focus our work on two aspects. In the image-retrieval system, an efficient method of calculating image features is implemented. To simulate real human soccer conditions, colourful advertisement is placed around the field which is similar to the real soccer field. Our method divides one image into several parts to calculate features respectively. To construct the image feature database, the robot learns the relationship between images and positions autonomously. This improvement eliminates the problems of image-retrieval method mentioned in previous research (Wolf et al., 2005). By using the efficient approach, robots can play soccer under more natural conditions towards real human soccer environment. In addition, to introduce collaboration among team members in localization module, we integrate the image-retrieval approach with collaboration. In real robot soccer, it

may not so easy to identify the specific robot who is nearby, especially in the dynamic environment of soccer competitions. In human soccer, players can localize in the field by the distance to ball and team members. Inspired by this technique, a dynamic reference object based method is implemented in the real robot competition. This collaborative approach can improve the self localization in the field with less artificial landmarks. Positive impact on localization through our approach is shown in experiments using the Sony Aibo ERS-7 robot.

2.1 Landmark & Experience Based Markov Localization

To improve the probabilistic approach, we created an efficient method to construct environment features as experience, which is collected by the robot autonomously. By using such experience, robot can localize in the field with less artificial landmarks towards real human soccer environment.

Most robot localization systems use landmarks as the tool to predict and correct current positions of mobile robots. For example, (Röfer et al., 2003) proposed and improved the landmark based Markov localization. In this approach, the current position of the robot is modelled as the density of a set of particles which are seen as the prediction of the location. Initially, at time t, each location l has a belief:

$$Bel_t(l) \leftarrow P(L_t^{(0)}) \tag{1}$$

To update the belief of robot possible location, at first, this approach uses the new odometry reading o_t:

$$Bel_t(l) \leftarrow \int P(l \mid o_t, l^-) Bel_t(l^-) dl^- \tag{2}$$

If robot receives new sensory information s_t, then it updates the belief with α being the normalizing constant:

$$Bel_t(l) \leftarrow \alpha P(s_t \mid l) Bel_t(l) \tag{3}$$

Considering the mobile robot with complex motions, let the geometric centre of robot body as the location vector ϕ, which contains the x/y- global coordinates of the centre point. Another vector θ is defined as the heading direction. Then every particle is updated by the motion model as follows when the robot moves:

$$\phi_t = \phi_{t-1} + \Delta_t \tag{4}$$

where Δ_t represents the displacement in x/y coordinates and heading direction.

To implement image retrieval system in Markov localization, we divide the sensory update into two parts: updating position probability by landmark perception and experience matching. If the robot recognizes landmarks well enough, landmark based sensor model will update the belief of position with the new landmark reading s_t:

$$Bel_t(\phi_t) \leftarrow \beta P(s_t \mid \phi_t) Bel_t(\phi_t) \tag{5}$$

where β is a normalizing constant. It is natural that the robot may miss some landmarks with real-time recognition for a period. Thus, we set $N_1(t)$ which is the amount of lasting frames of having no landmark perception from t as a condition to activate the experience system. If $N_1(t)$ is great enough, the experience based sensor model will update the probability as follows with e_t being the new reading experience with γ being the normalizing constant different from β:

$$Bel_t(\phi_t) \leftarrow \gamma P(e_t \mid \phi_t) Bel_t(\phi_t) \tag{6}$$

2.2 Experience Construction

The feature that is exploited from images with no landmark in the view, and represents the invariant character of images obtained at positions where collisions and other negative effects more likely occur is defined as *Experience*. In our system, we make the robot to collect the image features autonomously, which is named self-learning experience. The experience contains image features in divided areas and the whole image respectively. In the following paragraphs, we introduce our efficient method to construct experience in detail.

(a) Image Features in Divided Areas

In our method, we divide one image which is obtained by the robot camera into six parts. First, image features including average colour value $f_{i,j}$ and colour variance d_i in the divided areas are calculated by the following equations:

$$f_{i,j} = \frac{\sum_{x,y} M[y][j][x]}{N_i}; \{j = 0,1,2; i = 1,2,3,4,5,6\} \tag{7}$$

where $f_{i,j}$ is the average value in the colour channel j of the area i. $M[y][j][x]$ represents the value in the colour channel j at the position (x, y) in the image. N_i is the number of the pixels in area i. Clearly, $f_{i,j}$ is in the range from 0 to 255.

$$d_i = \frac{\sum_{x,y} (|M[y][0][x] - f_{i,0}| + |M[y][1][x] - f_{i,1}| + |M[y][2][x] - f_{i,2}|)}{N_i} \tag{8}$$

where i=1, 2, 3, 4, 5, 6. d_i is in the range from 0 to 382.5. When the value of colour variance in the certain area gets maximum, d_i is 382.5.

(b) Image Features in The Whole Image

After calculating features in divided areas, we collect average colour value F_j and colour variance D in the whole image which are calculated by the following equations:

$$F_j = \frac{\sum_i f_{i,j}}{S}; \{j = 0,1,2\} \tag{9}$$

where F_j represents the average value in the colour channel j of the whole image. S is the number of divided areas in the image.

$$D = \frac{\sum_i (|f_{i,0} - F_0| + |f_{i,1} - F_1| + |f_{i,2} - F_2|)}{S} \tag{10}$$

where D is in the range from 0 to 382.5.

(c) Experience Construction

In our system, the invariant features of images includes $f_{i,j}$, d_i , F_j , and D. All the features are calculated from images collected in certain places where the robot needs experience to help. We construct experience database embedded in robot's memory. This database stores the feature along with the global coordinates of the position where the image is taken. All the features are calculated off-line and stored in the database as experience. When the experience module is activated, the feature of current image taken by camera is computed on-line notated as *imageFeature* which includes average colour value $Q\_f_{i,j}$ and colour invariance Q_d_i in the divided areas, average colour value Q_F_j and colour invariance Q_D in the whole image. Meanwhile, the record notated as *bestRecord* whose feature is most similar to *imageFeature* is selected from the database. Fig. 1 shows the result of finding the best pose in database based on experience. The query image is on the left while its most similar image in the database is on the right. Their poses are represented by (x, y, θ). x, y are calculated in millimeter, while θ is in degree. Algorithm 1 presents how to calculate the difference *Diff* between the image for query and the image in database, where A_1, A_2, A_3, A_4, B, C_1, C_2 are control constants.

(a) (b) (c) (d)

Fig. 1. Examples for finding the best pose in image database. Images in the database are collected in the areas of the field where the robot can not see any landmark every 100mm in x, 100mm in y and 45° in θ. (a) is the current image taken by robot's camera when its real position is (−1660, 1520, 135°). (b) is the most similar picture to image (a) in the experience database which the corresponding position of the robot is (−1600, 1500, 135°). The location error is 60mm in x, 20mm in y, and 0° in θ. (c) is the random sample image taken after (a) when the real robot position is (−1040, 1220, 135°). The location error in experience image (d) is 240mm in x, 120mm in y, and 0° in θ

When the experience module is activated, difference between *imageFeature* and the feature of *bestRecord* is calculated. If the difference is small enough, the pose of *bestRecord* is transferred

into *bestPose* notated as l_{best} which is in the form of world coordinates in the robot system. With such *bestPose*, probabilities of all the sample poses are updated and new pose templates which are random poses near the *bestPose* are generated to perform the resample procedure in Markov localization. It is true that the more experience in database, the more precisely the calculation is. However, building such database is expensive in time cost and even unreachable in complex environments. As a part of the sensor update module, experience can help the Markov localization converge as soon as possible, which means the robot can know own position immediately. In our approach, we only need to construct the database in those really difficult situations. This method works well in real robot applications.

(d) Self Learning in Experience Collection

One of the difficulties in applying image-retrieval system into real robot localization is how to collect the experience efficiently and correctly. In our system, we create a self learning method for experience collection. The robot can collect images along with corresponding positions autonomously. When construct the experience database, we use the black-white stripes to adjust robot body which is similar to the one used in gait optimization mentioned in (Röfer, 2004). In the self learning procedure, at first, the robot adjusts its own body to the initial position which is preset by our control system. By using the stripes, the robot walks to the next position and stops to capture images in left and right view respectively as shown in Fig. 2. The black-white stripes help robot go to the preset position precisely.

(a) (b) (c)

Fig. 2. Self learning procedure in experience collection. (a) shows the Black-white stripes for body adjusting. The robot captures image in the left view and right view as shown in (b) and (c) respectively

Algorithm 1. Calculate the difference between the query image and the image in database

```
1: procedure Calculate the difference (query image, database)
2:    for all images in database do
3:       if  <A1 && <A2  then
4:          NumberOfAreasBeOK=0, Diff=0
5:          for  (i=1; i<S; i++)
6:             diff_f[i] =
7:             diff_d[i]=
8:             Diff+=C1*diff_f[i]+C2*diff_d[i]
9:             if  diff_f[i] < A3 && diff_d[i]< A4  then
10:               NumberOfAreasBeOK++
11:            end if
12:         end for
13:         if  NumberOfAreasBeOK > B then
```

```
14:             if  Diff < minDiff  then
15:                minDiff = Diff
16:                bestRecord = the current image in database
17:             end if
18:          end if
19:       end if
20:    end for
21: end procedure
```

2.3 Incorporating Experience in Markov Localization

In Markov localization, every sample pose has a belief which represents the probability of predicted position. In our approach, the sensor module updates the probability using the following equation:

$$p_i(t) = \prod_{j=1}^{K} q_i^{(j)}(t); i \in [1, S] \tag{11}$$

where K is the sum of sensor module types, while S is the number of all sample particles. Every $q_i^{(j)}(t)$ describes the position probability at time t using certain type of perception. Specifically, to incorporate the experience module in Markov localization, we set $q_i^{(j)}(t)$ as the quality for experience perception to every sample pose. The sum of the dimensionless distance and the dimensionless angle between *bestPose* and the sample pose is used as a criterion to update the quality with the fact that the quality is higher if the sample pose is nearer to *bestPose*. The experience quality of every sample pose is in the form of the following equation:

$$q_i^{(k)}(t) = \begin{cases} q_i^{(k)}(t-1) - \eta; v < q_i^{(k)}(t-1) - \eta \\ q_i^{(k)}(t-1) + \xi; v > q_i^{(k)}(t-1) + \xi \\ v; other. \end{cases} \tag{12}$$

where η and ξ are constants used for tuning quality not to change too fast. Thus, the quality can be controlled in a certain range. The criterion v is defined as follows:

$$v = e^{-(\sigma+\tau)^2} \tag{13}$$

Here σ is the dimensionless distance between *bestPose* and current sample pose, while τ is the dimensionless angle. Supposing that current sample pose is l_c (x_c, y_c, θ_c) and the *bestPose* is l_{best} (x_b, y_b, θ_b), then σ and τ are calculated in equations below:

$$\begin{aligned} \sigma &= \frac{\sqrt{(x_c - x_b)^2 + (y_c - y_b)^2}}{D_0} \\ \tau &= \frac{|\theta_c - \theta_b|}{A_0} \end{aligned} \tag{14}$$

where D_0 and A_0 are the constants which are used to control qualities of σ and τ. Normally, σ =0.05, τ =0.1. Moreover, we set sudden increases of both σ and τ in order to reduce greatly the qualities of the sample poses that are far away from *bestPose*. Using such method, the procedure of resample can be more effective and efficient. The useless particles can be eliminated as soon as possible. The time cost of the Markov localization convergence is relatively satisfied. Incorporating experience in Markov localization makes the probability update procedure more robust, especially when collisions or other negative effects occur.

2.4 Collaborative Localization

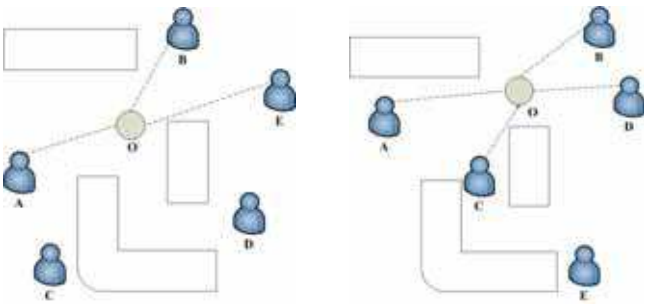
(a) The Notion Of Dynamic Reference Object

In RoboCup, static reference objects like beacon, and goal can be used to help localize in complex environments. However, global coordinates of such objects need to be known beforehand. Those static reference objects are not applicable in an unknown environment. To solve this problem, we propose the concept of *Dynamic Reference Object*. The object that can be detected by more than one robots among the team will be the candidate dynamic reference object. If the frequency of clearly recognizing the object is high enough, it may be set as the dynamic reference object. There is no need to know the object's position as a precondition. If a robot can localize itself accurately, the position of the dynamic reference object calculated by this robot is reliable. Meanwhile, another robot that has seen the reference object can use this calculated position of the object to measure own location. This information is useful for decreasing the time cost of Markov localization convergence and improve the result of position estimate especially for multiple robots collaboration.

There are several challenges to implement this approach in real robot systems. First of all, every robot that has detected the object will broadcast the calculated position to every other robot. Then the robot that needs help may be not able to figure out which position is correct. In addition, the result of the reference object position calculated by a robot may be wrong when another robot needs this information to measure own location. Time delay of the communication is another problem which may bring negative effect to the measurement. To solve problems mentioned above, with the assumption that robots can communicate with each other, our approach integrates *Reference Object Position Possibility* in the team message which will be broadcasted to every robot. The item which is relevant to the object position in team message includes calculated position, robot ID, time, and position possibility. This position possibility is due to the accuracy of the robot self localization. In our system, the object position possibility is notated as P_r is measured by the following equation:

$$P_r = e^{-\mu^2} P_l + e^{-\omega^2} P_e \tag{15}$$

respectively. μ is the sum of lasting frames after detecting the latest landmark, while ω is the sum of lasting frames after exploiting good experience. In real robot application, P_r will be normalized less than 1. If P_r is high enough, the calculated result by this robot will be the most reliable one among different robots perception. A robot that needs help always uses the most possible position of the reference object at the same time when it detects the object by itself. To illustrate the method, a common robot system is shown in Fig. 3 with five mobile robots. Object *O* is supposed to be the dynamic reference object. Table 1 is the real-time information in team message of the system in Fig. 3.



(a) (b)

Fig. 3. A simple system with five mobile robots and a dynamic reference object: (a) At time t_1, robot *A*, *B* and *E* can see the dynamic reference object *O*. They all use their own perception to calculate the position of the object and broadcast to every robot in the team. If at this time robot *A*, for example, needs the reference object to help, A will use the calculated position of the object from *B* or *E*. Querying the most possible position in team message shown in Table 1, *A* will take the calculated result by *B* as the reference. (b) At time t_2, *C* and *D* have not detected any landmark or experience for a period. Thus their answers to the object position are relatively unreliable. Position possibilities of them are shown to be low in Table 1. The reference object position will be set as *B* percepts.

Calculated Position	Robot ID	Time	Position Pssibility
(2388, 700)	A	t_1	0.71
(2264, 658)	B	t_1	0.92
(2530, 710)	E	t_1	0.86
(2368, 803)	A	t_2	0.81
(2401, 801)	B	t_2	0.91
(2103, 743)	C	t_2	0.32
(2215, 725)	D	t_2	0.43

Table 1. Team message relevant to dynamic reference object

(b) Multi-Robot Markov Localization

To illustrate how to integrate the dynamic reference object module in Markov localization, let us assume that robot *i* uses the reference object position calculated by robot *j*. Then robot *i* updates own position belief as follows with a normalizing constant ε :

$$Bel_t^{(i)}(\phi_t^{(i)}) \leftarrow \varepsilon Bel_t^{(i)}(\phi_t^{(i)}) P(\phi_t^{(i)} \mid r_t) Bel_t^{(j)}(\phi_t^{(j)}) \tag{16}$$

where r_t is the dynamic reference object position. The specific probability function using for collaborative approach is similar to the one in the experience model mentioned in equation (12).

In our approach, collaboration is a part of probability update modules in Markov localization. There is a problem that robots should known when to activate the collaboration

module using the dynamic object as a reference. To improve Markov localization using our collaborative approach, the collaboration module will be activated in two situations. We set $N_2(t)$ by using as the sum of lasting frames of having no landmark perception or experience as a condition to activate the collaboration system. If $N_2(t)$ is great enough and the robot has detected the dynamic reference object, the collaboration module will update the probability of every poses. In addition, if the robot has a perception of the object which has a relatively high position possibility, the robot will use this reference to improve the Markov localization in a collaborative way.

2.5 Real robot experiments

(a) Localization Environment

The experience-based collaborative approach presented above has been implemented on the Sony Aibo ERS7 legged robot in RoboCup environment. Fig. 4(a), (b) show the environment in 2006 and 2007 respectively. In our localization experiment field, we use the field similar to the standard field in four-legged soccer field 2007. However, we remove the beacons. As shown in Fig. 4(d), our field is surrounded by colorful advertisement which simulates the real human soccer environment.

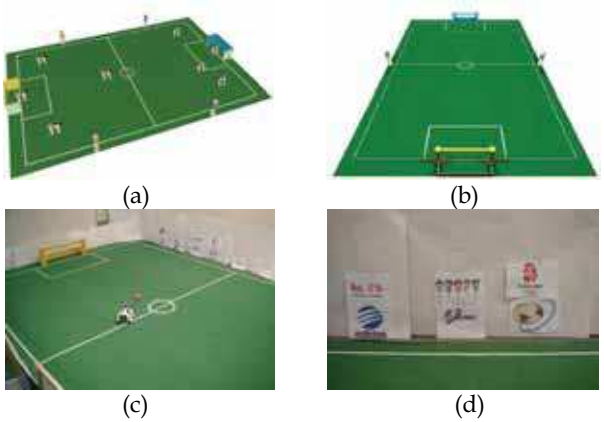


Fig. 4. Experimental field. (a) is the figure which shows the four-legged soccer field with four artificial beacons in 2006. (b) is the soccer field with two colorful beacons in 2007. (c) shows field with no beacon which is used to test our localization approach. (d) is the colorful advertisement placed around our test field which simulates the real human soccer environment

(b) Individual Robot Localization

Go to Certain Position: In the experiment, we use one four-legged robot to perform localization in our test environment shown in Fig.4 (c). Initially, the robot is placed at one center facing out of the field. Then the robot walks to a position with certain global coordinates and body facing angle. On the way to the destination, we pick the legged robot up for a while to effect the odometry in a negative way. This procedure makes the odometry not so reliable to imitate real dynamic environment in soccer competitions. In the experiment, the certain destination position is set to be $(-1450,-300,\ 0^\circ)$. After 42 seconds,

the robot walks to the position (−1380,−350, 6°). The localization error is 70mm in x, 50mm in y, and 6° in body angle.

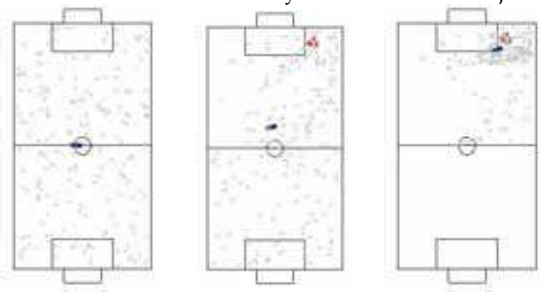
Randomly Walking: The robot is walking on the field with no beacon. We randomly select 8 points to test the self localization results. The robot is expected to go to the preset positions through localization. When it stops, we calculate the real positions on the ground. Table 2 shows the results in detail.

Point Number	Expected Postion (x, y, θ)	Real Postion (x, y, θ)	Error (x, y, θ)
1	(-1290, -440, 15)	(-1496, -713, 147)	(206, 273, 132)
2	(-1450, -300, 0)	(-1410, -150, 0)	(40, 150, 0)
3	(-180, -670, 45)	(-230, -610, 9)	(50, 60, 36)
4	(1430, -250, 55)	(-1909, -1162, 132)	(461, 912, 76)
5	(-650, 170, 0)	(-404, -427, 5)	(246, 597, 5)
6	(270, -480, -90)	(102, -402, -48)	(168, 78, 42)
7	(-1440, -340, 10)	(-1322, -332, 5)	(78, 8, 5)
8	(-2160, -390, 0)	(1979, -454, 8)	(181, 64, 8)

Table 2. Results of self localization in randomly walking. x, y are calculated in millimetre, while θ is in degree

(c) Collaborative Localization

In this experiment, the orange ball used in the four-legged league is considered as the dynamic reference object. We use three robots to perform multi-robot localization. Every robot uses the hybrid system tested in the individual experiment mentioned above. We set one of the three robots as a sample to estimate our collaborative approach. The other two robots move randomly to catch the ball and broadcast the ball position with position possibilities mentioned in section 3. We receive the calculated result from the sample robot. To imitate the outdoor environment, this robot stands in a certain position on the field where we eliminate the landmark which the robot can easily detect. Only experience and collaboration can help the robot localize. The localization result of the sample robot which has used the collaborative approach is shown in Fig. 5. The probability distribution can converges quickly after 3-9 seconds when the dynamic reference object is taken into account.



(a) t=0s (b) t=3s (c) t=9s

Fig. 5. The localization result of applying collaborative approach with dynamic reference object. Solid arrows indicate MCL particles(100). The calculated robot position is indicated by the solid symbol. (a) is the initial uniform distribution. (b) is the calculated result after 3 seconds. (c) is the well localization result after 9 seconds

2.6 Discussion

We have demonstrated an experience based collaborative approach that combines image database for experience without landmarks and real-time sensor data for vision-based mobile robots to estimate their positions under more natural conditions towards real human soccer environment. We used the team message of dynamic reference object to improve the Markov localization for multiple mobile robots. On the one hand, our approach presented a fast and feasible system for vision-based mobile robots to localize in the dynamic environment even if there is no artificial landmark to help. On the other hand, we showed the collaborative method with introduction of *Dynamic Reference Object* to improve the accuracy and robustness of self localization, even in the circumstance that the robot can not localize individually or has no idea of who is nearby. In real robot experiments, we have shown the positive result for legged robot localization using our experience-based collaborative approach. With limit experience, robot can perform better for localization in RoboCup environment. All the experience was collected by the robot autonomously through self learning process. In collaboration, the ball with unique colour was considered as the dynamic reference object. Experiments showed the reliability of our approach in dynamic environment with collisions and sudden position changes. Experiments will be continued in more complex environment with no symmetry.

3. Autonomous Gaits Evolution Using Particle Swarm Optimization

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. Alexander et al., 1980, 1983). To study and implement legged locomotion, various robot systems have been created (eg. Holmes et al., 2006; Raibert, 1986; Collins et al., 2005). 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 (Krasny & Orin, 2004) and real robot applications (Papadopoulos & Buehler, 2000; Hornby et al., 1999; Kim & Uther, 2003). 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 (Eberhart & Kennedy, 1995; Angeline, 1998; Naka et al., 2002) 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 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. (Stewart, 1996) mentioned that there exist eight types of gait patters for quadruped animals in nature. (Hornby et al., 1999) 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. (Röfer et al. 2004, 2005).

In this part, 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 with onboard processor. In robot experiments, we achieved an effective gait faster than previous hand-tuned gaits, using Aibo as the test platform.

3.1 Particle Swarm Optimization

(a) Overview of the Basic PSO

Particle Swarm Optimization (PSO) is a stochastic optimization technique, inspired by social behavior of bird flocking or fish schooling (Reynolds, 1987). It is created by Dr. Eberhart and Dr. Kennedy in 1995 (Eberhart & Kennedy, 1995). 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:

$$v_{id}^{k+1} = v_{id}^{k} + c_1 r_{1d}^{k}(pbest_{id}^{k} - x_{id}^{k}) + c_2 r_{2d}^{k}(gbest_{d}^{k} - x_{id}^{k}) \tag{16}$$

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 (16) 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:

$$v_d^{\max} = k x_d^{\max} \tag{17}$$

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

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{18}$$

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.

(b) Adaptive PSO with Changing Inertia Weight

In equation (16), 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:

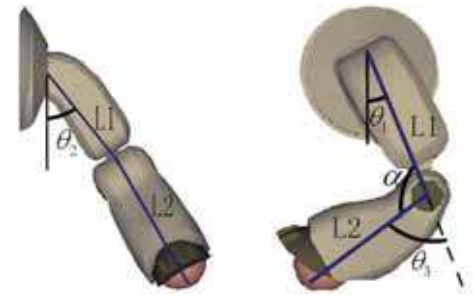
$$v_{id}^{k+1} = wv_{id}^{k} + c_1 r_{1d}^{k}(pbest_{id}^{k} - x_{id}^{k}) + c_2 r_{2d}^{k}(gbest_{d}^{k} - x_{id}^{k}) \tag{19}$$

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.

3.2 Optimization Problem

(a) 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.6 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.



(a) (b)

Fig. 6. 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

The following equations shows the inverse kinematics model:

$$\begin{aligned} x &= l_2 \cos\theta_1 \sin\theta_3 + l_2 \sin\theta_1 \cos\theta_2 \cos\theta_3 + l_1 \sin\theta_1 \cos\theta_2 \\ y &= l_1 \sin\theta_2 + l_2 \sin\theta_2 \cos\theta_3 \\ x &= l_2 \sin\theta_1 \sin\theta_3 - l_2 \cos\theta_1 \cos\theta_2 \cos\theta_3 - l_1 \cos\theta_1 \cos\theta_2 \end{aligned} \tag{20}$$

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

$$\begin{aligned} \theta_1 &= \cos^{-1} \frac{x^2 + y^2 + z^2 - l_1^2 - l_2^2}{2 l_1 l_2} \\ \theta_2 &= \sin^{-1} \frac{y}{l_2 \cos\theta_3 + l_1} \\ \theta_3 &= -\tan^{-1} \frac{a}{b} \pm \cos^{-1} \frac{x}{a^2 + b^2} \end{aligned} \tag{21}$$

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.

(b) 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 forwardleaning posture can speed up the walking, thus we constrain the range of stance parameters to keep robot in forwardleaning 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 3.

Parameter Name	Definition
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

Table 3. Control parameters in gaits evolution

3.3 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.

(a) 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 $(x^{\max}, x^{\min})$ 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 3.1. 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*.

(b) 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.

(c) 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 pbesti, which means the fitness value of P_i is higher than that of pbesti, pbesti 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 pbests 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.

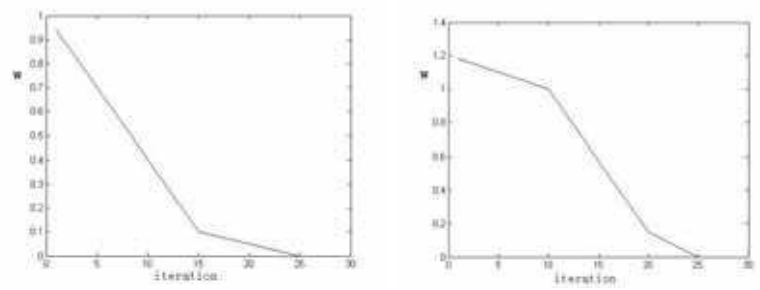
As mentioned in section 3.1, in order to realize global search in a broaden area at the beginning of the learning procedure and intensive search in a currently effective area at the end, we employ adaptive PSO with piecewise linearity declining inertial weight to perform the learning procedure. When inertial weight value ω is around 1, it presents global search characteristics and results in fast converge rate. When ω is a lot less than 1, intensive search is realized.

3.4 Real Robot Experiments

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 (22). Fig. 7(a) shows the vibration of ω through iterations. By iteration 15, ω has decreased to 0.1. The global search is diminished, while the intensive search is enhanced. Fig. 8(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 ω so fast. So we tried to use another equation (23) to update ω, Fig. 7(b) shows the vibration of ω, and Fig. 8(b) shows the learning result.

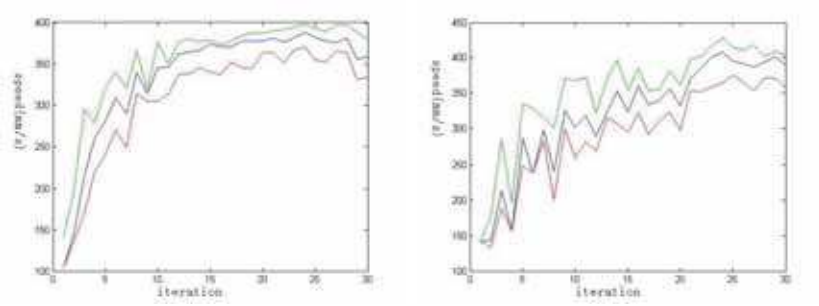
$$\begin{aligned} &\omega = 1 - 0.06 \times iter; (iteration \le 15) \\ &\omega = 0.1 - 0.01 \times (iter - 15); (15 < iteration \le 25) \\ &\omega = 0; (iteration > 25) \end{aligned} \tag{22}$$

$$\begin{aligned} &\omega = 1.2 - 0.02 \times iter; (iteration \le 10) \\ &\omega = 1 - 0.085 \times (iter - 10); (10 < iteration \le 20) \\ &\omega = 0.15 - 0.03 \times (iter - 20); (20 < iteration \le 25) \\ &\omega = 0; (iteration > 25) \end{aligned} \tag{23}$$



(a) (b)

Fig. 7. Vibration of inertial weight ω through iteration in real robot experiments. (a) shows the vibration of inertial weight ω through iterations in the first experiment. (b) shows the vibration of inertial weight ω through iteration in the second experiment



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

Fig. 8. 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

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, when ω becomes 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 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. 9 shows the best result of every iteration in both the two experiments.

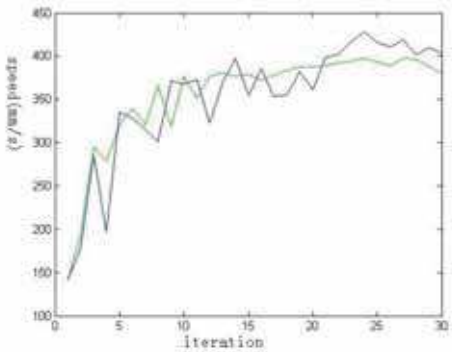


Fig. 9. The best result of every iteration in both the two experiments. The green line is the first one, while the blue line is the second one

3.5 Discussion

In this part, 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 highperformance 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.

4. Intelligent Behaviors

4.1 Obstacle Avoidance

In robot soccer competition, we introduce time-variable limit cycle to help robot avoid obstacles. To show the approach, we simply describe the shape of Aibo as a cycle in the two dimensional plane. Considering the following nonlinear system for dynamic limit cycle applying in Aibo:

$$\begin{aligned} \dot{\tilde{x}} &= \rho(\tilde{y} + \gamma\tilde{x}(\frac{1}{4}\overline{v}^2 - \tilde{x}^2 - \tilde{y}^2)) \\ \dot{\tilde{y}} &= \rho(-\tilde{x} + \gamma\tilde{y}(\frac{1}{4}\overline{v}^2 - \tilde{x}^2 - \tilde{y}^2)) \end{aligned} \tag{24}$$

where ρ is the character factor of the obstacle which is set to be a positive value. γ is the convergence factor. And v is the relative velocity to the obstacle which is dynamic when the robot moves. The size of limit cycle is changing when system (24) switches. To prove the

circle $\tilde{x}^2+\tilde{y}^2=\frac{1}{4}\overline{v}^2$ is the dynamic limit cycle of the switched system(1), we use the common Lyapunov function:

$$V(\tilde{x},\tilde{y})=\tilde{x}^2+\tilde{y}^2 \tag{25}$$

Such that:

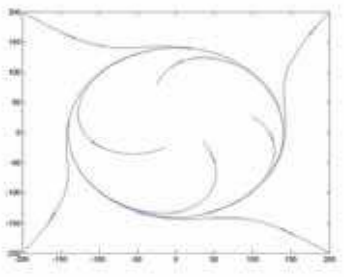
$$\dot{V}(\tilde{x},\tilde{y})=2\rho\gamma(\frac{1}{4}\overline{v}^2-\tilde{x}^2-\tilde{y}^2)(\tilde{x}^2+\tilde{y}^2) \tag{26}$$

For limit cycle, we can see that $\dot{V}(\tilde{x},\tilde{y})<0$ when $\dot{V}(\tilde{x},\tilde{y})>\frac{1}{4}\overline{v}^2$, while $\dot{V}(\tilde{x},\tilde{y})>0$ when $\dot{V}(\tilde{x},\tilde{y})<\frac{1}{4}\overline{v}^2$. This shows the following region is absorbing.

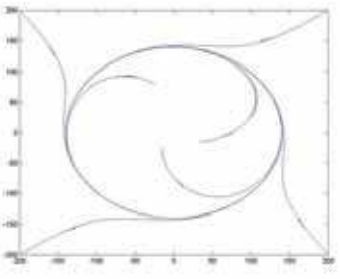
$$B=\{\rho_1\le V(\tilde{x},\tilde{y})\le\rho_2,|\,0<\rho_1<\frac{1}{4}\overline{v}^2,\rho_2>\frac{1}{4}\overline{v}^2\} \tag{27}$$

Since this argument above is valid for any $0<\rho_1<\frac{1}{4}\overline{v}^2$, and $\rho_2>\frac{1}{4}\overline{v}^2$, when ρ_1, ρ_2 get close to $\frac{1}{4}\overline{v}^2$, region B shrinks to the circle $V(\tilde{x},\tilde{y})=\frac{1}{4}\overline{v}^2$. This shows that the circle is a periodic orbit as shown in Fig. 10(a) when $\overline{v}$ = 280, ρ = 0:01, γ = 0:0001. This periodic orbit is called a limit cycle. We can see the trajectory from any point $(\tilde{x},\tilde{y})$ moves toward and converges to the limit cycle clockwise when close. The counterclockwise condition can be derived by the following system (shown in Fig. 10(b)):

$$\begin{aligned}\dot{\tilde{x}}&=\rho(-\tilde{y}+\gamma\tilde{x}(\frac{1}{4}\overline{v}^2-\tilde{x}^2-\tilde{y}^2))\\ \dot{\tilde{y}}&=\rho(\tilde{x}+\gamma\tilde{y}(\frac{1}{4}\overline{v}^2-\tilde{x}^2-\tilde{y}^2))\end{aligned} \tag{28}$$



(a) clockwise



(b) counterclockwise

Fig. 10. Phase portrait of limit cycle

Considering that the trajectory from any point $(\tilde{x},\tilde{y})$ inside the limit cycle moves outward the cycle, and the trajectory from any point $(\tilde{x},\tilde{y})$ outside the limit cycle approaches the cycle with distance determined by the relative speed $\overline{v}$, the limit cycle provides a method for obstacle avoidance among multiple mobile robots.

In RoboCup Four-legged League, there are many obstacles during the game. Robots can be considered as motive obstacles. When the robot approaches a teammate holding ball, it must stay out of the area where teammate handles ball, and be ready to perform cooperative strategies. If the robot holding ball encounters an opponent, it must control the ball and quickly avoid the approaching robot, especially when perform kicking ball in front of opponent goalie. Own penalty area is another one that can be taken for an obstacle. If the robot moves parallel to own ground line, it must avoid from walking into the own penalty area.

When the robot is in a safe region, by the dynamic limit cycle approach, it will move away the obstacle toward the safe circle with a radius relevant to the speed of the obstacle. Let α denote the orientation of the obstacle, (x_0,y_0) the centre point of the obstacle. With the following transformation, we get the expression of system (24) in the original frame:

$$\begin{aligned} x &= \cos\alpha(\tilde{x}+x_0)-\sin\alpha(\tilde{y}+y_0) \\ y &= \sin\alpha(\tilde{x}+x_0)+\cos\alpha(\tilde{y}+y_0) \end{aligned} \tag{29}$$

Let v denote the translational velocity of the robot in the original frame, θ the direction of the motion. The kinematic model of the robot is described by:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \frac{\dot{x}}{\dot{y}} &= \tan\theta \end{aligned} \tag{30}$$

Then we can see:

$$\begin{aligned} v &= \sqrt{\dot{x}^2+\dot{y}^2} \\ \theta &= \arctan\frac{\dot{x}}{\dot{y}}+\alpha \end{aligned} \tag{31}$$

Different obstacles have their own characters, with ρ matching to characters respectively. Using ρ in different values can control the magnitude of the absolute speed.

With the dynamic radius of the limit cycle, robot can perform more flexibly and rationally. Satisfactory results are obtained in robot experiments. The implementation of this method is introduced in (Wang, 2006b) in detail.

4.2 Perform Near Border

In real robot soccer, behaviors and strategies correlated to border line are important. Any inappropriate behavior near border may cause a negative impact. For example, if the ball is

near border in own half field, it is dangerous for the defender to handle ball inappositely to let it out of field. Because it may benefit the opponent striker to control the ball. To avoid this situation, we implement the near border behavior.
In competition of RoboCup Four-Legged League, we define that for a player, if the distance to border line is less than 600mm, it enters the *near border area*. It is simple that if the player handles ball near border, it can hold ball and move it along the direction vertical to borderline. However, actual test shows that different gaits along with grabbing ball motion may not help control ball well. Therefore, we divide the circle area around player into four parts. Fig. 11 shows the different parts of the *near border area* which may activate strategies respectively. We define the variable *robotPose.anlgle-to-border* which represents the absolute value of angle between robot's body direction and normal line to the border. Area 1 is the place where the *angle-to-border* is in range from 120° to 180°. In area 2 and 3, the angle is between 80° and 120°. Area 4 means the angle is less than 80°. In area 1, the robot grabs ball and adjusts its body direction first. Then the robot performs a sideways walk moving ball into field. In area 2 and 3, the robot performs a sideways walk directly. Player walks forward directly to the field if enters the area 4.

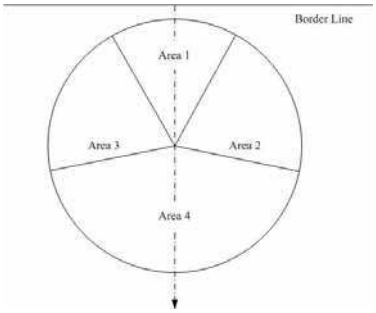


Fig. 11. Strategy field in near border area

5. Conclusion

We have made improvement in localization, locomotion and behavior modules. In RoboCup 2006, we perform our technical improvement in open challenge, passing ball and new goal challenge. After RoboCup 2006, we participated in RoboCup China Open 2006. Advantages in sharPKUngfu 2006 help our team make great success in this event. We got champions both in soccer competition and technical challenge. After the event, we focus our research on further study in collaborative localization, navigation and gaits optimization. All the improvement is explained above in detail. We have applied experience-based collaborative approach for localization which is important to make robots more rational and efficient. In gaits optimization, we implemented PSO based approach to get relatively high-speed forward gaits. To perform better under the soccer rule 2006, new behaviors and relevant actions have been created to hold ball in the field to get better performance. Besides, we tried to apply new approach to percept robots and avoid dynamic obstacles. Experiments in our lab show positive effect by using the real-time approach.
In the future, we plan to let robot play in the environment without any landmark towards real human soccer conditions. Further study should be continued to exploit enough

surrounding information to help self-localization. In vision module, we plan to implement color-edge based method to recognize beacons and goals which are newly defined in soccer rule. Beside of forward gaits optimization, we will implement PSO in other different walking types to gain optimized motion parameters. In multi-robot coordination, the research on formation control will continue. In addition, we will continue to get involved in challenges of passing ball and obstacle avoidance. The final version of our code 2006 is now available on our web site.

6. Acknowledgment

This work was supported by National Natural Science Foundation of China (No. 60404001 and No. 60274001), National Key Basic Research and Development Program (2002CB312200) and 985 Project of Peking University.
The authors gratefully acknowledge the contribution of other team members of the sharPKUngfu Legged Robot Team, including Lianghuan Liu, Yan Huang, Xin Guan, Jiajie Guo and Kaituo Zhou. The source code and experiment video can be downloaded from the sharPKUngfu official web site: http://www.mech.pku.edu.cn/robot/fourleg/.

7. References

Alexander, R. M.; Jayes, A. S. & Ker, R. F. (1980). Estimates of energy cost for quadrupedal running gaits, *J. Zoolog*., vol. 190, pp. 155C192, 1980.

Alexander, R. M. & Jayes, A. S. (1983). A dynamic similarity hypothesis for the gaits of quadrupedal mammals, *J. Zoology Lond.*, vol. 201, pp. 135C152, 1983.

Angeline, P. (1998). Using selection to improve particle swarm optimization, *Proceedings of the International Conference on Evolutionary Computation*, 1998, pp. 84-89

Arkin, R. C. & Balch, T. (1998). Cooperative multiagent robotic systems, In D. Kortenkamp, R. P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*, MIT/AAAI Press, Cambridge, MA

Collins, S.; Ruina, A.; Tedrake, R. & Wisse, M. Efficient bipedal robots based on passive dynamic walkers, *Science*, vol. 307, 2005, pp. 1082-1085.

Eberhart, R. & Kennedy, J. (1995). Particle swarm optimization, *Proceedings of the IEEEConference on Neural Network*, Pelfh, Australia, 1995, pp. 1942-1948

Fox, D.; Burgard, W.; Dellaert, F. & Thrun, S. (1999). Monte Carlo localization: Efficient position estimation for mobile robots, *Proceedings of the National Conference on Artificial Intelligence*, pp. 343-349

Fox, D.; Burgard, W.; Kruppa, H.; & Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization, *Autonomous Robots*, Vol. 8, No. 3, 2000, pp. 325-344.

Holmes, P.; Full, R. J.; Koditschek, D. & Guckenheimer, J. (2006). The dynamics of legged locomotion: Models, analyses, and challenges, *SIAM Review*, Vol. 48, No. 2, 2006, pp. 207-304.

Hornby, G. S.; Fujita, M.; Takamura, S.; Yamamoto, T. & Hanagata, O. (1999). Autonomous evolution of gaits with the Sony quadruped robot. In Banzhaf, W. et al. eds., *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2, 1297-1304. Orlando, Florida, USA: Morgan Kaufmann, 1999.

Kim, M. S. & Uther, W. (2003). Automatic gait optimisation for quadruped robots. *Australasian Conference on Robotics and Automation*, 2003.

Krasny, D. P. & Orin, D. E. (2004). Generating high-speed dynamic running gaits in a quadruped robot using an evolutionary search, *IEEE Transactions on Systems, Man, and Cybernatics - Part B: Cybernetics*, vol. 34, No. 4, 2004, pp. 1685-1696.

Naka, S.; Genji, T.; Yura, T. & Fukuyama, Y. (2002). Hybrid particle swarm optimization based distribution state estimation using constriction factor approach, *Proc. Int. Conf. SCIS & ISIS*, vol. 2, 2002, pp. 1083-1088.

Papadopoulos, D. & Buehler, M. (2000). Stable running in a quadruped robot with compliant legs, *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, 2000, pp. 444-449.

Raibert, M. H. (1986) *Legged robots that balance*, MIT Press, Cambridge; 1986.

Reynolds, C. (1987). Flocks, herds, and schools:Adistributed behavioural model, *Comp. Graph.*, vol. 21, no. 4, pp. 25-34, 1987.

Röfer, T. & Jüngel, M. (2003). Vision-Based Fast and Reactive Monte-Carlo Localization, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 856-861, 2003, Taipei, Taiwan

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. (2004). GermanTeam RoboCup 2004, technical report, Available online: http://www.germanteam.org/GT2004.pdf

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. (2005). *GermanTeam RoboCup 2005*, Technical report, 2005.

Schmitt, T.; Hanek, R.; Beetz, M.; Buck, S. & Radig, B. (2002). Cooperative probabilistic state estimation for vision-based autonomous mobile robots, *IEEE Transactions on Robotics and Automation*, 2002, Vol. 18, Issue 5, pp. 670-684

Shi, Y. & Eberhart, R. (1998). A modified particle swarm optimizer, *Proc. IEEE Int. Conf. Evolutionary Computation*, 1998, pp. 60-73

Stewart, I. (1996). Nature's Numbers. 1996.

Wang, Q.; Liu, L.; Xie, G. & Wang, L. (2006a). Learning from human cognition: collaborative localization for vision-based autonomous robots, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3301-3306, October 2006, Beijing, China

Wang, Q.; Rong, C.; Liu, L.; Li, H. & Xie, G. (2006b). The 2006 sharPKUngfu Team Report, technical report, http://www.mech.pku.edu.cn/robot/fourleg/en/resources.htm

Wang, Q.; Huang, Y.; Xie, G. & Wang, L. (2007). Let robots play soccer under more natural conditions: experience-based collaborative localization in four-legged league, *11th International Symposium on RoboCup 2007* (Robot World Cup Soccer Games and Conference), Lecture Notes in Artificial Intelligence, Springer, 2007.

Wolf, J.; Burgard, W. & Burkhardt, H. (2005) Robust vision-based localization by combining an image-retrieval system with Monte Carlo localization, *IEEE Transactions on Robotics*, Vol. 21, Issue 2, April 2005, pp. 208-216

Robotic Soccer

Edited by Pedro Lima

ISBN 978-3-902613-21-9

Hard cover, 598 pages

Publisher I-Tech Education and Publishing

Published online 01, December, 2007

Published in print edition December, 2007

Many papers in the book concern advanced research on (multi-)robot subsystems, naturally motivated by the challenges posed by robot soccer, but certainly applicable to other domains: reasoning, multi-criteria decision-making, behavior and team coordination, cooperative perception, localization, mobility systems (namely omni-directional wheeled motion, as well as quadruped and biped locomotion, all strongly developed within RoboCup), and even a couple of papers on a topic apparently solved before Soccer Robotics - color segmentation - but for which several new algorithms were introduced since the mid-nineties by researchers on the field, to solve dynamic illumination and fast color segmentation problems, among others. This book is certainly a small sample of the research activity on Soccer Robotics going on around the globe as you read it, but it surely covers a good deal of what has been done in the field recently, and as such it works as a valuable source for researchers interested in the involved subjects, whether they are currently "soccer roboticists" or not.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821