ELSEVIER

Robotics and Autonomous Systems 40 (2002) 69–77

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Behavior generation for a mobile robot based on the adaptive fitness function

Eiji Uchibe [a,*], Masakazu Yanase [b], Minoru Asada [c]

[a] *Human Information Science Laboratories, ATR International, Department 3, 2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan*

[b] *Production Technology Development Group, Sharp Corporation, Osaka, Japan*

[c] *Department of Adaptive Machine Systems, Graduate School of Engineering, Osaka University, Osaka, Japan*

Abstract

We have to prepare the evaluation (fitness) function to evaluate the performance of the robot when we apply the machine learning techniques to the robot application. In many cases, the fitness function is composed of several aspects. Simple implementation to cope with the multiple fitness functions is a weighted summation. This paper presents an adaptive fitness function for the evolutionary computation to obtain the purposive behaviors through changing the weights for the fitness function. As an example task, a basic behavior in a simplified soccer game (shooting a ball into the opponent goal) is selected to show the validity of the adaptive fitness function. Simulation results and real experiments are shown, and a discussion is given.

Keywords: Behavior acquisition; Adaptive fitness function; Multiple optimization problem; Genetic Programming; RoboCup

1. Introduction

One of the ultimate goals of robotics and AI is to realize autonomous robots that organize their own internal structure towards achieving their goals through interactions with dynamically changing environments. In applying some of evolutionary approaches to the robot in order to obtain purposive behaviors, the fitness (evaluation) function should be given in advance. There are two important issues when we attempt to design the fitness function.

First one is that the multiple fitness measures should be considered in order to evaluate the resultant performance. Since multiple objectives may be conflicting with each other, it is usually difficult to obtain the global minimum for each objective at the same time. In order to deal with multiple objectives, several methods are proposed [2]. The weighted sum method is most popular for multi-objective optimization since it is easy to implement and allows to scale objectives. However, this approach faces the essential problem of weighting itself, that is, how to decide the weight values. Another approach is to obtain the Pareto optimal solutions taking advantage of the parallel search of evolutionary computation. However, in robotic applications, it is sometimes meaningless to optimize one of the fitness measures. For example, one of the rational behavior of obstacle avoidance in a static environment is not to move, which is not our intentional result.

In addition, it seems ineffective to fix the weight values during the evolution. In general, when the given tasks are too difficult for the robot to accomplish them, the good evaluation is seldom obtained [1,7,8]. If we set up the severe fitness function at the beginning of evolution, the robot does not obtain

* Corresponding author.
E-mail address: uchibe@atr.co.jp (E. Uchibe).

0921-8890/02/$ – see front matter
PII: S0921-8890(02)00232-4

the good evaluation. As a result, the robot cannot accomplish the task. Therefore, we have to set up the appropriate fitness function according to the current ability of the robot when the robot can obtain the purposive behaviors in a finite learning time. In general, it seems difficult to accomplish the complicated task from the beginning. Asada et al. [1] proposed a paradigm called *Learning from Easy Mission*. Yang and Asada [10] proposed Progressive Learning which would learn a motion to be learned from slow to fast and apply it to a peg insertion task. Omata [6] applied genetic algorithms to acquire the neural network controller which can drive a bicycle. The designer give an initial velocity to the bicycle so as to control it easily. After the generation proceeded, the assist was slightly decreased.

In this paper, we propose an adaptive fitness function which changes the weights during the evolutionary processes. We focus on the correlation between objectives to change the weight. Based on the given priority, the robot modifies the fitness function to control the task complexity. In order to obtain the controller, we select a Genetic Programming (GP) method [4]. GP is a kind of genetic algorithms based on the tree structure with more abstracted node representation than gene coding in ordinary genetic algorithms. As example tasks in our work, we adopt the domain of soccer robots, RoboCup, which is an attempt to foster robotics and AI researches by providing a standard problem where a wide range of technologies can be integrated and examined [3]. In this task, six objectives are considered to evaluate the behavior. We show how the robot would acquire the purposive behaviors using the adaptive fitness function. Finally, the results of computer simulation, real experiments, and a discussion are given.

2. Adaptive fitness function

As described above, adaptive fitness function is effective to accelerate the speed of evolution. Here, we explain the advantage of this method schematically. Let the ith objective function of the individual k at the generation t be $f_i(k,t)$. We assume that f_i is standardized, and positive fitness representation. In other words, the smaller is the better (0.0 is the best). In addition, we introduce a priority function *pr* to define the priority of f_i. The fitness function $F(k,t)$ is computed by

$$F(k,t) = \sum_{i=1}^{n} w_i(t) f_i(k,t), \tag{1}$$

where $w_i(t)$ and n denote the non-negative weight for objective f_i and the number of objectives, respectively. We have to minimize the fitness function F. Obviously, criteria with large weights have more influence on the fitness than those with small coefficients. We focus on the change of each f_i and correlation so as to modify the weights. Technically, all weights are given an initial value and re-setting them happens by adding a value Δw after a certain predefined number of evaluations. We consider the change of the objective functions by

$$\Delta f_i(t) = \frac{1}{N} \sum_{k=1}^{N} \{f_i(k,t) - f_i(k,t-1)\}, \tag{2}$$

where N is the number of population. Since the objectives are standardized, $\Delta f_i < 0$ means that the objective f_i is improved based on the current fitness function F. Therefore, the robot does not modify the weight if $\Delta f_i < 0$ for all $i = 1, \ldots, n$.

The problem is that some objectives get worse, that is $\Delta f_i > 0$. In this case, the weight w_i should be modified to improve the performance of the controller. However, the weights for other objectives w_j $(j \not\equiv i)$ should also be considered since they might be related to each other. Now we consider the relations between two objective functions f_i and f_j. Linear correlation coefficient of f_i and f_j is given by

$$r_{ij}(t) = \frac{\sum_k (f_i(k,t) - \bar{f_i})(f_j(k,t) - \bar{f_j})}{\sqrt{\sum (f_i(k,t) - \bar{f_i})^2} \sqrt{\sum (f_j(k,t) - \bar{f_j})^2}}, \tag{3}$$

where $\bar{f_i}$ and $\bar{f_j}$ denote the average of f_i and f_j, respectively. Based on the linear correlation, the relations between two objectives can be roughly categorized into three: (a) no correlation, (b) positive correlation, and (c) negative correlation. Let $\boldsymbol{C}_i$ be the set of objectives which is related to the ith objective

$$\boldsymbol{C}_i = \{j \,|\, |r_{ij}| > \varepsilon, j = i+1, \ldots, n\}, \tag{4}$$

where ε is a threshold between 0 and 1, and r_{ij} is a correlation between f_i and f_j. In case of $\boldsymbol{C}_i = \emptyset$ (the

objective f_i is unrelated to other objectives), w_i can be modified independently. Therefore, w_i is increased so that f_i would be emphasized:

$$w_i(t+1) = w_i(t) + \Delta w, \tag{5}$$

where Δw is a small positive constant. In case of $\boldsymbol{C}_i \not\equiv \emptyset$, the weight is updated by

$$\Delta w_{j^*}(t) = \begin{cases} 1 & (r_{ij^*} > \varepsilon), \\ -1 & (r_{ij^*} < -\varepsilon), \end{cases} \tag{6}$$

where j^* is prior to other objectives in $\boldsymbol{C}_i$, that is

$$j^* = \arg\max_{j \in \boldsymbol{C}_i} pr(f_i).$$

The reason why w_i is not changed directly is that the weight of the upper objective would continue to be emphasized even if the corresponding objective is saturated. As a result, the lower objective related to the upper one is emphasized directly.

Finally, we summarize our proposed method to modify the weight of fitness function as follows:

(1) For $i = 1, \ldots, n$, update the weights as follows:
   A. In case of $\boldsymbol{C} = \emptyset$, update the ith weight by $w_i(t+1) = w_i(t) + \alpha$, where α is a step-size parameter.
   B. In case of $\boldsymbol{C} \not\equiv \emptyset$, update the j^*th weight by $w_{j^*}(t+1) = w_{j^*}(t) + \alpha \Delta w_{j^*}(t)$.
(2) Create the next population, and increment the generation by $t \to t+1$.

3. Task and assumption

3.1. Environment and robots

RoboCup [3] is an attempt to promote intelligent robotics research by providing a common task for evaluation of various theories, algorithms, and agent architectures. RoboCup has been increasingly attracting many researchers. In order for a robot to play soccer game reasonably well, many technologies need to be integrated and a number of technical breakthroughs must be accomplished. Therefore, we have selected a simplified soccer game consisting of two mobile robots as a testbed to show the validity of the adaptive fitness function. The task for the learner is to shoot a ball into the opponent goal without collisions with an opponent. At the beginning, the behavior is obtained in computer simulation, and we transfer the result of simulation to the real robot.

Fig. 1(a) shows an our mobile robot, a ball, and a goal. The environment consists of a ball, two goals (own and opponent goal), and two robots. The sizes of the ball, the goals and the field are the same as those of the middle-size real robot league of RoboCup Initiative. The robot cannot obtain the complete information about the environment because of limitation of its sensing capability and occlusion of the objects. For example, in order to capture the front view, the robot has a TV camera of which visual angles are 35° and 30° in horizontal and vertical directions, respectively. As motor commands, each mobile robot has two degrees of freedom.

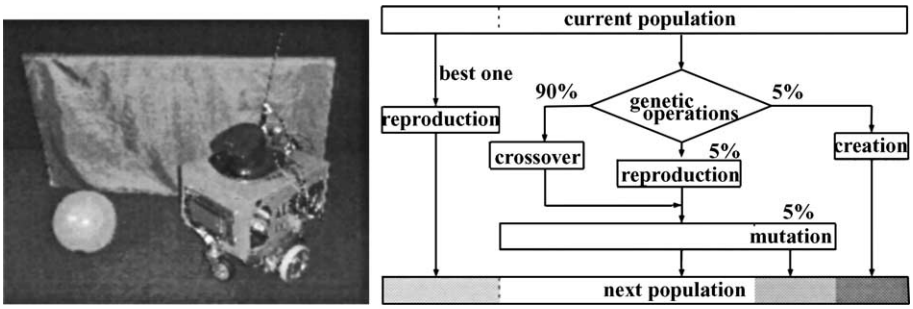

Fig. 1. GP implementation: (a) our mobile robot; (b) flowchart of GP.

3.2. Objective functions

Although shooting behavior is one of the fundamental ability to play a soccer game, many objectives have to be considered. In this experiment, we set up six objective functions: f_{opp} (the number of achieved goals), f_{own} (the number of lost goals), f_{kick} (the total number of ball-kicking), f_{c} (the total number of collisions), f_{step} (the total number of steps until all trials end), and f_{ov}. The first five objective functions are the same as what were used in our previous work [8]. The sixth objective f_{ov} can be regarded as a sub-goal to shoot the ball into the opponent goal effectively. If the ball and the opponent goal are observed in a line, f_{ov} become a small value.

In addition, our proposed method might depend on the priority that is given in advance. In order to check the dependence on the priority function, we prepare four priorities (case A, B, C, and D) as follows:

A. $f_{\mathrm{opp}} \rightarrow f_{\mathrm{own}} \rightarrow f_{\mathrm{ov}} \rightarrow f_{\mathrm{kick}} \rightarrow f_{\mathrm{c}} \rightarrow f_{\mathrm{step}}$
B. $f_{\mathrm{opp}} \rightarrow f_{\mathrm{own}} \rightarrow f_{\mathrm{kick}} \rightarrow f_{\mathrm{c}} \rightarrow f_{\mathrm{ov}} \rightarrow f_{\mathrm{step}}$
C. $f_{\mathrm{opp}} \rightarrow f_{\mathrm{step}} \rightarrow f_{\mathrm{ov}} \rightarrow f_{\mathrm{kick}} \rightarrow f_{\mathrm{c}} \rightarrow f_{\mathrm{own}}$
D. $f_{\mathrm{opp}} \rightarrow f_{\mathrm{own}} \rightarrow f_{\mathrm{ov}} \rightarrow f_{\mathrm{step}} \rightarrow f_{\mathrm{c}} \rightarrow f_{\mathrm{kick}}$

The initial weights for the six fitness measures are set as follows: $f_{\mathrm{own}} = f_{\mathrm{opp}} = 9.0$, $f_{\mathrm{kick}} = 8.0$, $f_{\mathrm{c}} = 4.0$, $f_{\mathrm{step}} = f_{\mathrm{ov}} = 2.0$. They are the best values in our previous experiments using the fixed weight fitness function. The policy to design the priority is explained as follows. Since the main purpose is to shoot a ball into the opponent goal, the objective f_{opp} is prior to all other objectives.

3.3. GP settings

In order to obtain the controller, we select the GP method [4]. In our case one individual corresponds to one controller, and each individual has two GP trees which generate the motor command of the left and right wheel, respectively. GP learns to obtain mapping function from the image features to the motor command. In other words, GP tries to obtain the simple feedback controller.

Then, we select the terminals as the center position of the objects in the image plane. For example, in a case of the ball, the current center position $(x_b(t), y_b(t))$ and the previous one $(x_b(t-1), y_b(t-1))$ are considered. The total number of the terminals is 4 (objects) $\times$ 4 (features) $=$ 16 since the objects in the environment are the ball, two goals and an opponent. Due to severe problems such as the limitation of sensing capabilities, the robot does not always perceive the correct information about the image features. As a function set, we prepare four fundamental operators such as $+$, $-$, $\times$ and $/$.

Fig. 1(b) shows a flowchart to create a new generation. The best performing tree in the current generation will survive in the next generation. The size of the population is set to 150. In order to select parents for crossover, we use tournament selection with size 10. The maximum depth by crossing two trees is 25. We perform $T = 30$ trials to evaluate each robot. The number of generations for which the evolutionary process should run is 200. One trial is terminated if the robot shoots the ball into the goal or the pre-specified time interval expires. The parameter ε in Eq. (4) is set to 0.5. The step-size parameter α is set to 0.02. A time interval is defined as a time period for one action execution corresponding to the sensory input of a robot (33 ms).

4. Experimental results

4.1. Comparison between the proposed method and the fixed weight method

At first, we perform a simulation using a stationary opponent. We compared the proposed method with the fixed weight method. Since the opponent did not move, this experiment can be regarded as an easy situation. Fig. 2(a) shows the result when the opponent is stationary. In the case of the fixed weight method, the performance was not improved after the 50th generation. On the other hand, the performance based on the proposed method was improved gradually.

Next, we show a simulation results using an active opponent. This experiment can be regarded as a more difficult situation. As an initial population for this experiment, we used the best population which was obtained in the previous experiment. Fig. 2(b) shows the histories of f_{opp}. In this experiment, although the opponent just chased the ball, the speed could be controlled by the human designer. Its speed was gradually increased at the 40th and 80th generations, respectively. According to the increase of the speed

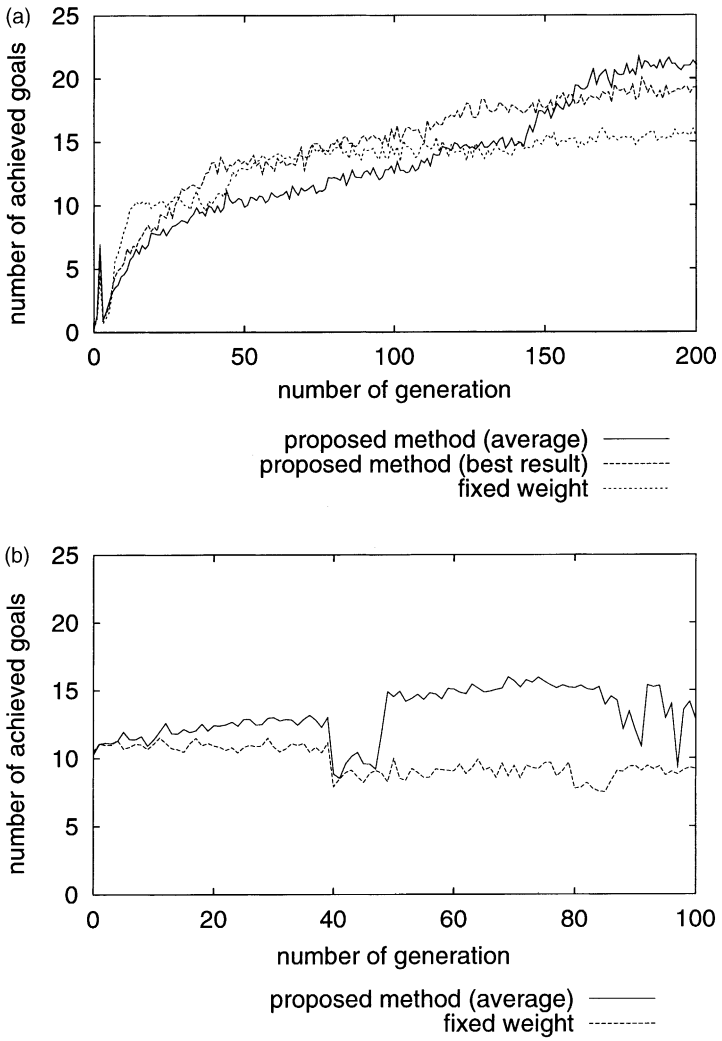

Fig. 2. Average of the number of achieved goals in the first experiment: (a) with the stationary opponent; (b) with the moving opponent.

of the opponent, the obtained scores f_{opp} was slightly decreased in a case of the fixed weight method. On the other hand, the robot using the proposed method kept the performance same in spite of the increase of the speed.

We checked the obtained behaviors based on both methods, and it was found the following issues: with respect to f_{opp} and f_{own}, both methods achieved the almost same performances. In cases of the number of collisions (f_c) and the steps (f_{step}), the performance of the proposed method is better than that of the fixed weight method. We suppose the reason why the robot would acquire such behaviors as follows. At the beginning of the evolution, the most important thing is to kick the ball towards the opponent goal even if it makes a collision with the opponent. Therefore, the weights for f_c and f_{step} are set to small values in a case of the fixed weight method. After a number of generation, the weight for f_{opp} affected the differences of fitness among individuals because most individuals accomplished the shooting behavior. Consequently, the robot using the fixed weight method did not consider f_c and f_{step} through the whole generations. On the other hand, the robot based on the

proposed method changed the weight for f_{step} to be considered. As a result, the robot using the proposed method obtained the shooting behavior more quickly.

4.2. Comparison among the different priority function

Next, we checked how the priority affects the acquired behaviors when we changed the order of the priority, because the priority of fitness measures must be given to the robot in advance. For the sake of the limitation of the space, we show the results of four fitness measures in Fig. 3 using the four priorities. From Fig. 3(a), although the learning curves were different among four cases, the final values converged to the almost same value.

Fig. 3(b) shows the weights of case C during the evolution. Since the weights for the number of achieved and lost goals were constant, only one line was shown in this figure. It follows from the initial weights described in Section 3.2, the important order of fitness measures were described as follows: $f_{\text{own}} \rightarrow f_{\text{opp}} \rightarrow f_{\text{kick}} \rightarrow f_{\text{c}} \rightarrow f_{\text{step}} \rightarrow f_{\text{ov}}$.

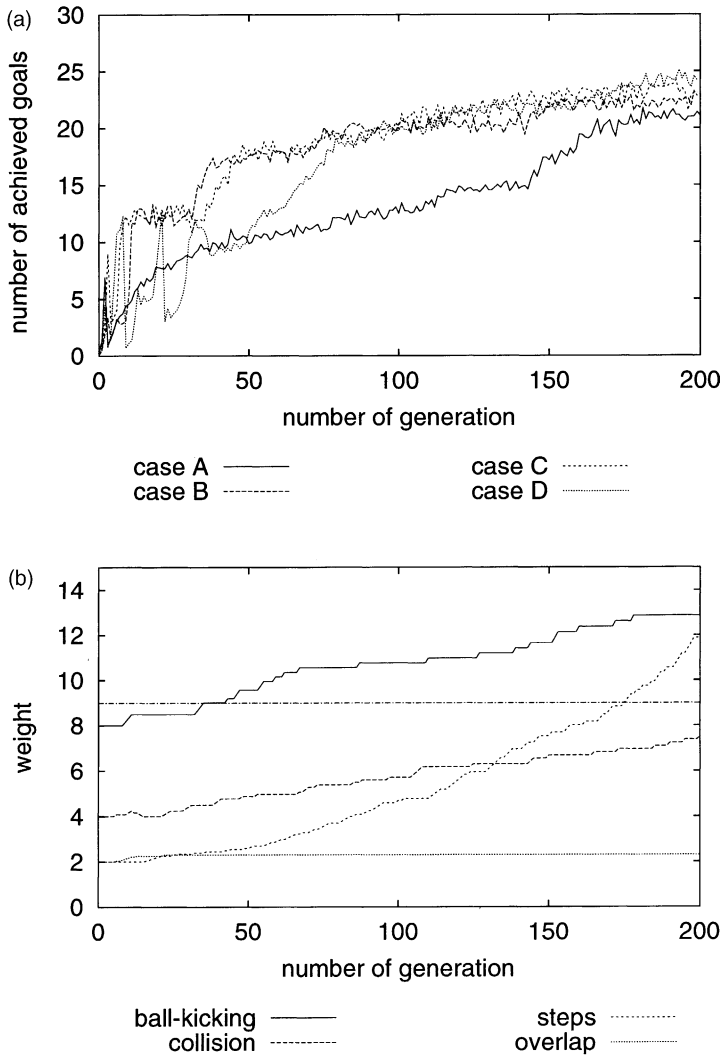

Fig. 3. Comparison among four priority functions: (a) achieved goals; (b) weight based on case C.

Using the adaptive fitness function, the resultant order were described as follows: $f_{kick} \rightarrow f_{step} \rightarrow f_{own} \rightarrow f_{opp} \rightarrow f_{c} \rightarrow f_{ov}$. In many cases we can see that settlement. That is, ball-kicking was emphasized through the evolution. On the other hand, it was not important for the robot to consider the measure about the overlapping degree f_{ov} directly.

4.3. Real experiments

We transfer the controller obtained in the computer simulation to the real robot. A simple color image processing (Hitachi IP5000) is applied to detect the objects in the image plane in real time (every 33 ms). In order to simplify and speed up the image processing time, we painted the ball, the own goal, and the opponent goal in red, blue, and yellow respectively. There is only one robot in the environment because of the limitation of image processing to detect the opponent robot.

Fig. 4 shows the preliminary result of the experiments, that is, one sequence of images where the robot accomplished the shooting behavior. As compared with the behaviors based on our previous methods [1,8], obtained behavior seems very smooth because of mapping from the continuous sensor space to the continuous action space. Our robot participated in the competition of RoboCup 1999 and 2000 which were held in Stockholm and Melbourne, respectively. The evolutionary processes among different priority were slightly different, but resultant performance were almost same in this experiment.

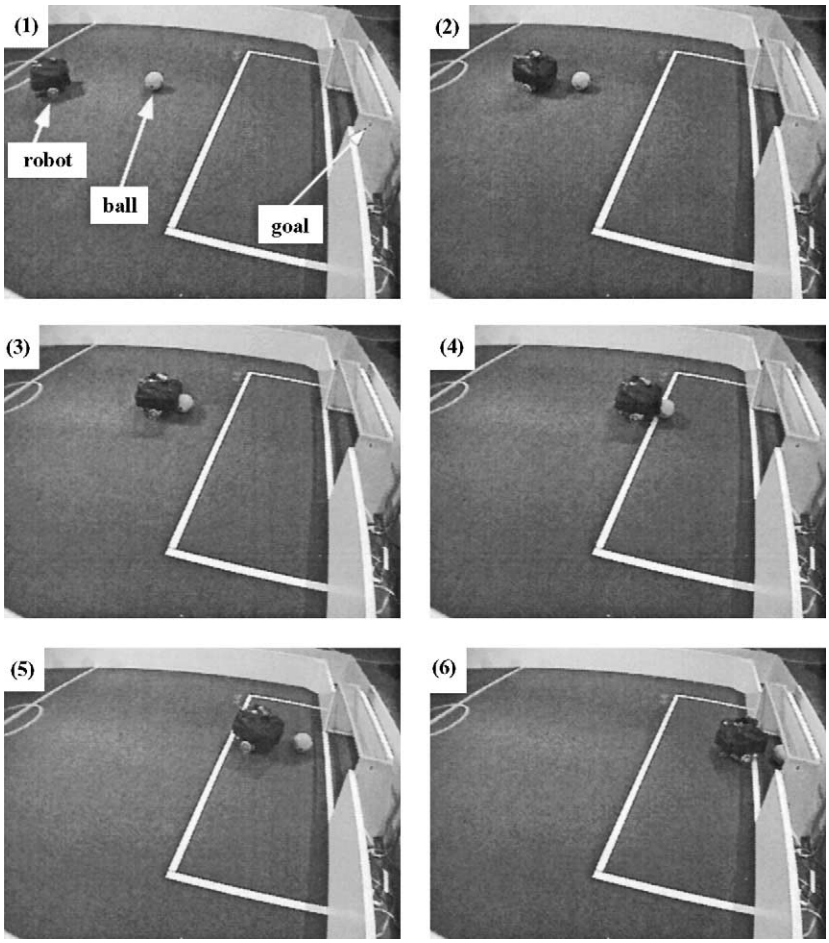

Fig. 4. Typical shooting behavior in the real environment.

Although the real experiments are encouraging, still there is a gap between the computer simulation and the real system. In other words, simulations are necessarily different from the real world. The best evolved controllers obtained by computer simulation may not work in the real world, and the best controllers for the real world may do poorly in simulation. One solution is to evaluate the algorithms using simulation based on the data collected in the real world. In other words, before the simulated evolution, the robot is run and sensor data is collected. Then, the evolution is done off-line in computer simulation. In Martin's work [5], they evolved a good vision filter to estimate the distance to the nearest object. Based on the estimated distance, the robot could avoid obstacles effectively. They reported good results in their own applications. We plan to adopt this approach in the future.

5. Conclusion

This paper presented the adaptive fitness function based on the changes of fitness through the evolution. In consideration of the correlation between multiple fitness measures, the weights for the combined fitness function are updated. We applied the adaptive fitness function method to the simplified soccer game, and showed the validity of the proposed method. The adaptive fitness function is used for more complicated behavior acquisition problem [9].

As a future work, we try to apply our adaptive fitness function to simultaneous evolution of multiple robots. We have already reported how the multiple robots could obtain the cooperative behaviors based on GP with the fixed fitness function [8]. Now, each robot utilize its own fitness to evaluate the behaviors. In order to realize successful co-evolution, fitness of other robots should be considered. This problem can be also regarded as multiple objective optimization problem, we think that we can apply our method into co-evolutionary tasks.

Acknowledgements

This research was supported by the Japan Society for the Promotion of Science, in Research for the Future Program titled Cooperative Distributed Vision for Dynamic Three-Dimensional Scene Understanding (JSPS-RFTF96P00501).

References

[1] M. Asada, S. Noda, S. Tawaratsumida, K. Hosoda, Purposive behavior acquisition for a real robot by vision-based reinforcement learning, Machine Learning 23 (1996) 279–303.
[2] C.M. Fonseca, P.J. Fleming, An overview of evolutionary algorithms in multiobjective optimization, Evolutionary Computation 3 (1995) 1–16.
[3] H. Kitano (Ed.), RoboCup-97: Robot Soccer World Cup I, Springer, Berlin, 1997.
[4] J.R. Koza, Genetic Programming I: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.
[5] M.C. Martin, Visual obstacle avoidance using genetic programming: First results, in: Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, 2001.
[6] T. Omata, Learning with assistance based on evolutionary computation, in: Proceedings of the IEEE International Conference on Robotics and Automation, Leuven, Belgium, 1998.
[7] E. Uchibe, M. Asada, K. Hosoda, Environmental complexity control for vision-based learning mobile robot, in: Proceedings of the IEEE International Conference on Robotics and Automation, Leuven, Belgium, 1998.
[8] E. Uchibe, M. Nakamura, M. Asada, Cooperative and competitive behavior acquisition for mobile robots through co-evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, FL, 1999.
[9] E. Uchibe, M. Yanase, M. Asada, Evolution for behavior selection accelerated by activation/termination constraints, in: Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, 2001.
[10] B.-H. Yang, H. Asada, Progressive learning for robotic assembly: Learning impedance with an excitation scheduling method, in: Proceedings of the IEEE International Conference on Robotics and Automation, Nagoya, Japan, 1995.

Eiji Uchibe received the Ph.D. degree in Mechanical Engineering from Osaka University in 1999. He worked as a Research Associate of the Japan Society for the Promotion of Science, in Research for the Future Program titled Cooperative Distributed Vision for Dynamic Three-Dimensional Scene Understanding. In April 2001 he became a Researcher at the Japan Science and Technology Corporation and worked on the ERATO project. Since October 2001, he has been a Researcher at the ATR Human Information Science Laboratories in Kyoto, Japan. His research interests are in reinforcement learning, evolutionary computation, computational neuroscience, and their applications. He is a member of IEEE, AAAI, RSJ, and JSAI.

Masakazu Yanase received the B.E. and M.Sc. degrees in Mechanical Engineering from Osaka University in 1999 and 2001, respectively. Since April 2001, he has been a Researcher at Production Technology Development Center, Production Technology Development Group, Sharp Corporation. His research interests are in the areas of evolutionary computation, vision-based mobile robots, and their applications.

Minoru Asada received the B.E., M.Sc., and Ph.D. degrees in Control Engineering from Osaka University, Osaka, Japan in 1977, 1979, and 1982, respectively. From 1982 to 1988, he was a Research Associate of Control Engineering, Osaka University, Toyonaka, Osaka, Japan. In April 1989, he became an Associate Professor of Mechanical Engineering for Computer-Controlled Machinery, Osaka University, Suita, Osaka, Japan. In April 1995 he became a Professor of the same department. Since April 1997, he has been a Professor of the Department of Adaptive Machine Systems at the same university. From August 1986 to October 1987, he was a Visiting Researcher at Center for Automation Research, University of Maryland, College Park, MD.

He has received the 1992 Best Paper Award of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-92), and the 1996 Best Paper Award of RSJ (Robotics Society of Japan). Also, his paper was one of the 10 finalists of IEEE Robotics and Automation Society 1995 Best Conference Paper Award. He was a general chair of IEEE/RSJ 1996 International Conference on Intelligent Robots and Systems (IROS-96). Since early 1990, he has been involved in RoboCup activities and his team was the first champion team with USC team in the middle-size league of the first RoboCup held in conjunction with IJCAI-97, Nagoya, Japan. In 2001, he received the Commendation by the Minister of Education, Culture, Sports, Science and Technology, Japanese Government as Person of Distinguished Services to enlightening people on science and technology.